UM ARCABOUÇO PARA A IMPLEMENTAÇÃO DE DATA WAREHOUSES ESTENDIDOS COM TIPOS ABSTRATOS DE DADOS

André Luiz Braga

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO .

Aprovada por:

_____
Prof. Jano Moreira de Souza, Ph.D.

_____
Prof. Geraldo Bonorino Xexéo, D.Sc.

_____
Prof. Claudio Esperança, Ph.D.

_____
Prof. Geraldo Zimbrao da Silva, D.Sc

_____
Prof. Claudia Maria Bauzer Medeiros, Ph.D.

_____
Prof.. Rubens Nascimento Melo, D.Sc.

RIO DE JANEIRO, RJ - BRASIL
SETEMBRO DE 2004

BRAGA, ANDRÉ LUIZ

Um Arcabouço para a Implementação de Data Warehouses Estendidos com Tipos Abstratos de Dados[Rio de Janeiro] 2004

XII, 280 p. 29,7 cm (COPPE/UFRJ, D.Sc., Engenharia de Sistemas e Computação, 1998)

Tese - Universidade Federal do Rio de Janeiro, COPPE

1. Data Warehouse

2. Tipos Abstratos de Dados

3 Data Warehouse Espacial

4. Sistemas de Informação Geográfica

I. COPPE/UFRJ II. Título (série)

DEDICATÓRIA

“Para minha filha Melissa, que você possa se inspirar no meu trabalho e dedicação para crescer ainda muito mais além”

AGRADECIMENTOS

Ao Prof. Geraldo Xexéo pela sua orientação e incentivo de novas idéias e descobertas.

A todo o time de desenvolvimento do DB2 Spatial Extender V8.1 do Silicon Valley Lab da IBM-CA-USA pelo suporte e discussão de idéias que fizeram parte deste trabalho, em especial ao engenheiro Rafael Coss.

Ao prof. Nelson Mattos, do Silicon Valley Lab da IBM-CA-USA, pela orientação e suporte dado no período de pesquisa de tese.

A CAPES pelo auxilio.

Ao CNPq pelo auxílio.

À toda a secretaria do Programa de Engenharia de Sistemas e Computação(PESC) em especial à Solange e à Claudia.

A Patrícia Leal secretária da linha de banco de dados pelo seu auxílio.

A Mercedes, bibliotecária do Centro de Documentação do PESC.

A Fred Furtado, revisor científico da revista Ciência Hoje, pelo trabalho de revisão do texto final.

À minha esposa Soraia, pelo seu apoio e auxílio, principalmente cuidando na nossa filha e da nossa vida em todos esses anos que tive de dedicar ao doutorado.

À minha mãe por todo o incentivo e apoio ao meu estudo e desenvolvimento ao longo de todos esses anos alem de cuidar muito bem da netinha.

Ao meu pai, que postumamente sempre esteve me acompanhando e me incentivando.

A todos os meus amigos e colegas pela paciência e alegria proporcionada nos momentos de intensivo trabalho e estudo.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

UM ARCABOUÇO PARA A IMPLEMENTAÇÃO DE DATA WAREHOUSES ESTENDIDOS COM TIPOS ABSTRATOS DE DADOS

André Luiz Braga

Setembro/2004

Orientador: Geraldo Bonorino Xexéo

Programa: Engenharia de Sistemas e Computação

Os sistemas de *data warehousing* têm a função de extrair e analisar os dados utilizados nos bancos de dados de negócios (vendas, folha de pagamento, estoque de produtos etc.). Existem vários tipos de dados não convencionais já pertencentes aos bancos de dados de negócios, como, por exemplo, tipos de dados espaciais, que podem fornecer valiosas e competitivas informações para estas corporações. Este trabalho propõe a inclusão e utilização de Tipos Abstratos de Dados (TADs) em sistemas de *data warehousing* através de três elementos:

- Um *framework* de definição formal das funcionalidades estendidas do processo de *data warehousing* para TADs.

- Uma metodologia para criação e classificação das funcionalidades estendidas dentro do *framework* proposto.

- Um sistema para geração automática de ferramentas de *data warehousing* para TADs definidos pelo *framework* proposto.

Por fim, mostramos o desenvolvimento completo de um processo de *data warehousing* espacial através da instância espacial do *framework* proposto.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

EXTENDED DATA WAREHOUSING MODEL FOR ABSTRACT DATA TYPES

André Luiz Braga

Setembro/2004

Advisor: Geraldo Bonorino Xexéo

Department: Computing and Systems Engineering

The main goal of data warehousing systems is to extract and analyze large business databases (sales, payrolls etc.). There several non-conventional data types already contained in the business databases, such as spatial data types, that can provide a lot of competitive information to business analysts.

This work shows how to integrate and use Abstract Data Types (ADTs) with the data warehousing process by using the following elements:

- A framework that defines the extended functionalities of the data warehousing process for ADTs.

- A methodology to create and classify the extended functionalities into the proposed framework.

- A system for the automated generation of data warehousing tools for any ADT defined by the proposed framework.

Finally, we detail the development of a complete spatial data warehousing system using the proposed framework.

ÍNDICE **Página**

Índice de figuras: **Página**

Índice de Tabelas: **Página**

PARTE I

Fundamentos e revisão bibliográfica

Capítulo 1 – Introdução

Nos últimos anos, a aplicação de sistemas de bancos de dados trouxe ferramentas que possibilitaram a exploração dos dados de negócios de forma a aumentar as vantagens competitivas. Um dos mais usados são os Sistemas de *Data Warehousing* (Gupta 1997). Eles tem a função de extrair os dados utilizados dos bancos de negócios (vendas, folha de pagamento, estoque de produtos etc.) e agregá-los num modelo analítico e temporal. Os dados agregados possibilitam aos analistas de negócios obter valiosas informações e projeções que irão guiar as estratégias para a corporação.

1.1 – *Data Warehousing*

Os Sistemas de *Data Warehousing* são compostos de duas partes, a extração e agregação dos dados, chamada de ETL (*Extraction, Transformation* e *Loading*), e a parte de análise multidimensional ou OLAP (*On-Line Analytical Process*). Os módulos que compõe a ETL cuidam basicamente de importar os dados de diversos bancos de dados. Algumas vezes, eles são bancos legados a executar junções (*JOINs*) nas várias tabelas do modelo de forma a construir um modelo centralizado que será usado para análise. O módulo de análise irá permitir o cruzamento de dados a partir dos vários elementos do modelo e sua visualização ao longo do tempo. Estes sistemas são otimizados para trabalhar com grandes volumes que crescem ao longo do tempo, assim como permitir a navegação entre valores agregados e detalhados muito rapidamente. Por exemplo, podemos ver uma planilha com todos os produtos vendidos por cada vendedor em cada loja ao longo do ano e rapidamente verificar o valor agregado por mês para cada vendedor para cada loja nos últimos dez anos.

1.2 – Tipos Abstratos de Dados (TADs)

O paradigma orientado por objetos se tornou o modelo mais aceito hoje em dia no desenvolvimento de aplicações. Ele chegou ao modelo relacional de bancos de dados através dos gerenciadores de bancos de dados objeto-relacionais (*ORDBMS*). Estes tipos de bancos de dados permitiram a inclusão de objetos de mais alto nível dentro do modelo de dados, assim como funções ou métodos de manipulação. Vários destes chamados Tipos Abstratos de Dados (TAD) têm sido construídos e usados no fluxo de transações de várias aplicações. Um processamento de informações de terminais de telefonia, por exemplo, pode modelar todas as regras de inserção e remoção de terminais baseados em um TAD representando um “Terminal”, o qual já conteria informações de localização, responsável, cliente etc.

1.3 – *Data Warehousing* para TADs

Alguns TADs já são conhecidos e bastante aplicados, tais como: tipos de dados espaciais, imagens, textos estruturados, objetos nebulosos etc. Existe um grande número de trabalhos que abordam a utilização específica de alguns destes tipos de dados no processo de *Data Warehousing,* mas sem uma formalização específica sobre a sua utilização. Os tipos de dados espaciais, por exemplo, são a base do desenvolvimento de Sistemas chamados de *Data Warehousing* Espacial. Eles têm por objetivo permitir a extração, manipulação e análise de informações espaciais contidas no banco de dados de negócios, como, por exemplo, obter dados agregados de vendas de um certo produto apenas para os clientes localizados até uma certa distância de cada loja.

A construção de sistemas de *Data Warehousing* voltados para TADs requer suporte para manipulação dos TADs no SQL, além de uma série de ferramentas, e suporte de funções para extração, transformação, análise e visualização dos dados. Atualmente, é possível encontrar estas ferramentas, ou partes delas, mas não temos um produto ou metodologia para integrar e organizar estas ferramentas e funções dentro de um processo geral e padronizado de *Data Warehousing* para TADs.

1.4 – *Data Warehousing* Espacial

Um dos TADs mais conhecidos, ainda que pouco utilizado no processo de *Data Warehousing*, são os Tipos de Dados Espaciais. A IBM(*International Business Machines Corporation*), através do **IBM Silicon Valley Lab** iniciou, em 2001, um projeto, do qual fiz parte, para estudar a integração de dados espaciais em sistemas de *Data Warehousing*. Neste período, verificamos, através de estudos de caso em vários clientes, uma grande necessidade e desejo de explorar melhor a dimensão espacial dos dados de negócios (Coss, Zeidenstein e Gonzales 2001). A solução inicialmente aplicada pela IBM naquele momento, e até hoje em muitos casos, é utilizar o suporte à manipulação espacial do banco de dados. Existem vários produtos atualmente no mercado que oferecem este suporte. Entre eles está o **ORACLE Spatial** (Oracle Corporation 2001), o **Informix Spatial Datablade** (IBM Corporation 2001) e a parceria da ESRI (ESRI 2004) e da IBM com o **DB2 Spatial Extender** (Davis 1998,IBM Corporation 1998,IBM Corporation 2002a). Parte deste trabalho foi utilizada pela IBM para demonstrações do seu conceito do processo de *Data Warehousing* Espacial.

Existem dois grandes problemas com a utilização destas ferramentas. O primeiro é que elas são, na verdade, um grande conjunto de pequenas funções que precisam ser

combinadas e utilizadas de forma a produzir algum resultado útil. Existem poucos trabalhos definindo uma metodologia consistente para orientar esta utilização (Helmer 2001). O segundo problema é que os usuários destes sistemas são analistas de negócios que não estão habituados e nem treinados a trabalhar com consultas espaciais, SQL etc. Por outro lado, estes analistas já conhecem os métodos de *data warehousing* e análise multidimensional. Uma solução é trazer o poder de manipulação espacial para dentro do processo de *data warehousing*.

1.5 – Objetivos do Trabalho

O objetivo final deste trabalho é criar um arcabouço para construção de *data warehouses* e para o processo de análise multidimensional envolvendo TADs e sua semântica específica. A utilização deste arcabouço será demonstrada através da sua aplicação para Tipos de Dados Espaciais, resultando numa arquitetura para processos de *data warehousing* espaciais.

Para atingirmos nossos objetivos, precisamos realizar as seguintes etapas:

1 - Definir uma arquitetura de *data warehousing* que contemple TADs, como, por exemplo, dados espaciais, imagem etc.

2 - Especificar um *framework* para construção de um processo de *data warehousing* para qualquer TAD.

3 - Criar uma ferramenta que permita a construção e visualização de consultas complexas utilizando este modelo.

4 - Mostrar uma aplicação do modelo para criação de um sistema de *data warehousing* Espacial utilizando tipos de dados espaciais.

1.6 – Metodologia

Para realizar estas etapas teremos de estudar e pesquisar várias áreas, bem como ferramentas. Não existe nenhum trabalho descrevendo a aplicação genérica de TADs em sistemas de *data warehousing*, mas existem pequenas fundações incompletas que se espalham em trabalhos de diversas áreas, como linguagens de consulta, ferramentas de ETL etc. Os diagramas da Figura 1 e Figura 2 foram desenvolvidos por este trabalho como uma forma de melhor explicar o relacionamento entre as diversas referencias e fundamentos pesquisados. A Figura 1 mostra um diagrama destas áreas e de como elas estão relacionadas. O diagrama da Figura 2 mostra a evolução do diagrama no sentido de atingir o objetivo desta tese.

Como podemos ver no diagrama da Figura 1, nossa pesquisa relativa a *data warehousing*(Figura 1**[A])** se divide em processos de construção do banco de dados de análise (Figura 1**[B]**) e a análise propriamente dita dos dados (Figura 1**[C]**). A utilização e o estudo dos metadados (Figura 1**[D])** serão aplicados nas duas áreas para descrever como utilizar os TADs nas ferramentas de ETL e na modelagem multidimensional. Iremos explicar mais detalhadamente seu papel no processo de *data warehousing* no Capítulo 2.

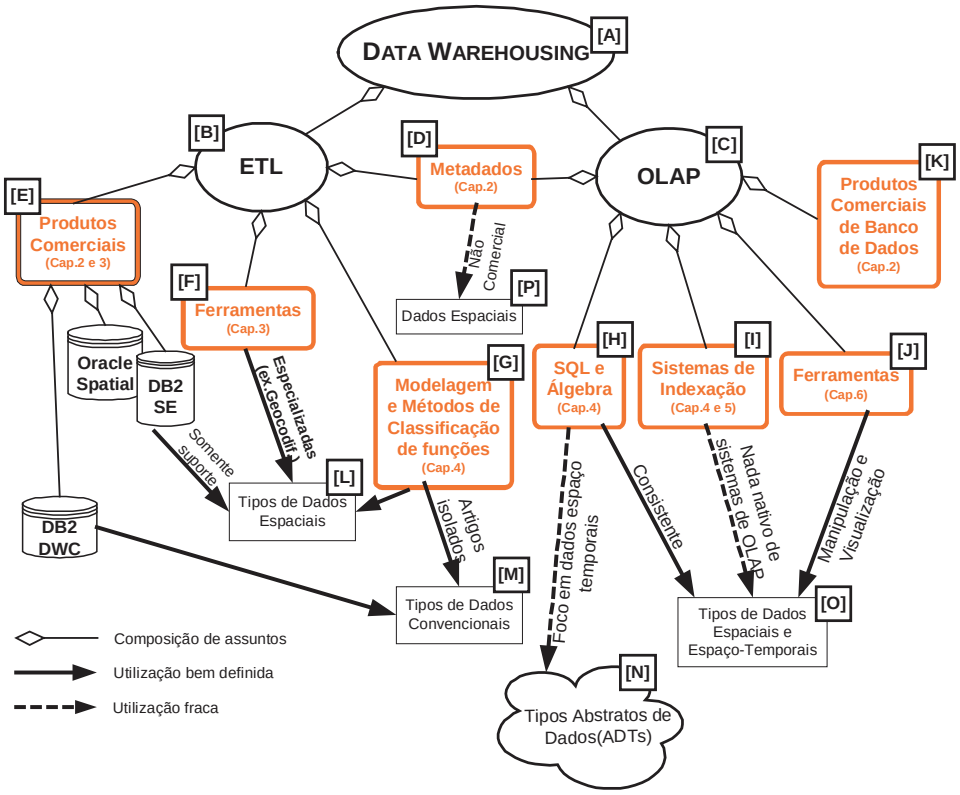

Figura 1 – Base de pesquisa para TAD Data Warehousing

Existem vários produtos comerciais (Figura 1**[E]**) para construção do processo de ETL. Nós iremos estudar as funcionalidades e propriedades deles no Capítulo 2. O exemplo que iremos mostrar para o modelo proposto utilizará um produto que fornece suporte para manipulação espacial no banco de dados (DB2 SE - IBM Corporation 2002a). Por isso, no Capítulo 3, detalharemos este produto, mostrando um modelo para o processo de *data warehouse* Espacial.

As ferramentas não comerciais para ETL (Figura 1**[F]**) serão melhor referenciadas no Capítulo 2 (Rifaieh e Benharkat 2002,Vassiliadis at al. 2000,Vassiliadis at al. 2003,Vassiliadis, Simitsis e Skiadopoulos 2002). De um modo geral, elas visam a

construção do processo em si e não dão suporte a TADs genéricos. Em alguns casos, temos ferramentas especializadas para a funcionalidade de ETL aplicadas a tipos de dados espaciais (Clarke 2000,Meador 2001,Trillium e Harte-Hanks Data Technologies 1999).

A proposta apresentada por nós não visa especificamente a construção do processo de *data warehousing,* mas a geração automática da ferramenta de construção e análise que irá usar as funções de manipulação de TADs definidos por pacotes de extensão do banco de dados. Estas ferramentas não farão parte da nossa proposta final, pois sua pesquisa teve por finalidade apenas descobrir se já existiam ferramentas que fornecessem a mesma solução deste trabalho. De fato, a nossa ferramenta será uma combinação dos tipos de ferramentas para ETL (Figura 1**[F])** e para OLAP(Figura 1**[J]),** como veremos a seguir.

As pesquisas correspondentes ao item **[G]** da Figura 1 estão divididas em métodos de modelagem multidimensional e ETL. Não encontramos nenhum trabalho abordando uma modelagem que utilizasse TADs genéricos(Figura 1**[N]**), mas temos pesquisas ligadas à classificação de funções e tipos de dados espaciais (Han, Stefanovic e Koperski 1998a,Kache 2001), e estudos relativos ao mapeamento complexo de atributos de tipo convencional (Rifaieh e Benharkat 2002). Estes estudos serão utilizados como base para a nossa classificação das funcionalidades de ETL e OLAP para TADs.

A análise multidimensional OLAP requer o estudo de álgebra e linguagens de consulta (Figura 1**[H]**), métodos de indexação (Figura 1**[I]**) e ferramentas de modelagem e análise (Figura 1**[O]**). As álgebras e linguagens de consulta voltadas para análise multidimensional, de um modo geral, fornecem formas de incluir as hierarquias e as dimensões dentro dos operadores da álgebra ou das sentenças da linguagem (Abelló, Samos e Saltor 2003,Pedersen, Riis e Pedersen 2002). Os trabalhos pesquisados em álgebras para OLAP não mostraram qualquer aplicação para TADs, mas pudemos encontrar, além de fundamentos gerais para utilização de TADs em bancos de dados (Guttag 1977,Liskov e Zilles 1974,Stonebraker 1986), alguns artigos voltados para tipos de dados espaciais (Lema e Güting 2002) e outros, mais recentes, não voltados para OLAP especificamente (Seshadri, Livny e Ramakrishna 1997). Em um trabalho de interesse (Skejellaug 1999), se faz uma análise do poder de manipulação de TADs no SQL através de funções definidas pelo usuário ou de extensão de operadores da álgebra. Em todos os casos temos enfoques diferentes, mas o *framework* de comparação apresentado usa tipos de dados temporais como exemplo, os quais são muito limitados se comparados às

possibilidades de um TAD mais geral. De todo modo, utilizaremos estes estudos para mostrar como os TADs e suas funções especializadas definidas pelo usuário podem ser inseridas dentro da linguagem de consulta ao banco de dados.

Existem várias questões complexas também no processo de indexação nos sistemas multidimensionais, pois cada valor analisado ("*Medida*" na tabela de fatos – seção 2.2.3) é indexado por, em média, três a quatro entidades ("Dimensões" – seção 2.2.3). Os sistemas de indexação são destinados a tipos de dados convencionais, mas podemos encontrar alguns estudos (Han, Xu e Lu 1990,Murai, Miyakoshi e Shimbo 1989,Papadias at al. 2002,Papadias, Kalnis e Mamoulis 1999,Rao at al. 2003,Rauber, Tomsich e Riedel 2000,Wang at al. 2003,Zhou e Prasher 2004) aplicados a tipos de dados espaciais, bem como nebulosos. Nos trabalhos de Chen at al. 1999 e DeBloch at al. 2001 podemos ver que é possível criar e inserir sistemas de índices definidos pelo usuário na maioria dos bancos de dados comerciais. Nós assumiremos que a generalidade dos TADs requer a construção de módulos específicos de indexação baseados nas suas características específicas de distribuição e ordenação.

Os produtos comerciais para OLAP (Figura 1**[K]**) estão geralmente associados às ferramentas de ETL, como no caso do DB2 OLAP Starter Kit. Estes produtos são fechados e otimizados para tratamento de dados convencionais, portanto estarão fora do escopo deste trabalho. Esperamos que nossa contribuição possa ser interessante para incorporação em produtos comerciais no futuro.

As ferramentas propostas para OLAP giram em torno de fornecer facilidades para a construção do modelo "estrela" e suas variações, como o modelo "floco de neve" (ver seção 2.2.3). Encontramos exemplos aplicados somente a tipos de dados espaciais ou espaço-temporais (Kemp e Lee 2000,Marchand, Bedard e Edwards 2001).

Este trabalho visa unificar as técnicas de classificação e os trabalhos sobre álgebras e sistemas de indexação para análise multidimensional (Figura 1), de forma a produzir um sistema gerador de ferramentas para ETL e OLAP para TADs (Figura 2). Iremos mostrar agora a convergência dos assuntos e ferramentas tratados ao longo do desenvolvimento deste trabalho (Figura 1) para o escopo final (Figura 2).

A estudo de metadados neste projeto (Figura 2**[D]**) visa desenvolver um modelo geral (Figura 2) para inclusão de TADs no banco de dados de *data warehousing*. Não encontramos qualquer trabalho de descrição de repositório, manipulação ou estrutura de

metadados genérica para TADs, mas alguns produtos, como o DB2 SE (IBM Corporation 2002a), possuem um catálogo para tipos de dados espaciais. Baseado neste tipo de estrutura, desenvolvemos um catálogo genérico para qualquer TAD. Os metadados também são utilizados para descrever o mapeamento de operações de transformações entre tabelas. O mapeamento proposto aqui é um pouco menos complexo que em Rifaieh e Benharkat 2002, mas, além de levar em conta TADs, a geração automática de transformações a partir do SQL pode ser feita a partir de funções primitivas definidas pelo usuário.

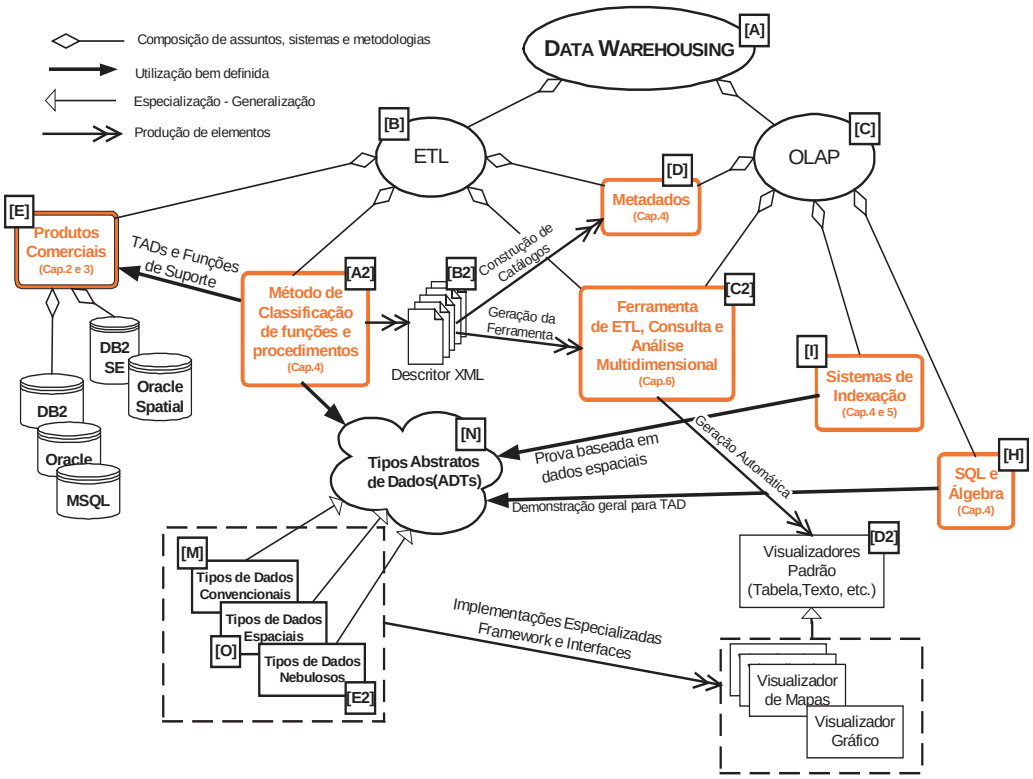

Figura 2 – Proposta de tese para TAD data warehousing

Os produtos estudados para construção de ETL (Figura 2**[E]**) serão usados como base para a ferramenta de ETL que é gerada automaticamente pelo nosso sistema. Os produtos de suporte a dados espaciais, por outro lado, serão usados para desenvolvimento da aplicação de exemplo do modelo.

O método de classificação de funções que vai fazer parte do *framework* de construção do processo de *data warehousing* para TADs, representado na Figura 1**[A],** será desenvolvido a partir do estudo dos métodos de classificação na Figura 1**[G]** e das ferramentas na Figura 1**[F]** e Figura 1**[J]**. A classificação apresentada em Han, Stefanovic e Koperski 1998a apenas especifica o retorno e a parametrização de funções para tipos de

dados espaciais, não mencionando também a cardinalidade dos parâmetros. Nós iremos considerar TADs genéricos, a cardinalidade máxima como binária e ainda as combinações entre TADs (Figura 2**[N]**) e tipos convencionais. O modelo apresentado poderá ser aplicado a qualquer conjunto de TADs (Figura 2**[O]**,**[M]** ou **[E2]**) e será definido por um arquivo descritor em formato XML (Figura 2**[B2]**). Este arquivo será enviado para o processador de metadados, para registro das informações, e para o gerador da ferramenta de ETL e consulta (Figura 2**[C2]**).

A ferramenta de ETL e OLAP gerada (Figura 2**[C2]**) irá também construir visualizadores padrão para os TADs. Visualizadores para as especializações dos TADs poderão ser desenvolvidos a partir de interfaces fornecidas junto com o sistema. Este sistema de geração e a ferramenta final produzida têm base em algumas filosofias dos trabalhos pesquisados, como, por exemplo, a construção automática dos procedimentos de transformação (Rifaieh e Benharkat 2002). A ferramenta gerada terá recursos para construção do processo como em Vassiliadis, Simitsis e Skiadopoulos 2002 e Vassiliadis at al. 2003, mas com o foco voltado para a utilização de TADs e suas funções de manipulação específicas. Por final, ofereceremos um sistema de geração, não existente em qualquer dos trabalhos ou produtos pesquisados, capaz de produzir a ferramenta de ETL e OLAP necessária para qualquer TAD.

1.7 – Organização do Texto

Na primeira parte (Capítulo 2 e Capítulo 3), vamos mostrar os fundamentos de *data warehousing*, assim como os de Bancos de Dados Espaciais. Este último tema será o foco principal do nosso exemplo de aplicação da metodologia e do sistema apresentados. Iremos mostrar a geração de uma ferramenta de *data warehousing* espacial a partir da nossa metodologia.

Este trabalho está dividido em três etapas. Na primeira etapa, mostrada no Capítulo 4 e desenvolvida no Capítulo 5, iremos mostrar uma metodologia e um *framework* para inclusão de qualquer TAD dentro de um processo de *data warehousing*. Na segunda etapa, no Capítulo 6, mostraremos um sistema de geração de ferramentas para processamento de *data warehousing* para TADs. Na última parte, no Capítulo 7, vamos mostrar a aplicação da metodologia e a ferramenta gerada para o exemplo de tipos de dados espaciais.

O Capítulo 8 apresentará as conclusões do trabalho.

Capítulo 2 – Sistemas de Data Warehousing

2.1 – Introdução

Data warehousing não pode ser completamente definido como uma "técnica", "modelo", "sistema", "ferramenta" ou até mesmo "conceito". As definições estão mais voltadas para cada campo de trabalho dentro de *data warehousing* (Devlin 1997a,Gray e Watson 1998,Kimball 1996,Tomlinson 1987). Na interpretação de Inmon 1992, considerado o "pai" do *data warehousing*, este é um "modelo orientado ao assunto de uma coleção de dados integrados e variantes no tempo". Por outro lado, Tomlinson 1987 e Kimball 1996 têm um foco muito forte na utilização de ferramentas de análise de *data warehousing*. Sob outra perspectiva, *data warehousing* pode ser considerado um sistema de suporte à decisão projetado para agregar e criar visões dos dados de forma a permitir um nível alto de gerência.

Nós consideramos que *data warehousing* pode ser definido de qualquer uma destas formas, dependendo de seu contexto. Neste trabalho usaremos o termo "processo de *data warehousing*" para especificar todos os passos para geração do banco de dados, incluindo a análise de negócios feita ao final. Enquanto que para "sistema", ou banco de dados, usaremos o termo "Banco de *Data Warehouse*".

Os itens abaixo constituem as características fundamentais do processo de *data warehousing* (Barquin e Edelstein 1997,Berson e Smith 1997,Devlin 1997b,Edelstein 1996,Edelstein 1997,Gupta 1997,Inmon 1992,Kimball 1996,Mimno 1997):

1 - **Orientado ao assunto** – O modelo do banco de dados não segue a modelagem convencional de negócios. O modelo segue um assunto específico que se deseja analisar. Por exemplo, se desejarmos tomar decisões sobre as vendas de algum produto, os dados de venda do produto estarão no centro do modelo com junções apenas para as entidades relevantes para minha análise, como, por exemplo, lojas e tipos de produto. Neste caso, a posição do gerente de cada departamento não tem a mínima importância na análise. A orientação ao assunto em sistemas de *data warehousing* requer perda de normalização e traz vários problemas de desempenho, visto que todos os dados de um assunto ficarão concentrados em uma só tabela. Isto vai ser resolvido por mecanismos de armazenamento especializados, como será visto mais adiante.

2 - **Integrado** – As fontes de dados podem se originar de diversas bases e modelos, por vezes heterogêneos. Portanto, o sistema de *data warehousing* deve ser capaz de integrá-los num modelo homogêneo provendo um acesso transparente ao processo de análise.

Existem várias funcionalidades e ferramentas que um sistema de *data warehousing* deve ter para realizar esta tarefa, como a formatação de dados segundo um padrão específico.

3 - **Temporal** – O banco de dados de *data warehouse* freqüentemente constitui-se de uma coleção de informações obtida incessantemente ao longo do tempo. Deste modo, o volume de dados torna-se muito grande, o que acarreta vários problemas de desempenho.

4 - **Não volátil** – O banco de dados de *data warehouse* é extraído do banco original de produção. Em vista disto, é necessário que os dados no banco de *data warehouse* sempre contenham o valor dos dados de produção. Esta característica também poderia ser chamada de "sincronismo".

5 - **Grande Volume de Dados** – A duas causas disso são a características temporais apresentadas no item (3) e também a grande quantidade de informação existente nos bancos de dados. Esta última é uma característica do alvo dos processos de *data warehousing*. Eles são geralmente destinados a bancos de dados de grandes corporações, cuja grande área de atuação naturalmente gera um enorme volume de informações e dados.

2.1.1 - Vantagens e Desvantagens

Sistemas de *data warehousing* e análise *on-line* (OLAP) representam metodologias e processos que nem sempre podem ser mensurados objetivamente com relação a sua utilidade e desempenho. Sakaguchi 1996 conduziu e classificou 456 publicações entre abril de 1992 e Julho de 1996, bem como compilou uma enorme lista de referências citando vantagens e desvantagens do processo de *data warehousing.* Nós citamos aqui algumas das mais importantes:

VANTAGENS

1 - **Simplicidade** - Ë a vantagem mencionada mais frequentemente. *Data warehousing* provê uma imagem única da realidade de negócios, integrando as várias dimensões de dados.

2 - **Rápido acesso aos dados –** Os usuários têm a capacidade de obter os dados necessários, centralizados num só lugar.

3 - **Isolamento do Suporte à Decisão –** O banco de *data warehouse* é montado separadamente do banco de produção. Desta forma, decisões podem ser tomadas sem interferir com as atividades correntes

4 - **Gerenciamento do Fluxo de Informação –** O banco de *data warehouse* é isolado, mas, por outro lado, é sincronizado com o banco de produção. Isto permite que os analistas possam responder rapidamente às mudanças e necessidades de negócios

DESVANTAGENS

1 - **Complexidade no desenvolvimento** – Os bancos de *data warehouse* que irão dar suporte ao processo de análise têm de ser desenvolvidos especificamente para cada negócio e para os vários assuntos a serem analisados. Este tempo inicial, apesar de só necessitar ser feito uma vez, precisa obter um conhecimento profundo dos assuntos de negócios

2 - **Custo de construção –** O modelo e o processo de *data warehousing* requerem planejamento e desenvolvimento que, na maioria das vezes, demanda profissionais especializados. Além disso, é necessário extrair todas as informações de negócios para construção do modelo.

3 - **Treinamento dos usuários finais –** Os usuários, no caso analistas de negócios, precisam aprender a trabalhar com este novo tipo de paradigma. Este tempo é necessário, apesar de hoje em dia termos grupos já bem formados de profissionais preparados para trabalhar com este tipo de sistema de informação

2.2 – Funcionalidades

Existem quatro passos no processo completo de *data warehousing* (Figura 3). Os primeiros três passos, Extração (*Extraction*), Transformação (*Transformation*) e Carga (*Loading*), ou ETL, são mais comumente associados ao processo de construção do banco de *data warehouse,* enquanto que a análise é referida como parte do OLAP (*On-Line Analytical Process*). A Extração e a Transformação são responsáveis por trazer os dados para o banco de *data warehouse* e integrá-los de forma consistente. A Carga se refere a inserção dos dados no modelo multidimensional (seção 2.2.3), levando em conta o histórico dos dados e os assuntos a serem analisados.

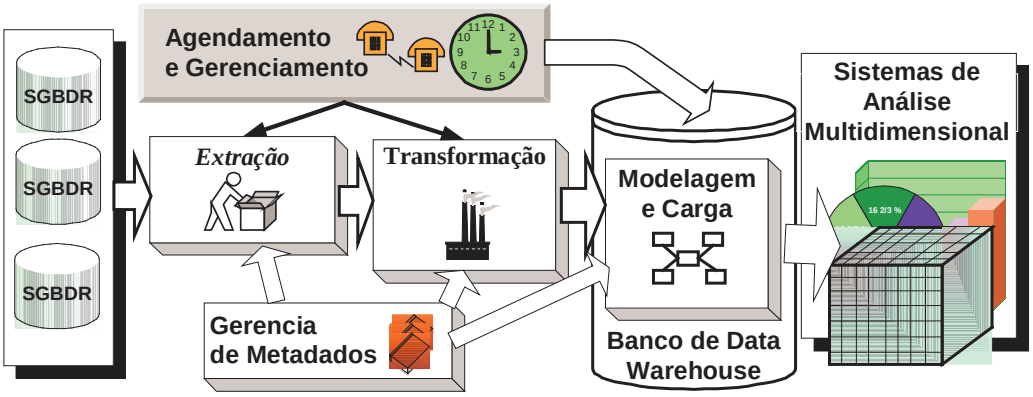

Figura 3 – Fluxo de processamento de *data warehouse*

A análise pode ser feita simplesmente através de consultas e ferramentas de análise estatísticas da ferramenta de *data warehousing,* ou usando sistemas avançados de análise multidimensional, ou ainda sistemas de OLAP (Berson e Smith 1997,Pendse 2001,Sarawagi, Agrawal e Gupta 1999b,Sarawagi, Agrawal e Gupta 1999a,Stefanovic 1997a). Estes sistemas utilizam um modelagem especial, cujo foco é a percepção do usuário dos dados. Este modelo é construído selecionando-se um atributo, chamado de *medida*, e todas as entidades que "*dimensionam*" esta medida. A tabela contendo a medida é chamada de tabela de fatos. Por exemplo, se tivermos uma medida "*vendas*", algumas dimensões poderiam ser "*tempo*", "*filial*" etc. Este modelo é chamado de "*Modelo Estrela*", porque a medida, ou tabela de fatos, que desejamos analisar fica no "centro" do modelo e as entidades de dimensão ficam a sua volta. Veremos este modelo, e suas variantes, como o "*floco de neve*", em maior detalhe na seção 2.2.3.

2.2.1 – Extração de Dados

A primeira fase do processo de *data warehousing* é a importação dos dados. Este processo consiste em ler os dados das múltiplas fontes, registrá-los no diário temporal, limpá-los, val idá-los e, por fim, integrá-los de forma consistente. Podemos dividir este processo em três passos (Berson e Smith 1997,GarterGroup 1998,Inmon 1992):

- Importação
- Limpeza
- Validação

IMPORTAÇÃO

Muitas tecnologias de bancos de dados vêm se desenvolvendo nos últimos 20 anos e os dados de grandes companhias estão invariavelmente espalhados em várias

tecnologias de banco de dados diferentes e nos mais diversos formatos. Um dos grandes desafios é integrar todos estes tipos de dados, mas o maior deles é entender o grande modelo de negócios. As definições dos dados estão, na maioria das vezes, perdidos ou as pessoas que as desenvolveram não estão mais disponíveis. Existem também dados que jamais foram gerenciados por qualquer sistema de banco de dados ou pertencem a tipos não convencionais de dados como, vídeo, áudio, dados espaciais, ou até mesmo a WEB (Lowe 2000,Morimoto at al. 2003). Dentre as questões mais importantes relativas à importação de dados, podemos citar (Berson e Smith 1997,Devlin 1997a):

Relativas à fonte de dados:

- **Mudança de Tecnologia -** Em muitos casos, não só o banco de dados, mas o próprio sistema operacional difere do usado no banco de dados de *data warehouse*, por isso são necessários meios de interconectar diferentes sistemas de arquivo, gerenciamento de índices e outras tecnologias.
- **Múltiplos Formatos –** Alguns dDSAfados poderão estar em formatos específicos de acordo com a sua tecnologia de armazenamento, ou mesmo por suas características complexas (vídeo, texto etc.)
- **Associação de Múltiplos Arquivos –** Os dados poderão estar espalhados em múltiplas fontes ou arquivos com regras específicas de associação. Por exemplo, se um tipo de pagamento é "mensal", isso pode indicar que ele tem que ser associado com os arquivos de clientes mensais.
- **Seqüência de Entrada –** Alguns arquivos podem requerer a entrada no sistema numa ordem específica e, em alguns casos, uma dependência cíclica pode ter de ser tratada.
- **Seleção Complexa de Dados Desejados** – Os dados podem estar armazenados num modelo complexo não normalizado, que pode requerer algoritmos especiais para obter os dados desejados.

Relativos à estrutura dos dados:

- **Extração dos dados alterados** – O processo de *data warehousing* deve armazenar os dados gerados ao longo do tempo, portanto não é desejável inserir dados que já tenham sido armazenados. Existem fontes de dados que não têm informação de

tempo, por isso as ferramentas de importação devem ter recursos para registrar o momento de inserção dos dados.

- **Critério de conversão** – Nós podemos ter diferentes fontes de dados com a mesma estrutura ou padrão de importação. A ferramenta de importação deve ter um registro que permita reutilizar o mesmo padrão para diferentes fontes de dados.
- **Nomes de Atributos -** Os dados extraídos e integrados no banco de *data warehouse* irão, muito provavelmente, mudar para um modelo diferente do modelo de produção. Este mapeamento precisa ser registrado e apresentado futuramente no processo de análise.

LIMPEZA

A limpeza dos dados tem por finalidade garantir que "*o mesmo pedaço de informação seja referenciado de uma mesma forma*" (Gray e Watson 1998). A situação mais comum está nas abreviações e sinônimos, por exemplo, "*Av*." ou "*Avenida*", "*M*" ou "*Masculino*". É necessário utilizar uma representação única para os dados em cada caso (Berson e Smith 1997):

- **Abreviações e Sinônimos** – Uma abreviação pode ser sido substituída pelo nome completo no processo humano de entrada de dados ou por especificações de formato. Em outros casos, a falta de informação pode ter permitido a entrada de palavras similares, como "*identidade*" ou "*registro*". É relevante dizer que a fase de limpeza é feita em nível de dados e não de nome de atributos. Se tivermos dois atributos, "*Sexo*" e "*Gênero*", por exemplo, não é função da fase de extração mixá-los.
- **Sistemas de dados incompatíveis** – A representação dos dados pode estar diferente por causa da variação em todo o domínio. Por exemplo, conjuntos de caracteres japoneses e ocidentais, ou escalas de medidas que estão em pés e precisam ser convertidas para metros.
- **Formatos dos dados** – Os dados podem estar na mesma estrutura e domínio, porém em diferentes formatos de representação, como duas datas estarem escritas como "*primeiro de dezembro de 1999*" ou "*1 DEZ 1999*".
- **Formatos dependentes de dados** – Em alguns casos o formato dos dados pode estar preso ao valor de um outro atributo. O próprio domínio de um atributo, como o padrão da moeda, pode estar definido em outro atributo.

VALIDAÇÃO

A maioria dos autores não menciona a fase de validação (Gray e Watson 1998). Sua finalidade maior é checar as falhas das fases anteriores. Ela garante que os dados extraídos irão manter o mesmo formato e domínio para cada carga de dados. Os principais itens a serem avaliados são:

- **Metamodelo da Aplicação** – Esta seria uma validação semântica. Os valores dos atributos não podem violar regras de formação do domínio. Por exemplo, "*idade*" não pode ter valores negativos.
- **Checagem de Versão** – Alguns tipos de dados podem ter diferentes versões de um mesmo formato, mas isto pode afetar também a semântica de alguns atributos. Dados relativos a impostos, por exemplo, podem variar alíquotas entre um período e outro.
- **Completude** – É necessário checar se todos os valores necessários estão presentes. Um certo atributo pode requerer por exemplo, que todos os dias da semana contenham dados.
- **Conformidade** – Este tipo de validação checa se houve mudanças bruscas ou inesperadas no fluxo dos dados. Por exemplo, os dados de venda de uma filial não poderiam deixar completamente de existir num certo mês.
- **Checagem de *DrillDown*** – Este tipo de validação ocorrerá mais freqüentemente de forma interativa com o usuário. Se este perceber que valores agregados parecem fora do padrão, ele terá a possibilidade de olhar os valores detalhados da agregação.

A fase de extração deve ser tratada com muito cuidado, pois todas as coisas que podem sair incorretas nesta fase irão se propagar para a análise dos dados. Este é um dos motivos pelos quais todos as ações e modificações feitas nessa fase devem ser registradas no módulo de metadados.

2.2.2 – Transformação e Integração

Este passo irá criar um modelo para o banco de dados de *data warehouse*. Cada fonte de dados irá ser transformada para construir um banco de dados único, integrado e consistente. Alguns autores definem várias categorias de transformação dentro do processo de ETL. Este trabalho compila várias classificações existentes (Computerwire 1996,Devlin 1997b,Inmon 1992,Vassiliadis at al. 2003,Vassiliadis, Simitsis e Skiadopoulos 2002), nos concentrando na proposta por Devlin 1997b, na seguinte lista:

- Seleção

- Separação e Concatenação
- Normalização e Desnormalização
- Sumarização e Agregação
- Conversão

Enriquecimento

SELEÇÃO

Esta é a forma mais simples de transformação, aonde os dados são particionados de acordo com um critério de seleção qualquer. Em alguns casos, esta operação pode já ser feita na processo de extração.

SEPARAÇÃO E CONCATENAÇÃO

Estas transformações são similares, respectivamente, a uma projeção e uma junção na álgebra relacional (Codd 1970). A separação extrai, ou divide, atributos dos dados ou da tabela do modelo. Não é incomum termos tabelas com grande quantidade de atributos desnecessários e desejarmos separar em várias tabelas que serão usadas em diferentes assuntos de análise.

A concatenação é mais comum quando importamos dados de bancos de dados já normalizados. A informação desejada está por vezes espalhada em várias tabelas e desejamos juntá-las sob um assunto específico.

NORMALIZAÇÃO E DESNORMALIZAÇÃO

Estas categorias são bastante similares às anteriores, mas a diferença jaz na manipulação das chaves de relacionamento. A normalização vai separar os dados em mais de uma tabela, mas mantendo um relacionamento consistente entre elas através das chaves.

A "*desnormalização*" ocorre quando fazemos a junção de tabelas numa grande tabela. É o caso mais usual, quando criamos a tabela de fatos a partir das tabelas de dimensões (veja seção 2.2.3).

SUMARIZAÇÃO E AGREGAÇÃO

Esta categoria corresponde à construção de consultas que resultarão na agregação de todo, ou parte, do banco de *data warehouse* ao longo do tempo, ou em outra dimensão. As ferramentas de *data warehousing* também deverão ser capazes de

desagrupar os dados de forma que eles possam ser vistos em resumo ou em detalhe. Isto é chamado de "*RollUp*" e "*DrillDown*", respectivamente.

A "*Granularização*" é a medida de nível de detalhe (*DrillDown*) que pode ser executada (Devlin 1997b). Quando nós agregamos dados, perdemos um nível de informação que só pode ser recuperado através dos dados originais, mas isto custa memória e desempenho. A "*granularização*" de um sistema de *data warehousing* será permitida em troca do desempenho desejado.

CONVERSÃO

A conversão ocorre quando um tipo de dado, por exemplo, texto, é convertido para um outro tipo, como numérico. Isto é bastante comum, dado que os tipos de dados de armazenamento nem sempre refletem a realidade do modelo.

ENRIQUECIMENTO

O enriquecimento ocorre mais freqüentemente na fase de limpeza (veja seção 2.2.1). A função desta categoria de transformação é aumentar a complexidade do atributo. O exemplo mais comum é unificar atributos para criar novos, mas também é possível ocorrer a troca de formatos de dados.

2.2.3 – Análise e Modelagem

O processo de análise em sistemas de *data warehousing*, genericamente chamado de OLAP, consiste em uma série de procedimentos para processamento dos dados e apresentação de resultados. Codd 1993 e Codd, Codd e & Salley 1993 propuseram algumas regras e funcionalidades básicas para todas as ferramentas de OLAP:

1 - O máximo de transparência para o usuário
2 - Manipulação intuitiva dos dados
3 - Agregação de resultados
4 - Permitir análise multidimensional

Os três primeiros itens desta lista são bastante diretos e podem ser levados a cabo em qualquer sistema. O último item está profundamente relacionado com a forma de modelagem dos dados (Ballard at al. 1998,Pedersen e Jensen 1998,Sarawagi, Agrawal e Gupta 1999b,Thomas 2001). O modelo relacional normalizado é muito eficiente para as transações no banco de produção (OLTP), mas não é adequado para o processo de análise de *data warehousing*. Os modelos usados em produção são feitos de dezenas de relacionamentos de 1:N para facilitar inserções, atualizações, armazenamento e

indexação, mas as consultas são menos óbvias de se construir. Os sistemas de *data warehousing* são projetados por analistas de negócios e estes modelos de produção não são tão intuitivos para eles. Inmon 1992 propôs uma abordagem utilizando um modelo bem mais simples, o modelo estrela, como mencionado na seção 2.2:

MODELO ESTRELA

O objetivo principal deste modelo é possibilitar o foco sobre uma assunto específico. Este modelo tem dois tipos de entidades: fatos e dimensões. O assunto a ser analisado é centralizado na tabela de fatos, e as dimensões (pontas da estrela) contém os atributos relacionados que irão qualificar os valores da tabela de fatos (Figura 4).

As linhas da tabela de fatos (Figura 5) têm um campo contendo o valor a ser observado e outros contendo chaves estrangeiras para as tabelas de dimensões. Cada linha, ou *tupla,* da tabela de fatos é chamada de "*medida*" do dado sendo analisado. Tomemos o seguinte exemplo: queremos medir as vendas de produtos de uma rede de supermercados. A tabela de fatos conterá a "*medida*" **Venda** e cada produto vendido por um supermercado corresponderá à dimensão **Produto**. O momento da venda corresponderá à dimensão **Tempo**, a dimensão **Cliente** corresponderá àa pessoa que comprou e a filial de compra corresponderá à dimensão **Filial** (Figura 4 e Figura 5). A "*medida*" representará então a renda obtida pela venda de um produto A, para um cliente B, na filial C, no momento D.

A grande vantagem do modelo estrela é que o desempenho de junção entre a tabela de fatos e as tabelas de dimensão é muito boa porque estas são geralmente pequenas o suficiente para ficar em memória (Thomas 2001). A tabela de fatos, por outro lado, é geralmente bem grande e crescente no tempo, e, portanto, o tamanho de cada *tupla* deve ser o menor possível. As tabelas de dimensão são atualizadas muito raramente, porque elas representam as informações de conteúdo fixo ou base para o modelo, como a localização das filiais, os tipos de produtos vendidos etc.

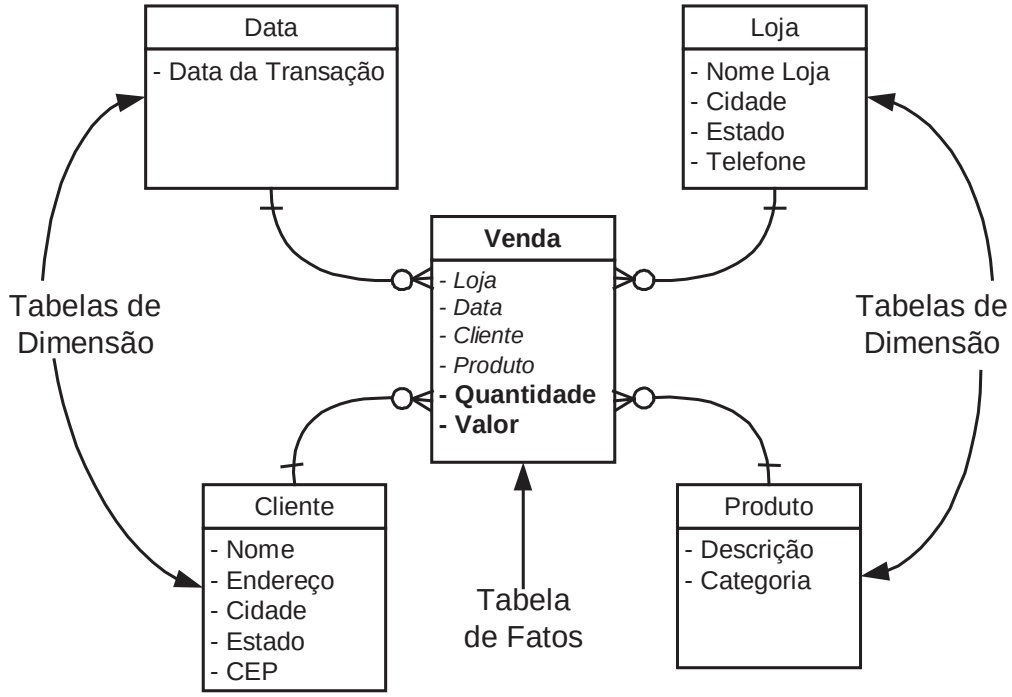

Figura 4 – Star Schema

O modelo estrela é bastante compacto, e portanto a integridade referencial entre a tabela de fatos e a de dimensões deve ser bastante reforçada. A maioria dos sistemas de *data warehousing* mapeiam as consultas ao modelo em SQL utilizando junções (*JOINs*) e técnicas avançadas de indexação para contornar os problemas relativos ao grande volume de dados.

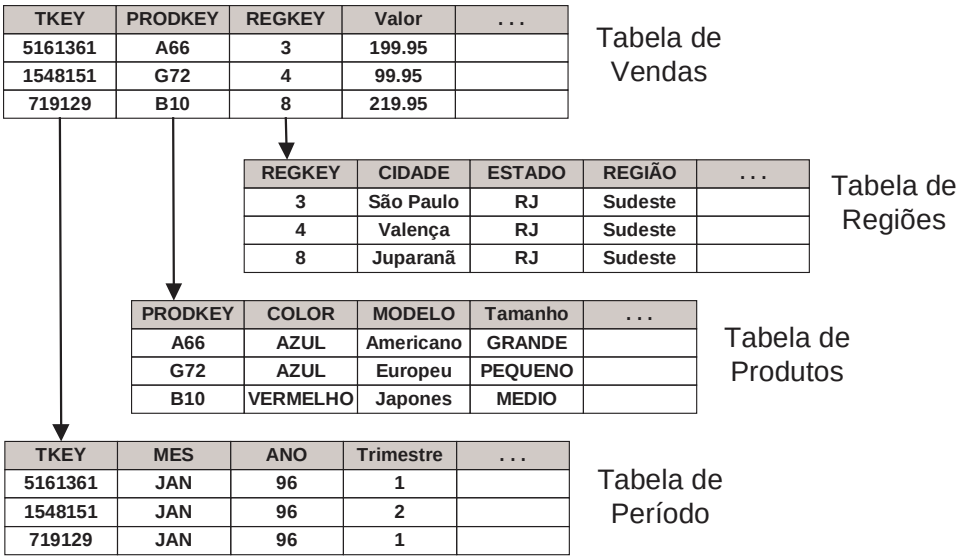

Figura 5 – Star Schema – Perspectiva de tabelas

MODELO “FLOCO DE NEVE”

Este modelo é uma alternativa ao Modelo Estrela e permite que múltiplas tabelas de fatos (Inmon 1992,Kimball 1996,Thomas 2001) associadas às mesmas dimensões criem “famílias de tabelas de fatos” (Figura 6). Esta alternativa é interessante quando temos múltiplos assuntos dentro da tabelas de fatos principais, ou quando a tabela é muito grande, causando problemas de desempenho na sua manipulação.

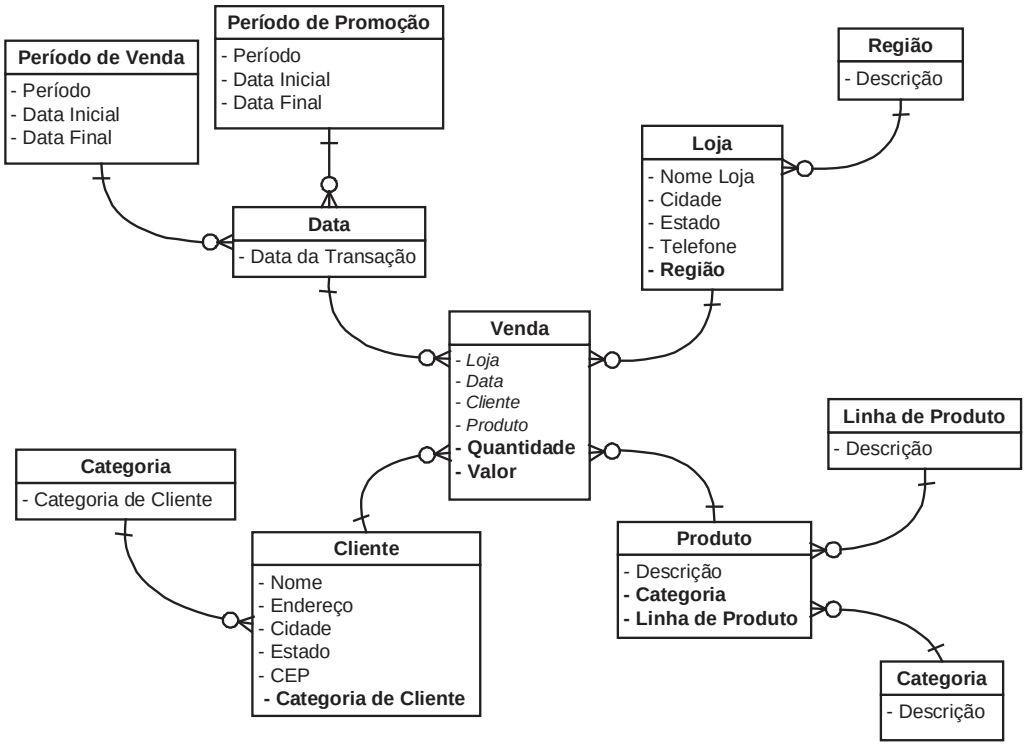

Figura 6 – SnowFlake Schema

DATA MARTS

Os *DataMarts* são pequenos bancos de *data warehouse* -- mais precisamente, grupos de cubos multidimensionais direcionados para um setor específico, como departamentos de uma grande empresa. Isto diminui o custo de um banco de *data warehouse* envolvendo todos os dados da empresa, contudo traz problemas de integração entre os vários *DataMarts*. Uma das soluções para este problema é montar o processo de *data warehousing* como uma coleção de *DataMarts* especializados para cada necessidade (Barquin e Edelstein 1997).

DIMENSÃO TEMPO

A dimensão correspondente ao tempo tem características especiais. Ela controla o crescimento dos dados do banco de *data warehouse* e é uma base para quase todas as

análises realizadas no cubo multidimensional (Devlin 1997a,Thomas 2001). Por exemplo, quase todos os modelos de dados de *data warehousing* estabelecem uma hierarquia padrão de agregação dos dados por ano, trimestre, mês e semana.

2.3 – OLAP

2.3.1 – Operações

Uma grande vantagem dos Modelos Estrela é a flexibilidade de fazer referências cruzadas. Cada dimensão, ou ponta da estrela, age como um foco de controle a partir do qual podemos agregar ou detalhar (*DrillDown*) os valores da tabela de fatos, tomando um valor de dimensão como parâmetro. Vamos supor que desejemos saber os valores das vendas de um certo produto numa certa cidade -- basta executar uma consulta para somar todos os valores da tabela de fatos para a chave das dimensões Cidade e Produto.

A percepção deste modelo é também denominada de cubo multidimensional. Podemos realizar as seguintes operações neste cubo (Abelló, Samos e Saltor 2003,Barquin e Edelstein 1997,Berson e Smith 1997,Carvalho 2004):

- **"*Fatiar*"(ou *slice*)** – Corresponde a uma projeção de apenas alguns vértices do cubo.
- **"Rotação"(ou *dice*)** – É quando projetamos e olhamos o cubo de uma outra perspectiva. Suponha, por exemplo, que queremos ver as vendas por produto ao invés de por filial.
- **Agregação (***Roll-up* ou *Drill-up*) – Como explicado acima, consiste de agregar valores obtendo um nível de generalidade maior.
- **"Detalhar" (***DrillDown***) –** É o inverso da agregação, aonde aumentamos o nível de detalhe de um dado. É importante salientar que a operação inversa não obrigatoriamente precisa retornar os valores originais não agregados. Podemos definir funções que criem um nível de detalhe a partir de um dado baseado em funções genéricas, como distribuições estatísticas ou média.

2.3.2 - Linguagens e Álgebra

Todas as operações realizadas no módulo de OLAP podem se tornar bastante complexas dependendo da sua modelagem (Pedersen e Jensen 1998) e são, de uma forma ou de outra, baseadas na álgebra relacional. Os sistemas chamados de "ROLAP" (*Relational On-Line Analytical Process*) usam de fato transformações das operações no cubo multidimensional em consultas em SQL. Recentes trabalhos de pesquisa propõem

linguagens de consulta específicas para o modelo multidimensional. Pedersen, Riis e Pedersen 2002, por exemplo, desenvolveram uma linguagem e álgebra que incorporam o conceito de níveis hierárquicos de agregação dentro de operadores genéricos de seleção e projeção no cubo multidimensional. A linguagem associa implicitamente as funções de agregação a cada medida da tabela de fatos, enquanto uma outra cláusula identifica quais os níveis que serão agregados. A linguagem apresentada por Abelló, Samos e Saltor 2003 não é tão detalhada com relação à álgebra, mas apresenta uma lista de descrição mais exaustiva da semântica das operações no cubo multidimensional. Eles justificam que uma melhor definição da semântica das operações pode aumentar a possibilidade de otimizar as consultas.

O estudo de linguagens de consulta multidimensionais utilizando TADs, ou *ADTs,* é bastante pequeno (Seshadri, Livny e Ramakrishna 1997,Skejellaug 1999) e, em sua maior parte, é aplicado em dados espaciais ou temporais. A utilização de TADs em linguagens de consulta mais gerais é mais explorada (Chen at al. 1999,Guttag 1977,Güting e Dieker 2000,Liskov e Zilles 1974,Stonebraker 1986), assim como a sua aplicação diretamente para dados espaciais ou espaço-temporais (Bauman 1999,Kolias, Kolias e Yassoglou 1981,Lema e Güting 2002,Papadias, Kalnis e Mamoulis 1999,Portier 1996,Stefanovic 1997a).

Os métodos de indexação para TADs são tratados em Chen at al. 1999 e DeBloch at al. 2001. De fato, a especificação de índices usando o SQL é simples, contudo a sua implementação irá requerer a construção de módulos e estruturas a parte para o gerenciamento específico de cada TAD. Os exemplos encontrados na literatura utilizam, em sua maioria, tipos de dados espaciais (Chen at al. 1999,Han, Xu e Lu 1990,Papadias, Kalnis e Mamoulis 1999) e alguns especificamente aplicados à OLAP espacial (DeBloch at al. 2001,Papadias at al. 2001,Papadias at al. 2002,Rao at al. 2003,Rauber, Tomsich e Riedel 2000,Wang at al. 2003,Zhou e Prasher 2004).

2.4 – Modelagem e Ferramentas

Existem várias ferramentas dos bancos de dados comerciais para modelagem ou construção de bancos de *data warehouse*. Eles podem ser destinados à construção do fluxo de ETL, à análise multidimensional, ou a ambos.

As ferramentas de modelagem multidimensional são mais destinadas à criação das variações do modelo estrela para o cubo multidimensional (Freitas 2001,Rizzi e

Golfarelli 2001). Elas não definem os processos, mas sim as estruturas, relacionamentos e hierarquias do modelo.

As ferramentas de construção de fluxo de ETL apenas permitem a definição dos passos que devem ser executados, assim como os mapeamentos das transformações entre as tabelas de entrada e saída. Os conceitos e especificações dos passos de ETL vêm sendo melhor definidos ao longo do tempo. Vassiliadis at al. 2000 apresentam uma ferramenta que é capaz de modelar todo o fluxo de ETL em um formato de arquivo XML (Light 1997,Maruyama, Tamura e Uramoto 1999) ou declarativo semelhante à DDL. Posteriormente, Vassiliadis, Simitsis e Skiadopoulos 2002 e Vassiliadis at al. 2003 definiram mais detalhadamente os tipos de operação do fluxo de ETL com uma simbologia de construção detalhada e elegante. Com relação ao mapeamento das transformações, Rifaieh e Benharkat 2002 sugerem que muitas vezes a ferramenta não é suficientemente flexível para realizar qualquer mapeamento complexo, como incluir um dado dependendo do valor de um atributo de uma outra tabela, e propõem uma ferramenta de mapeamento semântico.

Os produtos comerciais, por outro lado, parecem ter avançado bastante em flexibilidade. O DB2 *Warehouse Manager*, por exemplo, possui um produto visual para construção do fluxo de ETL (IBM Corporation 2002b), permitindo a inclusão de regras e até mesmo programas definidos pelo usuário. O sistema de controle de metadados (IBM Corporation 2002c) permite, similarmente, a Vassiliadis at al. 2000, a construção de todo o fluxo de ETL através de uma linguagem descritiva. A Tabela 1, modificada a partir da retirada de Rifaieh e Benharkat 2002 mostram alguns produtos mais conhecidos avaliados pelos seguintes atributos:

- O desempenho da ferramenta em relação aos bancos de dados que acessa para extração.
- A capacidade da atualização dos módulos ou funcionalidades da ferramenta.
- A capacidade de reaproveitamento de um plano de extração e transformação.
- A utilização de metadados na construção automática do plano de extração e transformação.
- As habilidades de automação do plano de extração
- Fontes de dados aceitas na extração

A ferramentas comerciais, bem como as propostas no meio de pesquisa, sofrem de duas grandes fraquezas. A primeira é a falta de tratamento de TADs, e a segunda é a falta de flexibilidade de alteração do produto (Rifaieh e Benharkat 2002). O *Query Assist* do DB2, por exemplo, não tem nenhuma forma de ser configurado a não ser por alteração de código e vários processos para requerer alteração no projeto.

Tabela 1- Produtos de ETL comerciais (modificada de Rifaieh e Benharkat 2002)

Produto	DTS	ORACLE DW Builder	Data Satage	SAGENT	DB2 DW Center
Desempenho	Melhor com MSQL Server	Melhor com o ORACLE	Depende da fonte de dados	Depende da fonte de dados	Boa com qualquer banco
Manutenção	Atualizando módulos ActiveX	Compilação	Modificando objetos	Modificando plano de contexto	Modificando plano de contexto
Reusabilidade	Funções Definidas	Funções definidas	Conjunto de objetos ou plano	Plano reutilizável	Plano reutilizável
Acesso de Metadados	Nenhum	Nenhum	Módulo de metadados	Nenhum	*Information Catalog Center*
Plano de Processo de Extração	Plano com *timer*	Plano com *triggers*	Gerenciamento de eventos	Automação SEAGENT	Automático e *timer*
Fontes de Dados e Integração	Txt,ODBC, OLEDB	Arquivos Flat, Oracle, Sybase, ODBC, OLDDB	MSQL Server, ORACLE, DB2, Txt, VSAM, Informix, etc.	MSQL Server, ORACLE, DB2, Txt, VSAM, Informix	MSQL Server, ORACLE, DB2, Txt, VSAM, Informix, ODBC, OLEDB

2.5 - Metadados

O grande número de operações realizadas no processo de *data warehousing* traz a necessidade de garantir a consistência de todas as formatações, transformações e integrações. Os metadados são informações sobre todos dados e processos que ocorrem no processo de *data warehousing*, como por exemplo, que atributos de uma tabela são preenchidos e com que tipos de dados, que transformações são feitas com os dados etc.

Os metadados também permitem que os analistas de negócios entendam os papéis e significados dos dados dentro do modelo de negócios. Um sistema de suporte a metadados que garanta todas estas funcionalidades precisa manter um registro preciso de todos os aspectos (dados, relacionamentos etc.) dos dados e do fluxo de transformações feitas no processo de *data warehousing* (Devlin 1997b).

Os focos principais de utilização de metadados em sistemas de *data warehousing* podem ser classificados em Baixo, Médio e Alto nível (Hufford 1997) nas respectivas categorias abaixo:

- **Construção** – São as informações básicas para construção do banco de *data warehouse* e correspondem às informações relativas às fontes dos dados extraídos, assim como ao registro das transformações realizadas em todo processo, como as **definições** dos tipos de dados, os relacionamentos, as estruturas, as tabelas, os atributos e ***até mesmo as funções e os procedimentos definidos pelo usuário para o tratamento de dados*** [1].

- **Controle** – Contém as informações necessárias para **controlar** o fluxo de processamento do *data arehousing*. Por exemplo, agendar operações, localização de fontes, códigos de acesso etc. Os metadados de controle estão profundamente ligados à dinâmica do processo de *data warehousing*.

- **Utilização** – Os usuários irão utilizar estas informações para obter a **semântica** dos dados de negócios e entender suas estruturas. Esta é a finalidade mais importante dos metadados do ponto de vista de análise. Os metadados de utilização permitem que os analistas de negócios conheçam o significado de cada atributo de cada tabela, ou as operações que podem ser feitas com cada dado. No caso do modelo estrela, é extremamente importante que o banco de *data warehouse* contenha a descrição de

[1] Estas funções e procedimentos são frequentemente implementados em casos onde exsitem Tipos de Dados Abstratos(TADs)

cada dimensão, bem como o significado de cada agregação no cubo multidimensional.

2.6 – Conclusão

Mostramos todas as etapas do processo de *data warehousing* e os elementos necessários para sua construção. A primeira etapa, de extração, tem sua complexidade relacionada ao quanto os formatos e sistemas fontes dos dados serão diferentes do banco de *data warehouse*, o qual utiliza o modelo relacional. As dificuldades da etapa de transformação dependem da necessidade de adaptação do modelo original ao modelo da análise. Alem disso, verificamos que muitas das transformações já constituem, em si mesmas, fonte de análise. Por exemplo, muitos analistas obteriam várias informações somente com a agregação de dados de suas filiais ao longo do tempo. Apresentamos uma classificação para as funcionalidades de transformação, que apesar de não absoluta, serve como guia para utilização das funções existentes na maioria dos sistemas.

A análise multidimensional é uma ferramenta poderosa para visualização das múltiplas perspectivas dos dados. O seu maior atrativo reside no fato de que as operações realizadas se resumem a agregações e seleções em tabelas, um procedimento que vem sendo utilizado pelos analistas de negócios há muito tempo. O grande gargalo desta tecnologia está no volume de dados, o que requer sistemas de bancos de dados especializados para realizar as operações de agregação e seleção de forma rápida e flexível.

Também observamos que ainda existe muito esforço para consolidar uma extensão da álgebra relacional para análise multidimensional. Mais ainda, pouco se tem considerado a existência de TADs dentro do processo. Os tipos de dados espaciais, por exemplo, já fazem parte integrante do banco de dados há algum tempo, porém não há qualquer ferramenta que extenda as operações de agregação e análise multidimensional com dados espaciais. No próximo capítulo, vamos dar uma breve introdução sobre dados espaciais e como eles estão sendo utilizados dentro dos sistemas de bancos de dados. Mas, no corpo principal deste trabalho, iremos mostrar, mais genericamente, como TADs podem ser inseridos dentro do contexto de análise multidimensional

Capítulo 3 – Data Warehouse Espacial

3.1 - Sistemas de Informação Geográfica

Sistemas de Informação Geográfica (SIGs) são definidos das mais variadas formas, dependendo, por exemplo, da sua área de atuação, utilização ou funcionalidades. As primeiras utilizações da dimensão espacial começaram nas batalhas aonde planos estratégicos eram feitos usando mapas (Tabela 2). A primeira análise espacial que se tem registrada foi o mapeamento de casos de cólera em Londres. De fato, a análise espacial de doenças é bastante explorada (ex. Bailey 1982,Braga at al. 1997,Braga, Nobre e Pinheiro 1994,Nobre e Carvalho 1996

A maioria dos fundamentos, estruturas e algoritmos para manipulação de dados espaciais já havia sido desenvolvida desde antes de 1950 e o inicio da década de 1980, mas o maior problema até aquele momento era a entrada de dados. Os mapas eram digitalizados manualmente, fornecendo-se os valores de posição de cada ponto, o que tornava o processo pouco acurado. As fotos aéreas eram mais precisas, mas sua transformação em pontos de localização e digitalização era por demais complexa naquele momento. No inicio da década de 1970, houve uma grande evolução nas ferramenta de extração: o primeiro Censo "*Geocodificado*". Este processo consiste na conversão de endereços em texto puro para coordenadas espaciais (ver seção 3.1.7).

O armazenamento e manipulação de estruturas de dados espaciais evoluiu significativamente na década de 1980. Os bancos de dados relacionais começaram a ser usados para armazenar estruturas de indexação especiais para lidar com dados espaciais, e os atributos de cada elemento espacial se separaram em *tuplas* do modelo relacional para flexibilizar o relacionamento entre os dados e sua representação espacial.

O desenvolvimento do hardware e interfaces na década seguinte permitiu aos usuários finais um acesso completo a bancos de dados espaciais e ferramentas de análise. As ferramentas de extração de dados espaciais também evoluiram em dispositivos capazes de medir de forma muito mais acurada qualquer ponto sobre a terra, os *GPSs* (*Global Positioning Systems*) (Jones 2001,Kaplan 1996).

Tabela 2 - Evolução histórica dos SIGs (Original deste trabalho e compilada de diversas fontes**)**

Data	Teorias	Pessoas e Organizações	Produtos e Projetos
Até Início do século XX	- Uso de múltiplos mapas temáticos - Análise geográfica	Louis-Alexandre Berthier	Batalha de Yorktown
		John Snow	Cólera em Londres
Década de 1950	Requisito de integração de rotas, destinos, origens e tempo.		Sistema de trasnporte de Detroit e Chicago
	Desenvolvimento da fotografia aérea		
	Cartografia computacional		
	Conceitos espaciais, distância, orientação	Nystuen	
	Algoritmos computacionais para projeções de mapas	Tobler	
	Base para geografia – pontos, linhas, áreas	Bunge	
	Sobreposição de camadas em mapas.	Berry's	
Início da década de 1960	- Requisito de nova tecnologia - Vetorização de imagens. - Partição de dados em camadas temáticas. - Separação do atributo e do dado de localização.		CANADA GEOGRAPHIC INFORMATION SYSTEM (CGIS)
Final da década de 1960	- Mapeamento de propósito geral - Facilidades de uso para os não-cartógrafos - Primeiras demonstrações da habilidade dos computadores de fazer mapas - Uso de *plotters* e perspectivas em 3D	Harvard Lab for Computer Graphics and Spatial Analysis - Howard Fisher	SYMAP CALFORM SYMVU GRID
	SACS: Sistema de conversão de endereços (*Street Address Conversion System*)	ESRI- Environmental Science Research Institute	
Década de 1970	- Teorias de análise espacial - O primeiro censo geocodificado - Formato espacial (*Digital Line Graph* (DLG)	USGS ERDAS	
Década de 1980	- Cartografia digital - Banco de dados relacional - Redes triangular-irregular - *Triangular - Irregular Networks* (TINs)		ARC/INFO
	Sistema de Posicionamento Global – GPS (*Global Positioning System*)	National Center for Geog. Inf. and Analysis (NCGIA)	GRASS, MapINFO, SPANS, IDRISI
	Aperfeiçoamento dos sistemas de censo	US bureau of Census 'TIGER'	
Década de 1990	- Sistemas mais orientados para o usuário final - Análise espacial aprimorada - Sistema de Informação Geográfica Brasileiro (Spring)		ATLAS GIS ARCVIEW ARC/INFO para/PCs Spring
2000	Extensores para sistemas de o banco de dados		DB2 Spatial Extender Informix Spatial Datablade ORACLE Spatial

Os produtos líderes no mercado de SIG hoje em dia são o **MapInfo** (MapInfo 2001) e a linha da **ESRI** (**ArcInfo**, **ArcGIS**, **ArcView** etc.)(ESRI 2004). Na área de *software* livre, temos o **GRASS** (Neteler e Mitasova 2002), que é amplamente utilizado em pesquisa e em organizações mundiais governamentais. Ele é um projeto mundial que recebe acréscimos e contribuições possibilitando a sua constante evolução. No setor nacional, o **Spring** (Camara at al. 1996), desenvolvido pelo Instituto Nacional de Pesquisas Espaciais (INPE), é bastante utilizado pelas organizações e fundações brasileiras. Ele é um SIG desenvolvido com filosofia orientada por objetos, utilizando a linguagem OQL (*Object Query Language*) e com um armazenamento que faz uso de vários bancos de dados, tais como o **ORACLE**. O *software* livre é uma solução recente, que tem sido usada por vários setores que não tem orçamento para compra de programas caros e mais poderosos, já que, em geral, são governamentais e sem fins lucrativos.

3.1.1 - Funcionalidades

A correta definição das funcionalidades necessárias a um SIG nos auxilia a classificar e especificar o tipo de SIG que desejamos implantar (Maguire e Raper 1992). As funcionalidades necessárias a um SIG são divididas no seu aspecto espacial e não espacial. Esta divisão deve-se ao fato das arquiteturas mais convencionais de SIGs serem compostas de gerenciamento espacial e não espacial separados. Um resumo dos componentes funcionais (Maguire e Dangermond 1991) divididos em espaciais e não espaciais pode ser visto na Tabela 3.

Tabela 3 - Aspectos funcionais de um SIG (extraída de Braga 1998a)

Grupos funcionais	Espacial	Não Espacial
Captura	- Imagens - Scanners, foto satélite etc. - Digitalização de dados, GPS etc. - Conversão de formatos - Edição e validação	- Teclado - Reconhecimento de voz e texto.
Estruturação	- Estruturas espaciais:(tesselações, mapas poligonais, híbridas) - Topologia	- Bancos de dados Orientados para Objeto, Relacionais etc. - Arquivos e listas
Manipulação	- Restruturação - Mudança das estrutruturas de dados, agregação de objetos. - Generalização - alisamento de contornos, eliminação de ruído. - Transformação: rotação, mudança de escala, projeções etc. - Consulta: busca, sobreposição de objetos etc.	- Restruturação: mudança das estrutruturas de dados - Generalização: resumos estatísticos - Transformação e Linearização - Consulta: busca, recuperação de dados
Análise	Adjacência, sobreposição de mapas, buffers etc.	- Estatísticas - Multivariadas
Representação	- Mapas cromáticos - Mapas texturizados - Mapas tridimensionais	- Tabelas - Relatórios - Gráficos

As análises feitas em SIGs são variadas, dependendo do tipo de aplicação. Os tipos de análises consistem, em geral, de consultas sobre a topologia da base espacial, como na Tabela 4, ou então, sobre análises estatísticas espaciais (ex.Nobre e Carvalho 1996, Dunn 1987).

Tabela 4 - Alguns tipos de consultas feitas em SIGs (extraída de Braga 1998a)

Tipo de Consulta	Consulta
Localização	O que existe em … ?
Condição	Onde está … ?
Modificação	O que foi alterado em …?
Roteamento	Qual o melhor caminho para…?
Padrão	Qual o padrão …?
Modelagem	O que acontece se…?

Openshaw 1991 propõe uma lista básica de análises a serem feitas em SIGs:

1. **Espalhamento de padrões** – Este tipo de análise mostra, através de visualização ou de matrizes de dados, como um certo atributo está distribuído ao longo do espaço. O exemplo mais comum deste tipo de análise são mapas cromáticos representando a densidade demográfica.

2. **Investigadores de relacionamentos** – Este tipo de análise visa determinar qual o relacionamento entre objetos no espaço, se duas linhas se interceptam ou se um ponto está dentro de uma área.
3. **Detectores de contornos** – Este tipo de análise tem por objetivo seguir linhas que se interceptam até obter contornos fechados.
4. **Análise de padrões nebulosos** – Este tipo de análise usa lógica nebulosa para representar valores imprecisos, como "alto", "baixo", etc (Braga 1998b).
5. **Visualização avançada** – Este tipo de análise visa mostrar dados multidimensionais de forma visual, como gráficos 3D.

A gama de possíveis análises em SIGs é muito grande e este conjunto apenas mostra um grupo das análises principais e mais usadas.

3.1.2 - Modelos de Dados

O objetivo da modelagem de dados consiste em criar esquemas capazes de definir a forma e o comportamento dos dados do mundo real. A modelagem de dados em bancos de dados convencionais é bastante conhecida e possui componentes já bem estabelecidos. A modelagem de dados espaciais, por outro lado, possui complexidade além da convencional. Por isso, existem muitas propostas de modelagem apropriada para um SIG. A maioria dos autores (Burrough 1992, Maguire e Dangermond 1991, Frank 1992, Egenhofer e Herring 1991a) tenta definir dois níveis básicos de modelagem:

1. **Modelos da realidade espacial** - feições espaciais.
2. **Modelagem geométrica ou lógica** - forma de se abordar as feições espaciais na geometria do espaço.

MODELAGEM DA REALIDADE

A modelagem da realidade, às vezes chamada de modelagem conceitual, consiste em descrever objetos que irão representar as entidades do mundo real. Estes objetos são associados com os dados não espaciais e então chamados de feições geográficas (Maguire e Dangermond 1991). As feições geográficas mais comuns são baseadas em estruturas de geometria euclidiana:

- **Ponto** - Utilizado para representação concentrada de elementos geográficos utilizando uma única coordenada, como cidades, fazendas, ocorrências de eventos etc.

- **Linha** - Utilizado para representar elementos que se estendem ao longo do espaço através de uma trajetória contínua, como rios, estradas, rotas, fronteiras, etc.
- **Área** - Utilizado para representar extensões de terra ou atributos que se estendam continuamente sobre ela, como lagos, colheitas, estados etc.
- **Superfície** - Utilizado para representar superfícies no espaço de 3 dimensões, como, por exemplo, morros.

MODELAGEM GEOMÉTRICA

A modelagem geométrica (Frank 1992), às vezes chamada de modelagem lógica (Maguire e Dangermond 1991), outras vezes de modelo gráfico (Burrough 1992), consiste na escolha de como iremos definir o espaço para descrever nossas entidades espaciais. As formas de modelagem estão divididas basicamente em duas classes: vetorial e tesselação espacial, também chamadas de formatos, ou modelos, *Vector* e *Raster*, respectivamente.

A *tesselação* espacial (Maguire e Raper 1992) é baseada em uma modelagem contínua do espaço em células (Figura 7), cada qual assumindo um valor que pode ser relacionado a um atributo da base não espacial, como a indicação da densidade populacional naquele elemento do espaço, ou uma característica própria do espaço, como a vegetação. Este formato é mais eficiente e flexível para sobreposição de camadas de mapas e, em alguns casos, análise de contigüidade. Este formato, contudo, não é tão adequado quando desejamos identificar objetos individuais, como rios, cidades etc. A razão disto é que as células estão dispostas no espaço sem nenhuma relação com os objetos geográficos nele existentes e, assim, se torna difícil identificar as células que correspondem a um elemento geográfico.

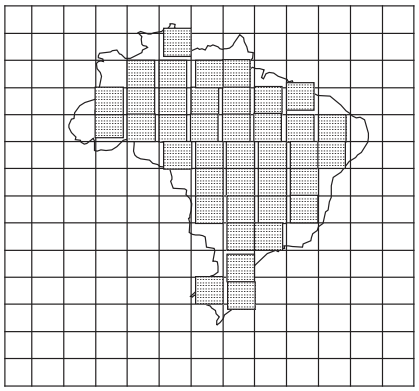

Figura 7 - Exemplo de tesselação espacial

No formato vetorial, as entidades geométricas da base espacial são descritas diretamente através das coordenadas de todos os pontos que as compõem. Um segmento de reta, por exemplo, passa a ser representado pelas duas coordenadas que delimitam seus extremos. O formato vetorial é mais usado para a modelagem propriamente dita de feições espaciais, pois permite que cada objeto ou feição seja descrito individualmente em sua forma e extensão.

3.1.3 – Uso do Paradigma Relacional

A questão da modelagem de dados está relacionada com os sistemas utilizados para implementação. Os bancos de dados baseados no modelo relacional (Codd,1970) já estão amplamente difundidos e estabelecidos, cuidando da segurança, integridade, recuperação e tratamento de concorrência. Em vista disso, tornam-se vantajosos em relação aos sistemas *ad hoc*.

Dada a estrutura complexa dos dados espaciais, o processo de armazenamento e modelagem utilizando o modelo relacional traz uma série de dificuldades. A solução destes problemas faz-se de muitas formas. Uma delas é criar sistemas híbridos, onde o gerenciador espacial beneficia-se das capacidades do banco de dados relacional, armazenando cada entidade num registro e definindo campos para especificação de coordenadas e extensão dos objetos, como o ARC/INFO (Morehouse 1992) e o GeoManager da IBM (Batty 1992).

Apesar da grande utilização do modelo relacional nos SIGs, as características dos dados espaciais, como a extensão ou a multidimensionalidade, trazem muitos problemas de armazenagem e manipulação, gerando inclusive uma distorção na modelagem dos dados (Worboys at al. 1994, Worboys 1994, Davis Jr e Borges 1994). Uma das soluções para essas questões é a utilização do modelo objeto-relacional para dar suporte a algumas capacidades do modelo orientado para objetos (Herring 1992).

Uma outra solução é a utilização do modelo orientado por objetos diretamente dentro do SIG, utilizando o banco apenas como armazenamento dos dados, como a iniciativa no Brasil do **Spring** (Camara at al. 1996). O lado ruim de abordagens como esta é que ela não tira vantagem do suporte espacial já existente no banco, como veremos mais adiante na seção 3.2.

3.1.4- Estruturas de dados

Dados espaciais possuem várias características particulares, como a multidimensionalidade, que trazem a necessidade de se utilizar estruturas de armazenamento e manipulação mais complexas que as estruturas convencionais, como listas, árvores binárias, árvores B, tabelas Hash etc. Não existe um tipo de estrutura espacial única para uma base espacial ou mesmo para um tipo de entidade espacial (pontos, linhas etc.). Cada estrutura espacial tem propriedades específicas para atender a um grupo de problemas específicos. A escolha de uma estrutura específica para os dados espaciais em um SIG é dependente da aplicação e influenciada pelo modelo de dados adotado.

As estruturas de dados utilizadas para implementar modelos espaciais está baseada nas duas modelagens geométricas principais (Maguire e Raper 1992, Goodchild 1992): vetoriais e tesselação espacial.

TESSELAÇÃO ESPACIAL.

Os métodos de tesselação dividem-se em regulares e irregulares. No formato regular, o espaço é dividido em células de mesmo tamanho. As estruturas espaciais mais comuns para trabalhar com tesselação utilizam a divisão hierárquica do espaço. As mais usadas baseiam-se nas *Quad-Trees* (Samet 1990) e suas variações.

No formato irregular, o espaço pode ser dividido em células de tamanhos diferentes. Temos como exemplos a TIN (Samet 1990) e os diagramas de voronoi (Franklin 1991).

FORMATO VETORIAL

As estruturas de dados utilizadas com este formato de dados são classificadas em não estruturadas, hierárquicas e topológicas.

Os métodos não estruturados são muito comuns em SIGs de menor porte. As estruturas mais comuns são chamadas de modelo “espaguete”. Este processo consiste em armazenar todos os pontos e linhas de uma base espacial em uma grande lista de coordenadas. Algumas versões incluem pontos especiais de ligação entre as linhas que se cruzam. As estruturas hierárquicas são baseadas em árvores como as *Quad-trees*.

Outras estruturas similares são utilizadas para linhas e polígonos. A mais conhecida delas utiliza-se de um princípio análogo ao das Árvores-B, denominadas de Árvores-R (Guttman 1984). Nesta estrutura, os nós da árvore possuem retângulos que

contém os objetos espaciais de sua sub-árvore (Samet 1990). As estruturas e algoritmos usados podem ter variações (Mediano, Casanova e Dreux 1995, Beckmann at al. 1990), mas seguindo o mesmo principio.

3.1.5 - Processamento de Consultas

Os tipos de consulta geralmente feitas em SIGs (seção 3.1.1) não são convencionais. Existem várias propostas de linguagens de consulta, ou dialetos (Egenhofer 1992, Portier 1996, Herring 1992). Cada dialeto pode ser caracterizado por alguns aspectos básicos:

1. **Tipo de dado espacial** - Alguns dialetos fornecem tipos de dados universais para o tratamento de entidades espaciais. Outros tipos fornecem classes pré-definidas, como pontos, linhas, áreas etc., além de fornecer tipos para as propriedades espaciais, como comprimento ou volume.
2. **Relacionamentos espaciais** - Cada implementação tenta oferecer capacidades para lidar com predicados espaciais de forma similar à convencional. Contudo, a complexidade das relações espaciais torna inadequadas linguagens similares ao SQL. Conceitos como proximidade ou contiguidade não são facilmente descritos. Na seção 3.1.6, descrevemos o modelo de relacionamento entre entidades espaciais mais detalhadamente.
3. **Visualização gráfica** - A representação de dados convencionais é feita através de tabelas. A representação de dados espaciais é bem mais complexa, pois envolve múltiplas formas de representação. Linguagens como o SQL não possuem sentenças específicas em sua linguagem para definir tais formas de representação.
4. **Seleção visual** - Outro aspecto é que em muitos casos o usuário deseja indicar objetos de seleção através do mapa sendo visualizado.

3.1.6 – Modelo de relacionamentos espaciais

Os relacionamentos entre elementos espaciais podem ter situações variadas dependendo dos tipos e dimensões dos elementos. Por exemplo, a interseção entre duas linhas poligonais pode ser um único ponto ou até mesmo uma outra linha. O modelo mais utilizado para estes relacionamentos espaciais é o DE-9IM (**Dimensionally Extended 9 Intersection Model**), desenvolvido por Clementini e Di Felice 1994 (Clementini e Di Felice 1994,Clementini e Di Felice 1996,Clementini, Di Felice e Egenhofer 1994,Denis e François 2001) como uma extensão do modelo de Egenhofer e Herring 1991b.

O modelo **DE-9IM** é uma abordagem matemática que expressa todos os relacionamentos entre o interior (I), o contorno (C) e o exterior (E) de geometrias de diferentes tipos e dimensões. Dada uma geometria **a**, I(a), C(a) e E(a) representam, respectivamente, o interior, o contorno e o exterior de a. Cada interseção pode resultar em geometrias de diferentes dimensões, como, por exemplo, a interseção dos contornos de dois polígonos pode ser um ponto ou uma linha poligonal. Uma geometria resultante com dimensão -1 vai indicar que não houve interseção, e as dimensões 0, 1 e 2 vão representar ponto, reta e linhas poligonais ou polígonos, respectivamente. A matriz do padrão **DE-9IM** consiste de valores aceitáveis para cada interseção:

- T – A interseção deve existir com dimensão = 0, 1, ou 2
- F – Nenhuma interseção deve existir: dimensão = -1
- * – A existência da interseção não importa: dimensão = -1, 0, 1 ou 2
- ***X*** – A interseção deve existir e a dimensão máxima deve ser ***X***

Tomemos com exemplo a função do padrão OGC (Open GIS Consortium e Ryden 1999) *ST_WITHIN*, cuja semântica determina se um elemento espacial está contido em outro. A matriz de interseção para a esta função (Tabela 5) tem valores "T" sempre que os interiores de ambas as geometrias se interceptarem, bem como quando o interior e o contorno da geometria **a** não interceptarem o exterior da geometria **b**. Todas as outras condições não importam (IBM Corporation 2001).

Tabela 5 - Matriz de padrão de relacionamento para a função *ST_Within*

		b		
		(I)nterior	**(C)ontorno**	**(E)xterior**
a	**(I)nterior**	T	*	F
	(C)ontorno	*	*	F
	(E)xterior	*	*	*

No modelo **DE-9IM**, cada função de relacionamento espacial vai possuir uma tabela similar à Tabela 5 indicando todos os possíveis resultados de cruzamento entre todos os tipos de elementos espaciais.

3.1.7– Extração de Dados Espaciais e Geocodificação

A maior dificuldade nos SIGs sempre foi o processo de entrada de dados. Posições no espaço são, na maioria das vezes, originadas de desenhos, mapas ou fotografias, as quais precisam ser convertidas em coordenadas espaciais numéricas, como latitude e longitude. Hoje em dia, temos dispositivos de obtenção de posição absoluta utilizando satélites -- os *GPSs* (Jones 2001,Kaplan 1996) --, mas eles requerem que o dispositivo seja levado diretamente para as posições que precisam ser mapeadas, o que nem sempre é possível.

A geocodificação é uma alternativa de obtenção de informação posicional na qual a informação codificada em algum tipo de formato, em geral texto, é convertida em coordenadas espaciais. O exemplo mais comum é a geocodificação de endereços, onde um endereço completo é convertido nas coordenadas espaciais do objeto.

A técnica mais simples e rudimentar de geocodificação é a manual, onde o endereço é verificado visualmente num mapa físico e as suas coordenadas são anotadas.

A utilização comercial mais freqüente do processo de geocodificação consiste na conversão de endereços armazenados no banco de dados corporativo em coordenadas espaciais ou geográficas. Esta utilização constitui-se se vários processos complexos (Clarke 2000,Meador 2001):

- **Limpeza** – Antes que se possa proceder a análise dos dados a serem geocodificados, devemos filtrar erros de digitação (como junções de letras sem nenhum significado), assegurar homogeneidade de nomenclatura (ex., r. e rua), corrigir dados incompletos etc. Isso é similar à limpeza feita no processo de *data warehousing* (seção 2.2.1).
- **Consistência de dados e validação** – Depois de pré-processados, a consistência dos dados é feita sob a forma de validação, onde os valores fornecidos são checados contra uma base com referenciais espaciais. Por exemplo, um número de rua pode estar fora dos limites mínimo e máximo de números para aquela rua, ou esta pode até mesmo não existir. Esta etapa pode dar margem a análises de similaridade de nomes. O nome exato de uma rua pode não existir porque ele não foi digitado corretamente. O índice de similaridade, ou tolerância, é geralmente um parâmetro fornecido ao processo. Existem muitas outras validações possíveis que dependerão da complexidade dos tipos de endereços do local em questão. O nome de uma rua, por exemplo, pode estar duplicado em dois lugares diferentes. Há também casos, como

em países asiáticos, onde o número das casas não segue uma ordem crescente, requerendo tabelas de validação.

- **Conversão de endereços** – Este é o último passo da geocodificação propriamente dita. O processo consiste em checar o elemento já filtrado, como o nome da rua, em uma grande base que irá converter o elemento nas suas coordenadas geográficas. No caso de conversão de endereços, a maioria dos métodos utiliza o número do prédio para interpolar o ponto dentro da linha que corresponde ao número, mas este método pode variar bastante dependendo das características específicas dos endereços do país em questão.

A geocodificação reversa (Meador 2001) é uma outra funcionalidade que pode ser encontrada em alguns geocodificadores. Ela converte posições espaciais em endereços. O processo utiliza mínimos de tolerância e distância para determinar os elementos espaciais mais próximos e retornar o elemento de dados (ex., endereço) mais próximo.

Os elementos de dados fontes do processo de geocodificação não precisam ser necessariamente endereços, mas qualquer elemento de dado que possa conter informações implícitas sobre a localização do elemento, como (Open GIS Consortium 2001):

- **Nome** – A conversão de um nome, por exemplo, de uma rua, pode produzir uma linha poligonal completa correspondente à rua.
- **Endereço de rede (IP)** – Uma forma bastante interessante de geocodificação é determinar a planta de localização física das máquinas servidoras de rede. Isto pode auxiliar no planejamento estratégico de redes ou até na busca criminal de indivíduos que estejam fazendo uso ilegal da rede.
- **Numero de telefone** – Esta forma tem utilidades similares às do endereço de rede, mas ainda tem outras perspectivas, pois números de telefones estão intimamente ligados com a residência de indivíduos.
- **Localizações de negócios** – Outra forma muito usada de geocodificação é a de nomes relacionados a uma aplicação de negócios específica. Por exemplo, um banco de dados pode conter nomes de pontos históricos ou de filiais de uma rede. O

processo de geocodificação iria automaticamente gerar as coordenadas espaciais destes pontos.

- **Imagens de satélite** (de Macedo e Fernandes 2003) – Este é um processo de geocodificação bem mais complexo, pois constitui-se na análise de uma fotografia cartográfica através de uma grande grade de células, seguida de comparações e conversões usando um banco de endereços.

Existem várias ferramentas comerciais especializadas em geocodificação. O **Trillium** (Trillium e Harte-Hanks Data Technologies 1999), por exemplo, é uma ferramenta criada originalmente para limpeza de dados, mas com módulos extras para a geocodificação. A **ESRI** (ESRI 2004) e a **MapInfo** (MapInfo 2001) são líderes neste mercado, porém seus geocodificadores costumam ser embutidos em seus produtos. Os extensores de manipulação espacial no banco de dados (ver seção 3.2) também possuem geocodificadores incorporados (IBM Corporation 1998,IBM Corporation 2001 e Oracle Corporation 2001). Estes produtos tentam seguir uma arquitetura aberta de tipo *plug-ins*, atualmente apoiada por padrões da OGC (Open GIS Consortium 2001), para que se possa inserir qualquer outro geocodificador externo. A razão disto advem de um dos maiores problemas existentes com o processo de geocodificação: a interpretação de endereços é extremamente particularizada ao país de origem. Desta forma, mesmo os mais poderosos geocodificadores que operam em um país podem não operar em outro. Nos Estados Unidos, por exemplo, endereços em geral começam com um número e depois o nome de rua(ex., *5591 Blossom Gardens Circle*), o que torna a geocodificação bastante simples. No Brasil, por outro lado (DIGIBASE 2004,MAPTEL 2004), os números ficam ao final, embora possam estar inclusos no nome (ex., rua 7 de setembro 122), o que torna a geocodificação um pouco mais complexa. No Japão, a situação é ainda mais grave, porque os números das casas não seguem seqüências crescentes, o que requer tabelas de mapeamento para cada rua para determinar a posição de um prédio específico.

De toda forma, o processo de geocodificação é o mais adequado para relacionar os dados de negócio com sua posição no espaço, e cada país terá que desenvolver seu processo de geocodificação específico.

3.2 –Sistemas de Informação Espacial e Bancos de Dados Espaciais

Existe uma diferença na literatura entre SIGs e Sistemas de Informação Espacial. Os SIGs estão sempre relacionados à geografia e cartografia, e seus produtos

manuseiam os dados espaciais de forma especializada, sem utilização mais intensa de um sistema gerenciador de bancos de dados convencional. Os Sistemas de Informação Espacial, além de nem sempre estarem ligados a fatos geográficos, como planejamento arquitetônico, exploram mais os dados armazenados num banco de dados convencional. Segundo Güting 1994, um Sistema de Informação Espacial pode ser definido levando-se em conta as seguintes características:

1 - Um Sistema de Banco de Dados Espacial é um Sistema Banco de Dados

2 - Ele deve oferecer tipos de dados espaciais no seu modelo de dados e linguagem de consulta.

3 - Suportar indexação nos dados espaciais e nos algoritmos de junção espacial

A necessidade de unir estas características ao universo dos bancos de dados comerciais (ESRI 2003a) e à área de SIGs levou a organização mundial para arquiteturas de SIGs, o **OpenGIS Consortium** (Open GIS Consortium 1997), a estabelecer uma série de padrões para arquitetura de SIGs. Um dos documentos mais importantes emitido por esta organização consiste na definição de uma API (*Application Programming Interface*) em SQL para permitir acesso, armazenamento e operações com elementos espaciais, como ponto, linha polígono etc. (Open GIS Consortium e Ryden 1999). Este padrão é aceito internacionalmente. Contudo, dos três maiores produtos de bancos de dados no mercado, que incorporam a manipulação espacial, somente os dois primeiros adotam completamente o padrão da OGC:

1 - **IBM**: **DB2 Spatial Extender** – O produto de extensão espacial para o banco de dados DB2 (Adler 2001,Davis 1998,IBM Corporation 1998,IBM Corporation 2002a,Kache 2001).

2 - **IBM**: **Informix Spatial DataBlade** – O produto de extensão espacial do banco de dados **INFORMIX**. A empresa **INFORMIX** foi adquirida pela **IBM** em 2002 e a intenção atual é anexar todas as funcionalidades do **Spatial DataBlade** no **DB2 Spatial Extender**, o qual se manteria como o produto principal. Atualmente, a equipe que desenvolve o **DB2 Spatial Extender** é composta em sua maioria por elementos da antiga equipe da **INFORMIX** (IBM Corporation 2001).

3 - **ORACLE**: **Oracle Spatial** – O produto de extensão espacial para o banco de dados **ORACLE** (Oracle Corporation 2001).

3.2.1– Oracle Spatial

O **Oracle Spatial**, solução da **ORACLE** para manipulação espacial, antes chamado de **Oracle Spatial Cartridge**, é bastante poderoso. Ele oferece um conjunto completo de

tipos de dados espaciais padrão, com pontos, linhas poligonais e polígonos, bem como coleções destes. Este produto, diferente dos demais, oferece objetos especializados para manipulação de arcos, círculos, retângulos e objetos de camada (*layer*), destinados a manipular as coleções de objetos.

O mecanismo de consulta oferece várias funções de relacionamento, inclusive de vizinhança próxima (*nearest neighborhood*), que são utilizadas em uma arquitetura de filtragem dupla. O primeiro filtro realiza operações de aproximação geométrica, enquanto o segundo aplica regras mais precisas para obter o resultado final. Por exemplo, no caso do cálculo de interseções entre polígonos, o primeiro filtro vai calcular as interseções entre os "retângulos de contorno" (*Max Bounding Rectangles*) dos polígonos, e o segundo vai calcular ponto a ponto para determinar as interseções.

O **Oracle Spatial** fornece duas formas de indexação espacial, *R-trees* e *Quad-trees* (Samet 1990), que podem ser escolhidas pelo usuário. Cada tipo de índice é adequado a diferentes características da massa de dados, como agrupamento dos dados, tipos de geometria etc. Uma tabela para decisão do melhor tipo de índice pode ser encontrada no manual do produto e em várias partes da literatura (Beckmann at al. 1990,Burrough 1986,Guttman 1984,Han, Xu e Lu 1990,Oracle Corporation 2001)

O conjunto de funções do produto não é muito grande, pois elas utilizam um formato único com parâmetros para indicar variações na funcionalidade. A função de teste de interseção, por exemplo, tem um parâmetro para indicar cada categoria. O pacote completo do produto ainda oferece uma biblioteca (*API*) para construção de aplicações em JAVA(Oracle Corporation 2000).

O último grande diferencial do **Oracle Spatial** é a disponibilidade de funções de agregação espacial. O produto possui funções similares às funções agregadoras do SQL, como AVG ou COUNT, para manipulação de geometrias. A função *SDO_AGGR_UNION*, por exemplo, é capaz de retornar a agregação de todas as geometrias de um conjunto ou coluna de uma tabela, resultando numa só geometria, do tipo coleção. Isto só é possível porque o ORACLE, versão 9i, permite a construção de funções de agregação definidas pelo usuário.

Pelo que mostramos neste breve resumo da ferramenta, podemos ver que o **Oracle Spatial** é um excelente instrumento de manipulação de dados espaciais. Sua única fraqueza, além do seu discutível desempenho perante os competidores, é que suas

funções não seguem o padrão da OGC, o que dificulta sua integração com ferramentas de SIGs.

3.2.2 - IBM DB2 Spatial Extender e IBM Informix Spatial DataBlade

O **Informix Spatial DataBlade** era o produto de manipulação de dados espaciais da **Informix** até 2002, quando concluiu-se a aquisição desta por parte da **IBM**. De todo modo, os dois produtos sempre foram bem similares, sendo o único grande diferencial o método de indexação por *R-Trees* do **Informix Spatial DataBlade**. O **DB2 Spatial Extender** utiliza um sistema de índices espaciais mais pobre, chamados por eles de indexação de grade (*grid indexing*). O espaço é simplesmente dividido em uma grade regular e as operações de seleção de objetos espaciais utilizam o MBR (*Max Bounding Rectangle*) de cada objeto espacial para determinar em que célula da grade ele está. A vantagem desta estrutura de indexação é o desempenho de melhor caso, dado que a estrutura é muito simples e existe uma ferramenta auxiliar do produto, o **index Advisor (versão 8)**, que determina, para cada conjunto de dados, a granularidade necessária da grade.

A arquitetura do **DB2 Spatial Extender** é constituída de quatro partes (Adler 2001,IBM Corporation 1998,IBM Corporation 2002a):

1 - Conjuntos de tipos de dados espaciais

2 - Funções de manipulação espacial

3 - Procedimentos armazenados (*Stored Procedures*) para importação de dados e manipulação de metadados.

4 - Interfaces com o usuário incorporadas ao Centro de Controle do DB2.

A arquitetura de ambos os produtos consiste de um conjunto de comandos de definição de tipos de dados e funções (DDL – *Data Definition Language*), uma parte de código DB2 SQL/PL (*SQL Programing Language*) e uma parte de código C incorporada às funções e procedimentos armazenados (*Stored Procedures*). A ativação do produto no banco nada mais é do que a execução destes comandos, que irão primeiro criar todas as classes de objetos espaciais, funções e procedimentos, e associar as funções e procedimentos nativos a códigos binários desenvolvidos em C.

O conjunto de funções de manipulação de dados espaciais dos dois produtos segue o padrão proposto pela OGC (Open GIS Consortium e Ryden 1999). Uma das boas razões para isto é que ambos sempre possuíram uma parceria com a ESRI (Adler e Higgs 2003,Davis 1998) no setor de mercado de SIGs, o que facilita a integração com seus produtos.

A categorização de funções apresentadas na Tabela 6 e Tabela 7 deste trabalho foi utilizada como base na classificação de funções adotada no manual da versão 8 do **Informix Spatial DataBlade (**IBM Corporation 2001**)** e do **DB2 Spatial Extender (**IBM Corporation 2002a**)**. Na Tabela 6 podemos ver a lista de funções que manipulam elementos espaciais individuais, como obtenção e propriedades, construtores e conversores de formato primitivo. A Tabela 7 mostra funções de relacionamento, conversão e transformação. As funções de transformação alteram o conteúdo das geometrias, mas sem alterar seu tipo, como acrescentar um ponto a mais numa linha poligonal. As funções de conversão podem produzir tipos de dados diferentes do parâmetro de entrada. Estas formas de classificação seguem princípios usuais, como em Han, Stefanovic e Koperski 1998a e Kache 2001, mas nós apresentamos aqui uma diferenciação na semântica de transformação em relação ao apresentado em Kache 2001 e IBM Corporation 2002a, além de um maior detalhamento das funções espaciais em Han, Stefanovic e Koperski 1998a.

A deficiência dos dois produtos, até a versão 8.1, pelo menos, é não possuir funções agregadoras (como AVG, COUNT etc.). Isto é uma conseqüência do próprio banco de dados (**DB2**) não possuir esta funcionalidade. Na equipe de desenvolvimento, da qual fiz parte, implementamos uma solução alternativa para o problema. O **DB2** permite que valores sejam armazenados entre chamadas das funções definidas pelo usuário. O grande problema é saber quando todos os valores do conjunto já passaram para a função e retornar o valor agregado final. A solução que desenvolvemos foi passar o valor de resultado através da seção de memória compartilhada de baixo nível, e utilizar o operador MAX() para garantir que a função seria chamada para todos os elementos do conjunto. A função agregadora retorna valores incrementais até a última chamada, desta forma, o índice de retorno é o da última chamada da função. O funcionamento deste mecanismo requer duas funções: uma específica de agregação e outra para obter o resultado. No exemplo, na Consulta 3-1, a função *ST_BuildMBRAggr* irá agregar os retângulos limitadores (*MBRs*) a cada chamada da função, retornando um número seqüencial crescente. O operador MAX() aplicado a ela irá garantir que a última chamada da função seja retornada, após a qual a função *ST_GetAggrResult* irá capturar o valor do MBR agregado na memória compartilhada.

```
SELECT  ST_GetAggrResult(MAX(ST_BuildMBRAggr(g))
FROM sample_points;
```

Consulta 3-1

Tabela 6 - Classificação das funções de manipulação individual de objetos espaciais do DB2 Spatial Extender (modificada de IBM Corporation 2002a)

	Funções de Acesso		Propriedades	Construtores e Conversores de Formato
	Geral	Booleanos		
Geometry	ST_GeometryType ST_SRID **ST_SrsId, ST_SrsName(V.8.1)** ST_Transform **ST_MaxM, ST_MinM, (V.8.1)** **ST_MaxX, ST_MinX, ST_MaxY, ST_MinY, (V.8.1)** **ST_MaxZ, ST_MinZ(V.8.1)** **ST_MBR(V.8.1)**	ST_IsValid Is3d **ST_Is3D(V.8.1)** IsMeasured ST_IsSimple ST_IsEmpty ST_Dimension	ST_Boundary ST_Envelope	**ST_Geometry** [ST_]**GeomFrom**[*Text/WKB/Shape*] [ST_]**As**[*Text/Binary/BinaryShape*] [ST_][*WKT/WKB/Shape*]**ToSQL** **ST_AsGML(V.8.1)**
Points	ST_CoordDim ST_X, ST_Y, Z(V.7), M(V.7) **ST_M, ST_Z(V.8.1)**			**ST_Point** [ST_]**PointFrom**[*Text/WKB/Shape*] [ST_]**As**[*Text/Binary/BinaryShape*]
Linestrings	ST_PointN, ST_NumPoints **ST_PerpPoints(V.8.1)**	ST_IsRing ST_IsClosed	ST_Length	**ST_Linestring** [ST_]**LineFrom**[*Text/WKB/Shape*] [ST_]**As**[*Text/Binary/BinaryShape*]
Polygons	ST_ExteriorRing ST_NumInteriorRing ST_InteriorRingN		ST_Area ST_Perimeter ST_Centroid ST_PointOnSurface	**ST_Polygon** [ST_]**PolyFrom**[*Text/WKB/Shape*] [ST_]**As**[*Text/Binary/BinaryShape*]
Multipoints	ST_NumGeometries ST_GeometryN			**ST_MultiPoint** [ST_]**MPointFrom**[*Text/WKB/Shape*] [ST_]**As**[*Text/Binary/BinaryShape*]
Multilinestrings	**ST_LineStringN, ST_NumLineStrings(V.8.1)** **ST_PerpPoints, (V.8.1)** ST_StartPoint, **ST_MidPoint(V.8.1)**, ST_EndPoint	ST_IsClosed	ST_Length	**ST_MultiLinestring** [ST_]**MLineFrom**[*Text/WKB/Shape*] [ST_]**As**[*Text/Binary/BinaryShape*]
Multipolygons	**ST_PolygonN, ST_NumPolygons(V.8.1)**		ST_Area ST_Centroid	**ST_MultiPolygon** [ST_]**MPolyFrom**[*Text/WKB/Shape*] [ST_]**As**[*Text/Binary/BinaryShape*]

Tabela 7 - Funções gerais de manipulação de dados espaciais do DB2 Spatial Extender (modificada de IBM Corporation 2002b)

Relacionamento		Conversão	Transformação	Combinação
Cruzamento	Contenção			
ST_Equals ST_OrderingEquals ST_Disjoint ST_Intersects ST_EnvelopesIntersect **ST_EnvIntersects(V.8.1)** **ST_MBRIntersects(V.8.1)** ST_Touches ST_Overlaps ST_Crosses	ST_Within ST_Contains ST_Relate	ST_Buffer ST_LocateAlong LocateBetween ST_Distance **ST_LocateBetween(V.8.1)** **ST_MeasureBetween(V.8.1)** **ST_FindMeasure(V.8.1)** **ST_ToGeomColl(V.8.1)** **ST_ToLineString(V.8.1)** **ST_ToMultiLine(V.8.1)** **ST_ToMultiPoint(V.8.1)** **ST_ToMultiPolygon(V.8.1)** **ST_ToPoint(V.8.1)** **ST_ToPolygon(V.8.1)**	ST_ConvexHull **ST_Generalize(V.8.1)** **ST_AppendPoint(V.8.1)** **ST_ChangePoint(V.8.1)**	ST_Intersection ST_Difference **ST_SymDifference(V.8.1)** **ST_SymmetricDiff(V.8.1)** ST_Union

Esta solução é eficiente em vários casos, com exceção de quando estamos utilizando a versão de múltiplas partições (EEE) do DB2. Neste caso, não temos como usar a área de memória compartilhada para passar os dados porque não há garantias de que uma consulta não esteja utilizando mais de uma partição de memória.

Nesta seção mostramos que estas duas ferramentas de manipulação espacial da **IBM** tem grandes vantagens e também fraquezas. Os dois produtos estarão se tornando um só a partir da versão 9, o que deve melhorar muito sua qualidade e desempenho. A única grande desvantagem, a falta de funções de agregação, na verdade não é tão crítica, dado que esta é uma funcionalidade facilmente implementável utilizando consultas um pouco mais complexas ou código SQL/PL[2].

3.3 – Sistemas de *Data Warehousing* Espacial

3.3.1 - Introdução

O objetivo principal de sistemas de *data warehousing* e informação geográfica tem sido o mesmo: permitir ao usuário acesso e análise de informação. Informação espacial no caso dos SIGs. A grande lacuna dos sistemas de *data warehousing* é a incapacidade de lidar com dados espaciais. Por outro lado, SIGs não exploram todas as vantagens do gerenciamento de dados dos bancos de *data warehouse*. Estes dois sistemas, enquanto sistemas de informação (Orman 1991), têm o comportamento similar de extrair, armazenar, processar e analisar dados. A Tabela 8 mostra as principais questões e requerimentos de um sistema em relação ao outro. As células sombreadas indicam as funcionalidades que o sistema já possui e as células brancas indicam as funcionalidades necessárias do sistema oposto.

O primeiro passo de cada sistema é a extração de dados. Sistemas de *data warehousing* são especializados em importar vários tipos de dados e fontes, contudo não possuem métodos para extração genérica de TADs como dados espaciais. Os SIGs estão equipados com várias funcionalidades para extração de dados espaciais, como a geocodificação (ver seção 3.1.7), contudo, não são tão especializados em extração de dados convencionais. O **Data Warehouse Center** do DB2 (IBM Corporation 2002d), por exemplo, é capaz de importar vários formatos de dados e registrar cada importação no catálogo de metadados.

[2] Linguagem de programação SQL do DB2, similar ao PL/SQL do ORACLE

A integração de fontes de dados espaciais em SIGs possui funcionalidades bastante complexas, com mudança de escalas, alteração de estruturas, sistemas de referência etc. Para permitir estas funcionalidades em sistemas de *data warehousing* é necessário incorporar as funcionalidades de manipulação de dados espaciais dentro do banco de dados.

Tabela 8 - Comparativo de funcionalidades de SIGs e sistemas de *data warehousing*(original deste trabalho**)**

Fases de Warehousing	*Data Warehousing* **Espacial**	*Data Warehousing* **em SIGs**
Extração	Extração de dados espaciais implícitos (geocodificação)	Geocodificação
	Importação espacial	Formatos espaciais complexos, digitalização
	Acesso a múltiplas fontes de dados	Integração com Bancos de *data warehouse*
Transformação e Integração	Manipulação de tipos de dados espaciais	Sistemas de referência, Escalas, Estruturas de rede
Armazenamento	Escalabilidade, transações e desempenho	Integração com bancos de *data warehouse*
	Extensões para estruturas de dados espaciais	Estruturas de dados e índices para dados espaciais
	Suporte para metadados espaciais	Metadados espaciais
Processamento	Armazenamento de dados no tempo	Manipulação de tempo
Interface com o usuário	Flexível e conhecida	Interfaces amigáveis
	Modelos intuitivos e orientados ao assunto de negócios	Um modelo padrão para relacionamentos espaciais equivalentes a “modelo estrela”
Análise	Análise multidimensional	Metodologia para análise multidimensional espacial
	Manipulação espacial multidimensional	Análise multicamadas e de rede

Os sistemas de *data warehousing* ainda levam grande vantagem na escalabilidade de volume de dados e na manipulação temporal dos dados. Neste último, os SIGs vêm avançando bastante (Al-Taha, Snodgrass e Soo 1993), mas os sistemas de *data warehousing* já possuem uma maior experiência e padronização no tratamento da dimensão temporal(ver seção 2.2.3).

As ferramentas e métodos de análise são o maior foco de diferenciação. Os SIGs são muito mais flexíveis e diversos, mas requerem um “off-set” de aprendizado e

utilização. Os sistemas de *data warehousing*, por outro lado, já utilizam modelos e métodos de análise multidimensional bem estabelecidos (seção 2.2.3).

A comparação de requisitos entre os dois sistemas mostra que a extensão espacial das funcionalidades de *data warehousing* adiciona os seguintes valores:

- A informação espacial pode trazer grande vantagem competitiva aos analistas de negócios.
- A metodologia utilizada pelos analistas pouco teria de modificação.
- Todo o poder de controle de transação, escalabilidade, gerenciamento de tempo e metadados poderia ser utilizado de forma integrada com ferramentas de SIGs.
- Os dados já existentes dentro dos bancos de *data warehouse* poderiam fornecer informações que não são percebidas pelos métodos convencionais.

3.3.2 – Conceitos e Abordagens

A integração dos dados e métodos de manipulação espacial com os sistemas de bancos de dados tem sido direcionada por soluções que são dadas a problemas específicos. Pouco se tem feito para estabelecer um modelo geral que defina um sistema e os processos de *data warehousing* espacial. A nossa pesquisa sobre o assunto nos levou a classificar os trabalhos e projetos que se referem ao "*data warehousing* espacial" em três linhas:

Data Warehousing Espacial

OLAP Espacial

MOLAP Espacial

A primeira, mais comum, envolve processos de geocodificação e importação da dados espaciais. A segunda abordagem, foca o acoplamento de métodos de análise espacial no fim do processo de ETL. Por último, o processo de análise multidimensional espacial ganha espaço na pesquisa no que se refere ao estudo de agregações de dados cuja a hierarquia é de fato espacial ou geográfica. Em muitos casos, não existe qualquer ferramenta específica desenvolvida, somente a implementação de uma solução para um dado caso de uso (Barclay, Gray e Slutz 2000,Chorel 2001,Grey 2001).

3.3.3 - *Data Warehousing* Espacial (*Spatial Data Warehousing*)

A abordagem dos artigos e trabalhos encontrados sob este enfoque (Barclay, Gray e Slutz 2000,Bedard 1999,Berkel 2000,Chorel 2001,CUBEWERX 1999,CUBEWERX 2004a,Grey 2001,Helmer 2001,Kimball 2000,Kimball 2001b,Kimball 2001a,Lowe 2000) consiste em estender o processo de ETL (Berson e Smith 1997,Lutz 2004). A extração consiste basicamente na geocodificação e limpeza dos dados. A carga e análise é substituída por SIGs conectados ao fim do processo de ETL (Adler e Higgs 2003,ESRI 1998,ESRI 2001).

3.3.4 - OLAP Espacial (*Spatial OLAP*)

O foco dos projetos e trabalhos nesta área é o processo de análise espacial dos dados. As técnicas envolvidas podem ser relacionadas à mineração de dados (Chawla at al. 2001,Han 1997a,Han, Kamber e Tung 2001,Han e Koperski 1995,Han, Koperski e Adhikary 1996,Han, Truffet e Zhou 1999,Koperski 1999,Koperski, Han e Adhikary 1998,Morimoto 2001,Morimoto at al. 2003,Morimoto, Kubo e Kanda 2001) ou análise multidimensional espacial (CUBEWERX 2004a,CUBEWERX 2004b,Kemp e Lee 2000,Papadias at al. 2001,Stefanovic 1997a,Zhang.L. at al. 2004). O conceito original de "OLAP" nem sempre representa operações interativas na base de dados espacial, mas, ao invés disso, apenas utiliza diversos algoritmos complexos de análise e mineração espacial diretamente em estruturas de dados espaciais, não utilizando qualquer suporte de OLAP proprietária do banco de dados (Morimoto, Kubo e Kanda 2001,Han 1997a,Han, Truffet e Zhou 1999,Koperski 1999,Koperski, Han e Adhikary 1998,Morimoto 2001)

3.3.5 - MOLAP Espacial (*Spatial MOLAP*)

Este foco é uma versão do OLAP Espacial que estende os tradicionais métodos de análises em cubos multidimensionais para modelos estrela (seção 2.2.3). Existem alguns conceitos definindo a semântica de uma dimensão e uma medida espacial em um cubo multidimensional (Kemp e Lee 2000,Shekhar, Lu e Zhang 2001)

A abordagem mais simples e comum é mapear os relacionamentos espaciais em relacionamentos convencionais e, então, pré-computar as agregações dentro do cubo. Por exemplo, se desejarmos pré-computar as contas bancárias de todos os clientes que moram numa certa cidade, basta gerar identificadores que relacionem cada cliente e a cidade onde mora, o que na maioria dos casos já faz parte dos dados por si só (Han, Stefanovic e Koperski 1998a,Han, Stefanovic e Koperski 1998b,Shekhar, Lu e Zhang 2001,Stefanovic 1997a)

Esta abordagem pode ser chamada de "*relacionamento espacial estático*" e não é eficiente se os parâmetros do relacionamento podem ser "*dinâmicos*". Vamos supor, por exemplo, que desejamos agregar as vendas ocorridas dentro de 2 Km de distância de cada uma das minhas lojas, e então aumentar esta distância para 4 Km. Não temos como pré-computar as agregações, pois não há como saber previamente o valor de distância que será requerido (CUBEWERX 1999,CUBEWERX 2004b,CUBEWERX 2004a,Ferreira e Tanaka 2001).

Para superar este problema, pode-se tirar vantagem das estruturas de indexação espacial já existentes (*R-Trees*, *Quad-Trees* etc). Estas estruturas naturalmente organizam os objetos espaciais de alguma forma hierárquica no espaço, assim pode-se pré-computar as agregações dos dados de acordo com esta hierarquia (Han, Xu e Lu 1990,Papadias at al. 2001,Papadias at al. 2002,Papadias, Kalnis e Mamoulis 1999,Rao at al. 2003,Rauber, Tomsich e Riedel 2000,Wang at al. 2003,Zhou e Prasher 2004,Rauber, Tomsich e Riedel 2000). O problema desta abordagem é que nem sempre a hierarquia de armazenamento espacial estará de acordo com um relacionamento específico desejado. De toda forma, a maioria das propostas de OLAP espacial menciona a utilização de estruturas de indexação espacial como parte de seu modelo, apesar de elas apenas servirem para suportar as operações espaciais padrão, independente das operações no cubo multidimensional (Rao at al. 2003,Rauber, Tomsich e Riedel 2000,Wang at al. 2003,Zhang.L. at al. 2004).

3.3.6 - Taxonomia de sistemas de *Data Warehousing* Espacial

A implementação de um sistema de *data warehousing* espacial deve considerar três componentes: o suporte espacial do banco de dados, a interação com SIGs e os tipos de ferramentas de análise empregadas. Para melhor visualizar todas as possíveis abordagens, este trabalho propõe uma taxonomia de classificação de cada uma destes componentes:

Componente 1 – Suporte Espacial do Banco de Dados – Este componente indicará as características do suporte espacial que o banco de dados possui.

A. **Nenhum** – Os dados espaciais são extraídos ou geocodificados e depois armazendos em algum lugar fora do banco de dados.
B. **Tipos de Dados Espaciais** – Oferece o suporte para tipos de dados espaciais, mas nenhuma função de manipulação.

C. **Funções de Manipulação Espacial** – Pode conter um conjunto limitado ou completo de funções para transformações espaciais, predicados espaciais, agregações etc.

Componente 2 – Interação com Sistemas de Informação Geográfica – Este componente mostra como o sistema de *data warehousing* espacial se integra com SIGs convencionais.

A. **Nenhuma** – Não existe nenhuma ferramenta de visualização espacial, ou apenas alguma aplicação espacial dedicada.
B. **Integrada** – O SIG irá armazenar todos os dados espaciais e não espaciais, e irá realizar todas as operações de manipulação.
C. **Conectada** – Podem haver vários níveis de conexão. O SIG, neste caso, não armazena os dados, mas sim o banco de dados principal. Se o banco de dados possuir completo suporte espacial, ele irá armazenar e também realizar operações de manipulação. O SIG apenas será responsável pela manipulação visual dos dados.

Componente 3 – Análise – Este componente irá indicar as ferramentas e categorias de análise que poderão existir no sistema.

A. **Aplicações Especializadas** – As soluções classificadas nesta categoria fornecem aplicações específicas para um certo problema ou fornecidas por um SIG.
B. **OLAP** – Nesta categoria temos recursos de análise espacial ou de mineração de dados, como mapeamento, otimização, análise de rede etc. Interatividade é requerida, mas nem sempre presente.
C. **MOLAP** – Existe análise multidimensional espacial e, no melhor caso, as operações de manipulação espacial estão contidas dentro do banco de dados multidimensional.

A descrição de cada componente pode não ser precisa. Em alguns casos, o sistema pode não ter todas as características descritas, mas, em geral, uma determinada abordagem de um sistema de *data warehousing* espacial pode ser representada como um ponto (**X**,**Y**,**Z**) em um cubo multidimensional (Figura 8). A lista a seguir mostra uma avaliação de cada abordagem:

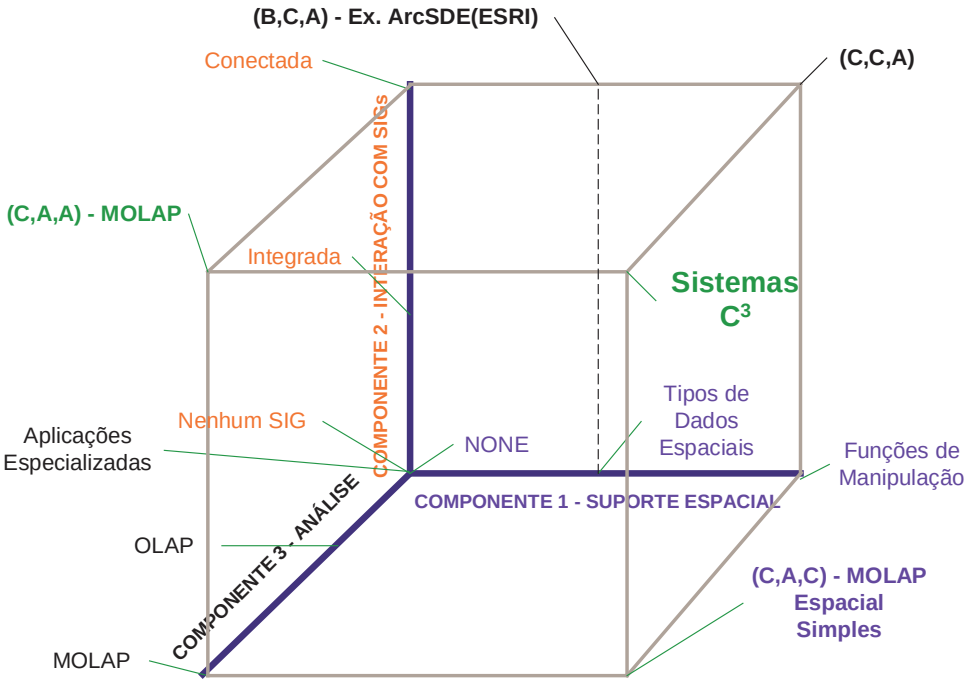

Figura 8 – Taxonomia de *data warehousing* espacial

- (A,A,A) representa um simples sistemas de mapeamento espacial (Braga at al. 1997), SIGs legados ou sistemas projetados para apenas uma funcionalidade espacial.
- (B,A,A) e (B,A,B) são especializados ou incluem aplicações especializadas que usam ou não o banco de dados para armazenamento, como o (A,A,B), no último caso (Morimoto 2001). Os sistemas (C,A,A) e (C,A,B) funcionam similarmente, mas tiram vantagem do suporte para funções espaciais do banco de dados.
- As abordagens (**X**,B,A) e (**X**,B,B) integram SIGs que possuem análise e mineração espacial, mas não exploram funcionalidades de armazenamento e manipulação espacial do banco de dados.

- Sistemas (**X**,C,A) e (**X**,C,B) utilizam todo o suporte espacial do banco de dados, com exceção das abordagens (A,C,**X**), que são sem significado dentro deste contexto. As abordagens (B,C,A) e (B,C,B) são mais comuns em arquiteturas tais como **ESRI ArcSDE** (ESRI 1998), e as arquiteturas (C,C,A) e (C,C,B) são modelos mais avançados para SIG - *data warehouse*.
- Sistemas (A,A,C) representam protótipos MOLAP (Papadias at al. 2001,Rauber, Tomsich e Riedel 2000), e todos os outros sistemas (**X**,**Y**,C) não existem realmente. A abordagem (B,A,C) pode armazenar tipos de dados espaciais no banco de dados, a despeito de ainda requererem a implementação de operações espaciais. Sistemas (C,A,C) são, geralmente, a ferramenta mais avançada de MOLAP, mas considerando a interação com SIGs nas categorias (**X**,B,C) e (**X**,C,C), não há um SIG com capacidades de MOLAP integradas. (C,B,C) e (B,B,C) são arquiteturas de pouca utilidade porque o suporte espacial do banco de dados não é explorado. Os sistemas (C,C,C) ou C^3 serão implementações bastante avançadas porque podem explorar o máximo das capacidades de todos os componentes.

As classificações acima demonstram que existem muitas formas de se criar uma arquitetura de *data warehousing* espacial, partindo de uma aplicação dedicada (A,A,A), passando por um simples SIG (A,B,A) e chegando ao presente modelo (B,C,A), onde o banco de dados é meramente usado como armazenamento. A arquitetura **ESRI ArcGIS** + **ArcSDE** + [**DB2 Spatial Extender/Informix Spatial Datablade/Oracle Spatial**] é um exemplo deste tipo de abordagem, mesmo sabendo-se que estes sistemas de bancos de dados oferecem suporte completo para funções espaciais.

O próximo passo para as arquiteturas de *data warehousing* espacial é implementar a abordagem (C,C,A) e encorajar as ferramentas de SIGs para usar o presente suporte de funcionalidades do banco de dados, tal como o **DB2 Spatial Extender** ou o **Informix Spatial Datablade**. O horizonte final destes sistemas está nas abordagens (C,C,B) e (C,C,C), onde a mineração de dados espacial e a análise multidimensional finalmente irá explorar o suporte espacial do banco de dados e chegar ao usuário final através dos SIGs.

O protótipo da ferramenta que será apresentada neste trabalho no Capítulo 6 e Capítulo 7 se encaixa na categoria (C,C,B) e também mostra uma prova de conceito para sistemas $\mathbf{C^3}$.

3.4 – Análise Multidimensional Espacial

3.4.1 – Introdução

Os conceitos que envolvem a criação um sistema da categoria (**X**,**Y**,C), para estender a análise multidimensional (MOLAP), são as dimensões e agregações espaciais. Existem várias questões a serem consideradas. A primeira delas é que o termo "dimensão espacial" não está ainda formalmente definido. Existem várias abordagens e em algumas delas uma dimensão espacial é simplesmente o mapeamento de entidades geográficas para seus nomes, como "ESTADOS" para "Identificador de Estados" (Shekhar, Lu e Zhang 2001). Neste trabalho, nós definimos dimensão espacial e mostramos como hierarquias espaciais podem ser criadas e agregações podem ser pré-computadas.

3.4.2 - Dimensões Espaciais

Existem duas formas de criar uma dimensão espacial. A primeira relaciona cada elemento da dimensão convencional com a posição no espaço. Por exemplo, o endereço de cada cliente seria convertido para coordenadas. A segunda forma é uma definição mais estrita de uma dimensão no cubo multidimensional -- ela posiciona cada valor de medida do modelo estrela no espaço. Suponha, por exemplo, que tenhamos um elemento de medida relacionado com as dimensões Tempo, Produto e Cliente. Cada medida expressa o número de produtos comprados por cada cliente num dado momento. Uma verdadeira dimensão espacial tem que produzir um conjunto de *tuplas* [Tempo, Produto, Cliente, Posição] que defina onde cada cliente estava no momento que cada produto foi comprado. Este seria o caso de compras feitas via dispositivos móveis, como celulares. Esta abordagem seria bastante custosa, dada a necessidade de registrar a posição de cada cliente. O mais provável seria registrar a posição da loja onde o cliente comprou o produto. A inserção de um atributo com dimensões espaciais dentro dos produtos comerciais de OLAP seria muito complexa e inapropriada. A solução mais interessante seria criar as dimensões espaciais dinamicamente, produzindo tabelas relacionadas às medidas através de identificadores.

3.4.3 - Hierarquias Espaciais

A agregação das medidas depende das hierarquias definidas entre as dimensões, por exemplo, Tempo, Cliente e Loja. As hierarquias espaciais geralmente se baseiam no relacionamento entre as dimensões, como "clientes dentro de uma área". Temos duas classificações possíveis para as hierarquias:

HIERARQUIAS ESPACIAIS ESTÁTICAS

A forma mais simples de construir uma hierarquia espacial é através de uma definição fixa do relacionamento entre as entidades espaciais. Suponha, por exemplo, que criemos uma chave A para cada estado e uma chave B para cada cidade que está localizada dentro de cada estado. A hierarquia entre *"cidades dentro de um estado"* seria determinada por uma tabela de relacionamento entre A e B. Esta tabela poderia ser criada uma só vez utilizando funções de relacionamento espacial. As entidades Estado e Cidade seriam mapeadas em dimensões no modelo estrela (Han, Stefanovic e Koperski 1998b,Silva e Campos 1998,Stefanovic 1997a,Zhou e Prasher 2004) e o cubo multidimensional seria construído usando-se a tabela de relacionamento entre A e B.

HIERARQUIAS ESPACIAIS DINÂMICAS

As hierarquias estáticas podem ser, de fato, bastante úteis e são aplicadas em diversas soluções. O maior problema desta abordagem é que não pode ser aplicada a relacionamentos espaciais que possam ser variáveis, como *"Ache as vendas de um certo produto para todos os clientes que moram até 10 km de cada uma de minhas filiais"*. Estes tipos de hierarquias podem ser chamadas de "hierarquias dinâmicas". As hierarquias dinâmicas possuem uma forma fixa e um parâmetro, geralmente não espacial. Vamos supor uma função genérica "*Inside*(*Dist*,Cliente,Filial)" que retorna os dados agregados do cliente a uma certa distância "*Dist*" das filiais. O exemplo acima ficaria:

"Ache as vendas de um certo produto Inside(Dist,Clientes,Filiais)"

No exemplo citado acima, poderíamos agregar o valor das vendas em diversas distâncias apenas substituindo o atributo distância "*Dist*". A agregação também pode ser de atributos espaciais, não só convencionais. Por exemplo, podemos querer gerar áreas agregadas aonde um certo volume de vendas ultrapasse um valor limite.

O maior problema das hierarquias dinâmicas é a sua implementação. Diferente das hierarquias estáticas, uma hierarquia dinâmica não pode ser simplesmente carregada no cubo multidimensional, pois os mecanismos convencionais dos bancos de dados multidimensionais não saberiam como lidar com o cálculo "*on the fly*", ou seja, em tempo de processamento da consulta dos dados do cubo. Dependendo do parâmetro, as junções entre as tabelas produziriam um conjunto de *tuplas* diferente. Uma solução possível, mas de pouco desempenho, é remontar a tabela de relacionamento para cada mudança do valor do parâmetro. Esta abordagem requer que todo o processamento seja

feito externamente e depois carregado no cubo multidimensional, perdendo, desta forma, o desempenho obtido pelo banco de dados multidimensional. Além disso, temos a perda no tempo de carga e recálculo do relacionamento.

3.4.4 - Sistemas de OLAP espacial

A taxonomia mostrada na seção 3.3.6 nos guia numa análise mais detalhada do avanço dos sistemas de *data warehouse espacial*. Não existe qualquer produto comercial que incorpore as características de análise multidimensional espacial dentro das ferramentas de banco de dados multidimensional. Estes sistemas estariam classificados na categoria (**X**,**Y**,C), sendo **X**,**Y** qualquer suporte ou método de comunicação com o banco de dados que possa ser usado. Existem alguns protótipos acadêmicos (Papadias at al. 2001,Rauber, Tomsich e Riedel 2000), mas que se restringem a propostas teóricas.

O objetivo final na criação de um sistema de *data warehouse espacial* completo, ou sistema C^3, é a construção de um banco de dados de *data warehouse* com suporte espacial completo e interativamente conectado com o SIG e o sistema de MOLAP. Nesta categoria de sistemas, o módulo de MOLAP seria totalmente capaz de utilizar o suporte espacial do banco de dados em suas consultas espaciais, enquanto que o SIG, em colaboração com o sistema de MOLAP, iria interativamente mostrar e manipular os dados multidimensionais. Poderíamos ter, por exemplo, hierarquias dinâmicas onde exibimos o resultado das agregações de vendas a **X** km de um certo ponto, e, além de visualizarmos os dados agregados, poderíamos também mostrar o mapa com a área selecionada.

As presentes capacidades dos sistemas (B,C,A), inclinando-se para (C,C,B), usando o **DB2 Spatial Extender** e o **Informix Spatial Datablade**, formam as fundações para a próxima geração de sistemas de arquitetura C^3. Neste momento, é necessário que os sistemas de MOLAP sejam conscientes de tipos de dados e funcionalidades espaciais. Isto ofereceria ao usuário capacidades espaciais que seriam direcionadas para o banco de dados de *data warehouse*. A interação entre SIGs, por outro lado, é bastante simples e direta, porque as capacidades de cada sistema não tem que ser expandidas, mas, de fato, conduzidas a interagir com os sistemas de MOLAP

3.5 – Conclusões

Mostramos aqui um panorama geral de SIGs e de como dados espaciais se inserem no contexto de banco de dados, com duas importantes contribuições:

1 - Uma melhora no entendimento e utilização das funções de manipulação espacial no SQL através da classificação apresentada na Tabela 6 e Tabela 7 da seção 3.2.2. Esta classificação, utilizada pela IBM nos manuais do **Spatial Extender**, foi baseada nas funções do **Spatial Extender** e do **Informix Spatial Datablade**. Contudo, estes dois produtos utilizam um padrão mundialmente aceito e proposto pelo OGC (Open GIS Consortium 1997).

2 - Uma taxonomia de integração entre SIGs e Sistemas de Bancos de Dados (seção 3.3.6). Esta taxonomia tem dois propósitos: mostrar o estado de evolução dos vários sistemas e guiar a construção de uma ferramenta capaz de tirar o máximo de vantagem do banco de dados no tratamento dos dados espaciais.

Nosso alvo neste trabalho é propor uma solução que possa ser aplicada e que tire vantagem de todos os recursos de um sistema gerenciador de bancos de dados. Iremos nos deter nos dois produtos líderes no mercado na área de banco de dados que já possuem suporte para funcionalidades espaciais. Mostraremos o desenvolvimento de uma metodologia e de uma ferramenta que irão utilizar estes produtos para validar o conceito de um sistema de *data warehousing* espacial de máxima evolução (**C**^{3}).

PARTE II

Sistemas de *Data Warehousing* para Tipos Abstratos de Dados

Capítulo 4 – Data Warehousing para Tipos Abstratos de Dados

4.1 – Introdução

A recente utilização do modelo objeto-relacional trouxe para dentro dos modelos de dados um grande número de tipos de dados complexos com semânticas específicas, como tipos de dados espaciais. Estes tipos de dados complexos, os TADs, já são bastante conhecidos dos bancos de dados objeto-relacionais atuais, os quais fornecem suporte mais do que adequado para trabalhar com eles.

Os sistemas de *data warehousing*, por outro lado, quase não levam em conta a manipulação e análise de modelos de dados contendo TADs. A pesquisa, bem como os produtos existentes para manipulação de TADs no processo de *data warehousing* e análise multidimensional, é apenas voltada para alguns tipos de dados específicos, como tipos de dados espaciais (CUBEWERX 2004a,Rao at al. 2003,Wang at al. 2003,Zhang.L. at al. 2004,Zhou e Prasher 2004) ou imagens (de Macedo e Fernandes 2003).

Neste trabalho, formalizamos um modelo para inclusão de TADs no processo de *data warehousing* e criamos um *framework* para sua construção. Este *framework* consiste em um conjunto de componentes e tabelas guias que organiza todas as funcionalidades necessárias para a realização do processo.

O fluxo de utilização do *framework* (Figura 9) começa na identificação dos TADs que desejamos incluir no processo de *data warehousing*. O passo seguinte utiliza as tabelas guias do *framework* para classificação das funções de manipulação de TADs já existentes no banco de dados, ou para construção de quaisquer outras necessárias. O *framework* possui ainda um conjunto de componentes e funcionalidades disponíveis para utilização com qualquer TAD, como catálogos de metadados genéricos ou procedimentos de transformação gerados automaticamente. Todas estas classificações e componentes do *framework* serão completamente traduzidas em um arquivo descritor em XML, que irá ser a entrada de um gerador automático de ferramentas de *data warehousing* para TADs. A ferramenta gerada irá fornecer as funcionalidades necessárias para estender o processo de ETL (ver seção 2.2) e análise multidimensional para os TADs de forma simples, consistente e flexível.

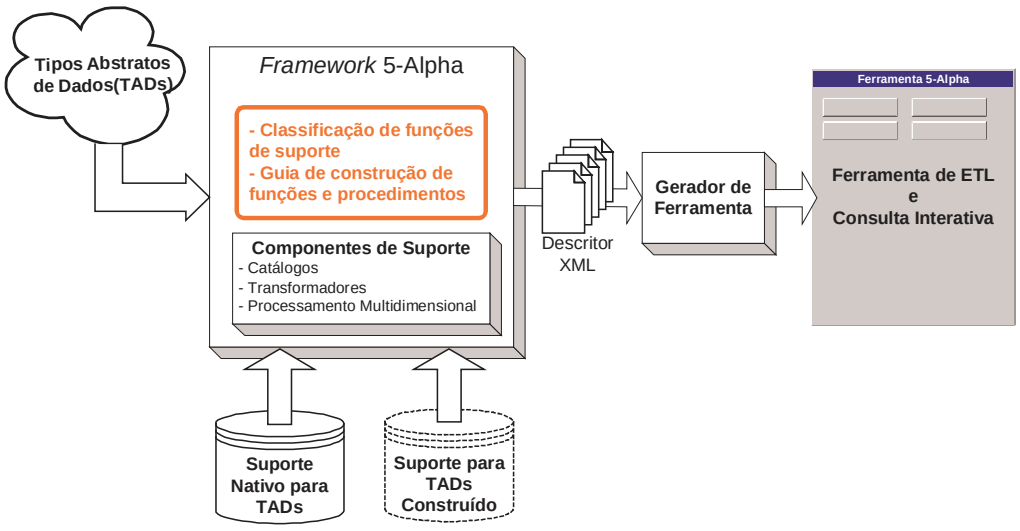

Figura 9 – Fluxo de utilização do *framework*

O *framework* é denominado de **5-Alpha** porque é composto de cinco itens mnemonicamente correspondentes às primeiras cinco letras do alfabeto (A, B, C, D, E). Eles contém uma metodologia que utilizam tabelas de classificação para guiar a inserção dos TADs dentro do processo de *data warehousing*. O resultado final desta construção será o descritor em XML usado como entrada para o gerador automático da ferramenta de consulta e análise. Os cinco itens do *framework* (A, B, C, D, E), mnemonicamente definidos, são:

1. Tipos (A)bstratos de Dados.
2. Funções usadas para expressar os operadores (B)ásicos (“=”, “<”, etc.) e conectivos lógicos (“AND”, “OR” etc.) aplicados a TADs.
3. Sistema de manipulação de (C)atálogos de metadados.
4. Suporte relacional para TADs do Banco de (D)ados.
5. Os procedimentos de *data warehouse* (E)stendido.

A Tabela 9 concentra o conjunto de guias e classificações que precisam ser realizadas para utilização do *framework*. Ela foi construída de forma a espelhar o seu fluxo de construção, bem como a relação de dependência entre cada item (Figura 10). O item **(A**) contém os TADs classificados hierarquicamente. O item (**C**) vem se sobrepor aos TADs para permitir sua manipulação e registro em nível de metadados. Ele será utilizado pelos itens (**B**) e (**D**) que constroem todo o suporte relacional. O processo de *data warehousing* estendido em si vai se apoiar diretamente sobre o suporte de

metadados e o suporte relacional. A extração apenas dependerá do suporte de metadados para o registrar os elementos de entrada, os mapeamentos, as transformações de formatos etc. As transformações são construídas através da definição de quais funções do suporte relacional utilizar. As funções de análise vão conter todos os procedimentos para agregação e *drilldown* já prontos, apenas dependendo das funções de relacionamento que irão mapear as hierarquias.

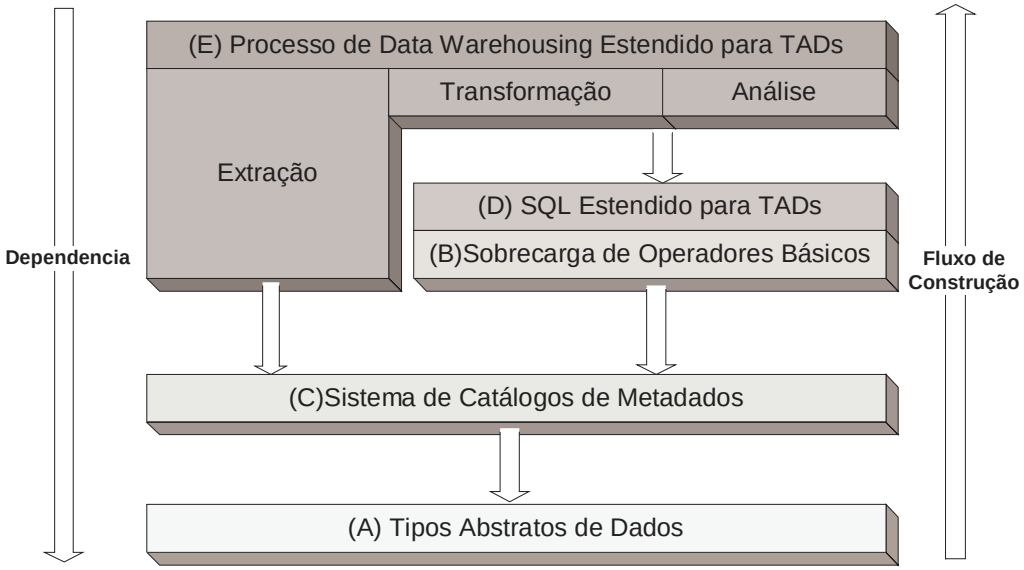

Figura 10 – Estrutura de construção das tabelas do *framework*

4.2 – Utilização do *framework*

4.2.1 – Item A – Classificação dos TADs

A lista de T(A)Ds envolvidos no processo de *data warehousing*, bem como os relacionamentos hierárquicos entre eles, deve ser classificada segundo a codificação especificada no *framework*. Os TADs podem estar relacionados através de hierarquias -- por exemplo, um polígono é a especialização de uma linha poligonal --, ou ainda através de relações "todo-parte", como por exemplo, um tipo "plano cromático" faria parte de TADs representando "*imagens*". A codificação utilizada pelo *framework* é bastante direta:cada item X principal é indicado pelo código [A- X -Y- ...], cada filho de ***X*** é definido pelo código ***Y*** e assim por diante. Um exemplo é:

[A-1-0] - *Veículo*

[A-1-1] - *Veículo Terrestre*

[A-1-1-1] - *Caminhão*

[A-1-2] - *Veículo Aquático*

Tabela 9 - Modelo de *Data Warehouse* Estendido

(**E**) – Processo de *Data Warehousing* Estendido para TADs		
XW[**PE**$_{\text{Extração}}$]	***X***W[**PT**$_{\text{Transformação}}$]	***X***W[**PA**$_{\text{Análise}}$]
Importação **PEI**$^{n}_{\text{Importação}}$([***ArqExt*** | ***TabList***$^{B}_{\text{fonte}}$], ***TabList***$^{A}_{\text{destino}}$[,DW$_{\text{parâmetro}}$]) **PEA**$^{n}_{\text{Agregação}}$([***ArqExt*** | ***TabList***$^{B}_{\text{fonte}}$], ***TabList***$^{A}_{\text{destino}}$[,DW$_{\text{parâmetro}}$]) **PES**$^{n}_{\text{Sequencializador}}$([***ArqExt*** | ***TabList***$^{B}_{\text{fonte}}$], ***TabList***$^{A}_{\text{destino}}$[,DW$_{\text{parâm.}}$]) **Préprocessamento** **PEP**$^{n}_{\text{Preprocessam.}}$(***TabList***$^{A}_{\text{fonte}}$, ***TabList***$^{A}_{\text{destino}}$[,DW$_{\text{parâmetro}}$]) **PEL**$^{n}_{\text{Limpeza}}$(***TabList***$^{A}_{\text{fonte}}$, ***TabList***$^{A}_{\text{destino}}$[,DW$_{\text{parâmetro}}$]) **PEV**$^{n}_{\text{Validação}}$(***TabList***$^{A}_{\text{fonte}}$, ***TabList***$^{A}_{\text{destino}}$[,DW$_{\text{parâmetro}}$])	**PT[S/M]R**$^{n}_{\text{Seleção}}$(***Tabs***$_{\text{fonte}}$,***Tabs***$_{\text{destino}}$, **F**$_{\text{sup}}$[,Param]) **PT[S/M]S**$^{n}_{\text{Separação}}$(***Tabs***$_{\text{fonte}}$, ***Tab***$_{\text{destino}}$, **F**$_{\text{sup}}$ [,Param]) **PT[S/M]U**$^{n}_{\text{Concatenação}}$(***Tabs***$_{\text{fonte}}$, ***Tabs***$_{\text{destino}}$, **F**$_{\text{sup}}$ [,Param]) **PT[S/M]A**$^{n}_{\text{Sumarização}}$(***Tabs***$_{\text{fonte}}$,***Tabs***$_{\text{destino}}$, **F**$_{\text{sup}}$ [,Param]) **PT[S/M]T**$^{n}_{\text{Transformação}}$(***Tabs***$_{\text{fonte}}$,***Tabs***$_{\text{destino}}$, **F**$_{\text{sup}}$ [,Param]) **PT[S/M]E**$^{n}_{\text{Enriquecimento}}$(***Tabs***$_{\text{fonte}}$, ***Tabs***$_{\text{destino}}$, **F**$_{\text{sup}}$ [,Param]) **PT[S/M]C**$^{n}_{\text{Conversão}}$(***Tabs***$_{\text{fonte}}$, ***Tabs***$_{\text{destino}}$, **F**$_{\text{sup}}$ [,Param]) **PT[S/M]D**$^{n}_{\textit{DrillDown}}$(***Tabs***$_{\text{fonte}}$, ***Tabs***$_{\text{destino}}$, **F**$_{\text{sup}}$[,Param])	***X***HW$_{\text{hierarquia}}$ **PASH1**$^{n}_{\text{Hierarquia}}$(***X***W$_{\text{col}}^{1}$,***X***W$_{\text{col}}^{2}$ [,**FRS2**$^{n}_{\text{Relação}}$[,**FPSD**$^{n}_{\text{DrillDown}}$]]) **PASH2**$^{n}_{\text{Hierarquia}}$(***X***HW$_{\text{hierarquia}}$,***X***W$_{\text{col}}$[,**FRS2**$^{n}_{\text{Relação}}$[,**FPSD**$^{n}_{\text{DrillDown}}$]]) **PAMH1**$^{n}_{\text{Hierarquia}}$(***X***W$_{\text{col}}^{1}$, ***X***W$_{\text{col}}^{2}$[,**FRM2**$^{n}_{\text{Relação}}$[,**FPMD**$^{n}_{\text{DrillDown}}$]]) **PAMH2**$^{n}_{\text{Hierarquia}}$(***X***HW$_{\text{hierarquia}}$, ***X***W$_{\text{col}}$[,**FRM2**$^{n}_{\text{Relação}}$[,**FPMD**$^{n}_{\text{DrillDown}}$]]) **PASA**$^{n}_{\text{AddData}}$(***X***HW$_{\text{hierarquia}}$, ***X***W$_{\text{col}}$, *FnAgg*[,*FnDrillDown*]) **PASS**$^{n}_{\text{Agregação}}$(***X***HW$_{\text{hierarquia}}$, Nível, DW$_{\text{parâmetro}}$) **PASD**$^{n}_{\text{DrillDown}}$(***X***HW$_{\text{hierarquia}}$, *Nível*) [**PASG**$_{\text{Análise}}$]
(**D**) - SQL Extendido para o TAD *X*		
XW[**F**$_{\text{Seleção}}$]	***X***W[**F**$_{\text{Projeção}}$]	***X***W[**F**$_{\text{Agregação}}$]
FRS1$^{n}_{\text{Relação}}$(***X***W$_{\text{tipo}}$) **FRS2**$^{n}_{\text{Relação}}$(***X***W$_{\text{tipo}}^{1}$,***X***W$_{\text{tipo}}^{2}$[,Param]) **FRS3**$^{n}_{\text{Relação}}$ (***X***W$_{\text{tipo}}$, Param) **FRM1**$^{n}_{\text{Relação}}$ (***X***W$_{\text{coluna}}$) **FRM2**$^{n}_{\text{Relação}}$ (***X***W$_{\text{col}}^{1}$, (***X***W$_{\text{tipo}}$|***X***W$_{\text{col}}^{2}$), [Param]) **FRM3**$^{n}_{\text{Relação}}$((***X***W$_{\text{tipo}}$|***X***W$_{\text{col}}^{1}$),[Par|Par$_{\text{Set}}$])	**FPST**$^{n}_{\text{Transformação}}$(***X***W$_{\text{tipo}}$,[Parâm]) **FPSC**$^{n}_{\text{Conversão}}$(***X***W$_{\text{tipo}}^{1}$,[***X***W$_{\text{tipo}}^{2}$|Parâm]) **FPSC**$^{n}_{\text{Combinação}}$(***X***W$_{\text{tipo}}^{1}$,***X***W$_{\text{tipo}}^{2}$[,Parâm]) **FPSD**$^{n}_{\text{DrillDown}}$(***X***W$_{\text{tipo}}^{1}$, [***X***W$_{\text{tipo}}^{2}$|Parâm]) **FPMT**$^{n}_{\text{Transformação}}$(***X***W$_{\text{coluna}}$, Parâm) **FPMC**$^{n}_{\text{Conversão}}$(***X***W$_{\text{coluna}}$, Parâm) **FPMD**$^{n}_{\text{DrillDown}}$(***X***W$_{\text{coluna}}$, Parâm)	**FA1**$^{n}_{\text{Agregação}}$(***X***W$_{\text{col}}^{1}$,[***X***W$_{\text{tipo}}$|***X***W$_{\text{col}}^{2}$| Param]) **FA2**$^{n}_{\text{Agregação}}$(***X***W$_{\text{col}}^{1}$,[***X***W$_{\text{tipo}}$|***X***W$_{\text{col}}^{2}$| Param]) **FA11**$^{n}_{\text{Agregação}}$(***X***W$_{\text{coluna}}$[,Param]) **FA12**$^{n}_{\text{Agregação}}$(***X***W$_{\text{coluna}}$[,Param]) **FA21**$^{n}_{\text{Agregação}}$(***X***W$_{\text{coluna}}^{1}$ [***X***W$_{\text{tipo}}$ | ***X***W$_{\text{coluna}}^{2}$]) **FA22**$^{n}_{\text{Agregação}}$(***X***W$_{\text{coluna}}^{1}$[,***X***W$_{\text{tipo}}$ |,***X***W$_{\text{coluna}}^{2}$])
(**B**) - ***X***W$_{\text{transparentes}}$ = Suporte de Operadores Básicos (“=”, “≠”, AND, OR, etc.)		
(**C**) - ***X***W$_{\text{MD}}$ – Sistema de Catálogos de Metadados		
XW$_{\text{CATALOGOS}}$	***X***W$_{\text{Manipulação de Catálogos}}$	***X***W$_{\text{Manipulação de Elementos}}$
PCS[D|M|H]$^{n}_{\text{Registro}}$(Parâmetros) **PCS[D|M|H]**$^{n}_{\text{Remoção}}$(Parâmetros)		**FCSC**$_{\text{Criação}}$(DW$_{\text{parâmetro}}$) **FCMC**$_{\text{Criação}}$(DW$_{\text{parâmetro}}$) **FC[S|M]A**$^{n}_{\text{Acesso}}$(***X***W$_{\text{tipo}}$) **FC[S|M]P**$^{n}_{\text{Propriedade}}$(***X***W$_{\text{tipo}}$)
(**A**) - ***X***W$_{\text{tipos}}$ = Tipos Abstratos de Dados(TAD)		

4.2.2 – Item B – Classificação dos operadores básicos

A classificação do conjunto de funções que sobrecarrega cada um dos operadores (**B**)ásicos de comparação ("=", "<", etc.) e dos conectores lógicos ("AND", "OR" etc.) é feita usando-se a Tabela 10, a qual mostra as funções que devem ser criadas ou classificadas para cada operador. Para classificar funções já existentes, não se depende de muitas regras, pois segue-se a semântica da função, e a criação é bastante direta.

A sobrecarga dos operadores básicos de comparação visa definir a semântica equivalente de cada um dos operadores quando aplicados aos TADs listados no item A. Por exemplo, se um TAD representa "*círculos*", o operador "<" pode indicar se um círculo é menor do que outro. A extensão dos conectores lógicos pode parecer redundante, dado que os seus operandos e resultados são sempre tipos de dados booleanos, mas ela visa atender a alguns casos específicos. Em primeiro lugar, as funções de suporte à seleção (item D) algumas vezes retornam valores condicionais, mas sem usar tipos de dados booleanos, e sim inteiros com valor 0 e 1, por exemplo. Neste caso precisamos de funções que estendam os operadores que entendem estes valores. Um outro caso mais complexo pode ocorrer quando o relacionamento entre TADs pode retornar valores lógicos multivalorados ou mesmo nebulosos. Suponha que no exemplo apresentado, a extensão do operador "<" não retorne verdadeiro ou falso, mas um valor de pertinência nebuloso (Yen 1999,Zadeh 1988,Zadeh 1965,Zimmermann 1990) entre 0 e 1. A função de extensão nebulosa do conectivo "AND" iria computar os valores de pertinência entre os dois operandos, resultando em um valor lógico aceitável. Uma descrição completa do funcionamento de conectivos lógicos nebulosos pode ser encontrada na tese de mestrado deste mesmo autor (Braga 1998a).

Tabela 10 - Sobrecarga de operadores básicos

		SQL	Semântica	5-Alpha
Operadores de Comparação	Igualdade	=	Igual	**FBEQ**$_{\text{Comparação}}$
		≠	Diferente	**FBNE**$_{\text{Comparação}}$
	Menor	<	Menor	**FBLS**$_{\text{Comparação}}$
		≤	Menor ou Igual	**FBLE**$_{\text{Comparação}}$
	Maior	>	Maior	**FBGT**$_{\text{Comparação}}$
		≥	Maior ou Igual	**FBGE**$_{\text{Comparação}}$
Conectivos Lógicos	Conjunção(∨)	**AND**	"E"	**FBAN**$_{\text{Conectivo}}$
	Disjunção(∧)	**OR**	"OU"	**FBOR**$_{\text{Conectivo}}$
	Negação(¬)	**NOT**	"NÃO"	**FBNO**$_{\text{Conectivo}}$

4.2.3 – Item C – Classificação do suporte de metadados

O sistema de manipulação de (**C**)atálogos de metadados contém três subconjuntos de elementos: a lista de catálogos de TADs, os procedimentos de manipulação de catálogos (inserir elementos, remover etc.) e as funções de criação e acesso a variáveis de TADs. O *framework* contém tabelas guias para classificação do suporte existente aos primeiros dois itens, o que pode ser visto na célula de manipulação de catálogos da Tabela 9, além de um modelo genérico de catálogo para TADs, que pode ser usado caso o suporte do banco não o possua. As classes de funções de manipulação de elementos podem ter uma classificação um pouco mais complexa dado que elas também podem ser usadas no suporte à projeção. A Tabela 11 contém as guias para classificação ou construção destas funções dependendo dos seus parâmetros e tipos de retorno. As funções do tipo **FCXC**$^{n}_{\text{Criação}}$ servem para a criação de elementos e as do tipo **FCXA**$^{n}_{\text{Acesso}}$ para obter seus atributos. As funções de tipo **FCXP**$^{n}_{\text{Propriedade}}$ retornam propriedades calculadas sobre os atributos dos elementos. A metodologia de classificação consiste em primeiro avaliar os parâmetros de entrada e retorno, e depois considerar a semântica específica de cada função. No caso das funções de construção, não há ambigüidade, e no caso das de acesso ou propriedade, a semântica irá decidir. Os detalhes de utilização desta tabela estão na seção 4.3.2 e o seu desenvolvimento na seção 5.5.2.

Tabela 11 - Categorias de funções de suporte de metadados

	Retorno			
	Elementos		**Conjuntos**	
Parâmetros	TAD	TPD	TAD_{set}	TPD_{set}
TAD	**FCSA**$^{n}_{\text{Acesso}}$ **FCSP**$^{n}_{\text{Propriedade}}$	**FCSA**$^{n}_{\text{Acesso}}$ **FCSP**$^{n}_{\text{Propriedade}}$		**FCMA**$^{n}_{\text{Acesso}}$ **FCMP**$^{n}_{\text{Propriedade}}$
TPD	**FCSC**$^{n}_{\text{Criação}}$		**FCSC**$^{n}_{\text{Criação}}$	
TPD_{set}	**FCMC**$^{n}_{\text{Criação}}$		**FCMC**$^{n}_{\text{Criação}}$	
TAD×TPD	**FCSA**$^{n}_{\text{Acesso}}$ **FCSP**$^{n}_{\text{Propriedade}}$	**FCSA**$^{n}_{\text{Acesso}}$ **FCSP**$^{n}_{\text{Propriedade}}$		**FCMA**$^{n}_{\text{Acesso}}$ **FCMP**$^{n}_{\text{Propriedade}}$
TPD × TPD	**FCSC**$^{n}_{\text{Criação}}$		**FCSC**$^{n}_{\text{Criação}}$	
TPD_{set}×TPD_{set}	**FCMC**$^{n}_{\text{Criação}}$		**FCMC**$^{n}_{\text{Criação}}$	
TPD_{set}×TPD	**FCMC**$^{n}_{\text{Criação}}$		**FCMC**$^{n}_{\text{Criação}}$	

4.2.4 – Item D – Classificação do suporte relacional

O suporte relacional no Banco de (**D**)ados contém tabelas para guiar a classificação ou construção de funções que estendam os operadores relacionais (seleção, projeção etc.) para manipulação dos TADs. Na seção (**D**) da Tabela 9, temos os três

grupos de classes de funções de suporte relacional do *framework*. A Tabela 12 deve ser utilizada como guia para classificação ou criação de funções nestes grupos. Esta tabela é orientada primeiramente por grupo (projeção, agregação ou seleção), depois pelos tipos dos parâmetros de entrada e saída, e, por último, pela semântica específica de cada função. Tomemos como exemplo a função "*ST_Buffer*", parte da especificação da OGC(Open GIS Consortium e Ryden 1999), descrita abaixo. A especificação define que ela deve criar um linha poligonal fechada (*Buffer*) em torno de geometrias (neste caso um TAD). Ela recebe uma geometria como parâmetro, e a distância em volta dela que determinará a "área" de "*Buffer*". A definição mais formal de *Buffer* na verdade inclui toda região da área e não somente a linha de contorno em si.

ST_Geometry ST_Buffer(ST_Geometry **TAD$_1$**, *Double* **Dist***)*

Primeiramente, identificamos na sexta linha da primeira coluna da Tabela 12os dois tipos de funções: transformação ou conversão. Analisando a semântica, verificamos que ela retorna o mesmo TAD que recebe como parâmetro. Desta forma, classificamos esta função como de conversão "**FPST1**$^{n}_{\text{Transformação}}$".

A utilização destas tabelas como guia de construção é menos definida. A tabela irá mostrar apenas as categorias de funções possíveis a serem construídas, mas fica a cargo do analista determinar a semântica de negócios específica. O analista deve escrever as categorias de funções e analisar o TAD em nível de negócio para descobrir que tipo de funcionalidade, dentro daquela categoria, poderia ter significado ou utilidade.

As funções de agregação podem ter variados comportamentos dependendo dos parâmetros de entrada e de sua semântica, por isso colocamos a sua nomenclatura à parte na Tabela 13.

Algumas funções do item (**B**) também podem ser usadas para suporte à junção, só que aqui as funções que vão suportar a junção possuem uma semântica mais ampla, envolvendo qualquer relacionamento entre TADs, ou mesmo entre TADs e TPDs. Um exemplo onde podemos visualizar ocorre entre uma função que avalie a igualdade entre dois TADs representando imagens, e uma outra que indique se um TAD do tipo "ponto" está contido dentro de um TAD do tipo "*círculo*". A primeira estende a semântica do operador de igualdade e a segunda representa um relacionamento não classificado entre dois TADs.

Tabela 12 - Categorias de funções de suporte relacional

Parâmetros		Retorno				
		Elementos		Conjuntos		Booleano
		TAD	TPD	TAD_{set}	TPD_{set}	
Unários	TAD	**FPST1**$^{n}_{\text{Transformação}}$ **FPSC1**$^{n}_{\text{Conversão}}$	**FPSC1**$^{n}_{\text{Conversão}}$	**FPST1**$^{n}_{\text{Transformação}}$ **FPSC1**$^{n}_{\text{Conversão}}$ **FPSD**$^{n}_{\text{DrillDown}}$	**FPSC1**$^{n}_{\text{Conversão}}$	**FRS1**$^{n}_{\text{Relação}}$
	TPD		**FPSC2**$^{n}_{\text{Conversão}}$ **FPST2**$^{n}_{\text{Transformação}}$		**FPSC2**$^{n}_{\text{Conversão}}$ **FPST2**$^{n}_{\text{Transformação}}$	**FRSTD**$^{n}_{\text{Relação}}$
	TAD_{set}	**FA11**$^{n}_{\text{Agregação}}$	**FA1**$^{n}_{\text{Agregação}}$	**FA12**$^{n}_{\text{Agregação}}$	**FA2**$^{n}_{\text{Agregação}}$	**FRM1**$^{n}_{\text{Relação}}$
	TPD_{set}		**FASTD1**$^{n}_{\text{Agregação}}$		**FASTD2**$^{n}_{\text{Agregação}}$	**FRMSTD**$^{n}_{\text{Relação}}$
Binários de Elementos	TAD×TAD	**FPSX**$^{n}_{\text{Combinação}}$	**FPSC1**$^{n}_{\text{Conversão}}$	**FPSD**$^{n}_{\text{DrillDown}}$	**FPSC1**$^{n}_{\text{Conversão}}$	**FRS2**$^{n}_{\text{Relação}}$
	TAD×TPD	**FPST1**$^{n}_{\text{Transformação}}$ **FPSC1**$^{n}_{\text{Conversão}}$	**FPSC1**$^{n}_{\text{Conversão}}$	**FPST1**$^{n}_{\text{Transformação}}$ **FPSC1**$^{n}_{\text{Conversão}}$ **FPSD**$^{n}_{\text{DrillDown}}$	**FPSC1**$^{n}_{\text{Conversão}}$	**FRS3**$^{n}_{\text{Relação}}$
Binários de Conjuntos	TAD_{set}×TAD_{set}	**FA21**$^{n}_{\text{Agregação}}$	**FA1**$^{n}_{\text{Agregação}}$	**FA22**$^{n}_{\text{Agregação}}$	**FA2**$^{n}_{\text{Agregação}}$	**FRM2**$^{n}_{\text{Relação}}$
	TAD_{set}×TAD	**FA21**$^{n}_{\text{Agregação}}$	**FA1**$^{n}_{\text{Agregação}}$	**FA22**$^{n}_{\text{Agregação}}$	**FA2**$^{n}_{\text{Agregação}}$	**FRM2**$^{n}_{\text{Relação}}$
	TAD_{set}×TPD	**FA11**$^{n}_{\text{Agregação}}$	**FA1**$^{n}_{\text{Agregação}}$	**FA12**$^{n}_{\text{Agregação}}$	**FA2**$^{n}_{\text{Agregação}}$	**FRM3**$^{n}_{\text{Relação}}$
	TAD_{set}×TPD_{set}	**FA11**$^{n}_{\text{Agregação}}$	**FA1**$^{n}_{\text{Agregação}}$	**FA12**$^{n}_{\text{Agregação}}$	**FA2**$^{n}_{\text{Agregação}}$	**FRM3**$^{n}_{\text{Relação}}$
	TPD_{set}×TAD	**FPMT**$^{n}_{\text{Transformação}}$ **FPMC**$^{n}_{\text{Conversão}}$	**FPMC**$^{n}_{\text{Conversão}}$	**FPMT**$^{n}_{\text{Transformação}}$ **FPMC**$^{n}_{\text{Conversão}}$ **FPMD**$^{n}_{\text{DrillDown}}$	**FPMC**$^{n}_{\text{Conversão}}$	**FRM3**$^{n}_{\text{Relação}}$
	TPD × TPD		**FPSC2**$^{n}_{\text{Conversão}}$ **FPST2**$^{n}_{\text{Transformação}}$		**FPSC2**$^{n}_{\text{Conversão}}$ **FPST2**$^{n}_{\text{Transformação}}$	**FRSTD**$^{n}_{\text{Relação}}$
	TPD_{set}×TPD_{set}		**FASTD1**$^{n}_{\text{Agregação}}$		**FASTD2**$^{n}_{\text{Agregação}}$	**FRMSTD**$^{n}_{\text{Relação}}$
	TPD_{set}×TPD		**FASTD1**$^{n}_{\text{Agregação}}$		**FASTD2**$^{n}_{\text{Agregação}}$	**FRMSTD**$^{n}_{\text{Relação}}$

Tabela 13 - Categoria de funções de agregação

	Parâmetro de Entrada	Operações(*X*)		Retorno(*Y*)	
FA*XY*$^{n}_{Agregação}$	TAD_{set}	**1**	- Conversão - Transformação	**1**	TAD
		2	**1** + Combinação	**2**	TAD_{set}
FA*Y*$^{n}_{Agregação}$	TAD_{set}	- Conversão		**1**	TPD
				2	TPD_{set}
FASTD*Y*$^{n}_{Agregação}$	TPD_{set}	- Conversão - Transformação		**1**	TPD
				2	TPD_{set}

4.2.5 – Item E – Construção do sistema de *Data Warehouse* Estendido

Os procedimentos de *data warehouse* (**E**)stendido estão divididos em extração, transformação e análise. O *framework* contém um conjunto de tabelas simples para classificar os procedimentos de extração já existentes (Tabela 14), dado que não há como prever a imensa quantidade de possíveis formatos externos e aplicações legadas às fontes de dados. Os procedimentos de transformação, por outro lado, estão classificados através da Tabela 15, a qual mapeia as funções de suporte relacional classificadas no item (**D**). Esta classificação permite definir transformadores que podem ser construídos automaticamente através da ferramenta mostrada no Capítulo 6, apenas indicando em quais funções de suporte relacional eles estão baseados. De modo similar, o sistema de análise multidimensional é construído indicando-se quais funções de suporte de relacionamento entre TADs (item (**D**) ou (**B)**) vão definir as hierarquias de agregação entre os TADs. O *framework* contém um formato de classificação que especifica cada hierarquia, sua função de relacionamento, os atributos das tabelas que serão agregados, bem como suas respectivas funções. A ferramenta de análise que será gerada irá utilizar estas informações para realizar a agregação e o *drilldown* do processo de análise multidimensional.

Tabela 14 - Categorias de funções de extração

Fonte		Destino - Banco A Lista de Tabelas
Componente Externo		**PEI**$^{n}_{Importação}$(***ArquivoExt***, ***TabList***$^{A}_{destino}$[,$DW_{parâmetro}$])
		PEA$^{n}_{Agregação}$(***ArquivoExt***, ***TabList***$^{A}_{destino}$[,$DW_{parâmetro}$])
		PES$^{n}_{Sequenciador}$(***ArquivoExt***, ***TabList***$^{A}_{destino}$[,$DW_{parâmetro}$])
Banco A	Tabelas	**PEP**$^{n}_{Preprocessamento}$(***TabList***$^{A}_{fonte}$, ***TabList***$^{A}_{destino}$[,$DW_{parâmetro}$])
		PEL$^{n}_{Limpeza}$(***TabList***$^{A}_{fonte}$, ***TabList***$^{A}_{destino}$[,$DW_{parâmetro}$])
		PEV$^{n}_{Validação}$(***TabList***$^{A}_{fonte}$, ***TabList***$^{A}_{destino}$[,$DW_{parâmetro}$])
Banco B	Tabelas	**PEP**$^{n}_{Preprocessamento}$(***TabList***$^{B}_{fonte}$, ***TabList***$^{A}_{destino}$[,$DW_{parâmetro}$])
		PEL$^{n}_{Limpeza}$(***TabList***$^{B}_{fonte}$, ***TabList***$^{A}_{destino}$[,$DW_{parâmetro}$])
		PEV$^{n}_{Validação}$(***TabList***$^{B}_{fonte}$, ***TabList***$^{A}_{destino}$[,$DW_{parâmetro}$])

Tabela 15 - Classificação de funções de transformação

Parametrização		Retorno: Elementos: TAD	Retorno: Elementos: TPD	Retorno: Conjuntos: TAD_{set}	Retorno: Conjuntos: TPD_{set}	Retorno: Booleano
Unária	TAD	**PTSS[1/2]**^{n} Separação **PTST1**^{n} Transformação **PTSE1**^{n} Enriquecimento	**PTSS1**^{n} Separação **PTSS2**^{n} Separação **PTSC1**^{n} Conversão	**PTSD**^{n} *DrillDown*	**PTSC1**^{n} Conversão	**PTSR**^{n} Seleção
	TPD	**PTSE2**^{n} Enriquecimento		**PTSE2**^{n} Enriquecimento		
	TAD_{set}	**PTSA1**^{n} Sumarização	**PTMS**^{n} Separação **PTSC2**^{n} Conversão	**PTST2**^{n} Transformação **PTME1**^{n} Enriquecimento **PTMS2**^{n} Separação	**PTMC2**^{n} Conversão **PTMS3**^{n} Separação	**PTMR**^{n} Seleção
	TPD_{set}	**PTME2**^{n} Enriquecimento		**PTME2**^{n} Enriquecimento		
Binária de Elementos	TAD×TAD	**PTSU**^{n} Concatenação	**PTSC1**^{n} Conversão	**PTSD**^{n} *DrillDow*	**PTSC1**^{n} Conversão	**PTSR**^{n} Seleção
	TAD×TPD	**PTST1**^{n} Transformação				
Binária de Conjuntos	TAD_{set}×TAD_{set}	**PTMU**^{n} Concatenação	**PTSC2**^{n} Conversão	**PTMU**^{n} Concatenação	**PTMC2**^{n} Conversão **PTMS3**^{n} Separação	**PTMR**^{n} Seleção
	TAD_{set}×TAD	**PTSA2**^{n} Sumarização		**PTMA2**^{n} Sumarização		
	TAD_{set}×TPD	**PTSA1**^{n} Sumarização		**PTMA1**^{n} Sumarização **PTME1**^{n} Enriquecimento **PTMT2**^{n} Transformação **PTMS2**^{n} Separação **PTMD**^{n} *DrillDown*		
	TAD_{set}×TPD_{set}					
	TPD_{set}×TAD	**PTMS1**^{n} Separação **PTMT1**^{n} Transformação **PTME3**^{n} Enriquecimento	**PTMS1**^{n} Separação **PTMC1**^{n} Conversão	**PTMD**^{n} *DrillDown*	**PTMC1**^{n} Conversão	
	TPD× TPD	**PTME2**^{n} Enriquecimento		**PTME2**^{n} Enriquecimento		
	TPD_{set}×TPD_{set}					
	TPD_{set}×TPD					

4.2.6 – Papéis de Utilização

O *framework* **5-Alpha** pode ser utilizado para duas finalidades: desenvolvimento de soluções específicas ou construção de ferramentas gerais. No primeiro caso, teremos um problema de negócio, um conjunto de TADs já modelados e as funções que já tiverem sido desenvolvidas ou estejam disponíveis por algum pacote comercial. O *framework* irá servir primeiramente como classificador de todas estas funções e, além de proporcionar as ferramentas de consulta e análise multidimensional, vai mostrar as funções e componentes que ainda faltarem na arquitetura.

A segunda abordagem de utilização é a construção de ferramentas gerais para *data warehousing* envolvendo TADs. Vamos supor, por exemplo, que desejemos criar uma ferramenta para realização de *data warehousing* para TADs representando entidades financeiras, como ações, opções, *commodities* etc. O *framework* irá servir como guia para construção de funções para determinar relacionamento entre ações ou a conversão de um tipo de ação em outra forma de aplicação. Já a ferramenta criada serviria para análise de evolução das ações no tempo e o relacionamento agregado entre elas.

O conjunto de passos necessários para um ou outro papel de utilização é bem direto e simples. A lista abaixo mostra quais as tarefas necessárias a cada caso:

1 - <u>Desenvolvimento de solução</u>

1.1 - Identificar todos os TADs envolvidos no processo de análise

1.2 - Verificar as operações e consultas que tenham que ser executadas sobre os TADs

1.3 - Classificar dentro do *framework* todas as funções de suporte já existentes no banco de dados, sejam as já implementadas ou aquelas advindas de algum produto comercial

1.4 - Seguir o sistema de codificação exaustiva do *framework* para levantar possíveis funcionalidades úteis, mas ainda não identificadas.

1.5 - Produzir o descritor em XML do *framework*

1.6 - Gerar a ferramenta de ETL e consulta a partir do descritor

2 - <u>Construção de ferramenta gerais</u>

2.1 - Levantar os TADs que se deseja oferecer

2.2 - Seguir o sistema de codificação exaustiva do *framework* para levantar todas as possíveis funcionalidades aplicáveis aos TADs identificados. Deve-se verificar, para cada possível função do *framework*, se a funcionalidade é aplicável ou útil.

2.3 - Produzir o descritor em XML do *framework*

2.4 - Gerar a ferramenta de ETL e consulta a partir do descritor

Mostramos uma visão geral de utilização do *framework* em dois papéis de utilização, mas acreditamos que outros papéis poderiam ser possíveis, como o de uma metodologia didática de análise de tipos complexos no processo de *data warehousing*. Na seção seguinte, iremos mostrar em maior detalhe cada etapa do *framework* e a construção do descritor em XML. No Capítulo 5, apresentaremos o processo de desenvolvimento e criação do *framework*.

4.3 – Descrição detalhada do *Framework* 5-Alpha

A construção de um modelo de *data warehouse* estendido requer a especificação de uma arquitetura de suporte, ou seja, as ferramentas que serão utilizadas para implementação do *data warehouse* estendido para o tipo de dado ***X***. Esta arquitetura de suporte deve fornecer um conjunto mínimo de funcionalidades para construir o processo de *data warehousing*. Evidentemente, uma maior ou menor disponibilidade das funções de suporte irá estabelecer o nível das capacidades de extração, transformação e análise (Seção 2.2). Estas funcionalidades consistem em funções e tipos embutidos no SQL, assim como métodos de processamento incorporados ao banco de dados.

4.3.1 – Requisitos da Arquitetura de Suporte

A arquitetura para implementar o modelo apresentado requer vários suportes dentro da linguagem SQL, bem como outras funcionalidades dentro do banco de dados, além de assistentes externos ao banco para criação e visualização das consultas. O suporte completo da arquitetura irá necessitar dos seguintes tipos de componentes, que estendem as funcionalidades e elementos do processo de *data warehousing*, como mostrado na Figura 11:

1 - **Tipos definidos pelo usuário** – As propriedades do modelo de banco de dados objeto-relacional viabilizam a construção dos tipos de dados estendidos dentro da linguagem SQL.

2 - **Funções definidas pelo usuário (***UDFs***)** – Cada tipo estará associado a um conjunto de funções para os diversos tipos de manipulação espacial que irão compor a extensão espacial do SQL.

3 - **Procedimentos armazenados no banco (***Stored Procedures***)** – Os procedimentos armazenados irão suportar a manipulação de conjunto de dados e serviços especiais.

4 - **Para uso interno** – Os procedimentos de uso interno irão cobrir todas as funcionalidades que não puderem ser feitas diretamente por funções definidas pelo usuário ou pelo suporte para as mesmas, como movimentação de dados entre tabelas.

5 - **Para uso por cliente de operação ou interface com o usuário** – Estes procedimentos irão constituir a maior parte do ETL, como importação de dados, limpeza etc.

6 - **Tabelas de metadados** – O sistema de *data warehousing* para TADs deve armazenar todas as informações relativas aos modelos e tipos de dados criados.

7 - **Módulos de carga dinâmica (*DLLs*)** – O modelo de *data warehouse* estendido é flexível o suficiente para permitir a incorporação de novas funcionalidades de manipulação. Estas funções poderão ser carregadas através de bibliotecas de carga dinâmica. O sistema irá possuir um conjunto de funções que irá carregar bibliotecas especificadas dinamicamente e definidas por interfaces genéricas.

8 - **Assistente de operação e construção de consultas** – Os produtos de *data warehousing* possuem, de um modo geral, ferramentas para construção de consultas, importação e transformação (Fernandes 2002,IBM Corporation 2002e etc.). Contudo, nem sempre estão preparados para construção e visualização de dados estendidos.

9 - **Módulo de Visualização** – É importante que os TADs possuam visualizadores específicos de acordo com sua semântica. Por exemplo, dados espaciais precisam ser visualizados como mapas.

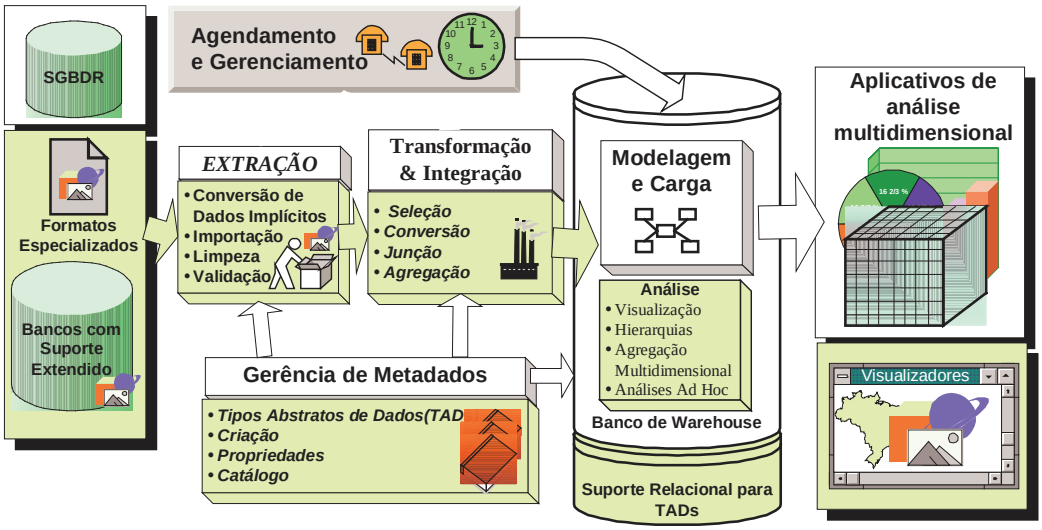

Figura 11 – Arquitetura de *data warehouse* estendido

4.3.2 – Elementos de Construção

Cada fase do processo de *data warehousing* vai ser agora montada em um sistema de camadas onde primeiro construímos os TADs de suporte, o suporte de metadados, as funções de suporte relacional e, por fim, as funcionalidades de *data warehousing* para TADs. Esta metodologia, assim como as ferramentas de classificação, definem o *Framework* **5-Alpha.**

O *framework* estabelece uma classificação para todos os elementos do modelo de *data warehousing* para TADs. Ele contém uma lista de códigos para guiar a classificação de seus elementos e um *template* para descrição do modelo. Este *template*, ou descritor, é escrito utilizando a linguagem XML (Light 1997,Maruyama, Tamura e Uramoto 1999), como similarmente feito por Vassiliadis at al. 2000. A codificação, denominada de **5-Alpha** (A, B, C, D, E) representa os cinco requisitos do modelo: Tipos (**A**)bstratos de Dados, Operadores (**B**)ásicos, (**C**)atálogo de metadados, Suporte relacional no Banco de (**D**)ados e Processo de *data warehouse* (**E**)stendido. Denominamos **5-Alpha-*X*** como a instância do *framework* para um conjunto de TADs ***X***. No Capítulo 7, iremos mostrar a instância **5-Alpha-*S***, que constitui a implementação deste *framework* para um conjunto de tipos de dados espaciais ***S***.

DESCRITOR XML

A Figura 12 mostra o diagrama geral do descritor em XML do *framework*. O descritor contém uma seção para cada um dos cinco requisitos do *fremework* (A, B, C, D e E). A raiz principal do modelo é o elemento **<Model>.** A especificação detalhada do descritor está no Apêndice B.

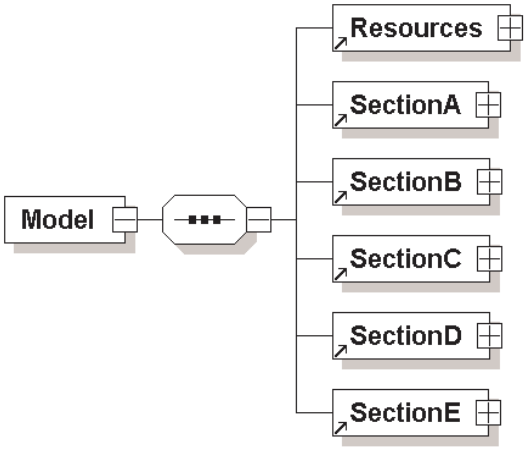

Figura 12 - Esquema geral do descritor do *framework*

```
<Model name="">
        <Resources>
                <BaseSchema SCHID="idvalue0" name="" />
                <XWExtFile ARQID="idvalue1" ExtDir="" ExtName="" />
        </Resources>
…
</Model>
```

Exemplo 1

Todos os tipos parâmetros de funções ou procedimentos estão mapeados através de cinco classes de elementos (Exemplo 2):

- (**<TADPar>**) para parâmetros de TADs
- (**<TPDPar>**) para parâmetros de TPDs
- (**<TPDParSet>**) para parâmetros de conjuntos de TADs
- (**<TADParSet**>) para parâmetros de conjuntos de TPDs
- (**<BOOLPar>**) para parâmetros de retorno booleano – Este tipo de parâmetro específico tem por finalidade permitir outros tipos de dados para indicar valores booleanos (ex., inteiros[0-1], caracteres[‘S’,‘N’]) e até mesmo TADs que possam representar valores lógicos, como tipos de dados nebulosos.

```
        <TADPar FakeNameAfter="" FakeNameBefore="" parDesc="" parName=""
                typeID="idvalue0" />
```

Exemplo 2

Os elementos de parâmetros ficam aninhados dentro dos elementos de definição de funções. Todos estes elementos possuem um atributo para indicar:

- O nome do parâmetro
- A descrição
- Um identificador único do tipo do parâmetro, correspondente aos tipos de dados definidos na seção A
- Rótulos de fantasia para serem exibidos nos controles da interface com o usuário. Estes rótulos opcionais estão destinados a serem colocados antes e depois do parâmetro. Por exemplo, uma função “Distância” com dois parâmetros, P1 e P1, poderia ser exibida das seguintes formas(Exemplo 3):

Rótulo Anterior	P1	Rótulo Posterior	Rótulo Anterior	P2	Rótulo Posterior
“Distância entre(“	A	“) E “	“(”	B	“)”
“(“	A	“) Distante “	“de (”	B	“)”

Exemplo 3

ELEMENTOS DO *FRAMEWORK*

A Tabela 9 apresenta um guia geral estruturado para criação de instâncias do *framework*. O conjunto completo de guias incorpora ainda as Tabela 10, Tabela 11, Tabela 12, Tabela 13, Tabela 14 e Tabela 15, já apresentadas previamente.

A lista a seguir mostra todas as funções já especificadas do modelo de *data warehouse* estendido para TADs dentro da codificação do *framework* **5-Alpha**, bem como a visão geral da estrutura do descritor em XML. A codificação da listagem utiliza o seguinte formato:

[<Seção>,<Categorias>,<Subcategoria 1>, <Subcategoria 2>...], onde **<Seção>** indica uma das cinco seções do *framework* e as categorias e subcategorias definem cada elemento necessário à construção do *framework*. As funções de projeção de elementos estariam na categoria **[D-2-...]**, por exemplo. Esta simbologia visa auxiliar à completude da construção do modelo, dado o grande conjunto de elementos e tarefas necessárias à sua construção.

O *framework* define então o processo de *data warehousing* estendido para um conjunto de TADs ***X*** como ***X***DW = {***X***W_{tipos}, ***X***$W_{transparentes}$, ***X***W_{MD}, ***X***W [$\mathbf{F}_{SuporteRelacional}$], ***X***W[$\mathbf{P}_{processamento}$]}, onde:

[A-0-0] - ***X***W_{tipos} – Todos os tipos de dados do conjunto X utilizados no sistema, apesar dos TADs serem o alvo principal desta seção. A identificação de quais serão os TADs eleitos para o processo de *data warehousing* está além de um procedimento fixo, contudo podemos indicar as seguintes orientações:

- Levantar os TADs que podem ser relacionados entre si estruturalmente, ou seja, hierarquicamente ou por relação todo-parte
- Levantar os TADs que podem ser relacionados entre si funcionalmente, como, por exemplo, identificar cidades (pontos) dentro de estados (polígonos).

- Levantar os TADs que podem ser agregados em relação a outra dimensão, como o tempo. Por exemplo, a soma de áreas de infestação de uma doença agregadas ao longo do tempo pode fornecer a sua direção de propagação.

A descrição dos tipos primitivos tem por finalidade identificá-los nos parâmetros das funções. O elemento (**SectionA**) inicializa esta seção no descritor, e a definição de TADs através do elemento (**XWTADType**) pode ser recursiva (Exemplo 4). Cada TAD possui um atributo de nome (**name**), que é o nome real no banco, e um identificador único do tipo (**typeID**).

[A-0-0] - ***X***$W_{tipo\ 1}$ – Um tipo de dado-pai dentro de uma hierarquia.

[A-1-0] - ***X***$W_{tipo\ 1.1}$ – Classe filha ou subclasse do ***X***$W_{tipo\ 1}$…

```
<SectionA>
        <XWTADType SCHID="idvalue0" name="" typeID="idvalue2">
                <XWTADType … />
        </XWTADType>
        <XWTPDType name="" typeID="idvalue4" />
</SectionA>
```

Exemplo 4

[B-0-0] - ***X***$W_{transparentes}$ – Esta seção define funções que vão oferecer operadores de comparação (Igual, Maior, Menor e Diferente) e conectivos lógicos (“E”, “OU” e “NÃO”). A Tabela 10 contém a lista de todos os operadores propostos no *framework*. A classificação de funções que implementam estes operadores deve seguir a seguinte metodologia:

- **Identificar operações de comparação entre TADs de mesmo tipo** – Para cada operação de comparação(<.= etc.), devemos verificar se existe uma semântica específica para os TADs a ser implementada.
- **Verificar se existem TPDs mapeando condições lógicas** – Em alguns casos podem existir funções retornando inteiros (0-1), ou outros tipos, para identificar valores de verdadeiro e falso.

Temos elementos do descritor para definir cada um dos grupos de operadores, como podemos ver no Exemplo 5, o operador de igualdade **<XWEqual>**.

```
<SectionB>
        <XWEqual BaseName="" SCHID="idvalue0" FUNCID="idvalue5" >
                <TADPar .../>
                <TADPar .../>
        </XWEqual>
</SectionB>
```

Exemplo 5

[C-0-0] - ***X***W_{MD} – Este item representa o suporte de metadados e está dividido em três seções que definem os catálogos e seus procedimentos de manipulação, além de uma lista de funções de manipulação de TADs dentro do modelo. Por exemplo, teremos catálogos indicando quais colunas de que tabelas possuem um certo tipo de dado estendido, e também funções para registro e remoção destas informações.

[C-1-0] - ***X***$W_{CATÁLOGOS}$ – Os produtos de suporte a TADs em geral já possuem catálogos de metadados próprios, mas eles devem conter um grupo básico de itens. Apresentamos um modelo de catálogo (Figura 13) contendo este grupo básico que pode ser usado para completar os elementos de catálogo não existentes no produto de suporte:

[C-2-0] - **Tipos Abstratos de Dados** – Este catálogo deve ter a lista de TADs, assim como as colunas e tabelas onde eles estão localizados.

[C-3-0] - **Modelos** – Este catálogo deve conter uma lista de nomes modelos que estejam relacionados aos catálogos de TADs, de módulos e de hierarquias. Cada um destes elementos deve poder pertencer a mais de um modelo, e cada modelo deve poder ter quaisquer números de elementos.

[C-4-0] - **Hierarquias** – O catálogo de hierarquias irá conter informações sobre o relacionamento hierárquico entre TADs, bem como a lista de cada atributo e de suas funções de agregação associadas.

[C-5-0] - **Módulos de processamento** – O catálogo de módulos vai conter a lista de cada módulo de importação ou pré-processamento do sistema. Ele conterá associação para uma tabela com uma lista de parâmetros para cada módulo. Podemos verificar pelo esquema da Figura 13 que as informações mínimas de cada módulo devem conter o nome externo do módulo no sistema operacional (“NOME EXTERNO”), assim como o nome interno do procedimento no banco (“NOME INTERNO PROC”). Além disso, temos que indicar que tipo de implementação externa foi usada (“TIPO IMPLEMENTACAO”), como, por

exemplo, C, PASCAL, SQL etc., bem como a categoria do módulo ("CATEGORIA MODULO")(ex., validação, importação etc.). O atributo "CRIADO" irá indicar se o módulo já foi carregado ou não, sendo utilizado para solicitar nova recarga.

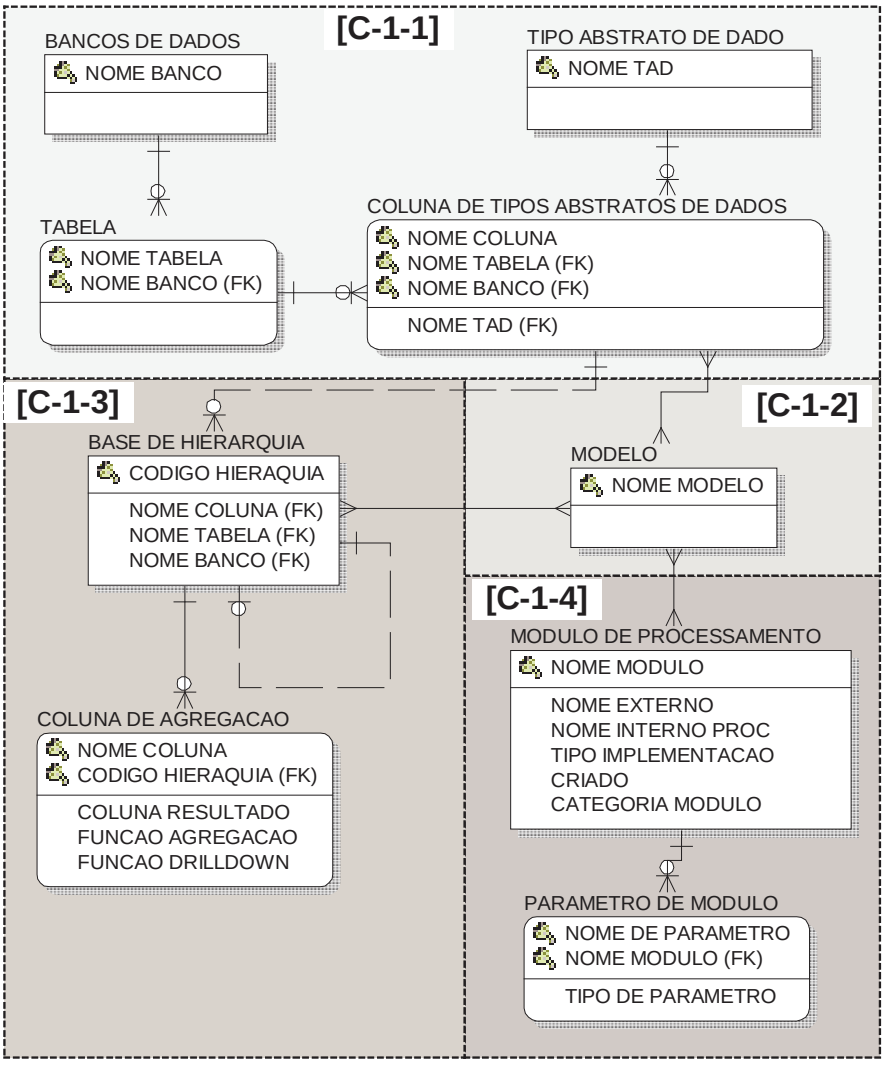

Figura 13 – Modelo de dados do catálogo de metadados

O procedimento de utilização do *framework*, neste ponto, requer que todos os catálogos já existentes nos produtos de suporte seja registrados nos itens acima utilizando o descritor XML. Estas informações serão utilizadas para geração automática da ferramenta de *data warehousing* mostrada no Capítulo 6**.** O Exemplo 6 mostra a especificação de catálogos para TADs (**XWTypeCatalog**), módulos de procedimentos (**XWModuleCatalog**), hierarquias (**XWHierarchyCatalog**) e modelos (**XWModelCatalog**).

```
<SectionC>
        <XWCatalogs>
                <XWTypeCatalog … />
                <XWModuleCatalog ... />
                <XWHierarchyCatalog   ...>
...
        </XWCatalogs>
...
</SectionC>
```

Exemplo 6

[C-6-0] - ***X***W$_{\text{Manipulação de Catálogos}}$ – Estes procedimentos servem para cadastrar dados, módulos e hierarquias nos catálogos de metadados. Os elementos do descritor (Exemplo 7) para estes procedimentos possuem um atributo para definir o nome no banco e outro para o código do esquema. Os parâmetros, já descritos no inicio desta seção, podem ter número indeterminado e são acrescentados como nós filhos.

[C-6-1] - **PCSD**$^{n}_{\text{Registro}}$(Parâmetros) – Procedimentos para registro de um TAD do catálogo de metadados. Os parâmetros devem ser o nome do TAD, a coluna e a tabela onde se encontra, e eventuais parâmetros adicionais.

[C-6-2] - **PCSD**$^{n}_{\text{Remoção}}$(Parâmetros) – Procedimentos para remoção de um TAD do catálogo de metadados.

[C-6-3] - **PCSM**$^{n}_{\text{Registro}}$(Parâmetros) – Procedimentos para registro de um módulo de processamento. A parametrização pode ser diversa, mas, de modo geral, deve incluir um nome no sistema operacional onde o módulo se encontra. Os atributos mínimos sugeridos para os módulos de processamento estão descritos no modelo de catálogo do item **[C-1-4]**.

[C-6-4] - **PCSM**$^{n}_{\text{Remoção}}$(Parâmetros) – Procedimentos para remoção dos módulos de processamento.

[C-6-5] - **PCSH**$^{n}_{\text{Registro}}$(Parâmetros) – Procedimentos para inclusão das informações de uma hierarquia no catálogo de metadados. Este registro deverá ser feito automaticamente quando da criação da hierarquia pelas funções de suporte de análise multidimensional (ver item E).

[C-6-6] - **PCSH**$^{n}_{\text{Remoção}}$(Parâmetros) – Procedimentos para remover as informações sobre uma hierarquia do catálogo de metadados.

```
<SectionC>
        <XWCatHandle>
                <PCD BaseName="" SCHID="idvalue0">
                        <TADPar … />
                        <TPDPar … />
                </PCD>
        </XWCatHandle>
</SectionC>
```

Exemplo 7

[C-7-0] - ***X***W$_{\text{Manipulação de Elementos}}$ – Este grupo compreende as funções de criação e manipulação de elementos de TADs, cuja seção é definida pelo elemento (**<XWDataHandle>**) do descritor (Exemplo 8). Elas retornam novos objetos de TADs ou propriedades dos mesmos, por este motivo estas funções são um subconjunto das funções de suporte à projeção descritas no item [D-X-X]. Contudo, esta seção do *framework* é mais apropriada devido a sua semântica.

[C-7-1] - **FCSC**$^{n}_{\text{Criação}}$(Parâmetros) – Subcategoria de funções que retorna uma nova instância de um TAD a partir de um valor de um elemento individual de uma *tupla*. Tomemos como exemplo a função de criação de pontos do **DB2 Spatial Extender**: *db2gse.ST_Point db2gse.ST_Point (Double X,Double Y, int SRID)*. Ela recebe duas coordenadas e mais um sistema de referência e retorna um TAD "ponto".

[C-7-2] - **FCMC**$^{n}_{\text{Criação}}$(Parâmetros) – Subcategoria de funções que retorna uma nova instância de um TAD a partir de um conjunto de elementos (várias *tuplas*) de TPDs. Poderíamos citar como exemplo uma função "Matriz CriaMatriz(PopCidades)" que retorna um objeto "Matriz" contendo todos os valores de população das cidades contidas na coluna "PopCidades".

[C-7-3] - **FCSA**$^{n}_{\text{Acesso}}$(***X***W$_{\text{tipo}}$) – Subcategoria de funções que retorna um atributo ou parte de um tipo de um TAD. A função "*Integer db2gse.ST_SRID(Geometria g)*" do **DB2 Spatial Extender** retorna o sistema de referencia de cada objeto "g".

[C-7-4] - **FCSP**$^{n}_{\text{Propriedade}}$(***X***W$_{\text{tipo}}$) – Subcategoria de funções que calcula e retorna uma propriedade de um TAD. As propriedades são características próprias do objeto de TAD, mas são obtidas a partir de um processamento ou cálculo, não sendo parte integrante do mesmo. Podemos citar, por exemplo, a

função do **DB2 Spatial Extender** "*Double db2gse.ST_Length(Geometria g)*", a qual calcula e retorna a o perímetro da geometria "g".

[C-7-5] - **FCMA**$^{n}_{Acesso}$(***X***W$_{coluna}$) – Esta categoria de função é de certa forma pouco comum de ser requisitada ou implementada em algum produto de suporte a TADs. Ela é similar as funções descritas no item **[C-3-3],** mas é capaz de retornar conjuntos de elementos, ao invés de elementos individuais. Um exemplo é a função "*ST_Pontos(Polígono p)*", que retorna um conjunto de elementos "ponto" para cada ponto que faça parte do polígono "p".

[C-7-6] - **FCMP**$^{n}_{Propriedade}$(***X***W$_{coluna}$) – Similar ao item anterior, este grupo de funções irá retornar um conjunto de propriedades de um objeto de TAD. Por exemplo, uma função que retornasse o conjunto de áreas de todos os polígonos internos de um polígono.

```
<SectionC>
        <XWDataHandle>
                <FCSC BaseName="" FUNCID="idvalue9" ISMETHOD="S"
                        RetType="idvalue0" SCHID="idvalue0" retMode="ONE">
                        <TPDPar …/>
                </FCSC>
        </XWDataHandle>
</SectionC>
```

Exemplo 8

[D-0-0] - ***X***W[**F**$_{Suporte\ Relacional}$] = O conjunto {**F**$_{Seleção}$, **F**$_{Projeção}$, **F**$_{Agregação}$} representa os conjuntos de funções de SQL estendidas para o suporte à álgebra relacional para manipular o conjunto de TADs ***X***. Esta seção é determinada pelo elemento (<**SectionD>**) do descritor que contém as seguintes sub-seções:

[D-1-0] - ***X***W[**F**$_{Seleção}$] – Este conjunto representa as funções de suporte à seleção, ou seja, de retorno de valores lógicos. A seção do descritor que especifica este grupo de funções é o elemento **<XWSelect>**, como podemos ver no Exemplo 9.

[D-1-1] - **FRS1**$^{n}_{Relação}$(***X***W$_{tipo}$) – Cada função **n** irá retornar verdadeiro ou falso de acordo com uma avaliação feita sobre cada elemento ***X***W$_{tipo}$ passado para a função. Por exemplo, a função do **DB2 Spatial Extender** "*Integer db2gse.ST_IsClosed(Curva c)*", a qual retorna verdadeiro ou falso se a curva "c" está fechada (ou seja, o primeiro e o último ponto são iguais) ou não.

[D-1-2] - **FRS2**$^{n}_{Relação}$($\boldsymbol{X}W_{tipo}^{1}$, $\boldsymbol{X}W_{tipo}^{2}$, [$DW_{parâmetro}$]) – Cada função **n** irá retornar verdadeiro ou falso de acordo com o relacionamento entre os dois elementos $\boldsymbol{X}W_{tipo}^{1}$ e $\boldsymbol{X}W_{tipo}^{2}$. Por exemplo, a função "*Integer ST_Contains*(*Geometria g1, Geometria g2*) retorna verdadeiro (1) se a geometria g1 está contida em g2. Possibilitamos, por conveniência, que esta categoria de funções possa receber algum parâmetro regulador, como obter verdadeiro se a geometria "g1" está a uma distância "d"(parâmetro) de "g2".

[D-1-3] - **FRS3**$^{n}_{Relação}$($\boldsymbol{X}W_{tipo}$, $DW_{parâmetro}$) – Cada função **n** irá retornar verdadeiro ou falso de acordo com uma avaliação feita sobre cada elemento $\boldsymbol{X}W_{tipo}$ e regulado pelo parâmetro $DW_{parâmetro}$ passados para a função. Por exemplo, uma função "*AreaMinima(Geometria g, Area a)*" iria retornar verdadeiro ou falso se a geometria g tem a área mínima determinada por "a".

[D-1-4] - **FRM1**$^{n}_{Relação}$($\boldsymbol{X}W_{coluna}$) – Cada função **n** irá retornar verdadeiro ou falso de acordo com o relacionamento entre todos os elementos dentro da coluna $\boldsymbol{X}W_{coluna}$. Por exemplo, a função "*SmallImage(Imagem I)*" iria receber o conjunto de TADs "*Imagem*" e retornaria a imagem de menor tamanho.

[D-1-5] - **FRM2**$^{n}_{Relação}$ ($\boldsymbol{X}W_{coluna}^{1}$, ($\boldsymbol{X}W_{coluna}^{2}$ | $\boldsymbol{X}W_{tipo}$), [$DW_{parâmetro}$]) – Cada função **n** irá retornar verdadeiro ou falso de acordo com o relacionamento entre todos os elementos da coluna $\boldsymbol{X}W_{coluna}^{1}$ e $\boldsymbol{X}W_{coluna}^{2}$ ou $\boldsymbol{X}W_{coluna}^{1}$ e um elemento individual de *tupla* $\boldsymbol{X}W_{tipo}$. De fato, esta categoria se refere a todas as funções de relacionamento entre mais de um TAD onde pelo menos um dos parâmetros é um conjunto. O exemplo mais comum seria uma função "*MaisSimilar(Pessoa p1,Pessoa p2)*" que retornaria verdadeiro para a pessoa dentro do conjunto "p1" mais similar a todas as pessoas do conjunto "p2". O parâmetro extra tem a mesma finalidade do descrito no item **[D-1-2]**.

[D-1-6] - **FRM3**$^{n}_{Relação}$(($\boldsymbol{X}W_{coluna}$ | $\boldsymbol{X}W_{tipo}$),($DW_{parâmetro}$|DW_{ParSet})) – Cada função **n** irá retornar verdadeiro ou falso de acordo com o relacionamento entre todos os elementos da coluna $\boldsymbol{X}W_{coluna}$ e um único elemento ou conjunto de TPD. Esta categoria de funções relaciona um TAD com um TPD considerando que pelo menos um deles será em forma de conjunto. Por exemplo, a função "*ST_ExisteIdade(Cidade g, ConjIdades ci)*" retorna verdadeiro para cada

cidade “g” que contenha pelo menos um habitante cuja idade esteja no conjunto “ci”.

```
<SectionD>
        <XWSelect>
                <FRS1 BaseName="" FUNCID="idvalue13" FalseValue="" ISMETHOD="S"
                        RetType="idvalue0" SCHID="idvalue0" TrueValue="">
                        <TADPar …./>
                </FRS1>
        </XWSelect>
</SectionD>
```

Exemplo 9

[D-2-0] - $\boldsymbol{X}$W[$\mathbf{F}_{\text{Projeção}}$] – Estas funções retornam elementos de TADs, TPDs ou conjuntos destes. As categorias são baseadas na classificação mostrada na seção 2.2.2 do Capítulo 2 com um acréscimo de uma análise nos tipos de parâmetros e retorno, como será explicado na seção 5.5.1 do Capítulo 5. A seção de funções de projeção do descritor usa o elemento **<XWProject>** (Exemplo 10), e os nomes dos elementos no descritor correspondem ao nomes das funções utilizadas no *framework*, acrescentando, eventualmente, algum índice ao final quando tivermos variação devido a semântica da função, como os elementos **<FPSC1>** e **<FPSC2>**.

[D-2-1] - $\mathbf{FPST}^{n}_{\text{Transformação}}(\boldsymbol{X}\text{W}_{\text{tipo}},[\text{DW}_{\text{parâmetro}}])$ – Estas funções recebem um elemento de TAD e, de acordo com a regulagem de um parâmetro ou não, retornam um outro elemento como o mesmo TAD, mas com valores modificados. A função *ST_Geometry ST_ConvexHull(ST_Geometry g)*, por exemplo, retorna um polígono convexo a partir de uma geometria qualquer. Ela recebe e retorna uma geometria, apenas alterando a posição interna dos seus pontos.

[D-2-2] - $\mathbf{FPSC}^{n}_{\text{Conversão}}(\boldsymbol{X}\text{W}_{\text{tipo}}{}^{1},[\ \boldsymbol{X}\text{W}_{\text{tipo}}{}^{2}|\ \text{DW}_{\text{parameter}}])$ – Funções que recebem como parâmetro valores individuais de cada *tupla* e retornam um tipo de dado diferente. O exemplo mais comum de conversão é utilizado em processos de armazenamento em funções que recebem um TAD e retornam um TPD do tipo texto ou binário.

[D-2-3] - $\mathbf{FPSX}^{n}_{\text{Combinacao}}(\boldsymbol{X}\text{W}_{\text{tipo}}{}^{1},\boldsymbol{X}\text{W}_{\text{tipo}}{}^{2},[\text{DW}_{\text{parâmetro}}])$ – Funções que recebem como parâmetros valores individuais de cada *tupla* e retornam um valor modificado combinado dos dois parâmetros, podendo ou não alterar seu tipo.

Por exemplo, a função do **DB2 Spatial Extender**, "*ST_Geometry ST_Union(Geometria g1, Geometria g2)*" vai retornar uma geometria única combinando todos os pontos da geometria g1 e g2.

[D-2-4] - $\mathbf{FPSD}^{n}_{DrillDown}(\boldsymbol{X}W_{tipo}\ [,DW_{parâmetro}])$ – Funções que recebem como parâmetros valores individuais de TADs de cada *tupla* e retornam um conjunto de elementos detalhados. Consideremos, por exemplo, uma função que receba como parâmetro uma imagem e retorne um conjunto de imagens contendo cada cor da imagem original separadamente.

[D-2-5] - $\mathbf{FPMT}^{n}_{Transformação}(\boldsymbol{X}W_{coluna}[,DW_{parâmetro}])$ – Funções que recebem como parâmetro valores de elementos de TADs e retornam um conjunto de elementos com valor modificado para cada um, sem alterar seu tipo. A diferença deste tipo de função é que ela pode receber como parâmetro conjuntos de TPDs. Um exemplo seria uma função que recebesse um objeto do tipo geométrico e um conjunto de valores de escalas, e retornasse um conjunto de objetos do mesmo tipo, mas com tamanho modificado para cada valor de escala fornecido.

[D-2-6] - $\mathbf{FPMC}^{n}_{Conversão}(\boldsymbol{X}W_{coluna}[,DW_{parâmetro}])$ – Funções que recebem como parâmetro valores de elementos de TADs e retornam um TAD diferente. A diferença deste tipo de função é que ela pode receber como parâmetro conjuntos de TPDs. Por exemplo, uma função que recebe um objeto do tipo geométrico "ponto" e um conjunto de valores de raios, e retorna um conjunto de objetos do tipo "círculo" para cada raio fornecido.

[D-2-7] - $\mathbf{FPMD}^{n}_{DrillDown}(\boldsymbol{X}W_{coluna}\ [,DW_{parâmetro}])$ – Funções que recebem como parâmetro valores de elementos de TADs e retornam um conjunto de elementos detalhados para cada elemento do conjunto de entrada. A diferença deste tipo de função é que ela pode receber conjuntos de TPDs. Um exemplo seria uma função que recebesse como parâmetro uma imagem e um conjunto de índices de cores, e retornasse um conjunto de imagens, cada uma contendo somente os índices de cores fornecido.

```
<SectionD>  …
        <XWProject>
                <FPST1 BaseName="" FUNCID="idvalue22" ISMETHOD="S"
                        RetType="idvalue0" SCHID="idvalue0" retMode="SET">
                        <TADPar … />
                        <TPDPar ... />
                </FPST1>
        </XWProject>
</SectionD>
```

Exemplo 10

[D-3-0] - $\boldsymbol{X}$W[$\mathbf{F}_{\text{Agregação}}$] –Estas funções serão suporte para todas as operações de agregação feitas no modelo utilizando conjuntos de TADs. Estas são funções que recebem como parâmetro, ou retornam, conjuntos de elementos e não elementos individuais, *tupla* por *tupla*. O elemento do descritor que define a seção de funções de agregação é **<XWAgregate>**(Exemplo 11).

[D-3-1] - $\mathbf{FA1}^{n}_{\text{Agregação}}(\boldsymbol{X}W_{\text{coluna}}{}^{1}, [\boldsymbol{X}W_{\text{tipo}} \mid \boldsymbol{X}W_{\text{coluna}}{}^{2} \mid DW_{\text{parâmetro}}])$ – Funções que realizam agregação de um conjunto de TADs, convertendo o resultado para um TPD. Por exemplo, a função "*Integer SomaNotas(TrechoMusical* M_{set}*, Integer TipoNota)*" receberia o conjunto de TADs de uma coluna do tipo "TrechoMusical" e retornaria a soma total das notas musicais do tipo "TipoNota" em cada um dos trechos.

[D-3-2] - $\mathbf{FA2}^{n}_{\text{Agregação}}(\boldsymbol{X}W_{\text{coluna}}{}^{1}, [\boldsymbol{X}W_{\text{tipo}} \mid \boldsymbol{X}W_{\text{coluna}}{}^{2} \mid DW_{\text{parâmetro}}])$ – Funções que realizam agregação de um conjunto de TADs, convertendo o resultado para um conjunto de TPDs. Seguindo o exemplo do item anterior, este tipo de função retornaria um conjunto de quantidades de notas de um mesmo tipo contidas no conjunto de trechos musicais de entrada.

```
<SectionD>
        …
        <XWAgregate>
                <FA1Section>
                        <FA1 BaseName="" FUNCID="idvalue30" ISMETHOD="S"
                                RetType="idvalue0" SCHID="idvalue0" retMode="SET">
                                <TADParSet …/>
                                <TADPar ... />
                        </FA1>
                </FA1Section>
```

Exemplo 11

[D-3-3] - $\mathbf{FA11}^{n}_{\text{Agregação}}(\boldsymbol{X}W_{\text{coluna}},[DW_{\text{parâmetro}}])$ – Funções que realizam agregação de um conjunto de dados, produzindo uma conversão ou

transformação que resulta em um único elemento de TADs. Por exemplo, a função "Imagem Sobreponha(Imagem I_{set})" retornaria uma outra imagem com a sobreposição de todas imagens dentro do conjunto I_{set}.

[D-3-4] - **FA12**$^{n}_{Agregação}$(***X***W_{coluna},[$DW_{parâmetro}$]) – Funções que realizam agregação de um conjunto de dados, produzindo uma conversão ou transformação que resulta em um conjunto de instâncias de TADs. Um exemplo deste tipo de função seria o mapeamento de um conjunto de TADs em outro, tal como na função "*$Geometry_{set}$ CalculaCentroides(*$Geometria_{set}$*, DistanciaMinima)*", que retorna o conjunto centróide dos pontos que estejam dentro da "*DistanciaMinima*" fornecida.

[D-3-5] - **FA21**$^{n}_{Agregação}$(***X***$W_{coluna}{}^{1}$,[***X***W_{tipo} | ***X***$W_{coluna}{}^{2}$],[$DW_{parâmetro}$]) – Funções que realizam agregação e combinação de dois conjuntos de TADs, produzindo uma conversão ou transformação que resulta em um único elemento de TAD. O exemplo mais comum, e geral, deste tipo de função está no cálculo do elemento mais similar dentro do conjunto fornecido na ***X***$W_{coluna}{}^{1}$ de todos os elementos da ***X***$W_{coluna}{}^{2}$.

[D-3-6] - **FA22**$^{n}_{Agregação}$(***X***$W_{coluna}{}^{1}$,[***X***W_{tipo} | ***X***$W_{coluna}{}^{2}$],[$DW_{parâmetro}$]) – Funções que realizam agregação e combinação de dois conjuntos de TADs, produzindo uma conversão ou transformação que resulta em um conjunto de TADs. Podemos estender o exemplo anterior de forma a atender este tipo de função. Neste caso, o exemplo seria uma função que retornaria o conjunto dos elementos mais próximos dentro de um valor máximo de distância fornecido.

[E-0-0] - ***X***W[$\mathbf{P}_{processamento}$] = {$\mathbf{PE}_{Extração}$, $\mathbf{PT}_{Transformação}$, $\mathbf{PA}_{Análise}$} – Representam um conjunto de funções que vão incorporar as funções de suporte relacional dos itens D e C em funções de *data warehousing* de alto nível.

[E-1-0] - ***X***W[$\mathbf{PE}_{Extração}$] – Este conjunto de funções é responsável por obter todos os dados externos ao banco de dados (ver seção **2.2.1**). Dividimos este grupo em categorias de funções de importação e funções de pré-processamento. A categoria de funções de importação é a responsável por trazer os dados para dentro do banco de *data warehouse.* A categoria de pré-processamento é responsável por preparar os dados para serem utilizados. Os procedimentos de extração são registrados dentro do elemento **<XWExtract>** do descritor.

[E-1-1] - **Importação** – Os procedimentos de importação, registrados na seção correspondente ao elemento **<XWImport>**(Exemplo 12). Eles podem conter qualquer número de parâmetros de TPDs. Além disso, todos eles possuem duas versões, uma contendo as informações de arquivos de entrada e outra contendo informações de uma tabela de um banco de dados externo.

[E-1-1-1] - $\mathbf{PEI}^{n}_{\text{Importação}}([\boldsymbol{ArqExt} \mid \boldsymbol{TabList}^{B}_{\text{fonte}}], \boldsymbol{TabList}^{A}_{\text{destino}}[,\text{DW}_{\text{parâmetro}}])$ – Cada função de importação **n** recebe um nome de um arquivo "***ArquivoExt***" contendo os dados de importação e transfere para a lista de tabelas ***TabList***. A importação também pode ser feita a partir de uma tabela de um outro banco de dados. Esta categoria de funções será primariamente responsável por tratar os diversos formatos particulares para cada tipo de dado estendido. Qualquer parâmetro necessário à importação é representado por $\text{DW}_{\text{parâmetro}}$.

[E-1-1-2] - $\mathbf{PEA}^{n}_{\text{Agregação}}([\boldsymbol{ArqExt} \mid \boldsymbol{TabList}^{B}_{\text{fonte}}], \boldsymbol{TabList}^{A}_{\text{destino}}[,\text{DW}_{\text{parâmetro}}])$ – Cada função de importação **n** recebe um nome de arquivo ou tabela de outro banco de dados e realiza uma agregação prévia dos dados antes de inserí-los nas tabelas de destino. Esta agregação também é feita sobre os TPDs, além das agregações especializadas para os tipos de dados extendidos

[E-1-1-3] - $\mathbf{PES}^{n}_{\text{Sequencializador}}([\boldsymbol{ArqExt} \mid \boldsymbol{TabList}^{B}_{\text{fonte}}], \boldsymbol{TabList}^{A}_{\text{destino}}[,\text{DW}_{\text{parâm.}}])$ – Esta subcategoria de funções de importação organiza a seqüência de entrada dos dados nas tabelas do banco de *data warehousing*. Estas funções cuidam para que os dados provenientes de múltiplos arquivos e tabelas sejam inseridos na ordem correta.

```
<SectionE>
        <XWExtract>
                <XWImport>
                        <PEII1 BaseName="" PEID="idvalue38" SCHID="idvalue0">
                                <XWFile ARQID="idvalue0" Seq="" />
                                <TgtTable TargetSchema="" TargetTable="" />
                                    <TPDPar …/>
                        </PEII1>
                </XWImport>
```

Exemplo 12

[E-1-2] - **Pré-Processamento** – Todos os dados importados para o banco de dados de *data warehouse* podem requerer serem pré-processados de alguma forma. Digamos, por exemplo, que seja necessário substituir diversos

acrônimos como Av., Avenida etc., por apenas um. O pré-processamento se caracteriza por não alterar a estrutura ou o modelo dos dados, mas apenas seu conteúdo (ver seção **2.2.1**). O descritor da seção de funções de pré-processamento utiliza o elemento **<XWPreprocess>**. As suas funções apenas visam operações dentro da própria base de *data warehouse*, portanto elas seguem o mesmo formato das funções de importação de tabelas de bancos externos. A única diferença é que as informações de usuário e senha não são necessárias.

[E-1-2-1] - $\mathbf{PEP}^{n}_{\text{Preprocessamento}}(\boldsymbol{TabList}^{A}_{\text{fonte}}, \boldsymbol{TabList}^{A}_{\text{destino}}[,\text{DW}_{\text{parâmetro}}])$ – Representa as **n** funções genéricas de pré-processamento.

[E-1-2-2] - $\mathbf{PEL}^{n}_{\text{Limpeza}}(\boldsymbol{TabList}^{A}_{\text{fonte}}, \boldsymbol{TabList}^{A}_{\text{destino}}[,\text{DW}_{\text{parâmetro}}])$ – Esta subcategoria de funções cuida de homogeneizar o conteúdo dos dados de forma que haja apenas uma representação para o mesmo valor. O sexo, por exemplo teria que ser sempre representado por "Masculino" ou "M", e não ambos.

[E-1-2-3] - $\mathbf{PEV}^{n}_{\text{Validação}}(\boldsymbol{TabList}^{A}_{\text{fonte}}, \boldsymbol{TabList}^{A}_{\text{destino}}[,\text{DW}_{\text{parâmetro}}])$ – Subcategoria que representa as **n** funções que garantem que um dado seja válido. Podemos verificar, por exemplo, se uma faixa de valores de idade não é superior a 200, ou então garantir que não existem endereços com números de casa fora de uma escala aceitável.

[E-2-0] - $\boldsymbol{X}\text{W}[\mathbf{PT}_{\text{Transformação}}]$ – Estes procedimentos utilizam o conjunto de funções de suporte relacional para produzir novas tabelas com diferentes estruturas e modelos. O tipo de transformação mais comum é carregar as tabelas de junção para preencher as tabelas de *Medidas* do modelo estrela (seção 2.2.3). Temos duas categorias de funções genéricas e mais oito subcategorias baseadas em definições da literatura. O modelo proposto permitiria que todas estas subcategorias de funções pudessem ser representadas genericamente. Contudo, nem todos os sistemas de bancos de dados são tecnologicamente adequados para fornecer os elementos necessários para construir todas as consultas SQL que serviriam de suporte para as funções genéricas de transformação. Atualmente, a construção de *UDFs* de agregação não está disponível em todos os bancos de dados, mas com o tempo esta capacidade deve estar disponível na maioria deles.

Esta seção (**<XWTranform>**) contém um elemento para cada categoria de transformação. O primeiro elemento (**<XWGeneral>**) define funções genéricas para transformação (Exemplo 13). Estas funções podem conter qualquer número de tabelas de entrada e de destino, assim como parâmetros. Os outros atributos do procedimento são iguais aos de extração, com exceção do mapeamento de colunas e de SQL configurável.

[E-2-1] - $\mathbf{PTG1}^{n}_{\text{Transformação}}(\boldsymbol{TabList}_{\text{fonte}}, \boldsymbol{TabList}_{\text{destino}}[,\text{DW}_{\text{Lista de Parâmetros}}])$ – Todas as **n** possíveis funções especializadas de transformação criadas, ou existentes no suporte, que sejam especializadas para alguma finalidade.

[E-2-2] - $\mathbf{PTG2}^{n}_{\text{Transformação}}(\boldsymbol{TabList}_{\text{fonte}}, \boldsymbol{TabList}_{\text{destino}}, \text{SQL}_{\text{list}}[,\text{DW}_{\text{Lista de Parâms}}])$ – Todas as **n** possíveis funções de transformação vão receber a lista de tabelas de fontes de dados, aplicar as declarações em SQL e produzir as tabelas de medidas. Devemos ter uma declaração em SQL, mais um possível parâmetro para cada tabela de destino. $\mathbf{P}^{n}_{\text{Transformação}}$ é uma forma geral para uma função de transformação, porém, cada declaração SQL diferente irá requerer uma nova função

```
<SectionE>
        <XWTranform>
                <XWGeneral>
                        <PTG BaseName="" SCHID="idvalue0">
                                <SrcTable   SourceSchema=""  SourceTable="" />
                                <TgtTable TargetSchema="" TargetTable="" />
                                <TPDPar />
                                <ColMapping SourceCol="" TargetCol="" />
                                <SQLMapping>SQLMapping</SQLMapping>
                        </PTG>
                </XWGeneral>
…
```

Exemplo 13

A lista abaixo mostra quais as categorias específicas de cada um dos procedimentos de transformação usando como base as funções de suporte relacional. Podemos encontrar exemplos para todas as categorias de funções abaixo junto das descrições das funções de suporte relacional nos itens **[C-X-X]** e **[D-X-X].**

[E-2-3] - **Seleção** – Os procedimentos específicos para transformação por seleção vão seguir a padronização dos procedimentos gerais de transformação, com a exceção de permitir até duas tabelas de entrada (para possibilitar a

realização de junções) e um identificador da função de suporte relacional que será usado para realizar a seleção.

[E-2-3-1] - **PTSR**$^{n}_{\text{Seleção}}$(***TabList***$_{\text{fonte}}$, ***TabList***$_{\text{destino}}$, **FRS_**^{k} [,DW$_{\text{Parâmetros}}$]) – Procedimentos que realizam a seleção de registros de tabelas fontes aplicando uma avaliação a cada registro selecionado através da função de relação **FRS_**$^{k}_{\text{Relação}}$.

[E-2-3-2] - **PTMR**$^{n}_{\text{Seleção}}$(***TabList***$_{\text{fonte}}$, ***TabList***$_{\text{destino}}$, **FRM_**^{k} [,DW$_{\text{Parâmetros}}$]) – Procedimentos que realizam a seleção de registros de tabelas fontes aplicando a avaliação em todo o conjunto de registros de uma vez através da função de relação **FRM_**$^{k}_{\text{Relação}}$

[E-2-4] - **Separação** – Estes procedimentos produzirão conjuntos de dados extraindo parte de cada elemento de TAD para produzir o dado de saída. Devemos ressaltar que as funções de suporte relacional retornam apenas um elemento da separação. Dentro da função de transformação, as funções de suporte de projeção têm de ser chamadas duas vezes para cada elemento para produzir as duas ou mais partes separadas do TAD.

[E-2-4-1] - **PTSS1**$^{n}_{\text{Separação}}$(***TabList***$_{\text{fonte}}$, ***TabList***$_{\text{destino}}$, **FPSC**^{k} [,DW$_{\text{Parâms}}$]) – Separa um TAD em duas ou mais partes aplicando a função de conversão uma vez para cada elemento da tabela fonte. Utiliza funções de conversão, de forma que as partes obtidas a partir do TAD fonte podem não ser exatamente uma parte dele, mas algum subprocessamento sobre o mesmo.

[E-2-4-2] - **PTSS2**$^{n}_{\text{Separação}}$(***TabList***$_{\text{fonte}}$, ***TabList***$_{\text{destino}}$, **FCSA**^{k} [,DW$_{\text{Parâmetros}}$]) – Esta subcategoria de Procedimentos de separação vai utilizar funções de acesso ao TAD para obter partes do mesmo. Desta forma, a separação é obrigatoriamente feita dividindo-se o TAD em suas reais partes

[E-2-4-3] - **PTMS1**$^{n}_{\text{Separação}}$(***TabList***$_{\text{fonte}}$, ***TabList***$_{\text{destino}}$, **FPMC**^{k} [,DW$_{\text{Parâms}}$]) – Funciona da mesma forma que o item acima, contudo, aplica a função de separação diretamente em todo o conjunto de dados fonte e produz todos os elementos separados de uma vez.

[E-2-4-4] - **PTMS2**$^{n}_{\text{Separação}}$(***TbList***$_{\text{fonte}}$,***TbList***$_{\text{destino}}$, **FA12**^{k} [,DW$_{\text{Parâmetros}}$]) – Similar ao item anterior, mas permite que todo um conjunto de elementos seja processado para produzir o conjunto final de saída.

[E-2-5] - **Concatenação** – O formato dos procedimentos de concatenação é igual ao de seleção: recebe duas tabelas de entrada e uma de destino. Os outros parâmetros são iguais.

[E-2-5-1] - **PTSU**$^{n}_{\text{Concatenação}}$(***TbList***$_{\text{fonte}}$, ***TbList***$_{\text{destino}}$, **FPSX**^{k} [,DW$_{\text{Parâmetros}}$]) – Procedimentos que realizam a concatenação de TADs em outro TAD combinado, aplicando a função de suporte de combinação **FS**$^{k}_{\text{Combinacao}}$ em um elemento de cada vez

[E-2-5-2] - **PTMU**$^{n}_{\text{Concatenação}}$(***TabList***$_{\text{fonte}}$,***TabList***$_{\text{destino}}$,**FA21**^{k} [,DW$_{\text{Parms}}$]) – Procedimentos que realizam a concatenação da mesma forma que a anterior, mas aplicam as funções de suporte de combinação em todo o conjunto de dados de uma vez.

[E-2-6] - **Sumarização** – Todos os procedimentos abaixo irão produzir a agregação do conjunto de TADs de um mesmo atributo. Por isso, todas as funções de suporte serão de agregação, tendo que ser aplicadas a um conjunto de dados e não a dados individuais. Os procedimentos de sumarização recebem somente uma tabela de entrada e uma de destino. Seu objetivo vai ser a agregação através das funções de suporte relacional indicadas.

[E-2-6-1] - **PTSA1**$^{n}_{\text{Sumarização}}$(***TbList***$_{\text{fonte}}$,***TbList***$_{\text{destino}}$,**FA11**^{k} [,DW$_{\text{Parâms.}}$])

[E-2-6-2] - **PTMA1**$^{n}_{\text{Sumarização}}$(***TbList***$_{\text{fonte}}$,***TbList***$_{\text{destino}}$,**FA12**^{k}[,DW$_{\text{Parâms.}}$])

[E-2-6-3] - **PTSA2**$^{n}_{\text{Sumarização}}$(***TbList***$_{\text{fonte}}$,***TbList***$_{\text{destino}}$,**FA21**^{k}[,DW$_{\text{Parâms.}}$])

[E-2-6-4] - **PTMA2**$^{n}_{\text{Sumarização}}$(***TbList***$_{\text{fonte}}$,***TbList***$_{\text{destino}}$,**FA22**^{k}[,DW$_{\text{Parâms.}}$])

[E-2-7] - **Transformação** – Este conjunto de procedimentos está destinado a funcionalidades genéricas de transformação que não possam se aplicar a uma categoria específica . Ele segue o mesmo padrão do descritor de procedimentos de sumarização e utiliza os identificadores de função especificados no *framework*.

[E-2-7-1] - $\mathbf{PTST1}^{n}_{\text{Transformaçao}}(\boldsymbol{TbList}_{\text{fonte}}, \boldsymbol{TbList}_{\text{destino}}, \mathbf{FPST}^{k}[,\text{DW}_{\text{Parms.}}])$ – Subcategoria que aplica funções de suporte em um registro de cada vez

[E-2-7-2] - $\mathbf{PTMT1}^{n}_{\text{Transformaçao}}(\boldsymbol{TbList}_{\text{fonte}}, \boldsymbol{TbList}_{\text{destino}}, \mathbf{FPMT}^{k}[,\text{DW}_{\text{Par}}])$ – Subcategoria que aplica as funções de suporte a todo um conjunto de dados de uma vez

[E-2-7-3] - $\mathbf{PTMT2}^{n}_{\text{Transformaçao}}(\boldsymbol{TbList}_{\text{fonte}}, \boldsymbol{TbList}_{\text{destino}}, \mathbf{FA12}^{k}[,\text{DW}_{\text{Parms}}])$ – Similar ao item anterior, mas é capaz de realizar sub-agregações dos dados, além do processo de transformação em si

[E-2-8] - **Enriquecimento** – Segue o padrão anterior. Estas funções realizam conversões que aumentam a complexidade das estruturas dos TADs.

[E-2-8-1] - $\mathbf{PTSE1}^{n}_{\text{Enriquecimento}}(\boldsymbol{TbList}_{\text{fonte}}, \boldsymbol{TabList}_{\text{destino}}, \mathbf{FPSC}^{k}\ [,\text{DW}_{\text{Par}}])$ – Realizam enriquecimento através de funções de conversão aplicadas a cada elemento individual do conjunto de dados

[E-2-8-2] - $\mathbf{PTSE2}^{n}_{\text{Enriquecimento}}(\boldsymbol{TbList}_{\text{fonte}}, \boldsymbol{TbList}_{\text{destino}}, \mathbf{FCSC}^{n}\ [,\text{DW}_{\text{Parâms}}])$ – Procedimentos que produzem um tipo particular de enriquecimento, onde TPDs são convertidos em TADs. Em outras palavras, eles criam TADs

[E-2-8-3] - $\mathbf{PTME1}^{n}_{\text{Enriquecimento}}(\boldsymbol{TbList}_{\text{fonte}}, \boldsymbol{TbList}_{\text{destino}}, \mathbf{FA12}^{k}\ [,\text{DW}_{\text{Parms}}])$ – Similar ao item anterior, mas lidam com todos os elementos de uma vez, podendo realizar agregações

[E-2-8-4] - $\mathbf{PTME2}^{n}_{\text{Enriquecimento}}(\boldsymbol{TabList}_{\text{fonte}}, \boldsymbol{TabList}_{\text{destino}}, \mathbf{FCMC}^{n}\ [,\text{DW}_{\text{Parm}}])$ – Similar ao item anterior, mas obtém os dados para criação de todos de uma vez.

[E-2-8-5] - $\mathbf{PTME3}^{n}_{\text{Enriquecimento}}(\boldsymbol{TbList}_{\text{fonte}},\ \boldsymbol{TbList}_{\text{destino}},\ \mathbf{FPMC}^{k}\ [,\text{DW}_{\text{Parâms}}])$ – Similar ao item anterior, mas sem possibilidade de sub-agregações

[E-2-9] - **Conversão** – Conjunto de funções genéricas de transformação para quando for necessária uma conversão do TAD original para outro TAD.

[E-2-9-1] - $\mathbf{PTSC1}^{n}_{\text{Conversão}}(\boldsymbol{TabList}_{\text{fonte}},\ \boldsymbol{TabList}_{\text{destino}},\ \mathbf{FPSC}^{k}\ [,\text{DW}_{\text{Parâms}}])$ – Realizam a conversão aplicando funções de suporte relacional de conversão em um elemento de cada vez

[E-2-9-2] - $\mathbf{PTSC2}^{n}_{\text{Conversão}}(\boldsymbol{TabList}_{\text{fonte}},\ \boldsymbol{TabList}_{\text{destino}},\ \mathbf{FA1}^{n}\ [,\text{DW}_{\text{Parâms}}])$ – Realizam a conversão aplicando, de uma vez, o conjunto de funções de

conversão em um grupo de elementos. Pode também realizar agregações e produzir um TAD como resultado

[E-2-9-3] - **PTMC1**$^{n}_{\text{Conversão}}$ (***TabList***$_{\text{fonte}}$,***TabList***$_{\text{destino}}$,**FPMC**$^{k}_{\text{Conversão}}$ [,DW$_{\text{Pars}}$]) – Realizam a conversão aplicando, de uma vez, funções de suporte relacional de conversão em um grupo de elementos .

[E-2-9-4] - **PTMC2**$^{n}_{\text{Conversão}}$ (***TabList***$_{\text{fonte}}$,***TabList***$_{\text{destino}}$,**FA2**^{n} [,DW$_{\text{Parâms}}$]) – Procedimentos similares ao item anterior podendo produzir um conjunto de TADs a partir da agregação de outro conjunto de TADs

[E-2-10] - ***DrillDown*** – Estas funções realizam *drilldown* através das funções de suporte relacional correspondentes.

[E-2-10-1] - **PTSD**$^{n}_{DrillDown}$(***TabList***$_{\text{fonte}}$, ***TabList***$_{\text{destino}}$, **FPSD**^{k} [,DW$_{\text{Parâms}}$]) – Utilizam funções que acessam cada elemento individualmente.

[E-2-10-2] - **PTMD**$^{n}_{DrillDown}$ (***TabList***$_{\text{fonte}}$, ***TabList***$_{\text{destino}}$, **FPMD**^{n} [,DW$_{\text{Parâms}}$]) – Utilizam funções que acessam, de uma vez, um conjunto de elementos.

[E-3-0] - ***X***W[**PA**$_{\text{Análise}}$] – As funções de análise são a fundação para as operações de OLAP estendidas. Nós às dividimos em dois grupos: um para uso restrito em análise multidimensional e outro, mais geral, cuja semântica é extremamente relacionada com o (TAD). As funções de análise, iniciadas pelo elemento (**<XWAnalysis>**), foram definidas no descritor de forma diferente. Foi mais conveniente especificar diretamente as hierarquias. O elemento **<XWHLeaf>** define a hierarquia entre dois elementos de TAD. A hierarquia pode conter qualquer número de colunas agregadas da tabela filha (**<XWHCol>**), definindo conjuntamente as funções para agregação ou *drilldown*. O processo de agregação pode conter vários elementos (**<TPDPar>**) de parâmetro para as funções de relacionamento que serão base da hierarquia.

Uma hierarquia pode ser encapsulada dentro de outra hierarquia de nível superior usando um elemento **<XWHBranch>**. Este elemento contém os mesmos dados da hierarquia básica, com exceção dos valores da coluna filha, que é substituída por um elemento básico de hierarquia (**<XWHLeaf>**). O Exemplo 14 mostra duas hierarquias, uma simples e outra de dois níveis, onde um elemento **<XWHBranch>** contém um **<XWHLeaf>**.

```
<XWAnalysis>
        <XWHLeaf …>
                <TPDPar …/>
                <XWHCol …./>
        </XWHLeaf>
        <XWHBranch    …>
                <XWHCol …/>
                <XWHLeaf …> …

        </XWHBranch>
..
</XWAnalysis>
```

Exemplo 14

Os procedimentos abaixo devem fornecem o suporte interno para criação de hierarquias que podem ou não fazer parte do produto de suporte utilizado. No caso do sistema proposto, estes procedimentos serão usados internamente para criar as hierarquias especificadas no descritor XML.

[E-3-1] - $\boldsymbol{X}HW_{hierarquia}$ – Entidade ou tipo de dado que irá definir uma hierarquia entre dois TADs. Se não houver suporte específico para este tipo, as hierarquias deverão ser armazenadas nas tabelas próprias dos catálogos de metadados.

[E-3-2] - $\mathbf{PASH1}^{n}{}_{Hierarquia}(\boldsymbol{X}W_{coluna}{}^{1}, \boldsymbol{X}W_{coluna}{}^{2} [,\mathbf{FRS2}^{n}[,\mathbf{FPSD}^{n}]])$ – Cada função **n** irá criar um objeto de hierarquia ($\boldsymbol{X}HW_{hierarquia}$) baseado em um tipo de relacionamento específico, como, por exemplo, um tipo de dado espacial "cidade", que está contido dentro de um outro representando estados. Cada hierarquia seria armazenada no catálogo de metadados e funcionaria como um parâmetro para operações de OLAP. Este grupo de funcionalidade irá se fundamentar em funções de suporte de relacionamento que são aplicadas a *tuplas* individuais (**FRS2**).

[E-3-3] - $\mathbf{PASH2}^{n}{}_{Hierarquia}(\boldsymbol{X}HW_{hierarquia}, \boldsymbol{X}W_{coluna}[, \mathbf{FRS2}^{n} [,\mathbf{FPSD}^{n}]])$ – Apenas complementando a função anterior, esta cria uma sub-hierarquia a partir da hierarquia $\boldsymbol{X}HW_{hierarquia}$ fornecida.

[E-3-4] - $\mathbf{PAMH1}^{n}{}_{Hierarquia}(\boldsymbol{X}W_{coluna}{}^{1}, \boldsymbol{X}W_{coluna}{}^{2}[,\mathbf{FRM2}^{n} [,\mathbf{FPMD}^{n}]])$ – Similar ao primeiro item, mas utiliza funções de suporte agregadoras, ou seja, que recebem conjuntos de TADs como parâmetros.

[E-3-5] - **PAMH2**$^{n}_{Hierarquia}$(***X***$HW_{hierarquia}$, ***X***W_{coluna}[,**FRM2**^{n} [,**FPMD**^{n}]]) – Similar ao anterior, apenas acrescentando elementos a uma hierarquia já criada.

[E-3-6] - **PASA**$^{n}_{AddData}$(***X***$HW_{hierarquia}$, ***X***W_{coluna}, *fAgg* [,*fDrillDown*]) – Adiciona uma nova variável de TPD com função de agregação e *drilldown* a uma hierarquia. Quando uma agregação for requisitada sobre um TAD, estas funções serão aplicadas a cada variável que será agregada junto com o próprio TAD. Por exemplo, se tivermos um TAD "estado", que agregue um TAD "*cidade*", sendo a variável "*população*" associada à cidade, podemos agregar as cidades dentro de um estado e ter o valor agregado da população de um estado.

[E-3-7] - **PASS**$^{n}_{Aggregação}$(***X***$HW_{hierarquia}$, Nível, $DW_{parâmetro}$) – Estas funções irão agregar os dados associados pelas funções **PASA**$^{n}_{AddData}$. O nível indicará quantas hierarquias deverão ser navegadas na agregação

[E-3-8] - **PASD**$^{n}_{DrillDown}$(***X***$HW_{hierarquia}$, *Nível*) – Estas funções irão realizar a navegação inversa da agregação (*drilldown*) para tantos níveis quanto for especificado e usarão as funções de *drilldown* definidas para a hierarquia na função **PASA**$^{n}_{AddData}$. O nível de detalhe calculado da população de um estado, descendo em direção a das cidades que ele contém, poderia ser calculado por uma função de média ponderada, a qual dividiria a população do estado de acordo com o tamanho das cidades.

[E-4-0] - [**PASG**$_{Análise}$] – Representam o conjunto de funções *ad hoc* que pode somente ser especificado no domínio dos TADs, como as funções de mapeamento para tipos de dados espaciais. O elemento (**<XWAdHoc>**) do descritor tem uma estrutura flexível que permite a inclusão de qualquer procedimento projetado com finalidades específicas.

4.4 – Conclusão

Neste capítulo apresentamos a descrição geral e detalhada de um *framework* para construção de sistemas de *data warehousing* incorporando TADs. A construção da metodologia desenvolvida para este *framework*, bem como suas ferramentas auxiliares, será mostrada no capítulo seguinte. No Capítulo 7, estaremos apresentando um sistema que vai utilizar a especificação gerada pela metodologia (através de um descritor XML) para a geração automática de ferramentas de *data warehousing* para TADs.

Capítulo 5 – Desenvolvimento do Framework 5-Alpha

5.1 – Fundamentos

Existem duas formas para um sistema acomodar dados não convencionais. A primeira é criar aplicações especializadas e *APIs* (*Application Program Interfaces*). A segunda é criar TADs representando os dados não convencionais e incorporar capacidades de manipulação destes dados dentro do banco de dados. A primeira técnica é apropriada para aplicações especializadas ou em casos onde os dados são adquiridos e montados especificamente para um certo propósito de análise. Por outro lado, em casos onde nós já temos os dados adquiridos e armazenados, parece muito mais razoável não interferir com a arquitetura do banco original e simplesmente estender o processo de *data warehousing* tradicional para trabalhar com estes tipos de dados não convencionais. O suporte para estes tipos de dados, que denominados de "*abstratos*", embasa as operações no banco de dados, de forma que o processo de *data warehousing* continue operando transparentemente. Os TADs passam, deste modo, a tirar vantagem do sistema de suporte de segurança, transação e índices do banco de dados. As propriedades específicas do tipo de dado estendido podem então ser exploradas, fornecendo novas informações não percebidas anteriormente.

A Figura 14 mostra uma extensão do modelo básico de *data warehousing* apresentado no Capítulo 2 - Figura 3. Cada módulo foi estendido para mostrar as funcionalidades expandidas para manipular TADs.

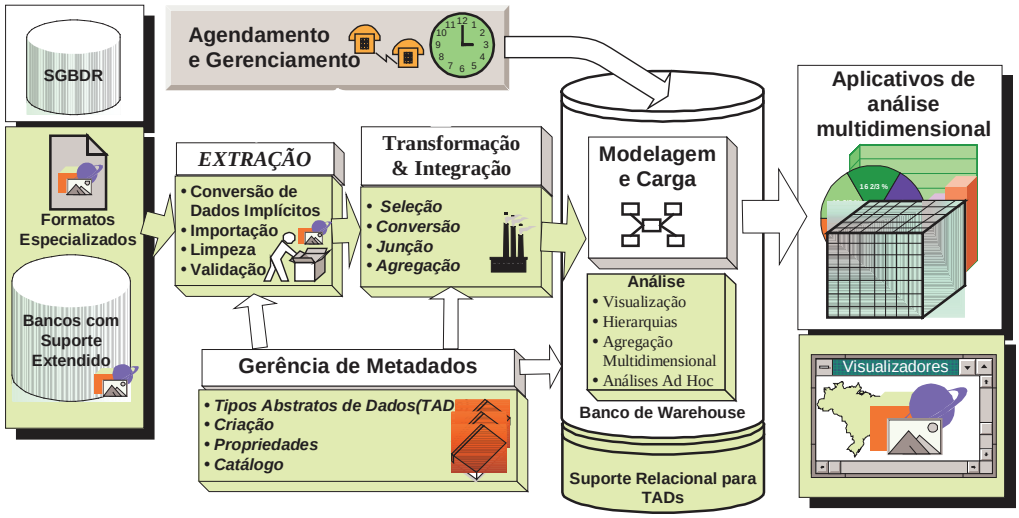

Figura 14 – Modelo geral de *data warehouse* estendido

Uma importante observação a ser notada é que, neste trabalho, entendemos o processo de *data warehousing*, assim como as atividades de OLAP, de uma forma mais ampla e conceitual. É fácil perceber que nem sempre é possível visualizar um tipo de dado estendido, como dados espaciais dentro de uma tabela de um cubo multidimensional, por exemplo.

A arquitetura necessária para estender o processo de *data warehousing* deve preencher dois requisitos iniciais: implementar os tipos estendidos no banco de dados objeto relacional e definir a semântica dos processos de *data warehousing* para este tipo de dado. O primeiro requisito consiste não só da implementação dos tipos de dados, mas também do suporte de baixo nível para os mesmos. A semântica específica vai nos guiar no desenvolvimento das funções de extração, transformação e análise. Todos os elementos de suporte para a construção do modelo de *data warehousing* estendido para um conjunto de TADs $\boldsymbol{X}$ pode ser definido em um modelo $\boldsymbol{X}$DW = { $\boldsymbol{X}W_{tipos}$, $\boldsymbol{X}W_{transparentes}$, $\boldsymbol{X}W_{MD}$, $\boldsymbol{X}W[\mathbf{F}_{Suporte\ Relacional}]$, $\boldsymbol{X}W[\mathbf{P}_{processamento}]$}, onde cada elemento é um subcategoria de funções e tipos descrita resumidamente como:

A - $\boldsymbol{X}W_{tipos}$ – Todos os TADs do conjunto $\boldsymbol{X}$.

$\boldsymbol{X}W_{transparentes}$ – Semântica estendida para os operadores convencionais de comparação e conectivos lógicos (=,≠, <, AND, OR etc)

$\boldsymbol{X}W_{MD}$ – Suporte de metadados.

$\boldsymbol{X}W[\mathbf{F}_{Suporte\ Relacional}]$ – Funções de SQL estendidas para o suporte à álgebra relacional para manipular o conjunto $\boldsymbol{X}$.

$\boldsymbol{X}W[\mathbf{P}_{processamento}]$ – Funcionalidades de *data warehousing* estendido.

5.2 – Seção A – Tipos Abstratos de Dados

Os TADs são utilizados para representar entidades complexas do mundo real no modelo do banco de dados. Os tipos mais comuns estão associados a espaço ou tempo, como elementos geográficos, imagens, vídeo etc. Contudo, os TADs podem também representar qualquer outro conceito, como uma doença, um produto a ser vendido etc. Neste trabalho consideramos a visão de um TAD como ampla, ou seja, qualquer TAD é passível de ser processado no banco de *data warehouse* e analisado de forma multidimensional.

Um TAD é composto de um conjunto de atributos que podem ser TPDs ou também outros TADs, se o modelo de dados contiver hierarquias entre os TADs. A diferença entre um TAD pai e um TAD filho pode ser o acréscimo de atributos ou uma simples troca de nome por razões de modelagem conceitual. Todas estas hierarquias entre TADs, além dos próprios TADs, precisam ser registrados num catálogo de metadados para utilização apropriada pelo processo de *data warehousing*. A metodologia proposta neste trabalho leva em conta os processos de transformação ou conversão entre TADs, o que torna necessário analisar os casos de polimorfismo. Ele ocorre quando dois TADs, filhos de um mesmo TAD pai, são tratados como sendo do tipo pai, escondendo-se assim os seus reais tipos. Suponha, por exemplo, que um círculo e uma linha poligonal, ambos filhos de "*geometria*", podem ser tratados como duas geometrias. Nestes casos, nem sempre é possível determinar se está existindo uma conversão entre tipos apenas observando-se os parâmetros de entrada e retorno.

O ponto fundamental da escolha dos TADs que farão parte do processo de *data warehousing* deve se basear nas necessidades de análise. Uma das razões de sua inclusão no processo é a existência de relacionamentos e hierarquias entre TADs, resultando em agregações, além de projeções ao longo do tempo. Esta escolha será resultado da análise de negócios e requisitos do problema ou aplicação específica em questão. Portanto, a análise completa dos TADs no processo de *data warehousing* deve realizar os seguintes passos:

- Escolha dos TADs que farão parte do processo de *data warehousing*
- Registro da estrutura interna de cada TAD.
- Registro de todas as hierarquias entre os TADs
- Registro dos casos de polimorfismo existentes

Nosso estudo para desenvolvimento do *framework* proposto irá se encarregar de guiar a execução de cada um destes passos, assim como inserir os TADs ao longo do processo de *data warehousing*.

5.3 – Seção B – Operadores Transparentes

Estes operadores consistem nos operadores de comparação (=, < etc.) e nos conectivos lógicos (AND, OR etc.). Embora eles façam parte do suporte relacional aos TADs, achamos conveniente estudá-los separadamente devido as suas características próprias. A denominação de "transparentes", adotada neste trabalho, se deve ao fato de

que as relações de comparação entre TADs através dos operadores padrão deve manter a sua semântica. Por exemplo, a comparação entre dois TADs, na forma:

"$O^1(TAD_x) = O^2 (TAD_x)$" onde,

$O^1(TAD)$ representa um objeto de Tipo Abstrato TAD_x, deve responder verdadeiro se os dois objetos forem iguais dentro da semântica de igualdade do TAD. Podem existir várias funções similares à "*igualdade*", mas as operações que nos referimos nesta seção devem manter a semântica estrita de que os dois objetos devem ser do mesmo tipo (sem polimorfismo), mesma cardinalidade (não representar conjuntos de objetos) e comparar apensas o conteúdo dos dois objetos. Estes operadores podem ser implementados através de funções ou diretamente com a sobrecarga dos operadores.

Os conectivos lógicos são de certa forma independentes da questão dos TADs do modelo. Entretanto, pode existir uma situação que requeira manipulação de TADs. Isto ocorrerá quando tivermos condições lógicas representadas por TADs, ao invés de valores lógicos "falso" ou "verdadeiro". Podemos, por exemplo, ter implementações do conectivo lógico "E" que ao invés de retornar "0" ou "1", para falso e verdadeiro, respectivamente, retornam um TAD "FUZZY" que contenha valores percentuais para indicar o índice de verdade da afirmação. Este é um exemplo típico em lógica nebulosa (Braga 1998a,Zadeh 1988,Zadeh 1965). Os operadores e conectivos tratados neste trabalho já foram apresentados na Tabela 10 da seção 4.2.2 e estão listados abaixo:

- **<u>Operadores de Comparação</u>**
    - **Igualdade** – Aqui incluímos os dois comparadores complementares, de igualdade e de diferença. Imaginemos, como exemplo, dois objetos espaciais, do tipo "*círculo*". Eles serão iguais se tiverem o mesmo centro e raio, e diferentes se qualquer destes atributos for diferente.
    - **Menor e Maior –** Estes operadores cuidam das comparações quantitativas, se houver, incluindo também as igualdades, ou seja, comparadores do tipo "*maior ou igual*" ou só "*maior*".
- **<u>Conectivos Lógicos</u>**
    - **Conjunção** – Representa o conector "E".
    - **Disjunção** – Representa o conector "OU".
    - **Negação** – Representa o conector "NÃO".

Os operadores de comparação são, de fato, um caso especial de funções de relacionamento binário, que serão tratadas adiante. Já os conectivos são funções de relacionamento que recebem parâmetros lógicos. Iremos fazer uma pequena menção a elas na parte de suporte relacional na seção 5.5.

5.4 – Seção C – Suporte de Metadados

Dividimos as funcionalidades necessárias ao suporte de metadados em três partes: catálogos, manipulação de catálogos e manipulação de dados. Os catálogos irão conter as informações sobre os TADs e os módulos do sistema. Os métodos de manipulação vão inserir e remover informações dos catálogos. Os procedimentos de manipulação de dados, por outro lado, não vão lidar diretamente com catálogos, mas vão criar e manipular os objetos de TAD.

5.4.1– Catálogos - Modelo básico

Na Figura 13 da seção 4.3.2, mostramos um modelo de suporte de metadados para um sistema de *data warehouse* estendido. Embora vários sistemas comerciais já possuam parte deste suporte, este modelo visa fornecer o mínimo de componentes necessários, os quais devem armazenar as seguintes informações:

1 - **Tipos Abstratos de Dados e sua localização** – O primeiro item necessário no catálogo de metadados é a lista de TADs, bem como as tabelas e colunas onde eles estão localizados. Esta informação será utilizada em diversos setores, podendo, por exemplo, indicar para um módulo de visualização qual o melhor modo de se mostrar o valor de uma célula contendo um TAD.

No modelo de catálogo básico proposto(Figura 15), todos os TADs ficam armazenados na entidade de “TIPO ABSTRATO DE DADO” e a localização de cada coluna de cada tabela (TABELA) de cada banco (BANCOS_DE_DADOS) está armazenada na entidade “COLUNAS DE TIPOS ABSTRATOS DE DADOS”. Esta última entidade pode ainda receber detalhes sobre um TAD em uma coluna específica.

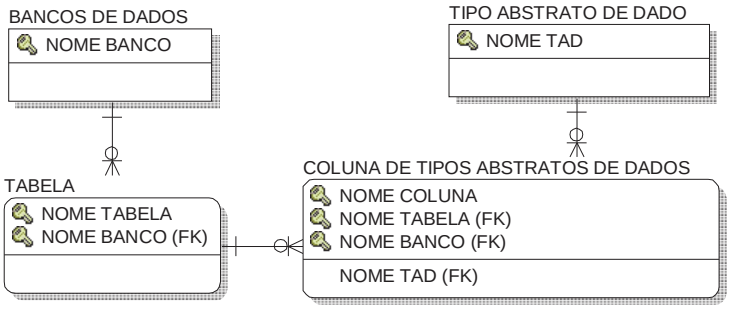

Figura 15 – Suporte de TADs do catálogo de metadados

2 - **Modelos** – Este elemento não é estritamente necessário, mas se torna conveniente para a construção de modelos de *data warehousing* para vários conjuntos de TADs diferentes, assim como diversas modelagens do mesmo TAD. Ele associa todos os elementos e módulos. Desta forma, podemos criar diversos TADs e inseri-los, assim como os módulos de processamento, em modelos diferentes..

O modelo do catálogo de metadados proposto, através da entidade MODELO(Figura 16), utiliza um relacionamento de muitos para muitos com os módulos de processamento (MODULO DE PROCESSAMENTO), com as colunas aonde os TADs estão localizados (COLUNAS DE TIPOS ABSTRATOS DE DADOS) e com a lista de hierarquias (BASE DE HIERARQUIA) para armazenar todos os modelos.

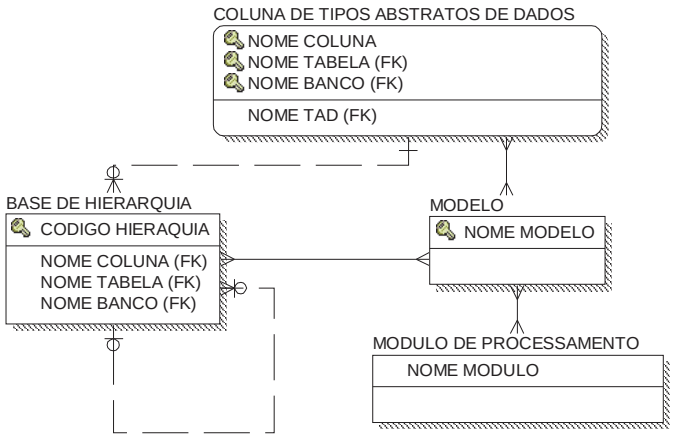

Figura 16 – Suporte de modelos do catálogo de metadados

3 - **Hierarquias** – As informações sobre cada hierarquia construída no sistema são necessárias para poder realizar as operações de análise multidimensional. Quando desejamos agregar um grupo de atributos através do relacionamento hierárquico dos seus TADs, temos que saber que atributos e que funções de agregação utilizar.

As hierarquias entre as colunas com TADs ficam armazenadas nas entidades "BASE DE HIERARQUIA" e "COLUNA DE AGREGACAO"(Figura 17). A primeira contém um auto-relacionamento que permite o encadeamento recursivo de hierarquias, ou seja, uma coluna com TADs está associada a uma coluna filho, que pode estar associada a outra coluna filho e assim por diante. A entidade "COLUNA DE AGREGACAO" serve para indicar cada coluna que será agregada pela hierarquia. Esta entidade contém a indicação da função que será usada na agregação, assim como o nível de detalhe (*drilldown*).

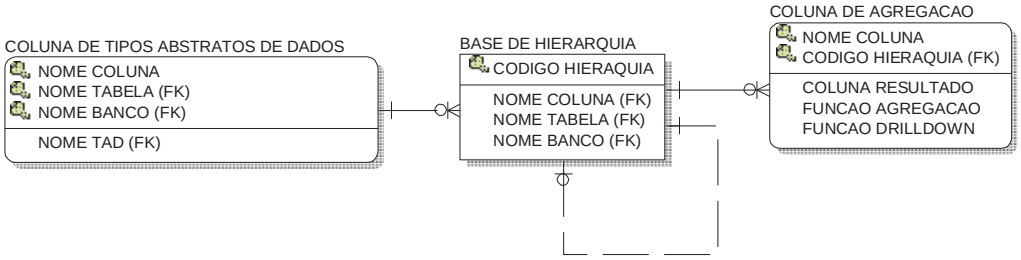

Figura 17 – Suporte de hierarquias do catálogo de metadados

4 - **Módulos de processamento** – Os módulos de extração e pré-processamento podem ser bastante diversos. Por esse motivo, torna-se necessário flexibilizar a incorporação dinâmica de módulos externos. Isto requer que os módulos possam incorporar a informação sobre qualquer número de parâmetros, detalhes sobre a implementação do módulo, o nome referenciado internamente e sua categoria (ver seção 4.2.3). Existem módulos que precisarão ser carregados no sistema apenas uma vez, para isto também é necessário a informação de que o módulo já foi carregado. Além disso, podemos usar esta informação para indicar ao sistema que o módulo atualmente carregado já não está mais válido.

O modelo de catálogo proposto contém duas entidades para armazenamento dos módulos de processamento (Figura 18): "MODULO DE PROCESSAMENTO" e "PARAMETRO DE MODULO". Este último permite a inclusão de listas ilimitadas de parâmetros para os módulos na entidade "MODULO DE PROCESSAMENTO". O atributo "NOME EXTERNO" referencia o nome do módulo no sistema operacional e o "NOME INTERNO PROC" é o nome que será dado ao procedimento armazenado no banco. O "TIPO IMPLEMENTACAO" indica o tipo de código no sistema operacional (C, DELPHI, SQL etc.). O atributo "CATEGORIA MODULO" indica que tipo de extração será executada pelo módulo, como validação, agregação, importação etc.

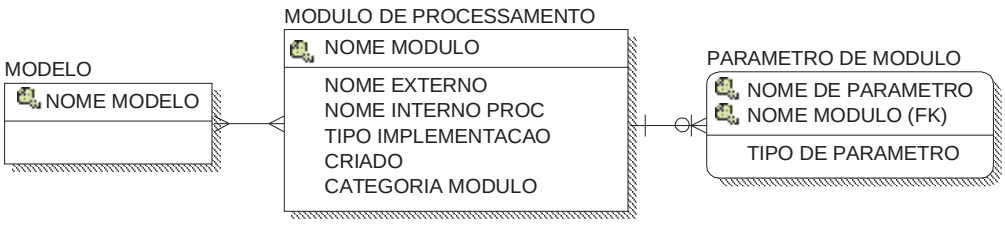

Figura 18 – Suporte de módulos de processamento do catálogo de metadados

5.4.2 – Procedimentos de Manipulação de Catálogos

A inserção e remoção de elementos do catálogo são realizadas pelo seguintes grupos de procedimentos:

1 - **Registro de Elementos de Dados** – Manipula, insere e remove o registro dos TADs que são criados para um modelo no banco de dados de *data warehouse*.

2 - **Registro de Módulos de Processamento** – Manipula, insere e remove o registro de módulos que realiza o processamento de elementos, como importação, pré-processamento e transformação.

3 - **Registro de Hierarquias** – Manipula, insere e remove o registro de novas hierarquias entre os tipos de dados estendidos ou os TADs.

A simbologia para todas estas categoria de procedimentos segue, genericamente, o formato:

"**PCS** <Código da Subcategoria>$_{\text{Registro / Remoção}}$(Parâmetro)",

como pode ser visto detalhadamente no Apêndice A. O procedimento para registro de um TAD em uma certa coluna do banco seria, por exemplo, "**PCSD**$_{\text{Registro}}$(Parâmetros)".

5.4.3 – Funções de Manipulação de Dados

Estas funções constituem um caso particular das funções de projeção, onde, dado um TPD usado como parâmetro, temos como resultado um TAD. Elas estão espalhadas pela Tabela 19, na seção 5.5.2, em todas as áreas onde um TPD dá origem a um TAD -- já mostrado na Tabela 11, da seção 4.2.3. Elas se dividem nas seguintes subcategorias:

1 - **Acesso** – Esta categoria de funções está relacionada à manipulação dos tipos de dados estendidos. Elas são chamadas de funções de acesso e são somente unárias. Elas recebem como parâmetro um elemento do tipo de dado estendido e retornam algum elemento ou sub-componente do mesmo.

2 - **Propriedade** – Estas funções são similares às anteriores, contudo diferem no fato de retornarem uma propriedade do tipo de dado, mesmo que esta não faça parte física do dado. Estas propriedades têm que ser muitas vezes calculadas no momento de sua obtenção, como a área de um objeto, por exemplo.

3 - **Criação** – Estas funções também estão relacionadas à manipulação dos tipos de dados estendidos. Seu objetivo é criar instâncias de tipos de dados estendidos a partir de tipos de dados primitivos. Esta versão da categoria recebe como parâmetros os valores individuais dos atributos de uma *tupla*. A outra versão (**`FCMC`**) difere no fato de receber como parâmetro todo um conjunto de valores. Uma das perspectivas desta versão é a criação de uma normalização estendida para os de tipos de dados estendidos. Suponha, por exemplo, que tenhamos uma tabela com uma lista de coordenadas, cada uma associada a uma cidade. Poderíamos criar diretamente um TAD "Estado" a partir de todas as coordenadas e associar cada cidade a este "Estado".

A classificação destes tipos de função deve se guiar primeiramente no tipo de parâmetro e retorno.O segundo passo é verificar, caso haja parâmetros e retorno de TADs, se os mesmos pertencem a mesma hierarquia, como mencionado na seção 5.2. A simbologia para todas estas categoria de funções segue, genericamente, o formato:

"**`FC[S/M]`** <Código da Subcategoria>$_{\text{Subcategoria}}$(Parâmetro)",

como pode ser visto detalhadamente no Apêndice A. Uma função para criação de elementos recebendo parâmetros individuais de *tupla* seria, por exemplo, "**`FCSC`**$_{\text{Criação}}$(Parâmetros)".

5.5 – Seção D – Suporte Relacional para TADs

A construção do suporte relacional para os TADs vai requerer primeiramente um estudo da extensão da álgebra relacional. Veremos a seguir que este estudo nos leva a um grande conjunto de funções e a uma metodologia para classificá-los.

5.5.1 – Extensão da álgebra relacional

A álgebra relacional se baseia em um conjunto de operadores que utilizam como parâmetro predicados aplicados às relações ou tabelas do banco de dados (Navathe e Elmasri 1994). Estes predicados podem ser obtidos através de funções que retornem valores lógicos (verdadeiro ou falso) ou um processamento sobre os atributos das relações, como veremos logo a seguir. O suporte necessário para tipos de dados estendidos requer a adição de um novo conjunto de funções. Iremos mostrar a seguir as

categorias destas funções para cada operador da álgebra. Também devemos ressaltar que iremos considerar funções de relacionamento binário, ou seja, que recebem no máximo dois parâmetros que representem atributos de uma relação ou tabela. Consideramos que relações ternárias ou n-árias tem poucas aplicações práticas no uso dos TADs, e é sempre possível simular relações de maior cardinalidade através do aninhamento de funções.

SELEÇÃO – FUNÇÕES DE RETORNO LÓGICO

A operação de seleção é usada para selecionar um subconjunto de *tuplas* em uma relação que satisfaça a condição de seleção. A simbologia da álgebra relacional para a operação de seleção é:

$$\sigma_{\text{<Função Condicional>}} \text{ (Relação) onde,}$$

- σ - denota o operador de seleção e
- <Função Condicional> - é uma expressão booleana definida sobre os atributos da relação.

O suporte relacional à seleção irá somente requisitar a construção de funções que substituam a expressão booleana do operador. Elas irão retornar um parâmetro booleano (verdadeiro ou falso) de acordo com os relacionamentos entre os seus parâmetros. As subcategorias destas funções, abaixo, irão definir as combinações dos tipos de parâmetros que as mesmas receberem. Por exemplo, um TAD e um TPD (binária), somente um TAD (unária), somente TPDs etc. As funções também poderão receber valores individuais de *tuplas* de uma relação ou conjuntos de valores de um atributo:

1 - **Subcategoria 1** – Subcategoria de funções que retornam uma condição booleana sobre um TAD sem receber nenhum outro parâmetro. Funções deste tipo podem, por exemplo, retornar a validade de um TAD. Elas também podem receber um conjunto de TADs e retornar uma condição booleana sobre o conjunto total de elementos, como uma função que avalia uma condição de organização entre os elementos do conjunto.

2 - **Subcategoria 2** – Subcategoria de funções que retornam uma condição booleana sobre os valores de dois elementos de TADs de duas relações. Estas funções são usadas para estabelecer junções baseadas em características específicas dos TADs. Por exemplo, se o TAD é uma imagem, poderíamos ter uma função que representasse a semelhança entre duas imagens

3 - **Subcategoria 3** – Esta subcategoria de funções de relacionamento recebe alternadamente TADs e TPDs como parâmetros. Os TPDs serão usados como parâmetros para seleção dos TADs.

4 - **Subcategoria STD** – Esta subcategoria de funções recebe TPDs e retorna uma seleção qualquer sobre eles. Ela representa funções de suporte para seleção e junção convencional com TPDs.

A simbologia para todas estas categoria de funções segue, genericamente, o formato:

"**FR[S|M]**<Código da Subcategoria> $_{\text{Relação}}$(Parâmetros)",

como pode ser visto detalhadamente no Apêndice A. Uma função de relacionamento entre dois TADs recebendo parâmetros individuais de *tupla* seria, por exemplo, "**FRS2**$_{\text{Relação}}$(Parâmetros)".

PROJEÇÃO – FUNÇÕES DE RETORNO DE ATRIBUTOS

O operação de projeção seleciona certas colunas para todas as *tuplas* de uma relação e pode ser denotada por:

$$\pi_{<\text{Atributos, . }>} \text{ (Relação) onde,}$$

π - é o operador de seleção e

f ($\pi_{<\text{Atributos,. }>}$ (Relação)) – representa os valores resultantes de uma função qualquer *f* aplicada aos valores dos atributos retornados pelo operador de projeção.

Seguindo a operação de seleção, temos somente que criar funções que serão aplicadas sobre so atributos. Esta categoria de funções é bem mais extensa que a anterior, pois, de fato, irá retornar todos os possíveis processamentos sobre os valores dos atributos da relação. Um subconjunto destas será usado inclusive para a criação de instâncias dos TADs.

Todas as funções de projeção são escritas no formato abaixo (veja Apêndice A). Elas recebem como parâmetros os valores dos atributos de uma *tupla* ou valores nominais. A quantidade de parâmetros que representam atributos de uma relação (nomes de colunas) pode ser de no máximo dois, pois estamos considerando apenas funções binárias.

FP[S|M][T|C|D|X]<subcategoria>(Parâmetros...)

A categoria de parâmetros (**S**) recebe parâmetros como elementos individuais de *tuplas,* enquanto que na categoria de parâmetros (**M**), pelo menos um dos seus parâmetros representará um conjunto de valores de um atributo da tabela ou relação. As subcategorias (**T|C|D|X**) podem ser de:

Conversão – As funções cujo tipo de dado de retorno difere do tipo de dado fornecido como parâmetro são classificadas nesta categoria. Algumas situações poderão parecer dúbias, quando vários objetos fazem parte de uma hierarquia de classes. Suponhamos a função A = f(B), onde B é uma subclasse de A. Existe uma conversão de tipos, mas o modelo objeto-relacional A ainda pode existir dentro do mesmo domínio de B. No caso contrário (B = f(A)) existe um "enriquecimento" do tipo de dado A (ver seção 2.2.2), de forma que B não pode mais ser tratado como A.

Existe ainda uma situação onde os tipos de dados são diferentes, mas uma conversão não se aplica. Isto ocorre quando os dois TADs não fazem parte de uma mesma hierarquia semântica. Suponhamos, por exemplo, que um tipo de dado represente um *Sistema de Coordenada* e outro seja *Círculo*. Um *Sistema de Coordenada* não se converte em um *Círculo*, mas é um parâmetro para a construção do *Círculo*. Neste caso, é fácil observar que *Sistema de Coordenada* não faz parte da árvore de objetos espaciais à qual pertence o *Círculo*.

Transformação – As funções que realizam transformações num TAD podem mudar os valores dos atributos do TAD, mas o TAD, ou seja, a classe dos objetos, deve permanecer a mesma. De toda forma, dentro do paradigma orientado por objetos, um TAD poderia, conceitualmente, alterar sua própria estrutura, como acrescentar um atributo, e continuar sendo o mesmo TAD -- desde que sua interface continuasse a mesma. Isso, porém, tem poucas aplicações práticas.

Combinação – Esta categoria se aplica apenas às funções binárias. O retorno destas funções pode ser do tipo de um dos parâmetros (Transformação) ou de um tipo diferente dos dois (Conversão). O processamento deve utilizar o conteúdo de cada um dos parâmetros para produzir o valor e o tipo do dado de retorno.

DrillDown – Esta categoria está mais comumente associada a grupos de processos mais complexos, mas seu objetivo é aumentar o nível de detalhe a respeito de um elemento que foi primariamente gerado a partir de uma agregação (ver seção 2.2.2).

JUNÇÃO

A operação de junção, denotada abaixo, é usada para combinar *tuplas* relacionadas em uma única *tupla*.

$$(\text{Relação}_1) \triangleright\!\triangleleft_{\text{<Função Condicional>}} (\text{Relação}_2)$$

O relacionamento é estabelecido através do retorno de uma expressão lógica que deve envolver as duas relações. Desta forma, um subconjunto das funções utilizadas no suporte de seleção será usada para o suporte à junção. Este subconjunto envolve todas as funções que estabelecem relacionamentos entre dois objetos de um TAD.

Um exemplo comum de uma junção ocorre quando desejamos saber todos os indivíduos que residem dentro de uma certa área. Os TADs, neste caso, são tipos de dados espaciais, como ponto e área. A função de relacionamento vai retornar verdadeiro ou falso se o ponto se encontra dentro do polígono, efetuando assim a junção do conjunto de dados dos indivíduos com os dados de cada área.

FUNÇÕES DE AGREGAÇÃO

As funções de agregação têm por finalidade lidar com todas as *tuplas* de um certo atributo de uma só vez, como, por exemplo, tirar a média de todos os valores de uma coluna (AVG).

A agregação não está definida na proposta original da álgebra relacional (Codd 1970), mas vários trabalhos (Cabibbo e Torlone 1999,Klug 1982) têm sido feitos propondo extensões da álgebra para inclusão de operadores de agregação. Eles se fundamentam na especificação de funções que são chamadas no início, no fim e para cada *tupla* de uma relação. O resultado final da agregação é dado após a chamada da última função. Este é o principio usado pelos poucos bancos de dados comerciais, como o ORACLE (Fernandes 2002), que suportam funções de agregação definidas pelo usuário. De fato, a maioria dos bancos de dados comerciais não possuem este suporte. Apesar da nossa escolha de banco de dados para este trabalho também não possuir este suporte, isto não influenciará a generalidade do nosso modelo.

Classificamos as funções de agregação nas seguintes subcategorias, de acordo com os parâmetros que recebem e retornam, e com a sua semântica:

- **Subcategoria 1** – Esta subcategoria de funções executa conversão, dado que recebe TADs e retorna TPDs, como uma função que retorna o valor da área total ocupada por todos os objetos do tipo “Imagem”.
- **Subcategoria 2** – Esta subcategoria é similar à anterior, contudo retorna conjuntos de TPDs (TPD_{set}). Podemos supor como exemplo uma função que retorna o conjunto dos valores das áreas de todos os polígonos que se interceptam.
- **Subcategoria 1<Y>** – O produto da agregação dentro desta subcategoria de funções pode realizar conversão ou transformação, cuja semântica(da conversão e transformação) já foi devidamente explicada anteriormente. O fator diferencia se o retorno será um elemento individual ou um conjunto:
    - **Subcategoria <X>1** – Funções com retorno de TAD – Um exemplo desta subcategoria seria o retorno de um objeto de contorno envolvendo um conjunto de objetos geométricos. Neste exemplo, a função estará executando, muito provavelmente, uma conversão, assumindo que o conjunto de objetos de parâmetros terá vários tipos, mas que o resultado terá de ser polígono.
    - **Subcategoria <X>2** – Funções com retorno de TAD_{set} – Similarmente ao exemplo anterior, poderíamos ter uma função cujo retorno consistisse em conjuntos de objetos de contorno que envolvessem os grupos de objetos que se interceptassem.
- **Subcategoria 2<Y>** – Funções tipo 1Y com Combinação – Esta subcategoria de funções é capaz de realizar combinações entre conjuntos de elementos de TADs. Ela se divide em duas subcategorias dependendo do tipo de retorno:
    - **Subcategoria <X>1** – Funções com retorno de TAD – Este tipo de função recebe dois conjuntos de TADs e retorna um único elemento de TAD. Uma função de exemplo poderia retornar o objeto geométrico mais próximo, dentro de um grupo de objetos, de um outro grupo de objetos.
    - **Subcategoria <X>2** – Funções que retornam a instância de um TAD_{set} – Esta outra categoria, similar à anterior, recebe conjuntos de TADs como parâmetro, mas retorna um outro conjunto de TADs. Podemos ter, por

exemplo, uma função que retorne os objetos geométricos de um grupo mais próximo que os objetos do segundo.

As seguintes subcategorias diferem das anteriores por receberem TPD_{set} como parâmetros. Elas representam as funções de agregação convencionais.

- **Subcategoria STD1** – Esta subcategoria representa funções de Conversão ou Transformação de TPDs com retorno de TPD, como a função AVG (média do SQL).
- **Subcategoria STD2** – Funções de Conversão ou Transformação de TPDs com retorno TPD_{set}.

A simbologia para as funções de agregação terá a seguinte forma geral:

FA*XX*$^{n}_{\text{Agregação}}$, onde,

a simbologia ***XX*** irá indicar as subcategorias das funções de agregação, como mostrado na Tabela 13, da seção 4.2.4. Mostraremos adiante como estas funções estarão incluídas no processo de construção do *framework* **5-Alpha**.

OPERADORES DE UNIÃO, INTERSEÇÃO E DIFERENÇA

As operações de união, interseção e diferença são independentes do tipo de dado, pois trabalham com conjuntos de elementos como um todo, transparentemente aos tipos de dados que contém.

5.5.2 - Classificação de funções de suporte

Todas as funções de suporte consideradas aqui poderão ter dois formatos distintos relacionados à quantidade de parâmetros fornecidos. Esta classificação não difere do principio aplicado em Han, Stefanovic e Koperski 1998a para a caracterização de objetos espaciais e suas funções de manipulação. No entanto, aqui, estaremos aplicando este principio de uma forma mais geral:

Funções Unárias: Tipo de Retorno **FUNÇÃO**(Parâmetro)

Funções Binárias: Tipo de Retorno **FUNÇÃO**($Parâmetro_1$, $Parâmetro_2$)

TIPOS DE RETORNO

As funções que representam predicados no suporte à seleção só poderão retornar valores lógicos. Já as funções de suporte podem retornar os seguintes tipos de dados:

Tipos Abstratos de Dados (TAD) – São tipos definidos pelo usuário no modelo objeto-relacional (ADTs – *Abstract Data Types*) que serão usados no processo de *data warehousing*.

Tipos Primitivos de Dados (TPD) – São tipos convencionais e endógenos do banco de dados, como inteiros, *Strings* etc. As funções de suporte de projeção podem também retornar tipos lógicos.

Conjuntos (TAD_{set}, TPD_{set}) – Representam conjuntos de valores, tanto de TADs quanto de TPDs.

Booleano – Representam o tipo lógico e são utilizados somente nas funções que irão suportar as operações de seleção, apesar deste tipo de fato pertencer ao conjunto de TPDs. Em alguns casos, este tipo pode ser implementado fisicamente com tipos primitivos (ex., 1=Verdadeiro, 0 = Falso).

PARÂMETROS

Os tipos dos parâmetros das funções unárias e binárias são os mesmos do retorno explicado no item anterior. Eles podem ser provenientes de um atributo (ou coluna), de uma relação (ou tabela) ou mesmo serem criados dinamicamente no momento da chamada da função (*on the fly*). Frequentemente, as funções poderão conter vários parâmetros de TPDs. Uma coordenada (x,y,z), por exemplo, já representa mais de dois parâmetros. Desta forma, nosso sistema de classificação de funções irá considerar que um parâmetro do tipo TPD pode de fato estar representando um grupo de parâmetros (Parâmetro 1, Parâmetro 2,...Parâmetro n) sem perda de generalidade do processo de classificação.

Os parâmetros do tipo conjuntos podem vir do nome do atributo e da relação de onde eles provêm, ou através de parâmetros do tipo conjunto.

Tipos Abstratos de Dados(TAD)

Tipos Primitivos de Dados(TPD)

Conjuntos(TAD_{set}, TPD_{set})

CLASSIFICAÇÃO GERAL

O primeiro passo para construção exaustiva de todas as categorias de funções é dispor todas as combinações possíveis de tipos de retornos e parâmetros de entrada apresentados acima. Para as funções binárias, construímos na Tabela 16 todas as combinação dos tipos de parâmetros apresentados. Várias categorias particulares estão destacadas à parte (Tabela 17). A combinação de tipos primitivos entre si indicarão funções para manipulação de dados convencionais. As combinações simétricas foram retiradas da tabela, porém as combinações que envolvem apenas TPDs não foram removidas, pois ainda poderão estar relacionadas aos tipos estendidos através do retorno

das funções de suporte. A explicação mais detalhada destas combinações de parâmetros será apresentada mais adiante, junto com o detalhamento da Tabela 18.

Tabela 16 – Parametrização de funções binárias

	TAD	**TPD**	***TAD$_{set}$***	***TPD$_{set}$***
TAD	TAD × TAD	TAD × TPD	TAD$_{set}$ × TAD	TPD$_{set}$ × TAD
TPD		TPD × TPD	TAD$_{set}$ × TPD	TPD$_{set}$ × TPD
TAD$_{set}$			TAD$_{set}$ × TAD$_{set}$	TAD$_{set}$ × TPD$_{set}$
TPD$_{set}$				TPD$_{set}$ × TPD$_{set}$

Tabela 17 - Legenda de parâmetro de funções binárias

Parâmetros envolvendo somente elementos
Parâmetros envolvendo conjuntos
Combinação simétrica já apresentada
Combinações envolvendo apenas tipos convencionais

A Tabela 18 mostra a lista exaustiva das categorias de funções de suporte possíveis para os parâmetros e tipos de retorno apresentados, considerando funções unárias e binárias. Temos seis áreas genéricas de tipos de funções que serão classificadas dentro desta tabela. O resultado final, mostrado na Tabela 19, conterá algumas categorias a mais, bem como exceções. Esta tabela nos possibilita fazer uma análise inicial dos tipos de funções que podemos gerar a partir dos tipos de dados apresentados.

Tabela 18 - Categorias exaustivas de tipos de funções

<table>
<tr><th colspan="3" rowspan="3"></th><th colspan="5">Retorno</th></tr>
<tr><th colspan="3">Elementos</th><th colspan="2">Conjuntos</th><th>Booleano</th></tr>
<tr><th colspan="2">Parametrização</th><th>TAD</th><th>TPD</th><th>TAD$_{set}$</th><th>TPD$_{set}$</th><th></th></tr>
<tr><td rowspan="4" colspan="2">Unária</td><td>TAD
TPD</td><td colspan="4">Funções de
suporte de projeção e
metadados</td><td>Funções de suporte
de relacionamento
de
elementos</td></tr>
<tr><td>TAD$_{set}$
TPD$_{set}$</td><td colspan="4">Funções de
suporte de agregação</td><td>Funções de
suporte
de relacionamento
de conjuntos</td></tr>
<tr><td rowspan="10">Binária</td><td rowspan="2">Elementos</td><td>TAD × TAD
TAD × TPD</td><td colspan="4">Funções de suporte
de projeção e combinação</td><td>Funções de suporte
de relacionamento
de elementos</td></tr>
<tr><td rowspan="8">Conjuntos</td><td>TAD$_{set}$ × TAD$_{set}$
TAD$_{set}$ × TAD
TAD$_{set}$ × TPD
TAD$_{set}$ × TPD$_{set}$
TPD$_{set}$ × TAD
TPD × TPD
TPD$_{set}$ × TPD$_{set}$
TPD$_{set}$ × TPD</td><td colspan="4">Funções de suporte
de agregação
e combinação</td><td>Funções de
suporte de
relacionamento
de conjuntos</td></tr>
</table>

Funções de suporte de projeção – Estas funções irão retornar instâncias ou conjuntos de instâncias de TADs e TPDs. O seu parâmetro de entrada, contudo, será sempre em elementos individuais abstratos ou primitivos. Em outras palavras, estas funções irão tomar o valor de cada elemento, processá-los e retornar um elemento ou conjunto correspondente a cada um. Desta forma, caracterizamos estas funções como funções de suporte à projeção

Funções de suporte de metadados – Estas funções são um caso particular das funções de suporte à projeção, pois seus parâmetros de entrada e retorno são os mesmos. Entretanto, a sua semântica difere no sentido em que são usadas para criação e manipulação de TADs (ver seção 5.4.3).

Funções de suporte à agregação – O retorno destas funções se comporta exatamente como o de suporte à projeção, porém recebe conjuntos de elementos e não valores individuais como parâmetros. Torna-se óbvio que, para o retorno de elementos (TAD e TPD), é necessário a agregação dos conjuntos para poder produzir um único elemento a partir de um conjunto. O retorno de conjuntos (TAD_{set} e TPD_{set}) pode ou não produzir elementos agregados. Se tivermos uma diferença de cardinalidade, ou seja, entre o número de elementos do conjunto de entrada e o do conjunto de retorno, a agregação ou expansão é inevitável. De todo modo, mesmo quando a cardinalidade for igual, temos que verificar se as funções receberão todo o conjunto de dados como parâmetro e se o processamento agregado se fará necessário.

Funções de suporte de projeção e combinação – O comportamento destas funções segue o de suporte de projeção, dado que os parâmetros de retorno são os mesmos. A diferença aqui é que, como temos parâmetros binários e o retorno obviamente unário, as funções terão de executar a combinação dos parâmetros de entrada. Esta combinação pode incluir uma conversão de tipos de dados ou somente uma transformação.

Funções de suporte de agregação e combinação – Esta categoria de funções junta as características das funções de suporte à agregação com as de suporte à projeção e combinação. Dado as características de parâmetros de conjuntos explicada acima, agregações quase certamente se farão necessárias. Por outro lado, todos os parâmetros aqui são binários, o que representa a necessidade de combinação de elementos na maioria dos casos.

Funções de suporte de relacionamento de elementos – O retorno de parâmetros do tipo booleano poderia ser tratado como retorno de TPDs, porém - e propositadamente destacado aqui -, este tipo de retorno é a base para a construção das funções de suporte à seleção e junção (seção 5.5.1). Todo este conjunto de categorias está baseado na entrada de parâmetros como instâncias e não conjunto de instâncias.

Funções de suporte de relacionamento de conjuntos – Este conjunto possui todas as características do item anterior, somente adicionando a possibilidade de receber conjuntos como parâmetros de entrada.

CLASSIFICAÇÃO EXAUSTIVA

A Tabela 19 mostra uma classificação exaustiva das funções de suporte ao modelo relacional para dados estendidos. Esta tabela detalha as categorias mostradas na Tabela 18 e foi construída usando todas as possíveis combinações de parâmetros e tipos de retorno mostrados na Tabela 16. Iremos analisar a seguir esta tabela linha por linha.

As funções de agregação foram classificadas separadamente (Tabela 20) para facilitar a compreensão da Tabela 19. As funções de agregação que recebem conjuntos de TADs (primeira e segunda linha da Tabela 20) podem, primeiramente, incluir conversão do tipo de dado. Caso o retorno seja um TPD (segunda linha) está é a única opção. Se for também um TAD ou conjunto de TADs, temos a possibilidade de conversão ou transformação. Ainda nesta última situação, podemos ter funções com parâmetros binários, o que vai implicar em combinação dos TADs, além de possível conversão ou transformação. A última linha da Tabela 20 representa funções de agregação padrão envolvendo somente TPDs. Estas poderão ser de conversão ou transformação, seguindo o mesmo caso das funções (**FSTD**^{n}...).

Tabela 19 - Categorias de funções de suporte relacional

Parametrização		Retorno				
		Elementos		Conjuntos		Booleano
		TAD	TPD	TAD_{set}	TPD_{set}	
Unária	TAD	**FPST1**$^{n}_{\text{Transformação}}$ **FPSC1**$^{n}_{\text{Conversão}}$ **FCA**$^{n}_{\text{Acesso}}$ **FCP**$^{n}_{\text{Propriedade}}$	**FPSC1**$^{n}_{\text{Conversão}}$ **FCA**$^{n}_{\text{Acesso}}$ **FCP**$^{n}_{\text{Propriedade}}$	**FPST1**$^{n}_{\text{Transformação}}$ **FPSC1**$^{n}_{\text{Conversão}}$ **FPSD**$^{n}_{\text{DrillDown}}$	**FPSC1**$^{n}_{\text{Conversão}}$	**FRS1**$^{n}_{\text{Relação}}$
	TPD	**FCSC**$^{n}_{\text{Criação}}$	**FPSC2**$^{n}_{\text{Conversão}}$ **FPST2**$^{n}_{\text{Transformação}}$	**FCSC**$^{n}_{\text{Criação}}$	**FPSC2**$^{n}_{\text{Conversão}}$ **FPST2**$^{n}_{\text{Transformação}}$	**FRSTD**$^{n}_{\text{Relação}}$
	TAD_{set}	**FA11**$^{n}_{\text{Agregação}}$	**FA1**$^{n}_{\text{Agregação}}$	**FA12**$^{n}_{\text{Agregação}}$	**FA2**$^{n}_{\text{Agregação}}$	**FRM1**$^{n}_{\text{Relação}}$
	TPD_{set}	**FCMC**$^{n}_{\text{Criação}}$	**FASTD1**$^{n}_{\text{Agregação}}$	**FCMC**$^{n}_{\text{Criação}}$	**FASTD2**$^{n}_{\text{Agregação}}$	**FRMSTD**$^{n}_{\text{Relação}}$
Binária de Elementos	TAD×TAD	**FPSX**$^{n}_{\text{Combinação}}$	**FPSC1**$^{n}_{\text{Conversão}}$	**FPSD**$^{n}_{\text{DrillDown}}$	**FPSC1**$^{n}_{\text{Conversão}}$	**FRS2**$^{n}_{\text{Relação}}$
	TAD×TPD	**FPST1**$^{n}_{\text{Transformação}}$ **FPSC1NCONVERSÃO** **FCA**$^{n}_{\text{Acesso}}$ **FCP**$^{n}_{\text{Propriedade}}$	**FPSC1**$^{n}_{\text{Conversão}}$ **FCA**$^{n}_{\text{Acesso}}$ **FCP**$^{n}_{\text{Propriedade}}$	**FPST1**$^{n}_{\text{Transformação}}$ **FPSC1**$^{n}_{\text{Conversão}}$ **FPSD**$^{n}_{\text{DrillDown}}$	**FPSC1**$^{n}_{\text{Conversão}}$	**FRS3**$^{n}_{\text{Relação}}$
Binária de Conjuntos	TAD_{set}×TAD_{set}	**FA21**$^{n}_{\text{Agregação}}$	**FA1**$^{n}_{\text{Agregação}}$	**FA22**$^{n}_{\text{Agregação}}$	**FA2**$^{n}_{\text{Agregação}}$	**FRM2**$^{n}_{\text{Relação}}$
	TAD_{set}×TAD	**FA21**$^{n}_{\text{Agregação}}$	**FA1**$^{n}_{\text{Agregação}}$	**FA22**$^{n}_{\text{Agregação}}$	**FA2**$^{n}_{\text{Agregação}}$	**FRM2**$^{n}_{\text{Relação}}$
	TAD_{set}×TPD	**FA11**$^{n}_{\text{Agregação}}$	**FA1**$^{n}_{\text{Agregação}}$	**FA12**$^{n}_{\text{Agregação}}$	**FA2**$^{n}_{\text{Agregação}}$	**FRM3**$^{n}_{\text{Relação}}$
	TAD_{set}×TPD_{set}	**FA11**$^{n}_{\text{Agregação}}$	**FA1**$^{n}_{\text{Agregação}}$	**FA12**$^{n}_{\text{Agregação}}$	**FA2**$^{n}_{\text{Agregação}}$	**FRM3**$^{n}_{\text{Relação}}$
	TPD_{set}×TAD	**FPMT**$^{n}_{\text{Transformação}}$ **FPMC**$^{n}_{\text{Conversão}}$	**FPMC**$^{n}_{\text{Conversão}}$	**FPMT**$^{n}_{\text{Transformação}}$ **FPMC**$^{n}_{\text{Conversão}}$ **FPMD**$^{n}_{\text{DrillDown}}$	**FPMC**$^{n}_{\text{Conversão}}$	**FRM3**$^{n}_{\text{Relação}}$
	TPD × TPD	**FCSC**$^{n}_{\text{Criação}}$	**FPSC2**$^{n}_{\text{Conversão}}$ **FPST2**$^{n}_{\text{Transformação}}$	**FCSC**$^{n}_{\text{Criação}}$	**FPSC2**$^{n}_{\text{Conversão}}$ **FPST2**$^{n}_{\text{Transformação}}$	**FRSTD**$^{n}_{\text{Relação}}$
	TPD_{set}×TPD_{set}	**FCMC**$^{n}_{\text{Criação}}$	**FASTD1**$^{n}_{\text{Agregação}}$	**FCMC**$^{n}_{\text{Criação}}$	**FASTD2**$^{n}_{\text{Agregação}}$	**FRMSTD**$^{n}_{\text{Relação}}$
	TPD_{set}×TPD	**FCMC**$^{n}_{\text{Criação}}$	**FASTD1**$^{n}_{\text{Agregação}}$	**FCMC**$^{n}_{\text{Criação}}$	**FASTD2**$^{n}_{\text{Agregação}}$	**FRMSTD**$^{n}_{\text{Relação}}$

Tabela 20 - Categoria de funções de agregação

	Parâmetro de Entrada	Operações(*X*)		Retorno(*Y*)	
FA*XY*$^{n}_{\text{Agregação}}$	TAD_{set}	**1**	- Conversão - Transformação	**1**	TAD
		2	**1** + Combinação	**2**	TAD_{set}
FA*Y*$^{n}_{\text{Agregação}}$	TAD_{set}	- Conversão		**1**	TPD
				2	TPD_{set}
FASTD*Y*$^{n}_{\text{Agregação}}$	TPD_{set}	- Conversão - Transformação		**1**	TPD
				2	TPD_{set}

MÉTODO DE CLASSIFICAÇÃO

Vamos descrever abaixo a metodologia de classificação de cada linha da Tabela 19:

	TAD	TPD	TAD_{set}	TPD_{set}	Booleano
TAD	**FPST1**$^{n}_{\text{Transformação}}$ **FPSC1**$^{n}_{\text{Conversão}}$	**FPSC1**$^{n}_{\text{Conversão}}$	**FPST1**$^{n}_{\text{Transformação}}$ **FPSC1**$^{n}_{\text{Conversão}}$ **FPSD**$^{n}_{\text{DrillDown}}$	**FPSC1**$^{n}_{\text{Conversão}}$	**FRS1**$^{n}_{\text{Relação}}$
	FCSA$^{n}_{\text{Acesso}}$ **FCSP**$^{n}_{\text{Propriedade}}$	**FCSA**$^{n}_{\text{Acesso}}$ **FCSP**$^{n}_{\text{Propriedade}}$		**FCMA**$^{n}_{\text{Acesso}}$ **FCMP**$^{n}_{\text{Propriedade}}$	

Primeira Linha - A primeira e a segunda linha da tabela representam funções que recebem como parâmetro um único elemento. Na primeira, as funções recebem de um TAD e na segunda um TPD. Este parâmetro é passado para as funções a cada chamada das mesmas. Por causa disso, nesta linha, apenas as funções da categoria **FPS_**$^{n}_{\text{<subcategoria>}}$ e **FRS_**$^{n}_{\text{Relação}}$ se aplicam.

Na primeira célula, as funções que recebem o TAD retornam um outro elemento de TAD. Se este TAD for do mesmo tipo do parâmetro recebido, então teremos uma **Transformação**, caso contrário, teremos uma **Conversão**. Esta combinação pode ainda permitir uma outra categoria de funções, os *accessors*. Estas funções retornam partes, atributos ou propriedades do TAD, , quando um TAD é composto de outros TADs, neste caso especificamente.

Na segunda célula, como o retorno desta categoria de funções é um TPD, e obviamente de domínio diferente do TAD, as funções só poderão ser de conversão ou, quando o TAD é composto de TPDs.

A terceira célula desta primeira linha retorna conjuntos de instâncias de TADs a partir de uma única instância de um TAD. Havendo ou não mudança no tipo de dado,

teremos conversão ou transformação, respectivamente. A característica de aumento de cardinalidade de elemento para conjunto levanta a subcategoria de função de *drilldown*. Este tipo de função seria capaz de aumentar o nível de detalhamento associado a cada elemento de uma *tupla*.

A quarta célula ganha a classe de funções de conversão, dado que o seu retorno são de TPDs com parâmetro de TADs. O *drilldown* não se aplica aqui, pois semanticamente o detalhamento está associado a uma mesma categoria de tipo de dado, ou seja, TAD para TAD.

A quinta célula vai representar todas as funções cujo retorno é lógico e recebem como parâmetro uma instância de um TAD. Esta categoria específica (**FRS1**$^{n}_{\text{Relação}}$) não faz verdadeiramente um relacionamento entre elementos de duas relações, porém realiza testes de relacionamento interno dentro do TAD. Um exemplo deste tipo de função seria testar se um TAD está vazio ou inválido.

	TAD	TPD	TAD_{set}	TPD_{set}	**Booleano**
TPD		**FPSC2**$^{n}_{\text{Conversão}}$ **FPST2**$^{n}_{\text{Transformação}}$		**FPSC2**$^{n}_{\text{Conversão}}$ **FPST2**$^{n}_{\text{Transformação}}$	**FRSTD**$^{n}_{\text{Relação}}$
	FCSC$^{n}_{\text{Criação}}$		**FCSC**$^{n}_{\text{Criação}}$		

Segunda Linha - Na primeira célula da segunda linha, temos a conversão de um TPD em um TAD. Isto sugere que um TAD foi construído a partir de elementos primitivos básicos (TPDs), como não poderia deixar de ser para qualquer processo de criação inicial de um TAD. As funções relacionadas à conversão de TPDs em TADs pertencem à subcategoria **FCSC**$^{n}_{\text{Criação}}$, dado que o parâmetro neste caso é um valor de elemento de uma *tupla*.

Na segunda e quarta células, temos a possível conversão ou transformação entre TPDs, não envolvendo nenhum dos tipos de dados estendidos de *data warehouse*. Desta forma, classificamos estas funções como funções "padrão", que recebem valores de elementos de *tupla* (**FPSC2**^{n} e **FPST2**^{n}). Na quarta célula temos a possibilidade de *drilldown* padrão, ou seja, entre TPDs, devido ao possível aumento de cardinalidade de TPDs para TPD_{set}.

Na terceira célula, as funções de criação de TADs novamente se aplicam, com a diferença de que aqui a mesma gera conjuntos de TADs a partir de valores de

TPDs. Podemos imaginar estas funções como métodos de criação de TADs em demanda.

Na quinta célula da segunda linha, temos funções de relacionamento que envolvem apenas TPDs, ou seja, funções de suporte à seleção e junção convencionais.

	TAD	TPD	TAD_{set}	TPD_{set}	**Booleano**
TAD_{set}	**FA11**$^{n}_{\text{Agregação}}$	**FA1**$^{n}_{\text{Agregação}}$	**FA12**$^{n}_{\text{Agregação}}$	**FA2**$^{n}_{\text{Agregação}}$	**FRM1**$^{n}_{\text{Relação}}$

Terceira Linha - Na terceira linha da Tabela 19, nos deparamos com um parâmetro de entrada em forma de conjunto de TADs. Os parâmetros de conjunto requerem agregações conjuntamente a outras operações. A Tabela 20 mostra as possíveis combinações das funções de agregação com as outras subcategorias já discutidas acima.

A primeira e a terceira célula da terceira linha da Tabela 19 apresentam a mesma configuração da primeira linha da Tabela 20, onde um conjunto de TADs irá produzir outros TADs. As subcategorias implicitamente contidas na agregação são, portanto, de conversão, quando o TAD de retorno diferente do TAD do conjunto de entrada, ou transformação, caso contrário. A diferença entre a primeira e a terceira células é somente o tipo de retorno, de individual para conjunto, respectivamente. A segunda e a quarta células produzirão uma conversão com certeza, devido à mudança do parâmetro de entrada de TAD para o retorno de TPD, sendo que, na quarta célula, os TPDs aparecem em conjunto.

Na quinta célula da terceira linha, temos a categoria de funções de relacionamento similar a da primeira linha, só que aqui os parâmetros são conjuntos de TADs. Nesta subcategoria, podemos ter funções que avaliem relacionamentos entre os elementos do conjunto de TADs.

	TAD	TPD	TAD_{set}	TPD_{set}	**Booleano**
TPD_{set}		**FASTD1**$^{n}_{\text{Agregação}}$		**FASTD2**$^{n}_{\text{Agregação}}$	**FRMSTD**$^{n}_{\text{Relação}}$
	FCMC$^{n}_{\text{Criação}}$		**FCMC**$^{n}_{\text{Criação}}$		

Quarta Linha - Na quarta linha, assim como nas duas últimas, da Tabela 19, temos novamente um conjunto como parâmetro de entrada, contudo ele não é de um TAD. Logo, as únicas agregações que serão feitas nesta linha são na segunda e quarta células, seguindo o padrão de agregação da terceira linha da Tabela 20.

A primeira e a terceira células da quarta linha são similares às mesmas da segunda linha, onde um TPD se converte em um TAD. Parâmetros de tipos primitivos estão se convertendo em um tipo abstrato, desta forma temos novamente funções de criação de TADs. A diferença encontrada nesta quarta linha está nos parâmetros de conjunto que nos levam às funções da categoria (**FCMC**$^{n}_{\ldots}$). Na quinta célula da quarta linha, temos uma função de relacionamento padrão para tipos primitivos, similar a da segunda linha, só que utilizando agora como parâmetro de entrada um conjunto de TPDs. A única diferença entre a quarta linha e as duas últimas linhas da Tabela 19 são os parâmetros binários e compostos com conjuntos. Entretanto, não consideramos aqui a combinação entre TPDs e as agregações padrão de TPDs com transformações e conversões, já especificadas na Tabela 20.

	TAD	TPD	TAD_{set}	TPD_{set}	**Booleano**
TAD × TAD	**FPSX**$^{n}_{\text{Combinação}}$	**FPSC1**$^{n}_{\text{Conversão}}$	**FPSD**$^{n}_{\text{DrillDown}}$	**FPSC1**$^{n}_{\text{Conversão}}$	**FRS2**$^{n}_{\text{Relação}}$
TAD × TPD	**FPST1**$^{n}_{\text{Transformação}}$ **FPSC1**$^{n}_{\text{Conversão}}$	**FPSC1**$^{n}_{\text{Conversão}}$	**FPST1**$^{n}_{\text{Transformação}}$ **FPSC1**$^{n}_{\text{Conversão}}$ **FPSD**$^{n}_{\text{DrillDown}}$	**FPSC1**$^{n}_{\text{Conversão}}$	**FRS3**$^{n}_{\text{Relação}}$
	FCSA$^{n}_{\text{Acesso}}$ **FCSP**$^{n}_{\text{Propriedade}}$	**FCSA**$^{n}_{\text{Acesso}}$ **FCSP**$^{n}_{\text{Propriedade}}$		**FCMA**$^{n}_{\text{Acesso}}$ **FCMP**$^{n}_{\text{Propriedade}}$	

Quinta e Sexta Linhas - A quinta e sexta linhas da Tabela 19 apresentam as categorias de funções com parâmetros binários, mas apenas de elementos individuais. Na primeira célula da quinta linha, temos como parâmetros de entrada dois TADs e como resultado um TAD. Desta forma, temos funções que combinam

os dois TADs (Combinação), convertem o tipo de dado (Conversão) ou mantém o tipo de dado de um dos dois operandos (Transformação). Na primeira célula da sexta linha, as funções de combinação são substituídas pela Transformação e Conversão explícitas. Dado que um dos parâmetros não é um TAD, a combinação não pode ser feita. Mas a conversão poderá ocorrer se o TAD de retorno for de tipo diferente do parâmetro de entrada. Dentro dos objetivos deste trabalho, consideramos como base primária das funções os parâmetros de TAD.

Podemos também ter *accessors* na sexta linha, considerando que o TPD pode ser um parâmetro de sintonia para obtenção de algum atributo interno do TAD.

Na segunda, terceira e quarta células, temos características similares nas categorias das funções. Na segunda e quarta células, similarmente à primeira linha, os parâmetros de entrada são TADs e os de saída TPDs, o que estabelece claramente uma conversão. Temos uma diferença apenas na quarta célula, onde a função retorna conjuntos de TPDs. Na terceira célula, o *drilldown* tem as mesmas características da primeira linha, mas, neste caso, possíveis operações de combinação podem estar inclusas.

Na quinta célula, temos as funções que farão parte do suporte a operações de junção, assim como de seleção. Na quinta linha, as funções de relacionamento $\mathbf{FRS2}^{n}_{\text{Relação}}(...)$ serão responsáveis pela junção de elementos de duas relações, considerando características de relacionamentos específicas dos tipos de dados estendidos do modelo de *data warehouse*. O conceito de igualdade especializado para estes TADs será implementado aqui. Na sexta linha, as funções de relacionamento $\mathbf{FRS3}^{n}_{\text{Relação}}(...)$ também poderão ser usadas para operações de junção, mas dado que um de seus parâmetros é um TPD, será mais comum existirem funções de suporte à seleção dentro de uma relação. O TPD seria um parâmetro associado ao TAD. Podemos supor, por exemplo, uma função que retorne verdadeiro para todos os TADs menores que um certo tamanho (TPD).

	TAD	TPD	TAD_{set}	TPD_{set}	**Booleano**
$TAD_{set} \times TAD_{set}$	**FA21**$^{n}_{\text{Agregação}}$	**FA1**$^{n}_{\text{Agregação}}$	**FA22**$^{n}_{\text{Agregação}}$	**FA2**$^{n}_{\text{Agregação}}$	**FRM2**$^{n}_{\text{Relação}}$
$TAD_{set} \times TAD$	**FA21**$^{n}_{\text{Agregação}}$	**FA1**$^{n}_{\text{Agregação}}$	**FA22**$^{n}_{\text{Agregação}}$	**FA2**$^{n}_{\text{Agregação}}$	**FRM2**$^{n}_{\text{Relação}}$
$TAD_{set} \times TPD$	**FA11**$^{n}_{\text{Agregação}}$	**FA1**$^{n}_{\text{Agregação}}$	**FA12**$^{n}_{\text{Agregação}}$	**FA2**$^{n}_{\text{Agregação}}$	**FRM3**$^{n}_{\text{Relação}}$
$TAD_{set} \times TPD_{set}$	**FA11**$^{n}_{\text{Agregação}}$	**FA1**$^{n}_{\text{Agregação}}$	**FA12**$^{n}_{\text{Agregação}}$	**FA2**$^{n}_{\text{Agregação}}$	**FRM3**$^{n}_{\text{Relação}}$

Sétima a Décima Linhas - A análise da sétima à décima linha da Tabela 19 vai mostrar o mesmo padrão da terceira linha da tabela. A razão disto não é coincidência, mas o fato de todas estas linhas receberem como parâmetros conjuntos de TADs. Esta característica reforça a necessidade de agregarmos o resultado, assim todas as células irão executar algum tipo de agregação definida na Tabela 20. Na primeira e terceira células da sétima e oitava linhas, a existência de dois parâmetros de TAD leva à possibilidade de combinação (funções do tipo **FA2**_$^{n}_{\text{Agregação}}$ da Tabela 20). Nas mesmas células das linhas nove e dez das, temos apenas um parâmetro de conjunto de TAD, logo as funções de agregação só adicionam a possibilidade de Conversão ou Transformação. As diferenças entre a primeira e segunda células, assim como a terceira e quarta células, é o efeito de retorno de TPD, o que obrigatoriamente nos leva a funções de conversão.

Todas as funções de retorno condicional (quinta célula) da sétima à décima linha são do tipo de parâmetro de conjunto (**FRM**_$^{n}_{\text{Relação}}$). A categoria de funções da sétima e oitava linhas recebe dois conjuntos de TADs como parâmetros, e realizará seleções ou junções baseada no relacionamento do produto cartesiano entre todos os elementos dos conjuntos de TADs. A função de relacionamento da nona e décima linhas vai apenas testar um elemento de TAD contra todos os outros do conjunto, aceitando uma possível parametrização por TPDs, como, por exemplo, selecionar o elemento mais similar dentro de certos limites. Esta categoria de funções será mais usada para seleção de elementos internos à relação.

	TAD	TPD	TAD_{set}	TPD_{set}	**Booleano**
TPD_{set} × TAD	**FPMT**$^{n}_{\text{Transformação}}$ **FPMC**$^{n}_{\text{Conversão}}$	**FPMC**$^{n}_{\text{Conversão}}$	**FPMT**$^{n}_{\text{Transformação}}$ **FPMC**$^{n}_{\text{Conversão}}$ **FPMD**$^{n}_{\text{DrillDown}}$	**FPMC**$^{n}_{\text{Conversão}}$	**FRM3**$^{n}_{\text{Relação}}$

Décima Primeira Linha - A décima primeira linha vai mostrar características semelhantes à sexta. Nesta última, temos elementos individuais de TADs contra parâmetros TPDs. Na décima primeira linha, a diferença é que os parâmetros são conjuntos de TPDs. A conseqüência disso é tornar todas as funções do tipo **F_<S>**$^{n}_{\text{<subcategoria>}}$ no tipo **F_<M>**$^{n}_{\text{<subcategoria>}}$. As funções de retorno condicional também se comportarão da mesma forma que na nona e décima linhas.

	TAD	TPD	TAD_{set}	TPD_{set}	**Booleano**
TPD × TPD		**FPSC2**$^{n}_{\text{Conversão}}$ **FPST2**$^{n}_{\text{Transformação}}$		**FPSC2**$^{n}_{\text{Conversão}}$ **FPST2**$^{n}_{\text{Transformação}}$	**FRSTD**$^{n}_{\text{Relação}}$
	FCSC$^{n}_{\text{Criação}}$		**FCSC**$^{n}_{\text{Criação}}$		

Décima Segunda Linha – Os parâmetros de entrada das funções desta linha são TPDs, o que estabelece que as funções de retorno booleano (quinta célula) e de retorno de TPDs (segunda e quarta células) serão convencionais, não envolvendo TADs. Neste último caso, podemos ter conversão de tipos entre TPDs ou simples alteração de valores ou transformação. Por outro lado, Na primeira e terceira células, o retorno é de TADs, o que nos conduz a funções de criação de TADs, como já descrito na quarta linha.

	TAD	TPD	TAD_{set}	TPD_{set}	**Booleano**
TPD_{set} × TPD_{set}		**FASTD1**$^{n}_{\text{Agregação}}$		**FASTD2**$^{n}_{\text{Agregação}}$	**FRMSTD**$^{n}_{\text{Relação}}$
	FCMC$^{n}_{\text{Criação}}$		**FCMC**$^{n}_{\text{Criação}}$		
TPD_{set} × TPD		**FASTD1**$^{n}_{\text{Agregação}}$		**FASTD2**$^{n}_{\text{Agregação}}$	**FRMSTD**$^{n}_{\text{Relação}}$
	FCMC$^{n}_{\text{Criação}}$		**FCMC**$^{n}_{\text{Criação}}$		

Décima Terceira e Décima Quarta Linhas – Estas linhas acompanham as características da quarta linha, como já explicado acima.

As linhas 1, 2, 4, 6, 12, 13 e 14 apresentam funções de projeção com uma semântica diferenciada, pois são usadas para criação e manipulação dos TADs. Nós as destacamos do conjunto total de funções (Tabela 21) e as inserimos na seção de suporte de metadados (seção 5.4.3). Esta categoria de funções é usada para criação de TADs ou acesso às suas propriedades, apesar de também poderem funcionar como funções de projeção.

Tabela 21 – Classificação de funções de manipulação de dados

	Retorno				
	Elementos		**Conjuntos**		**Booleano**
Parâmetros	TAD	TPD	TAD_{set}	TPD_{set}	
TAD	**FCSA**$^{n}_{Acesso}$ **FCSP**$^{n}_{Propriedade}$	**FCSA**$^{n}_{Acesso}$ **FCSP**$^{n}_{Propriedade}$		**FCMA**$^{n}_{Acesso}$ **FCMP**$^{n}_{Propriedade}$	
TPD	**FCSC**$^{n}_{Criação}$		**FCSC**$^{n}_{Criação}$		
TAD_{set}					
TPD_{set}	**FCMC**$^{n}_{Criação}$		**FCMC**$^{n}_{Criação}$		
TAD×TAD					
TAD×TPD	**FCSA**$^{n}_{Acesso}$ **FCSP**$^{n}_{Propriedade}$	**FCSA**$^{n}_{Acesso}$ **FCSP**$^{n}_{Propriedade}$		**FCMA**$^{n}_{Acesso}$ **FCMP**$^{n}_{Propriedade}$	
TAD_{set}×TAD_{set}					
TAD_{set}×TAD					
TAD_{set}×TPD					
TAD_{set}×TPD_{set}					
TPD_{set}×TAD					
TPD × TPD	**FCSC**$^{n}_{Criação}$		**FCSC**$^{n}_{Criação}$		
TPD_{set}×TPD_{set}	**FCMC**$^{n}_{Criação}$		**FCMC**$^{n}_{Criação}$		
TPD_{set}×TPD	**FCMC**$^{n}_{Criação}$		**FCMC**$^{n}_{Criação}$		

Mostramos aqui a metodologia de classificação para a listagem exaustiva das funções de suporte relacional e metadados. Uma descrição detalhada com exemplos já foi mostrada na seção 4.3.2 do Capítulo 4.

Esta classificação é a base de quase todas as funcionalidades do sistema de *data warehousing* estendido para TADs. Através desta classificação podemos especificar todo o suporte relacional e de metadados que será usado como base para os procedimentos de transformação e análise,mostrados na seção seguinte.

5.6– Seção E - Funcionalidades de *Data Warehousing* para TADs

As funcionalidades do processo de *data warehouse* propriamente dito se fundamentarão em várias das funções embutidas no banco de dados. Contudo, existem vários grupos de procedimentos e funcionalidades que pertencem a um conjunto mais diverso de padrões. Por isso, sua classificação não é tão precisa.

5.6.1 – Extração

O processo de extração de dados em *data warehousing* (seção 2.2.1) é o mais diverso com respeito à forma e à semântica. A razão disto é sua função de converter dados nos mais diversos formatos e estruturas para o formato e estrutura do banco de dados de *data warehousing*. Existem duas etapas neste processo: a primeira apenas traz os dados externos para dentro do banco de *data warehousing*; a segunda realiza pré-processamentos necessários para acomodar os dados no banco.

A classificação que utilizamos para as funcionalidades de extração leva em conta a forma de entrada e saída dos dados, bem como a sua semântica básica. A Tabela 22 mostra a classificação, onde podemos verificar dois tipos básicos de conversão: de um meio externo para o banco ou de um banco externo para o banco de dados de *data warehousing*. A classificação quanto ao uso deve ser feita através da análise da semântica das operações.

Tabela 22 - Categorias de funções de extração

Fonte	Destino - Banco 1
	Lista de Tabelas
Componente Externo	**PEI**$^{n}_{\text{Importação}}$(***CompExt***, ***TabList***$^{1}_{\text{destino}}$[,Parâmetros...])
	PEI$^{n}_{\text{Agregação}}$(***CompExt***, ***TabList***$^{1}_{\text{destino}}$[,Parâmetros...])
	PEI$^{n}_{\text{Sequencializador}}$(***CompExt***, ***TabList***$^{1}_{\text{destino}}$[,Parâmetros...])
Banco 1	**PEP**$^{n}_{\text{Pré-processamento}}$(***TabList***$^{1}_{\text{fonte}}$, ***TabList***$^{1}_{\text{destino}}$[,Parâmetros...])
	PEP$^{n}_{\text{Limpeza}}$(***TabList***$^{1}_{\text{fonte}}$, ***TabList***$^{1}_{\text{destino}}$[,Parâmetros...])
	PEP$^{n}_{\text{Validação}}$(***TabList***$^{1}_{\text{fonte}}$, ***TabList***$^{1}_{\text{destino}}$[,Parâmetros...])
Banco 2	**PEP**$^{n}_{\text{Pré-processamento}}$(***TabList***$^{2}_{\text{fonte}}$, ***TabList***$^{1}_{\text{destino}}$[,Parâmetros...])
	PEP$^{n}_{\text{Limpeza}}$(***TabList***$^{2}_{\text{fonte}}$, ***TabList***$^{1}_{\text{destino}}$[,Parâmetros...])
	PEP$^{n}_{\text{Validação}}$(***TabList***$^{2}_{\text{fonte}}$, ***TabList***$^{1}_{\text{destino}}$[,Parâmetros...])

IMPORTAÇÃO

O processo de importação dos dados para dentro do banco de *data warehousing* pode receber dois tipos de fonte de dados:arquivos, *web* etc.,; ou tabelas de um outro

banco de dados. Coletamos três possíveis semânticas principais (seção 2.2.1) baseadas na literatura:

PEI$^{n}_{\text{Importação}}$ – Procedimentos para importação, considerando arquivos externos ou tabelas de um outro banco de dados.

PEA$^{n}_{\text{Agregação}}$ – Procedimentos que realizam agregações nos dados antes de inseri-los no banco, como somar a conta de diversos pacientes de um hospital e inserir somente os valores consumidos de material por médico.

PES$^{n}_{\text{Sequencializador}}$ – Procedimentos que realizam a concatenação serializada de dados em uma certa ordem. Quando temos dados espalhados em múltiplos arquivos, sem a consistência do modelo relacional, a ordem de inserção destes arquivos no banco é relevante, porque uma inserção pode depender de um registro que ainda não foi inserido.

PRÉ-PROCESSAMENTO

Uma vez inseridos os dados no banco de *data warehouse,* podemos necessitar de manipulações para integrá-los, cujas possíveis semânticas são:

- **PEP**$^{n}_{\text{Pré-processamento}}$ – Procedimentos de pré-processamento geral, dependente do tipo de dado, bem como da sua origem.
- **PEL**$^{n}_{\text{Limpeza}}$ – Procedimentos de limpeza (*Cleansing*), consistindo especificamente em homogeneizar os dados de forma que cada valor tenha apenas uma representação.
- **PEV**$^{n}_{\text{Validação}}$ – A validação consiste em verificar se os valores dos dados estão consistentes com o seu domínio, modelo de dados ou sua regra de formação. Por exemplo, em um TAD do tipo "som", exigimos que os valores de freqüência sejam superiores a faixa audível.

5.6.2 – Transformação

As funcionalidades de transformação irão se apoiar nas funções de suporte relacional para realizar os vários processamentos entre as tabelas do banco de *data warehousing* e para produzir as análises desejadas, ou então carregar os subsistemas de análise multidimensional.

As funções de transformação podem recair, de maneira geral, nos dois formatos abaixo. O primeiro representa uma função especializada com um algoritmo específico.

A segunda, mais geral, recebe uma sentença SQL como parâmetro de filtragem para a transformação.

PTG$^{n}_{\text{Transformação}}$(***TabList***$_{\text{fonte}}$, ***TabList***$_{\text{destino}}$[,DW$_{\text{Lista de Parâmetros}}$])

PTG$^{n}_{\text{Transformação}}$(***TabList***$_{\text{fonte}}$, ***TabList***$_{\text{destino}}$, ***SQL$_{list}$*** [,DW$_{\text{Lista de Parâmetros}}$])

A despeito dos formatos gerais acima, selecionamos um subconjunto dos principais tipos de transformações de acordo com as referências descritas na seção 2.2.2. Estas funções irão se basear nas funções de suporte relacional já descritas na seção 5.5.2, e seu formato geral vai sempre indicar a função principal que as suporta:

PT<**S**|**M**><Subcategoria>(***TabList***$_{\text{fonte}}$, ***TabList***$_{\text{destino}}$, **F_**$^{k}_{\text{<semântica>}}$ [,Parâmetros]),

a nomenclatura detalhada para cada subcategoria pode ser encontrada no Apêndice A.

A indicação da função de suporte não significa que a real implementação da função irá utilizá-la. Por motivos tecnológicos ou de desempenho, pode ser interessante criar uma função de transformação especializada para cada uma das funções abaixo:

Seleção

Separação e Normalização

Concatenação e Desnormalização

Sumarização

Transformação

Enriquecimento

Conversão

DrillDown

MÉTODO DE CLASSIFICAÇÃO

A classificação das funcionalidades, ou procedimentos, de transformação citadas requer a análise da semântica específica de cada uma, e, depois, uma categorização através dos tipos de parâmetro e retorno dos dados. Baseado nisto, buscamos, na Tabela 19 da seção 5.5.2, pelas células correspondentes aos parâmetros e tipos de retorno e construímos a Tabela 23. Esta última será detalhada na Tabela 24, que mostrará a construção dos procedimentos de transformação.

Tabela 23 - Categoria de funcionalidades de transformação

Parametrização		Retorno				
		Elementos		Conjuntos		Booleano
		TAD	TPD	TAD_{set}	TPD_{set}	
Unária	TAD	Separação Transformação Enriquecimento	Separação Conversão	DrillDown	Conversão	Seleção
	TPD	Enriquecimento		Enriquecimento		
	TAD_{set}	Sumarização	Separação Conversão	Transformação Enriquecimento Separação	Conversão	
	TPD_{set}	Enriquecimento		Enriquecimento		
Binária de Elementos	TAD×TAD	Concatenação	Conversão	DrillDown	Conversão	
	TAD×TPD	Transformação				
Binária de Conjuntos	TAD_{set}×TAD_{set}	Concatenação Sumarização		Concatenação Sumarização		
	TAD_{set}×TAD					
	TAD_{set}×TPD	Sumarização		Sumarização Transformação Enriquecimento Separação DrillDown		
	TAD_{set}×TPD_{set}					
	TPD_{set}×TAD	Separação Transformação Enriquecimento	Separação Conversão	DrillDown		
	TPD × TPD	Enriquecimento		Enriquecimento		
	TPD_{set}×TPD_{set}					
	TPD_{set}×TPD					

Tabela 24 - Classificação de funções de transformação

Parametrização		Retorno: Elementos: TAD	Retorno: Elementos: TPD	Retorno: Conjuntos: TAD_{set}	Retorno: Conjuntos: TPD_{set}	Retorno: Booleano
Unária	TAD	**PTSS[1/2]**^{n} Separação **PTST1**^{n} Transformação **PTSE1**^{n} Enriquecimento	**PTSS1**^{n} Separação **PTSS2**^{n} Separação **PTSC1**^{n} Conversão	**PTSD**^{n} *DrillDown*	**PTSC1**^{n} Conversão	**PTSR**^{n} Seleção
	TPD	**PTSE2**^{n} Enriquecimento		**PTSE2**^{n} Enriquecimento		
	TAD_{set}	**PTSA1**^{n} Sumarização	**PTMS**^{n} Separação **PTSC2**^{n} Conversão	**PTST2**^{n} Transformação **PTME1**^{n} Enriquecimento **PTMS2**^{n} Separação	**PTMC2**^{n} Conversão **PTMS3**^{n} Separação	**PTMR**^{n} Seleção
	TPD_{set}	**PTME2**^{n} Enriquecimento		**PTME2**^{n} Enriquecimento		
Binária de Elementos	TAD×TAD	**PTSU**^{n} Concatenação	**PTSC1**^{n} Conversão	**PTSD**^{n} *DrillDow*	**PTSC1**^{n} Conversão	**PTSR**^{n} Seleção
	TAD×TPD	**PTST1**^{n} Transformação				
Binária de Conjuntos	TAD_{set}×TAD_{set}	**PTMU**^{n} Concatenação	**PTSC2**^{n} Conversão	**PTMU**^{n} Concatenação	**PTMC2**^{n} Conversão **PTMS3**^{n} Separação	**PTMR**^{n} Seleção
	TAD_{set}×TAD	**PTSA2**^{n} Sumarização		**PTMA2**^{n} Sumarização		
	TAD_{set}×TPD	**PTSA1**^{n} Sumarização		**PTMA1**^{n} Sumarização **PTME1**^{n} Enriquecimento **PTMT2**^{n} Transformação **PTMS2**^{n} Separação **PTMD**^{n} *DrillDown*		
	TAD_{set}×TPD_{set}					
	TPD_{set}×TAD	**PTMS1**^{n} Separação **PTMT1**^{n} Transformação **PTME3**^{n} Enriquecimento	**PTMS1**^{n} Separação **PTMC1**^{n} Conversão	**PTMD**^{n} *DrillDown*	**PTMC1**^{n} Conversão	
	TPD× TPD	**PTME2**^{n} Enriquecimento		**PTME2**^{n} Enriquecimento		
	TPD_{set}×TPD_{set}					
	TPD_{set}×TPD					

SELEÇÃO

Semântica Específica: Vai produzir um conjunto de elementos selecionado a partir de uma condição aplicada sobre os parâmetros. O resultado das funções de suporte será sempre booleano, desta forma, os procedimentos de transformação por seleção irão se basear em todas as funções de seleção da Tabela 19.

Retorno: Booleano **Parâmetros**: Todas as combinações.

PTSR$^{n}_{\text{Seleção}}$(***TabList***$_{\text{fonte}}$, ***TabList***$_{\text{destino}}$, **FRS_**$^{k}_{\text{Relação}}$ [,DW$_{\text{Lista de Parâmetros}}$])

PTMR$^{n}_{\text{Seleção}}$(***TabList***$_{\text{fonte}}$, ***TabList***$_{\text{destino}}$, **FRM_**$^{k}_{\text{Relação}}$ [,DW$_{\text{Lista de Parâmetros}}$])

SEPARAÇÃO E NORMALIZAÇÃO

Semântica Específica: Extração de partes dos TADs. De acordo com a semântica da separação, um TAD pode se "partir" em dois outros TADs de tipos diferentes, ou em um TAD e um TPD, considerando também os retornos em forma de conjuntos. As funções que podem dar suporte a estes procedimentos são as de conversão ou as de acesso a propriedades internas do TAD. Neste último caso, podemos imaginar um TAD se dividindo em suas propriedades atômicas.

Retorno: TAD **Parâmetros:** TAD, ou TAD $\times$ TPD$_{\text{set}}$

Retorno: TPD **Parâmetros:** TAD, ou TAD $\times$ TPD$_{\text{set}}$

Retorno: TAD$_{\text{set}}$ **Parâmetros:** TAD$_{\text{set}}$, TAD$_{\text{set}}$ $\times$ TPD$_{\text{set}}$ ou TAD$_{\text{set}}$ $\times$ TPD

Retorno: TPD$_{\text{set}}$ **Parâmetros:** TAD$_{\text{set}}$, TAD$_{\text{set}}$ $\times$ TPD$_{\text{set}}$ ou TAD$_{\text{set}}$ $\times$ TPD

Os procedimentos abaixo, classificados na Tabela 24, servirão de base para as operações de transformação com separação. As funções de retorno de conjunto têm que ser analisadas pela ótica da Tabela 20, onde precisamos ter conversão com entrada e saída em forma de conjunto.

PTSS1$^{n}_{\text{Separação}}$(***TabList***$_{\text{fonte}}$, ***TabList***$_{\text{destino}}$, **FS**$^{k}_{\text{Conversão}}$ [,DW$_{\text{Lista de Parâmetros}}$])

PTSS2$^{n}_{\text{Separação}}$(***TabList***$_{\text{fonte}}$, ***TabList***$_{\text{destino}}$, **FMD**$^{k}_{\text{Acesso}}$ [,DW$_{\text{Lista de Parâmetros}}$])

PTMS1$^{n}_{\text{Separação}}$(***TabList***$_{\text{fonte}}$, ***TabList***$_{\text{destino}}$, **FM**$^{k}_{\text{Conversão}}$ [,DW$_{\text{Lista de Parâmetros}}$])

PTMS2$^{n}_{\text{Separação}}$(***TabList***$_{\text{fonte}}$, ***TabList***$_{\text{destino}}$, **FA12**$^{k}_{\text{Agregação}}$ [,DW$_{\text{Lista de Parâmetros}}$])

PTMS3$^{n}_{\text{Separação}}$(***TabList***$_{\text{fonte}}$, ***TabList***$_{\text{destino}}$, **FA2**$^{k}_{\text{Agregação}}$ [,DW$_{\text{Lista de Parâmetros}}$])

CONCATENAÇÃO E DESNORMALIZAÇÃO

Semântica Específica: Combinação de TADs. Este processo vai produzir um novo TAD ou conjunto de TADs a partir de um TAD ou conjunto deles. Os parâmetros de entrada serão, portanto, sempre TADs e binários. O retorno obviamente só pode ser um TAD.

Retorno: TAD **Parâmetros**: TAD × TAD

Retorno: TAD_{set} **Parâmetros**: $TAD_{set} \times TAD_{set}$ ou $TAD_{set} \times TAD$

As classes de procedimentos abaixo irão realizar a concatenação usando como base as funções de suporte relacional de combinação. As situações de parâmetro de conjunto serão tratadas por funções de agregação com combinação (**FA2*Y***), como descrito na Tabela 20.

PTSU$^{n}_{\text{Concatenação}}$(***TabList***$_{\text{fonte}}$, ***TabList***$_{\text{destino}}$, **FS**$^{k}_{\text{Combinação}}$ [,$DW_{\text{Lista de Parâmetros}}$])

PTMU$^{n}_{\text{Concatenação}}$(***TabList***$_{\text{fonte}}$, ***TabList***$_{\text{destino}}$, **FA2_**$^{k}_{\text{Agregação}}$ [,$DW_{\text{Lista de Parâmetros}}$])

SUMARIZAÇÃO

Semântica Específica: Agregação de TADs. Os TADs de um mesmo atributo são agregados em um TAD ou em subgrupos destes, por isso podemos produzir um conjunto de TADs a partir de um outro conjunto. Os parâmetros da função de suporte relacional devem, portanto, sempre incluir um conjunto de TADs, mesmo que, a partir de parâmetros binários, um dos parâmetros seja um TPD.

Retorno: TAD **Parâmetros**: TAD_{set}, $TAD_{set} \times TAD_{set}$, $TAD_{set} \times TAD$ ou $TAD_{set} \times TPD$

Retorno: TAD_{set} **Parâmetros**: TAD_{set},$TAD_{set} \times TAD_{set}$, $TAD_{set} \times TAD$ ou $TAD_{set} \times TPD$

A lista abaixo foi totalmente baseada nas funções de agregação, pois todos os parâmetros de retorno são conjuntos. De acordo com a Tabela 20, teremos todo o conjunto de funções de suporte (**FA*XY***) já que a agregação de TADs pode resultar em transformação ou combinação, e produzir como resultado tanto um TAD quanto um conjunto de TADs.

PTSA1$^{n}_{\text{Sumarização}}$(***TabList***$_{\text{fonte}}$, ***TabList***$_{\text{destino}}$, **FA11**$^{k}_{\text{Agregação}}$ [,$DW_{\text{Lista de Parâmetros}}$])

PTMA1$^{n}_{\text{Sumarização}}$(***TabList***$_{\text{fonte}}$, ***TabList***$_{\text{destino}}$, **FA12**$^{k}_{\text{Agregação}}$ [,$DW_{\text{Lista de Parâmetros}}$])

PTSA2$^{n}_{\text{Sumarização}}$(***TabList***$_{\text{fonte}}$, ***TabList***$_{\text{destino}}$, **FA21**$^{k}_{\text{Agregação}}$ [,$DW_{\text{Lista de Parâmetros}}$])

PTMA2$^{n}_{\text{Sumarização}}$(***TabList***$_{\text{fonte}}$, ***TabList***$_{\text{destino}}$, **FA22**$^{k}_{\text{Agregação}}$ [,$DW_{\text{Lista de Parâmetros}}$])

TRANSFORMAÇÃO

Semântica Específica: TADs convertidos para outros TADs. Não há parâmetros binários de TADs, pois eles já estão cobertos nas funções de combinação e não há mapeamentos de conjuntos para elementos individuais, e vice versa. Estas funções não realizam conversão, por isso o retorno e os parâmetros devem ser TADs de mesmo tipo.

Retorno: TAD **Parâmetros**: TAD ou TAD $\times$ TPD_{set}

Retorno: TAD_{set} **Parâmetros**: TAD_{set},TAD_{set} $\times$ TPD ou TAD_{set} $\times$ TPD_{set}

Os procedimentos de transformação irão se fundamentar nas duas funções de transformação, tanto para elementos simples e conjuntos quanto para as funções agregadoras de transformação cujo parâmetro seja um TAD.

PTST1$^{n}_{\text{Transformação}}$(***TabList***$_{\text{fonte}}$, ***TabList***$_{\text{destino}}$, **FS**$^{k}_{\text{Transformação}}$ [,$DW_{\text{Lista de Parâmetros}}$])

PTMT1$^{n}_{\text{Transformação}}$(***TabList***$_{\text{fonte}}$, ***TabList***$_{\text{destino}}$, **FM**$^{k}_{\text{Transformação}}$ [,$DW_{\text{Lista de Parâmetros}}$])

PTMT2$^{n}_{\text{Transformação}}$(***TabList***$_{\text{fonte}}$, ***TabList***$_{\text{destino}}$, **FA12**$^{k}_{\text{Agregação}}$ [,$DW_{\text{Lista de Parâmetros}}$])

ENRIQUECIMENTO

Semântica Específica: TADs convertidos para outros TADs mais complexos, ou TPDs convertidos para TADs. Não há redução de conjunto para elemento individual, o que já foi classificado como sumarização, e nem expansão de um elemento individual para um *set* o que configuraria um *drilldown*. O enriquecimento pode também significar um TPD se convertendo em um TAD, ou seja, temos funções de construção de TADs.

Retorno: TAD **Parâmetros**: TPD, TAD , TPD_{set}, TPD $\times$ TPD_{set} ou TAD $\times$ TPD_{set}

Retorno: TAD_{set} **Parâmetros**: TAD_{set}, TPD_{set}, TPD $\times$ TPD_{set},TAD_{set} $\times$ TPD_{set} ou TAD_{set} $\times$ TPD

Todas as funções de enriquecimento serão baseadas em conversão, criação de TADs ou em agregações que gerem TADs a partir de conversão (**FA12**).

PTSE1$^{n}_{\text{Enriquecimento}}$(***TabList***$_{\text{fonte}}$, ***TabList***$_{\text{destino}}$, **FS**$^{k}_{\text{Conversão}}$ [,$DW_{\text{Lista de Parâmetros}}$])

PTSE2$^{n}_{\text{Enriquecimento}}$(***TabList***$_{\text{fonte}}$, ***TabList***$_{\text{destino}}$, **FSMD**$^{n}_{\text{Criação}}$ [,$DW_{\text{Lista de Parâmetros}}$])

PTME1$^{n}_{\text{Enriquecimento}}$(***TabList***$_{\text{fonte}}$, ***TabList***$_{\text{destino}}$, **FA12**$^{k}_{\text{Agregação}}$ [,$DW_{\text{Lista de Parâmetros}}$])

PTME2$^{n}_{\text{Enriquecimento}}$(***TabList***$_{\text{fonte}}$, ***TabList***$_{\text{destino}}$, **FMMD**$^{n}_{\text{Criação}}$ [,$DW_{\text{Lista de Parâmetros}}$])

PTME3$^{n}_{\text{Enriquecimento}}$(***TabList***$_{\text{fonte}}$, ***TabList***$_{\text{destino}}$, **FM**$^{k}_{\text{Conversão}}$ [,$DW_{\text{Lista de Parâmetros}}$])

CONVERSÃO (EMPOBRECIMENTO)

Semântica Específica: TADs convertidos para TPDs. As funções de suporte utilizadas aqui serão somente as que envolvem conversão de TAD para TPDs. As conversões entre diferentes TADs já foram cobertas nos procedimentos de enriquecimento. As funções responsáveis por acesso a propriedades (*acessors*) estão descritas nas funções de separação. Estas funções irão cobrir somente as conversões de significado mais geral.

Retorno: TPD **Parâmetros**: TAD, TAD $\times$ TPD, TAD $\times$ TPD_{set},

Retorno: TPD **Parâmetros**: TAD_{set}, TAD_{set} $\times$ TAD_{set}, TAD_{set} $\times$ TAD, TAD_{set} $\times$ TPD, TAD_{set} $\times$ TPD_{set}

Retorno: TPD_{set} **Parâmetros**: TAD, TAD $\times$ TPD ou TAD $\times$ TPD_{set}

Retorno: TPD_{set} **Parâmetros**: TAD_{set}, TAD_{set} $\times$ TPD_{set} ou TAD_{set} $\times$ TPD

PTSC1$^{n}_{\text{Conversão}}$(***TabList***$_{\text{fonte}}$, ***TabList***$_{\text{destino}}$, **FS**$^{k}_{\text{Conversão}}$ [,$DW_{\text{Lista de Parâmetros}}$])

PTSC2$^{n}_{\text{Conversão}}$(***TabList***$_{\text{fonte}}$, ***TabList***$_{\text{destino}}$, **FA1**$^{n}_{\text{Agregação}}$ [,$DW_{\text{Lista de Parâmetros}}$])

PTMC1$^{n}_{\text{Conversão}}$ (***TabList***$_{\text{fonte}}$, ***TabList***$_{\text{destino}}$, **FM**$^{k}_{\text{Conversão}}$ [,$DW_{\text{Lista de Parâmetros}}$])

PTMC2$^{n}_{\text{Conversão}}$ (***TabList***$_{\text{fonte}}$, ***TabList***$_{\text{destino}}$, **FA2**$^{n}_{\text{Agregação}}$ [,$DW_{\text{Lista de Parâmetros}}$])

DRILLDOWN

Semântica Específica: TADs convertidos para outros TADs, mais complexos ou com maior cardinalidade, ou TPDs,. O *drilldown* é uma funcionalidade de análise multidimensional e se fundamentará especificamente nas funções de suporte relacional ao *drilldown*.

Retorno: TAD_{set} Parâmetros: TAD, TAD $\times$ TPD , TAD $\times$ TPD_{set} ,TAD_{set}, TAD_{set} $\times$ TPD ou TAD_{set} $\times$ TPD_{set}

PTSD$^{n}_{\textit{DrillDown}}$(***TabList***$_{\text{fonte}}$, ***TabList***$_{\text{destino}}$, **FS**$^{k}_{\textit{DrillDown}}$ [,$DW_{\text{Lista de Parâmetros}}$])

PTMD$^{n}_{\textit{DrillDown}}$ (***TabList***$_{\text{fonte}}$, ***TabList***$_{\text{destino}}$, **FM**$^{n}_{\textit{DrillDown}}$ [,$DW_{\text{Lista de Parâmetros}}$])

5.6.3 – Análise

As funcionalidades de análise de *data warehousing* seguem dois caminhos, a análise multidimensional e a análise especializada *ad hoc*. Neste último caso, as funcionalidades são bastante diversas, pois são criadas para aplicações de negócios específicas. No que se refere a TADs, estas funcionalidades dependerão da sua semântica. Se o tipo de dado for áudio, por exemplo, poderemos ter análises de similaridade a partir do espectro de freqüência. Para dados espaciais, a lista apresentada na seção 3.1.1 mostra os vários exemplos de análises especializadas.

As funcionalidades de análise multidimensional, por outro lado, possuem um processo mais padronizado que utiliza o modelo estrela, cubos multidimensionais etc. (ver seção 2.4). O processo de análise multidimensional está fundamentado em três itens:

Hierarquias – Listas de relacionamentos de agregação entre atributos de entidades. O catálogo de metadados deverá armazenar a informação do tipo de relacionamento de agregação entre dois atributos de TADs, assim como o modo como os atributos associados a este TAD serão agregados. Vamos supor, por exemplo, tipos de dados abstratos "*pacote*" e "*lata*", para os quais criamos uma relação entre latas que possam ser contidas dentro de pacotes. Se quisermos montar o valor agregado de um pacote em função das latas que o mesmo contém, temos que definir uma informação de metadados do tipo:

$$\boldsymbol{X}\text{HW}_{\text{hierarquia}}(\text{Atributo Agregador, Atributo Agregado, Relacionamento, } \textit{FunçãoAgg}(\text{Atributo}))$$

Exemplo: ***P*** – Tipo de dado estendido "pacote"

$$\boldsymbol{P}\text{HW}_{\text{hierarquia}} = (\text{Pacote, Lata, Contenção, Soma}(\textit{Preço\_lata})$$

Agregação – A agregação dos atributos determinados se dará em função da agregação entre os dados abstratos. Os sistemas de OLAP já possuem mecanismos para otimizar as funcionalidades de agregação, porém tais sistemas devem acrescentar os critérios de agregação utilizando os relacionamentos entre os TADs. Para isto, propomos a utilização das funções de suporte relacional para os tipos estendidos.

DrillDown – Entrar em nível de detalhe tomando como base um ponto na hierarquia multidimensional é simples se começarmos pela navegação inversa da agregação.

Entretanto, poderá existir um conceito extra no que se refere ao *drilldown* -- o nível de detalhe de um TAD pode consistir em produzir novos TADs a partir de um outro TAD. Por exemplo, o nível de detalhe de um mapa de cidades pode ser em nível de ruas. Para cobrir estas possibilidades, nós propomos que funções de suporte relacional de *drilldown* possam ser adicionadas como base do processo.

HIERARQUIAS

Uma hierarquia é definida de duas formas possíveis: relacionando duas colunas como agregadora e agregada, e relacionando uma hierarquia pré-existente e outro atributo, segundo o formato abaixo:

$$\boldsymbol{X}\mathrm{HW}_{\mathrm{hierarquia}} = (\boldsymbol{X}\mathrm{W}_{\mathrm{coluna}}{}^{\mathrm{agregador}}, \boldsymbol{X}\mathrm{W}_{\mathrm{coluna}}{}^{\mathrm{agregado}}, \mathbf{FR\_\_\_}{}^{\mathrm{n}}{}_{\mathrm{Relação}}, \mathbf{F\_\_}{}^{\mathrm{n}}{}_{\mathrm{DrillDown}},$$
$$\mathit{FunçãoAgg}^{1}(\mathrm{Coluna}^{1}), \mathit{FunçãoAgg}^{2}(\mathrm{Coluna}^{2}),...)$$

$$\boldsymbol{X}\mathrm{HW}_{\mathrm{hierarquia}} = (\boldsymbol{X}\mathrm{HW}_{\mathrm{hierarquia}}, \boldsymbol{X}\mathrm{W}_{\mathrm{coluna}}{}^{\mathrm{agregado}}, \mathbf{FR\_\_\_}{}^{\mathrm{n}}{}_{\mathrm{Relação}}, \mathbf{F\_\_}{}^{\mathrm{n}}{}_{\mathrm{DrillDown}},$$
$$\mathit{FunçãoAgg}^{1}(\mathrm{Coluna}^{1}), \mathit{FunçãoAgg}^{2}(\mathrm{Coluna}^{2}),...)$$

Os outros parâmetros são a função de suporte relacional que especifica como as colunas da hierarquia estarão relacionadas, a função de suporte relacional que especifica o *drilldown* e as funções de agregação para cada coluna associada a hierarquia.

As hierarquias são estabelecidas entre dois TADs, entre dois conjuntos de TADs ou entre um conjunto de TADs e um TAD[3]. Por isso, somente as funções de suporte relacional de relacionamento com estes parâmetros binários servirão de suporte para criação de hierarquias. As funções para realizar o *drilldown* já estão devidamente especificadas na seção 4.3.2 e 5.5.2. Elas servem para determinar como uma hierarquia pode caminhar do nível agregado para o nível de detalhe "criando" os TADs a partir dos TADs do nível hierárquico mais agregado.

Os procedimentos para a criação das hierarquias tem o seguinte formato geral, melhor detalhado no Apêndice A:

$$\boldsymbol{X}\mathrm{HW}_{\mathrm{Hierarquia}} = \mathbf{PA[S|M]H\_}{}^{\mathrm{n}}{}_{\mathrm{Hierarquia}}(\boldsymbol{X}\mathrm{W}_{\mathrm{col}}{}^{1}, \boldsymbol{X}\mathrm{W}_{\mathrm{col}}{}^{2}, \mathbf{FR[S|M]\_}{}^{\mathrm{n}}[, \mathbf{FP[}\mathrm{S}|\mathrm{M}\mathbf{]D}^{\mathrm{n}}])$$

$$\boldsymbol{X}\mathrm{HW}_{\mathrm{hierarquia}} = \mathbf{PA[S|M]H\_}{}^{\mathrm{n}}{}_{\mathrm{Hierarquia}}(\boldsymbol{X}\mathrm{HW}_{\mathrm{hierarquia}}, \boldsymbol{X}\mathrm{W}_{\mathrm{col}}{}^{2}, \mathbf{FR[S|M]\_}{}^{\mathrm{n}}[, \mathbf{FP[}\mathrm{S}|\mathrm{M}\mathbf{]D}^{\mathrm{n}}])$$

As hierarquias entre TPDs não foram consideradas neste trabalho, dado que elas já fazem parte de todos os modelos e sistemas de análise multidimensional convencionais.

AGREGAÇÃO E DRILLDOWN

Para realizarmos a agregação dos dados relacionados a uma hierarquia, ou seja, os atributos, precisamos especificar uma função de agregação. O número de funções é ilimitado, por isso apresentamos uma função para adicionar funções de agregação a uma hierarquia. O *drilldown* pode ser feito pela navegação reversa ou também pode-se fornecer alguma função específica.

$$\textbf{PASA}^{n}_{\text{AddData}}(\boldsymbol{X}\text{HW}_{\text{hierarquia}}, \boldsymbol{X}\text{W}_{\text{coluna}}, \textit{FunçãoAgg}[,\textit{FunçãoDrillDown}])$$

A agregação em si é realizada apenas referenciando a hierarquia e a que nível desejamos agregar. O mesmo ocorre com o *drilldown* -- uma vez que todas as funções e relacionamentos já estão pré-definidos, basta definir o nível de navegação reversa desejada. Os seguintes procedimentos realizarão a agregação ou o nível de detalhe para uma hierarquia qualquer $\boldsymbol{X}\text{HW}_{\text{hierarquia}}$:

$$\textbf{PASS}^{n}_{\text{Agregação}}(\boldsymbol{X}\text{HW}_{\text{hierarquia}}, \text{Nível}, \text{DW}_{\text{parâmetro}})$$

$$\textbf{PASD}^{n}_{\text{DrillDown}}(\boldsymbol{X}\text{HW}_{\text{hierarquia}}, \textit{Nível})$$

5.6.4 - Funções *Ad hoc*

Estas funções podem ter uma enorme variedade e diversidade, portanto torna-se difícil estabelecer um padrão para as mesmas. Especificaremos apenas a seguinte forma geral, na qual qualquer funcionalidade possa ser modelada:

$$\textbf{PASG}^{n}_{\text{AdHoc}}(\text{Parâmetros Gerais})$$

5.7 – Conclusão

Mostramos aqui a metodologia que levou à criação do *framework* **5-Alpha** através de um exaustivo processo de classificação das funcionalidades necessárias ao processo de *data warehousing*. Esta metodologia não é necessária para utilização do *framework,* apesar de contribuir para um melhor entendimento no momento da sua utilização. Também oferecemos uma contribuição mostrando um método de classificação e organização de processos contendo um grande número de possíveis funções e módulos de *data warehousing*.

Capítulo 6 – Sistema de Consulta e Análise

6.1 – Introdução

Todos os sistemas que permitem a extensão do banco de dados objeto-relacional para um tipo de dado específico sofrem dos seguintes problemas básicos:

5 - Existe uma grande quantidade de funções para manipulação de tipos de dados não convencionais e são, em geral, de difícil memorização.

6 - Consultas mais complexas envolvendo funções para dados não convencionais podem ser muito difíceis de se elaborar.

7 - Os usuários dos tipos de dados estendidos têm muito pouco conhecimento de SQL avançado.

8 - Os atributos específicos de um tipo de dado podem trazer novas semânticas aos processos de análise do sistema de *data warehouse*.

9 - Algumas funções podem não ter suporte do banco de dados, como funções agregadoras definidas pelo usuário.

10 - Existe falta de flexibilidade de alteração da própria ferramenta de *data warehouse* (Rifaieh e Benharkat 2002). O *Query Assist* do DB2, por exemplo, não tem nenhuma forma de ser configurado a não ser por alteração de código e muita burocracia de alteração de projeto.

As funções para manipulação espaciais apresentadas na seção 7.3 são um exemplo da complexidade da qual falamos no item 1. Podemos verificar que, somente para criação de um tipo de geometria, temos mais de três construtores diferentes. Se quisermos criar uma consulta simples, digamos, "obter a média das contas de todos os clientes do meu banco com mais de 50.000 e que moram dentro de 5km de cada filial", teríamos que realizar a consulta abaixo:

```
SELECT BRANCH.BRANCH_ID, AVG(ACCOUNT.BALANCE)
FROM   BANKDEMO.ACCOUNT,BANKDEMO.CUSTOMER,BANKDEMO.BRANCH
WHERE ((ACCOUNT.CUSTOMER_ID = CUSTOMER.CUSTOMER_ID) AND
       (db2gse.ST_Within(CUSTOMER.FEATURE,
        db2gse.ST_Buffer(BRANCH.FEATURE,5.000))=1 AND
           (ACCOUNT.BALANCE >= 50000)))
     GROUP BY BRANCH.BRANCH_ID;
```

Consulta 6-1

Como podemos ver, para realizar uma simples agregação, é necessário usar subterfúgios, dado que o SQL99 não realiza agregações de TADs, nem de *LOBs*. Este tipo de "complexificação" da consulta para obter "elementos mais próximos" geograficamente, assim como outros, requer um maior conhecimento de manipulação

do SQL. Em alguns casos é possível empacotar consultas complexas em funções dentro do banco, mas, em outros casos, o próprio banco não permite isso, como as *UDFs* agregadoras do DB2 (item 5). De qualquer forma, não é sempre interessante empacotar consultas complexas em nível mais baixo. Podem surgir novos recursos do SQL que venham a melhorar o desempenho das consultas, e sempre devemos tentar transferir para o banco de dados a decisão de planejamento e otimização de consultas.

Os usuários das aplicações que lidam com tipos de dados não convencionais estão bastante distanciados do universo do banco de dados. Eles olham para os dados com uma semântica totalmente própria e realizam operações que nem sempre estão mapeadas dentro de um processo de *data warehousing*. Tomando como base a análise espacial, a agregação de dados através da distância de um cliente a uma filial não está prevista no processo de OLAP, por exemplo.

Um sistema capaz de transpor estes problemas e fornecer uma solução mais geral para uma ferramenta de *data warehousing* estendido deve conter os seguintes requisitos:

- **Facilidade de utilização de funções especializadas** – A lista de funções estendidas para dados não convencionais pode ser extensa e com muitos parâmetros. O sistema deve possuir *menus* e atalhos para escolha de funções mais prováveis a serem usadas em cada situação.
- **Integração entre as aplicações específicas e** *data warehousing* – É importante que módulos deste sistema possam ser utilizados para integração com aplicações específicas dos tipos de dados não convencionais, como Sistemas de Informação Geográfica.
- **Modularidade de processamento e visualização** – O sistema deve possuir interfaces genéricas para obter tipos de dados não convencionais e os diversos tipos de visualização. Cada tipo de dado não convencional pode possuir uma forma particular de visualização.
- **Transparência na construção de consultas complexas** – Algumas funções, como “perto de”, no caso espacial, podem requerer o uso de subconsultas em vários níveis. Devemos poder escrever predicados de consulta de nível mais alto, que poderão ser convertidos em consultas mais complexas se necessário.

- **Portabilidade** – O sistema deve poder ser executado em diversos sistemas operacionais.
- **Compatibilidade com padrões internacionais** – Além da compatibilidade óbvia com o SQL/99, a linguagem gerada deve cumprir os padrões existentes para os tipos de dados não convencionais sendo tratados. No caso de dados espaciais, o OGC (Open GIS Consortium e Ryden 1999) descreve um padrão a ser utilizado para funções espaciais dentro do SQL, como já mencionado na seção 3.2.
- **Interatividade de execução de consultas** – Para que possamos testar e observar o resultado das consultas, o sistema precisa poder executá-las e enviar o resultado diretamente para os visualizadores. As consultas requerem variáveis que possam ser alteradas no processo de execução interativa.Desta forma, o usuário poderia ajustar os valores de parâmetros para melhor sintonia e desempenho de suas consultas.

Na seção seguinte iremos mostrar um protótipo projetado para atender a consultas espaciais.

6.2 – O Sistema

6.2.1 – Visão Geral

O sistema tem por finalidade auxiliar na criação e execução das consultas e procedimentos de *data warehousing* definidas através do *framework* **5-Alpha** (seção 4.3.2) para um conjunto de TADs. A Figura 19 apresenta a modelagem dos casos de uso do sistema. O **Usuário** inicializa a criação de uma consulta, ou procedimento, fornecendo o descritor do *framework* para seu modelo de TADs. O **Sistema** irá então montar o painel de construção de consultas e conectar-se ao banco de dados. Após a montagem do painel de construção, o **Usuário** cria sua consulta ou procedimento e pode escolher executá-los e visualizá-los diretamente ou interativamente. O **Sistema** então executa e envia a consulta ou procedimento diretamente para os visualizadores, ou fornece a opção de execução interativa. A montagem do painel de construção é feita utilizando-se as funções e procedimentos definidos através do *framework* **5-Alpha**. O **Sistema** disponibiliza todas as funções e procedimentos em painéis para que o usuário possa inseri-las nas consultas de forma interativa e flexível.

Os visualizadores fornecem as perspectivas de tabela, do texto e da WEB para o resultado das consultas, contudo, visualizadores específicos para a semântica de cada TAD devem ser implementados de acordo com uma interface de visualização pré-definida no sistema.

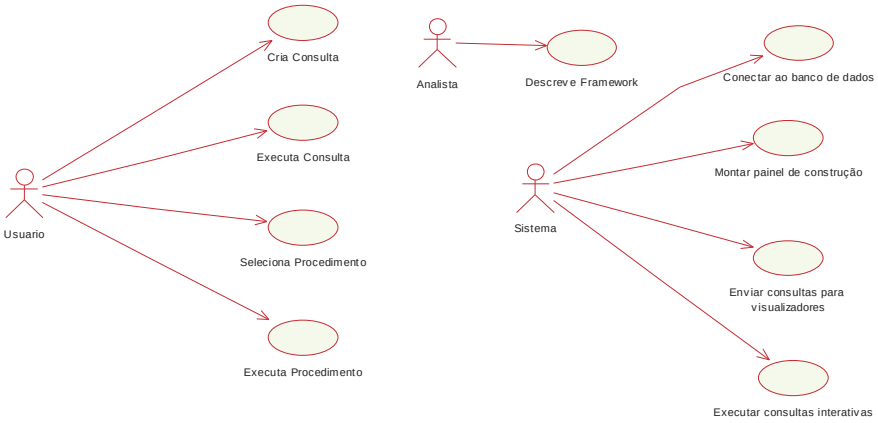

Figura 19 - Casos de uso da ferramenta

6.2.2 – Arquitetura e Funcionamento

O sistema é composto de quatro módulos mais um conjunto extensível de visualizadores (Figura 20). A linguagem utilizada para o desenvolvimento foi JAVA, tanto por suas características de portabilidade e segurança quanto por ter sido uma linguagem amplamente utilizada em SIGs (ESRI 2003b) e sistemas de bancos de dados espaciais (Oracle Corporation 2000). Cada módulo a seguir foi desenvolvido utilizando arquitetura de *JAVA Beans* (Rodrigues 1998) e os seus métodos expostos são acessados através da implementação de *listeners* de conexão:

- Módulo de geração do *framework*
- Módulo de acesso ao banco de dados
- Módulo de construção de consultas e procedimentos
- Módulo de processamento interativo
- Módulos visualizadores

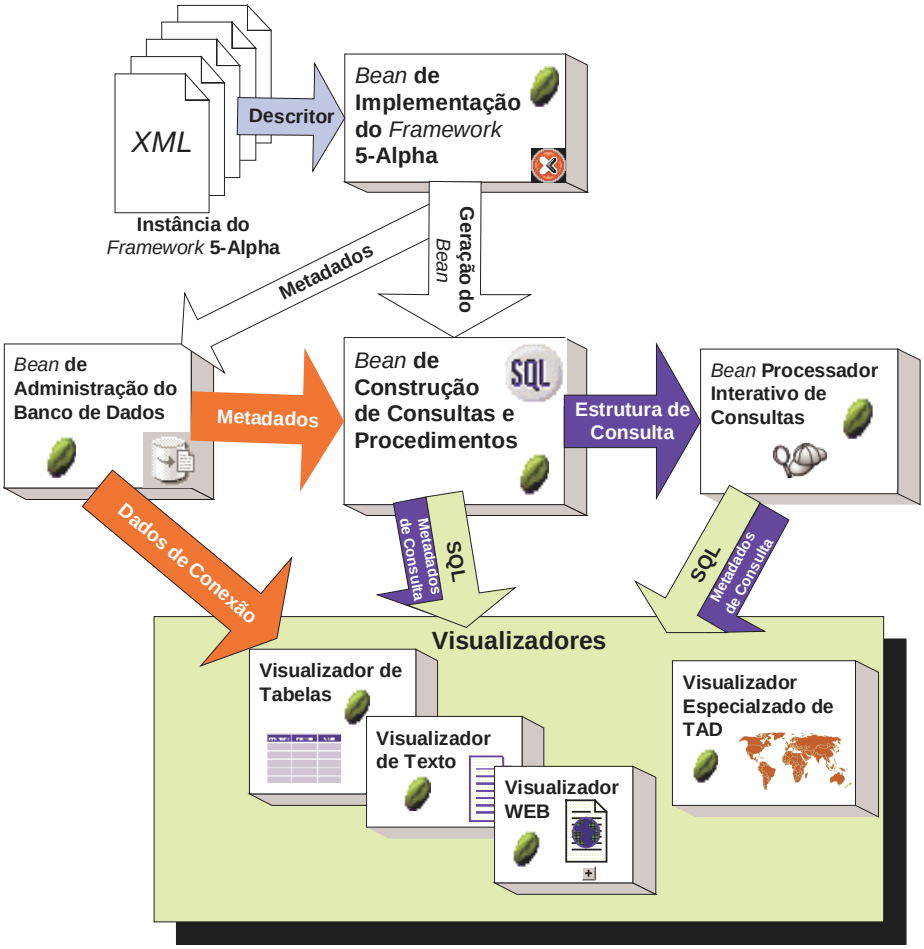

Figura 20 - Arquitetura geral do sistema de consulta e análise

O módulo de implementação do *framework* **5-Alpha** não funciona interativamente com o sistema, mas apenas gera a ferramenta em si. Ele lê o arquivo em XML (Seção 4.3.2) com a descrição completa de todos os TADs, catálogos, funções e procedimentos, e o envia para todos os outros módulos. O processo serve para gerar as interfaces com o usuário e para configurar as funcionalidades necessárias para compor a ferramenta de *data warehousing* para os TADs.

O módulo de acesso ao banco de dados fornece o serviço de conexão com o banco, e recebe do módulo de implementação uma lista dos tipos reconhecidos pelo sistema, bem como a informação sobre as tabelas de catálogo. Ele utiliza esta informação para obter os metadados relativos aos TADs e TPDs em cada coluna de cada tabela do banco de d*ata warehousing* e enviá-los para o módulo de construção de consultas. Este último irá utilizar os metadados para apresentar ao usuário listas de tabelas e colunas com cada tipo de TAD e TPD separadamente.

O módulo de construção de consultas permite ao usuário criar, armazenar e executar consultas e procedimentos de forma flexível, consistente com a minimização de erros. As consultas são criadas em um painel aonde cada lacuna é apresentada na forma de um botão, que ativa todas as funções e atributos da tabela passíveis de serem inseridos naquela posição. As funções e atributos que podem ser inseridos também são, por sua vez, apresentados na forma de botões que, uma vez acionados, acrescentam novas funções ou novos botões de lacunas no painel. Os elementos da consulta, assim construídos, são armazenados em uma estrutura que gera a consulta SQL final em modo texto, podendo ser enviada para os visualizadores ou para o módulo de processamento interativo. O diagrama completo de funcionamento deste módulo será mostrado na seção 6.2.3.

O módulo de processamento interativo é destinado a flexibilizar a execução de consultas contendo parâmetros variáveis fornecidos pelo usuário. Suponhamos, por exemplo, uma consulta que seleciona os clientes dentro de um círculo de raio X e que desejamos executar esta consulta para vários valores de X. Este módulo permite uma alteração rápida e flexível de X que seja reenviada para execução e visualização.

Os visualizadores são responsáveis por mostrar os resultados de uma consulta ou procedimento. Eles de fato utilizam o manipulador (*handle*) de conexão com o banco que é enviado pelo módulo de acesso ao banco. A consulta é enviada para o banco junto com informações sobre os parâmetros de retorno do resultado da consulta a ser visualizada. Através destas informações, o módulo de visualização tem como decidir de que forma apresentará os atributos de TADs da consulta. Os módulos de visualização básicos de tabela e texto implementados utilizam um método padrão de apresentação. Outros módulos, desenhados especificamente para um TAD, devem implementar sua própria estratégia de tradução dos resultados da consulta.

O conjunto completo de módulos do sistema está convenientemente conectado e inserido em quadros de interface com o usuário escritos em JAVA (*frames*), e que podem ser invocados através de uma barra principal de opções (Figura 21). Estes quadros de interface controlam a visibilidade dos módulos, não permitindo a repetição de sua criação caso os mesmos já tenham sido invocados, mas permitindo que os canais de conexão possam ser bloqueados temporariamente. Este mecanismo impede a alteração dos resultados de uma consulta, mesmo que a sua execução seja requerida.

Figura 21 – Barra principal de menu

A Figura 22 mostra o diagrama de classes de todos os *listeners* de conexão que são utilizados para enviar as mensagens entre os módulos. O módulo de administração do banco de dados conhece uma interface (Classe abstrata) para um *listener* **SEDBConnectListener**. O módulo administrador do banco envia mensagens através de classes que implementam esta interface e são adicionadas a sua lista de *listeners*. Existem duas classes que implementam esta *interface*, uma para enviar os metadados para o módulo de construção de consultas (**SEDBConnectQueryBuilderListener**), e outra para enviar os dados de conexão para os visualizadores (**SEDBConnectViewListener**). O módulo de construção de consultas, por sua vez, conhece uma outra interface (**QueryBuilderListener**) que tem uma implementação para enviar a consulta montada ao processador interativo de consultas (**QueryBuilderTunerListener**), e outra implementação para enviar aos visualizadores (**QueryBuilderViewListener**). O processador interativo de consultas conhece outra interface para enviar consultas para os visualizadores (**QueryViewListener**).

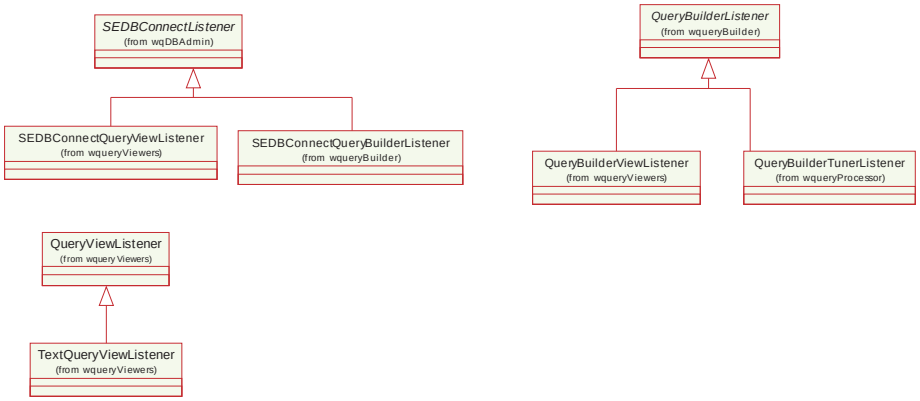

Figura 22 - Diagrama dos *listeners* de conexão

A Figura 23 apresenta o diagrama de seqüência das mensagens enviadas entre os módulos, após o pedido de conexão com o banco de dados. O *listener* de conexão com o módulo de consulta primeiro envia os dados de conexão e depois os metadados. As informações de conexão incluem também a versão do banco de dados. Esta informação

é relevante para possíveis variações que possam haver de montagem das funções para os diversos tipos de bancos de dados. A presente versão foi projetada somente para operar com o **DB2 Spatial Extender V7.2** (IBM Corporation 1998) e **V8.1** (IBM Corporation 2002a).

Os metadados são enviados em dois pacotes separados, um com as informações da colunas e tabelas dos TPDs e outro com os TADs. Após o envio de cada um destes pacotes, o módulo de consulta configura as listagens de tabelas e colunas correspondentes a cada tipo de dado.

O *listener* de conexão com os visualizadores envia duas informações de conexão com o banco, uma contendo a conexão física e os metadados, e outra com os atributos de conexão (nome do banco, usuário, senha). Alguns visualizadores podem requerer fazer sua própria conexão com o banco, não reutilizando aquela já aberta.

As mensagens entre os outros *listeners* ocorrerá apenas quando for requisitada a execução de uma consulta, como veremos ao final da seção 6.2.3.

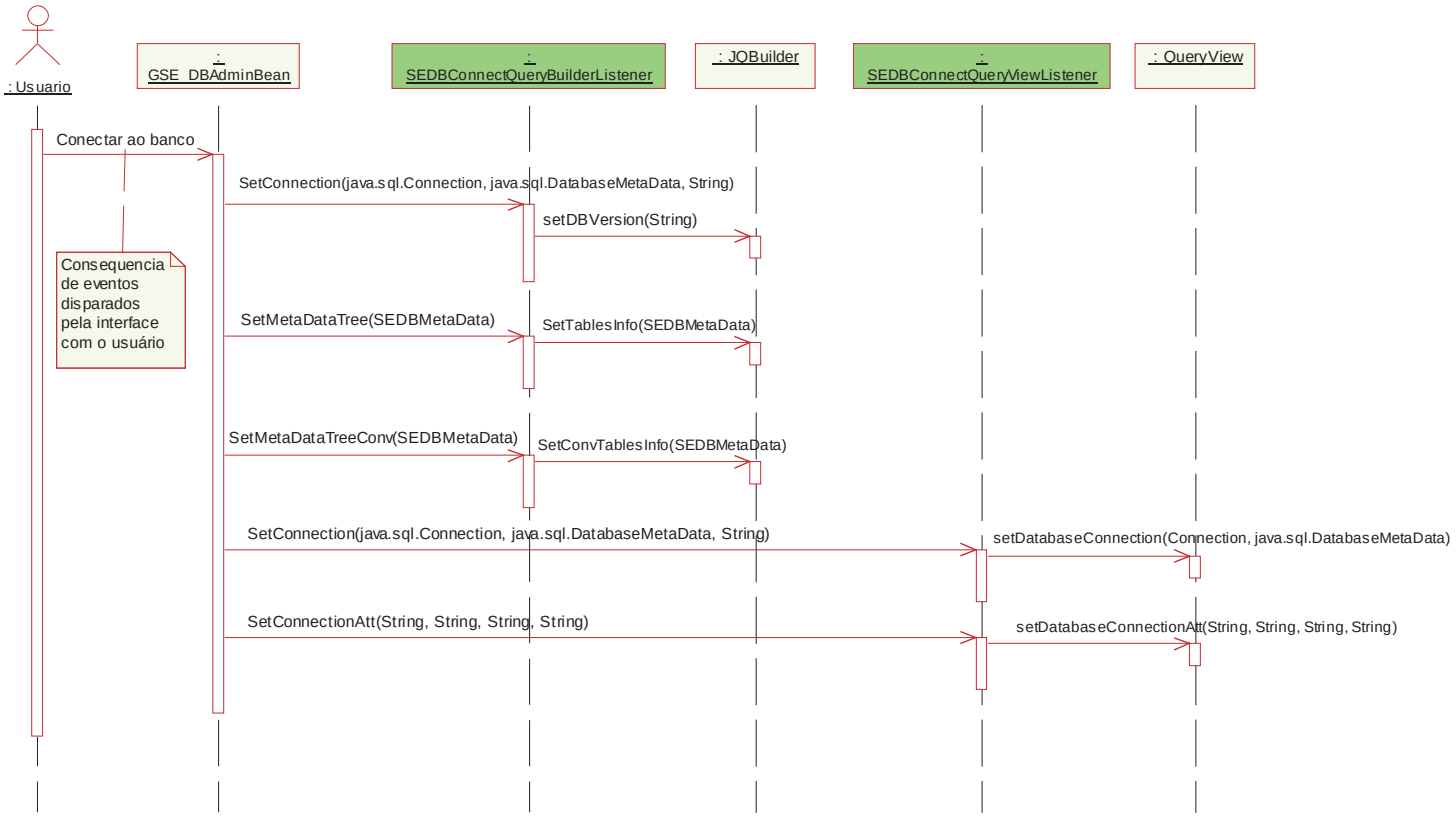

Figura 23 - Diagrama de sequência de conexão e inicialização da ferramenta

6.2.3 – Módulo de Construção de Consultas e Procedimentos

A construção flexível é baseada no conceito de preenchimento adequado de lacunas (veja topo da Figura 24). Isto reduz a possibilidade de erros de sintaxe pois somente as funções cujo retorno corresponde ao tipo da lacuna são apresentadas para o usuário na forma de botões. Quando estes são acionados, o painel de construção acrescenta esta nova função dentro da lacuna.

A disposição básica destes painéis pode ser vista na Figura 24. A quantidade de grupos de funções e procedimentos nos levaram a optar por uma arquitetura visual de *"tabbed panels"*, ou seja, painéis sobrepostos, que são trazidos para o nível de visualização principal quando selecionados. O painel principal (Figura 25) é dividido em três grandes grupos de painéis:

- Construção (**JQConstructionPanel**) – Apresenta as partes da consulta em construção.
- Consulta (**JQQueryPanel**) – Contém as funções para construção de consultas SQL.
- *Warehousing* (**JQWarehousePanel**) – Contém todos os procedimentos executáveis de *data warehousing* definidos na **seção E** do ***framework* 5-Alpha**.

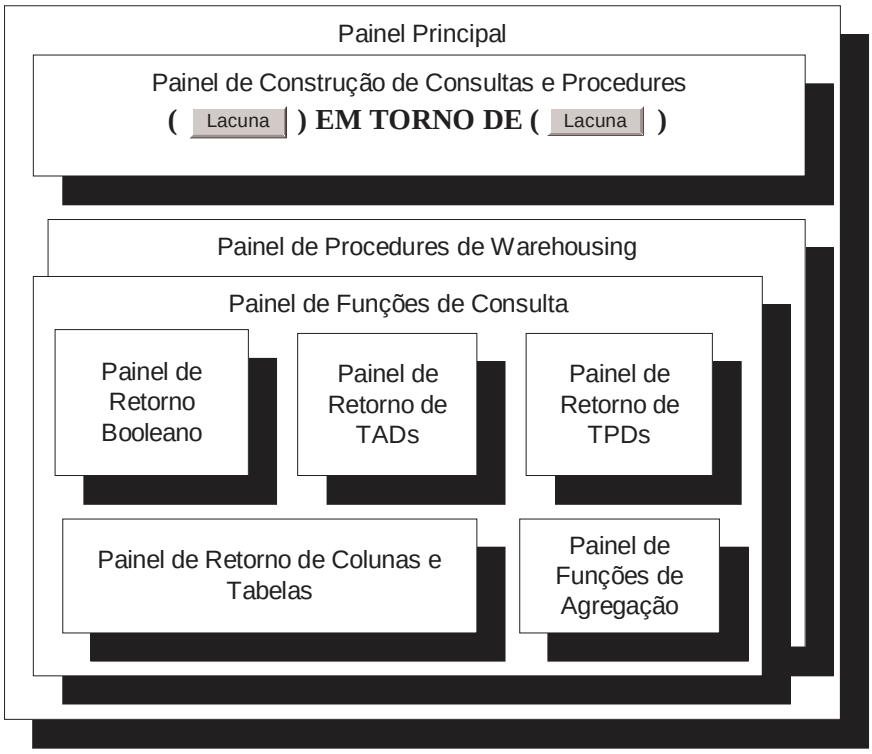

Figura 24 - Projeto da disposição visual dos painéis da ferramenta de consulta

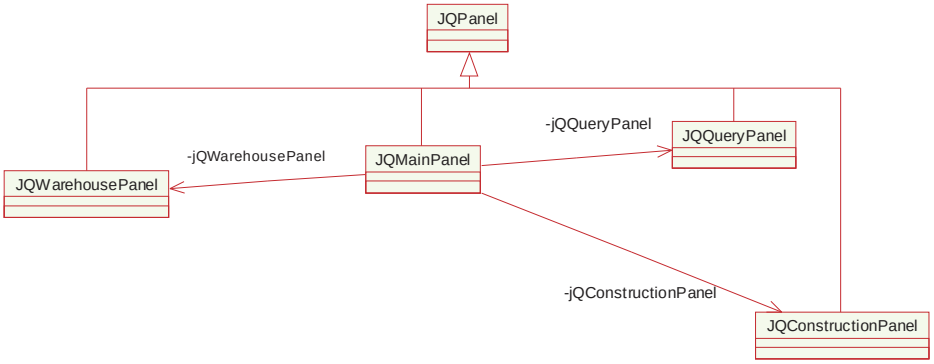

Figura 25 – Diagrma de classes dos painéis principais de consulta

PAINEL DE CONSTRUÇÃO

Este grupo de painéis sobrepostos é utilizado para construir consultas ou procedimentos em partes separadas. A construção de uma sentença padrão em SQL utiliza um painel para cada uma das cláusulas SELECT (**JQSelectPanel**), FROM (**JQFromPanel**), WHERE (**JQWherePanel**) e GROUP BY ou ORDER BY (**JQxByPanel**). As consultas completas são automaticamente criadas, em formato de texto, no painel **JQSQLEditPanel,** onde elas podem ser editadas diretamente. Uma vez editada, a versão é armazenada, perdendo o vínculo com os painéis de geração. Esta versão do sistema não possui *parsing* reverso para os painéis de construção. Estes possuem um painel adicional para indicação das representações visuais da consulta (**JQPresentationPanel**), como indicar que uma variável da cláusula SELECT será representada com cores em faixas de valores homogêneos. Por último, o painel de controles possui comandos para apagar, executar, enviar para execução interativa e carga, e armazenamento de consultas (**JQControlsPanel**).

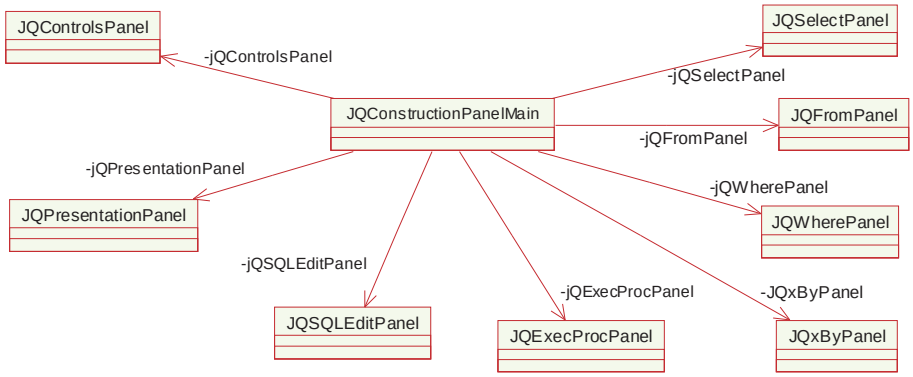

Figura 26 - Painéis de construção de consultas e procedimentos

As classes de botões, que são utilizados para representar as lacunas nas cláusulas das consultas e procedimentos(Figura 27), correspondem a todos os tipos de retorno das funções e variáveis disponíveis para serem inseridas na consulta. Os botões que retornam TPDs (**JQButtonTPD**) e TADs (**JQButtonTAD**) possuem duas versões equivalentes para retorno de conjuntos. A primeira interpreta conjuntos através do nome de uma coluna que contém o conjunto de elementos de TPDs (**JQButtonTPDSetCol**) ou TADs (**JQButtonTADSetCol**). A segunda versão corresponde a funções que de fato retornam conjuntos na forma de uma tabela do banco, sendo **JQButtonTPDSet** a classe para retorno de conjuntos de TPDs, e **JQButtonTADSet** a classe para conjuntos de TADs. Essas duas são especializaçõesde uma classe geral para retorno de nomes de tabelas (**JQButtonTable**), a qual será usada, em grande parte, pelos procedimentos de *data warehousing* que precisam receber nomes de tabelas como parâmetros.

As classes de botões possuem métodos para habilitar a visualização dos painéis contendo funções correspondentes ao seu tipo de retorno, bem como um atributo para indicar qual o TAD ou TPD que ele está representando. Por isso, os painéis ativados também omitirão as funções que não retornam exatamente o mesmo TAD.

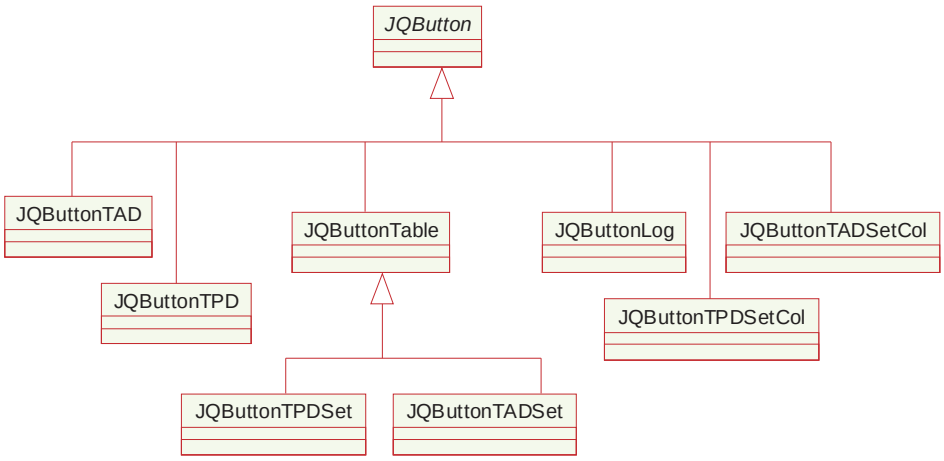

Figura 27 - Botões utilizados pelo painel de construção de consultas

PAINÉIS DE FUNÇÕES DE CONSULTA

Este grupo de painéis contém funções em grupos associados a cada tipo de retorno, TADs, TPDs, conjuntos, tabelas etc. (Figura 28). A maioria destes agrega outros painéis para agrupar as funções de um mesmo tipo de retorno em categorias distintas, como conversão, construção, transformação etc. (Figura 29). Os painéis de retorno de tabelas e colunas de tabelas de TADs (**JQTableTADPanel)** e TPDs (**JQTableTPDPanel**), diferentemente dos outros, não contêm funções, mas apenas o nomes de colunas e de suas respectivas tabelas.

O painel de retorno de valores nominais (**JQPanelNomValues**) tem por função inserir qualquer valor fornecido diretamente pelo usuário e mantê-lo como parâmetro variável da consulta.

O painel de funções agregadoras (**JQAggPanel** - Figura 30) contém todas as funções que tratam seus parâmetros de entrada como conjuntos, independente do seu retorno. Este painel será redundante, quando a respectiva função agregadora for de um retorno já definido em outro painel, como TAD, por exemplo. Entretanto, por praticidade, dispomos esta categoria de funções separadamente.

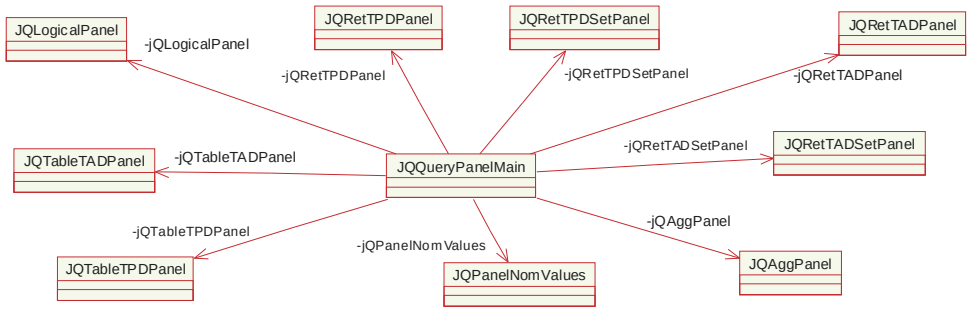

Figura 28 - Painéis fonte das funções de consulta

Figura 29 - Painéis fonte de funções de retorno de TAD e TPD

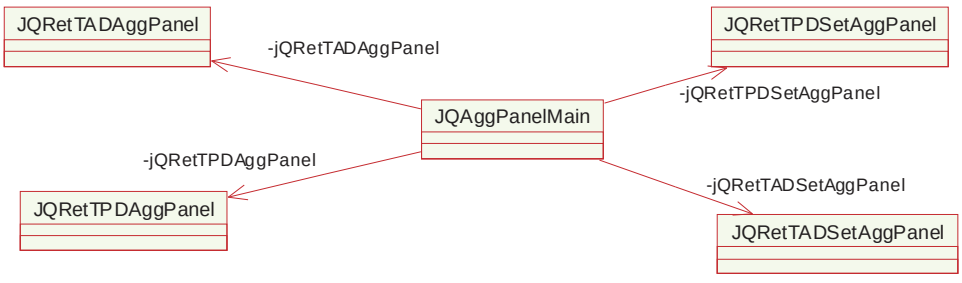

Figura 30 - Painéis fonte das funções de agregação

O grupo de painéis de retorno de valores booleanos (Figura 31) contém apenas um painel para armazenar todas as funções de retorno booleano, mas possui vários outros com funções para comparação (Seção B do *framework* **5-Alpha**) e conectivos lógicos (AND, OR etc.). Os painéis de comparação (**JQCompConnPanelTPD** e

JQCompConnPanelTAD) possuirão funções de igualdade, diferença e ordem parcial para TPDs (=,<,>,≠ convencionais) e TADs. O painel de conectivos lógicos é o mesmo para as duas situações (**JQLogConnPanel**), contudo, o painel **JQExtLogConnPanel** está disponível para conectivos lógicos estendidos e para trabalhar com outros tipos de lógica, como a nebulosa.

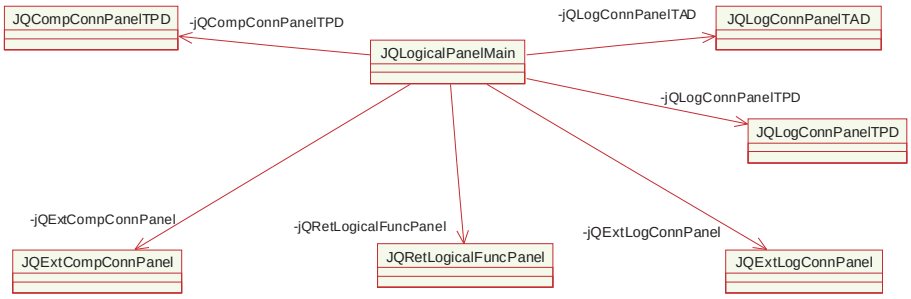

Figura 31 - Painéis fonte das funções booleanas

PAINÉIS DE PROCEDIMENTOS DE *DATA WAREHOUSING*

Os painéis contendo a lista de procedimentos de *data warehousing* se dividem nos 3 subgrupos principais correspondentes ao *framework* **5-Alpha** (Figura 32), e cada um destes nas suas respectivas subcategorias do *framework*.

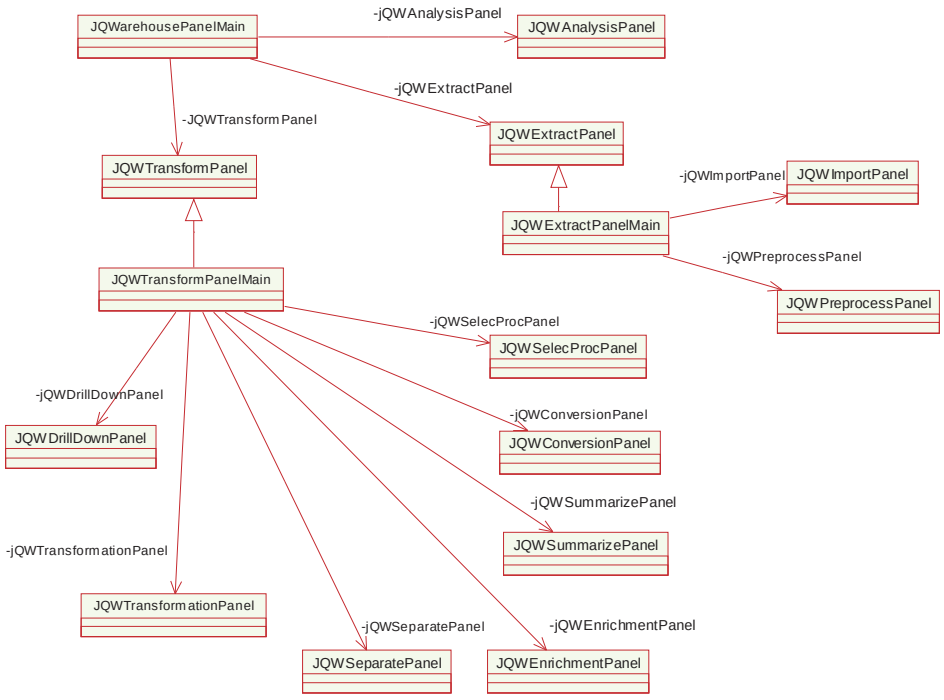

Figura 32 - Painéis fonte de procedimentos de *warehousing*

ESTRUTURA DE ARMAZENAMENTO

A consultas construídas são armazenadas e manipuladas através de uma estrutura em árvore (Figura 33). Os nós desta árvore são representados pela classe **QueryNode**. Esta estrutura pode conter qualquer sentença em SQL, seja uma consulta (**XWQuery**) ou a chamada a um procedimento armazenado (**XWProcedure**). A raiz principal desta estrutura (**XWProgram**) representa um conjunto qualquer de sentenças que pode ainda conter cláusulas de pré-inicialização (**BeginClause**)

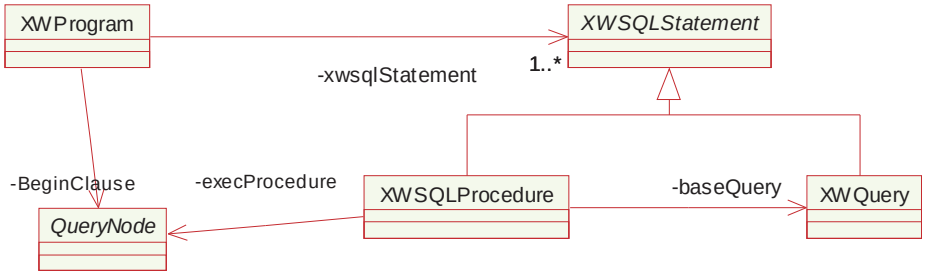

Figura 33 - Estrutura de armazenamento de consultas e procedimentos

A subárvore que contém sentenças de consulta (SELECT...FROM..) tem uma raiz da classe **XWQuery** e cada um de seus nós é composto de especializações da classe abstrata **QueryNode** (Figura 34). A classe contém uma subárvore correspondente a cada cláusula da consulta (SELECT, FROM, WHERE etc.), mais uma cláusula inicial que pode ser escrita antes do SELECT (como WITH), e ainda uma cláusula para ser usada pelo sistema de representação visual de valores (**PresentationAttributes**). Cada nó pode conter outros nós por referência indireta usando objetos da classe **RefNode**. Esse mecanismo é necessário para permitir que novas funções possam ser inseridas recursivamente na lacuna representada pelo objeto da classe **RefNode**. Elas são fornecidas como parâmetros nas diversas partes do sistema para que os novos nós sejam inseridos a partir delas.

Esta estrutura de árvore é a responsável pela montagem do painel visual de construção de consultas. O objeto principal **XWQuery** conhece uma interface para um *listener* que é usada para acessar cada painel de cláusula do módulo de construção. Os métodos **RebuildPanels()** da classe acessam cada subárvore de cláusula da consulta, e repassam o *listener* correspondente chamando o método **BuildPanel()** recursivamente a partir do nó raiz. Cada nó utiliza a interface para o *listener* para acrescentar rótulos, botões ou texto ao painel de construção.

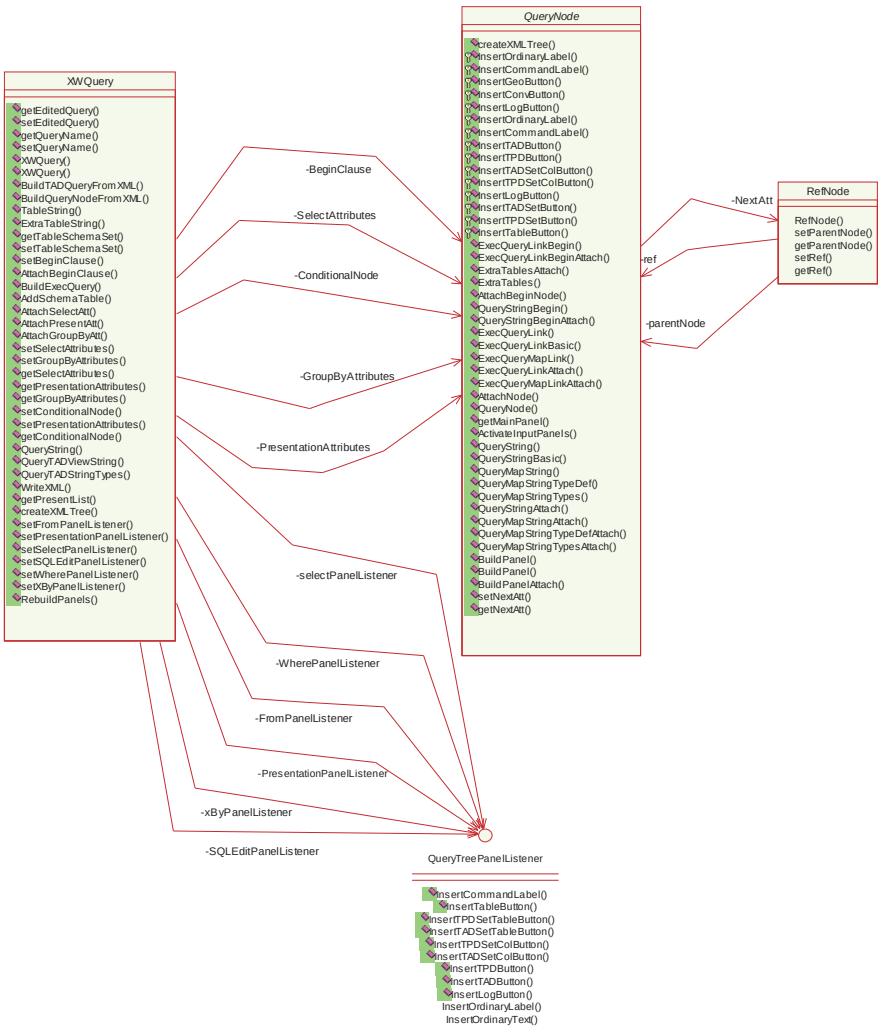

Figura 34 - Estrutura geral de uma consulta

A hierarquia principal dos nós se divide em nós de retorno booleano (**LogicalNode**), retorno de valores (**DataNode**) e retorno de conjuntos (**QuerySetNode**). A implementação de cada nó segue o padrão de retorno de funções de acordo com o *framework* **5-Alpha** (Figura 35), e ainda se especializam em nós representando funções que recebem parâmetros individuais (**TPDSParXXX** e **TADSParXXX**). Os nós utilizados para armazenar nomes de colunas de tabelas estão especializados separadamente por conveniência. Já os nós específicos para procedimentos armazenados são implementados pela classe **ProcNode**. A utilização desta hierarquia de nós pode ser feita diretamente ou através da especialização dos mesmos em novos nós, representando funções específicas.

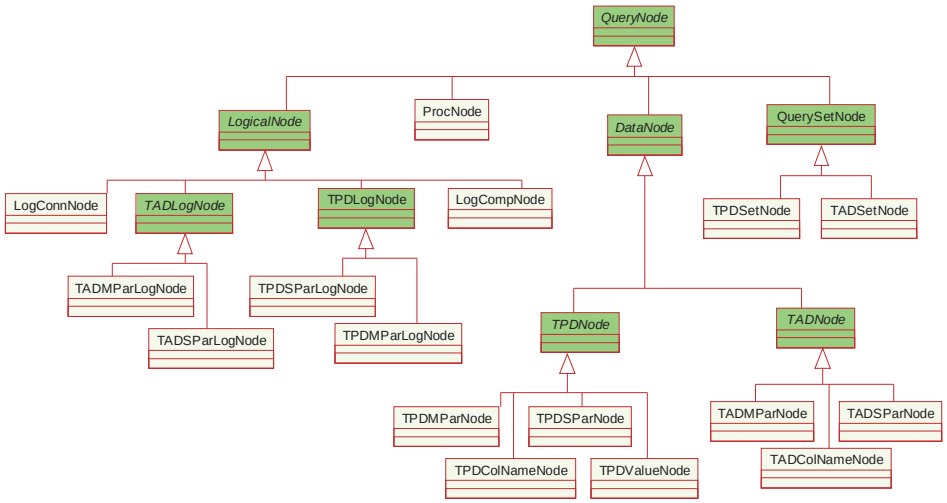

Figura 35 - Estrutura dos nós de consulta

CONSTRUÇÃO E EXECUÇÃO DE CONSULTAS

O primeiro passo do usuário para montar uma consulta é selecionar um dos painéis de construção correspondentes a cláusula da consulta que ele deseja escrever, como, por exemplo, a cláusula SELECT. O diagrama de seqüência da Figura 36 mostra as possíveis ações para cada painel selecionado. O acionamento de um certo "*tab*" do painel de construção dispara um método da forma geral **ActivateXXXClause(),** que, além de tornar o painel do "*tab*" visível, invoca um método **ActivatePanel** do painel para ativar todos os painéis de função adequados à construção daquela cláusula. A primeira função, variável ou procedimento que será inserido no painel não utiliza nenhum objeto de lacuna, já que a cláusula ainda está vazia. Assim, para cláusulas vazias, os painéis de funções e variáveis que são ativados podem ser mais amplos. Uma cláusula SELECT, por exemplo, pode inserir todos os tipos de função ou variáveis disponíveis, inclusive as de retorno booleano.

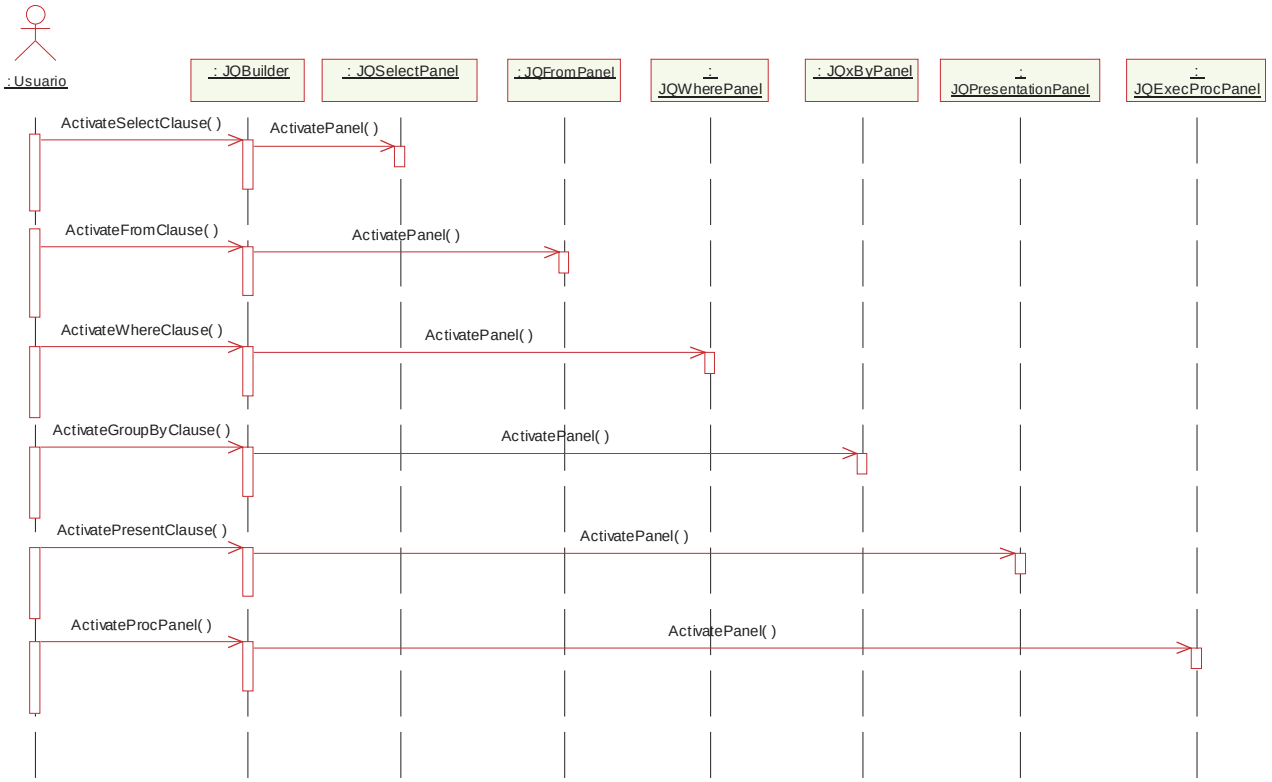

Figura 36 - Iniciando a criação de uma consulta

A seqüência de inserção de uma nova função em uma lacuna qualquer de uma cláusula (Figura 37) se inicia pelo acionamento de um botão de **lacuna,** ou *placeholder*. A seguir, as especializações da classe de botão de lacuna **JQButton** acionarão os métodos de ativação (**JQActivateXXXPanels()**) do módulo principal **JQBuilder,** que irá acionar as classes de painel **XXX** correspondentes. Por exemplo, um botão **JQButtonLog** ativará todos os painéis do grupo **JQLogicalPanel**. O botão também passará o TAD correspondente, que servirá de parâmetro de ativação para as funções a serem ativadas dentro do painel, àquela lacuna.

O último passo do acionamento do botão de lacuna será configurar o seu objeto de nó de referência (**RefNode**) da estrutura de consulta(**QWQuery**) como ponto corrente de inserção de função. Desta forma, qualquer nova função selecionada será inserida a partir desse nó.

Após a seleção da lacuna, o usuário seleciona uma das funções disponíveis acionando qualquer um dos botões da classe **JQWButtonPanelFunc,** que estarão nos painéis de seleção. O método de ativação (**actionPeformed**) do botão obterá, primeiramente, o nó de referência corrente para inserção da nova função. A seguir, um novo nó, obtido a partir de uma cópia do nó de consulta correspondente que o botão carrega, será inserido no nó de referência. O último passo é remontar o painel de consulta, agora incluindo esta nova função inserida. O método **RebuildCurrentPanel** obtém a árvore correspondente à sentença sendo construída no momento (**getCurrentQWStatement**), e depois chama o método recursivo de remontagem de painéis da estrutura **XWSQLStatement**. Este método percorrerá, recursivamente, todos os nós da cláusula correspondente ao painel corrente de construção e remontará toda a consulta.

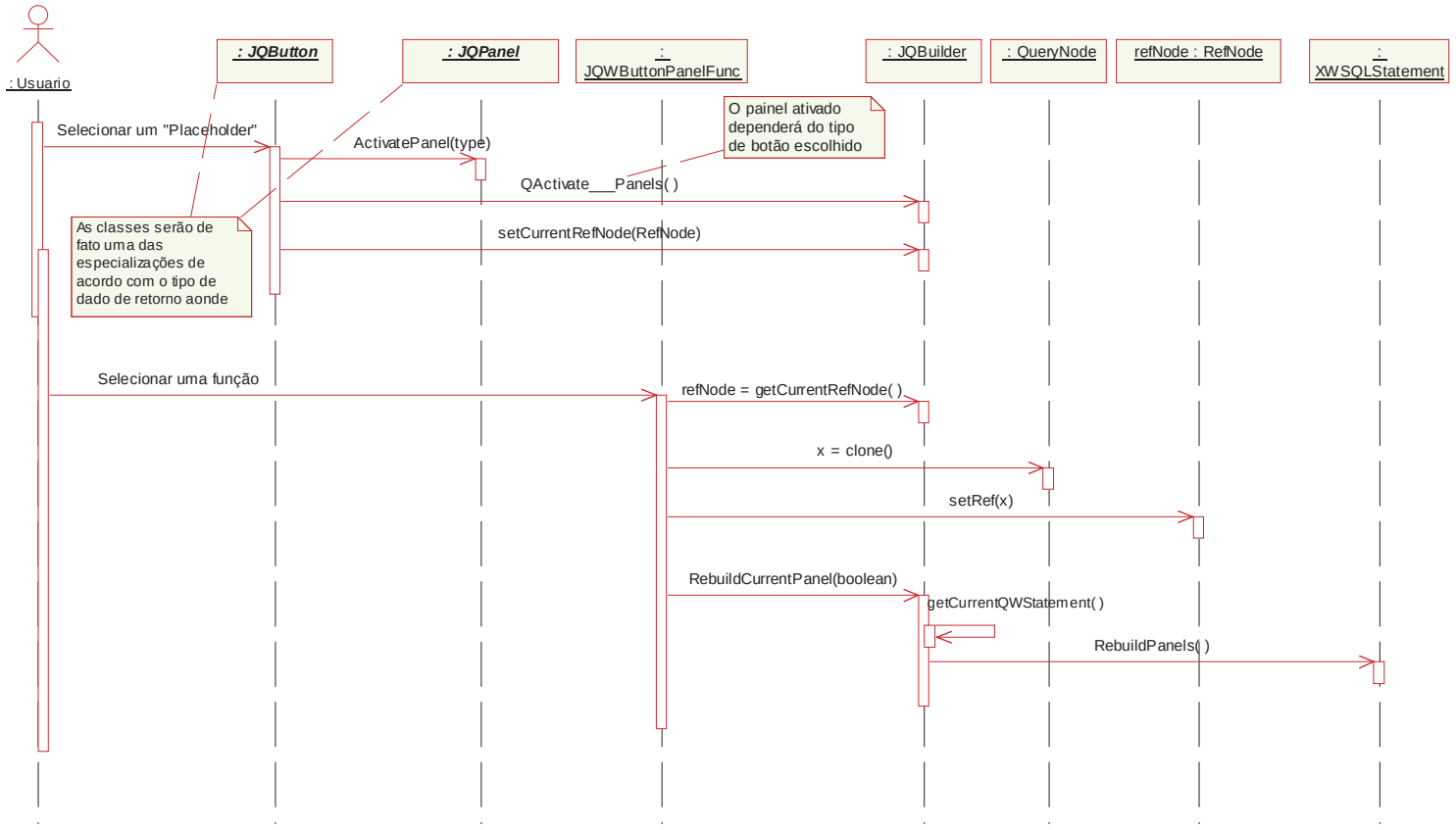

Figura 37 - Seleção e inserção de uma função no painel de construção

A Figura 38 mostra um exemplo da seqüência de eventos para a montagem de um dos painéis de construção. O método **RebuildPanels,** da estrutura de consulta **XWQuery,** chama o método de montagem de painel **BuildPanel** da sub-árvore da cláusula correspondente, passando a referência para o *listener* de conexão com o painel de construção da cláusula. Cada nó usará o *listener* para inserir elementos no painel. No exemplo mostrado na Figura 38, o nó da estrutura correspondente à função "Q1 AROUND Q2" primeiro insere um texto no painel, e depois obtém a referência para o nó inserido na lacuna de "Q1". Se esta referência não for nula, o método de montagem do painel para este nó é chamando recursivamente (**Q1.BuildPanel**). Se houvesse outra função, digamos, f1(z) inserida em Q1, o painel estaria sendo montado como "(f1(z)) AROUND Q2". Caso a referência seja nula, um botão de lacuna é inserido no painel de acordo com o tipo de dado correspondente (TAD, TPD,TADset etc.), como podemos ver no método **InsertTPDButton** do *listener*.

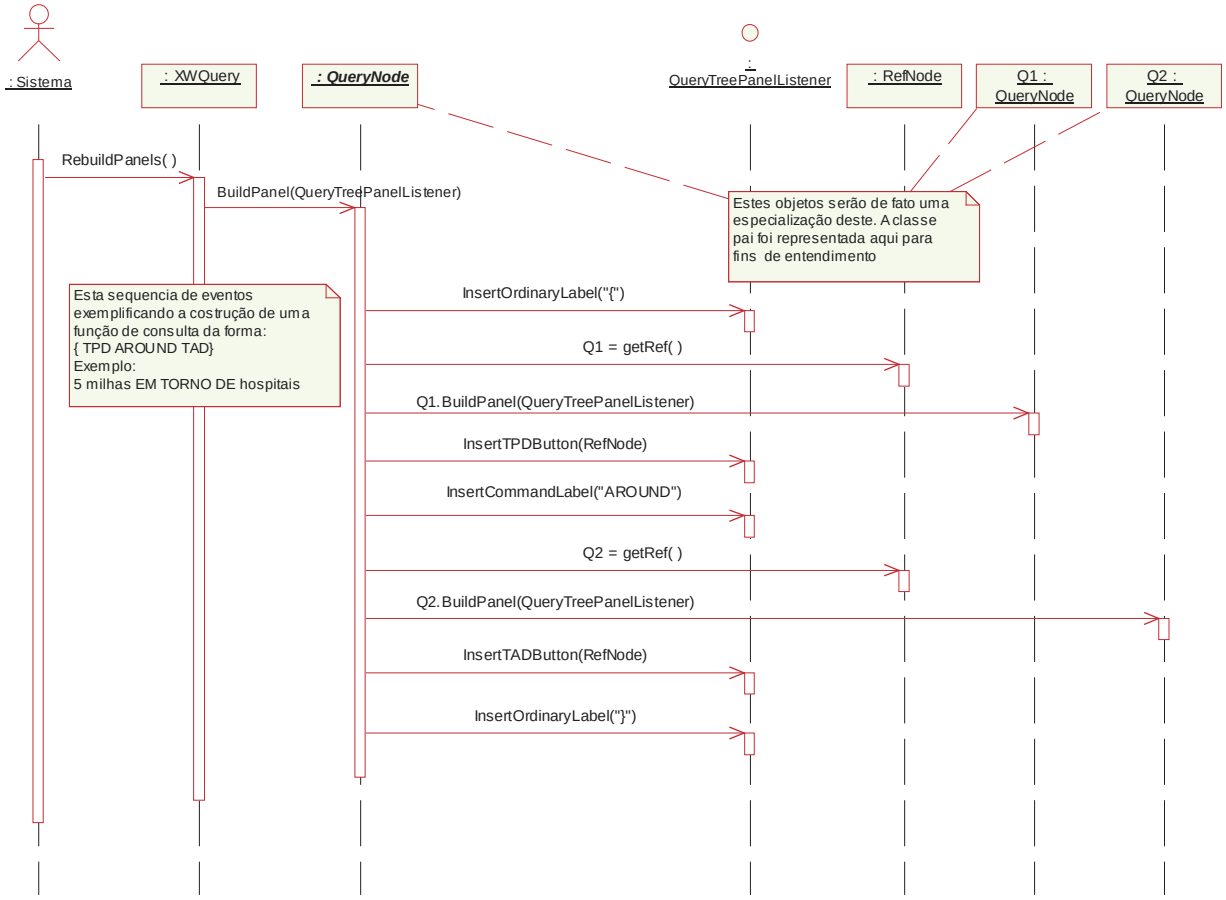

Figura 38 - Inserção de uma nova função no painel de construção

A execução de uma consulta, após completada a sua construção, é bastante simples. Existem duas possíveis seqüências (Figura 39). A primeira envia a consulta diretamente do módulo construtor de consultas para os visualizadores através do *listener* de conexão correspondente (seção 6.2.2). A segunda envia a consulta para o processador de consultas interativas (**QueryTunerPane**). Este módulo recebe a estrutura de consulta e converte os valores nominais ou variáveis definidos pelo usuário como caixas de preenchimento, que podem ser alteradas de forma flexível. A consulta é então enviada aos visualizadores.

.

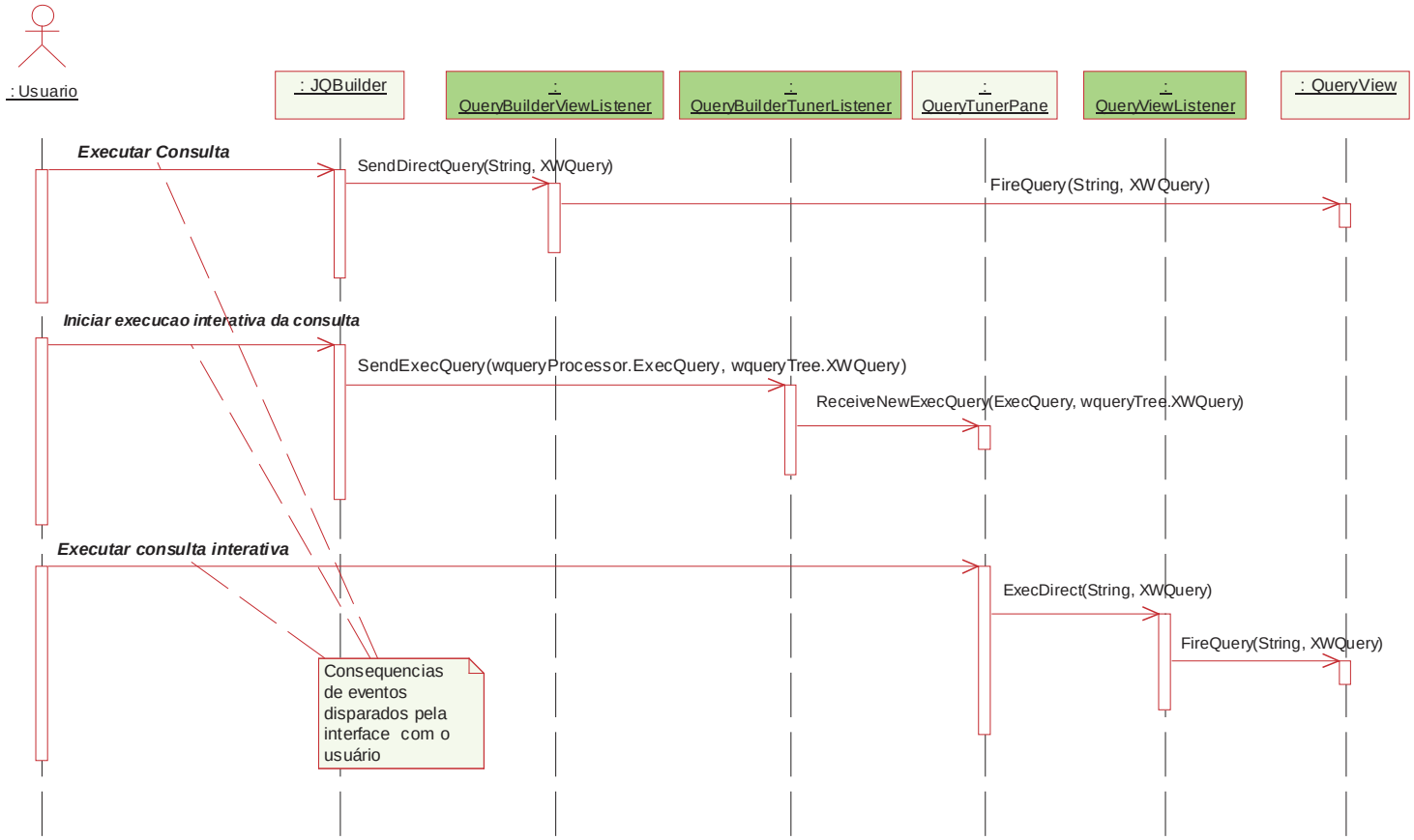

Figura 39 - Execução e visualizaçãao de uma consulta

6.2.4 – **Módulo de Implementação do *Framework* 5-Alpha**

Este módulo é responsável por construir a ferramenta de consulta e fornecer informações aos módulo de administração do banco de dados sobre os TADs e seus catálogos (metadados).

As seções A e C do descritor XML do *framework* **5-Alpha**, ou seja, catálogos e TADs do modelo, são enviados para o módulo de administração do banco de dados que cria os metadados enviados para o resto do sistema

As seções B e D correspondem a funções de suporte à álgebra relacional. Cada uma será mapeada em um nó da estrutura de armazenamento de consultas (**XWQuery**). Para cada função, um nó é criado e associado a um botão da classe **JQButtonPanelFunc** (Figura 37). Quando uma nova função, correspondente a este nó, precisar ser criada, uma clonagem deste nó será feita e acrescentada na estrutura de consulta.

A Figura 40 mostra um exemplo de mapeamento de algumas funções do descritor XML do *framework* em nós de consulta. A organização das funções no *framework* e, conseqüentemente, no descritor refletem muito mais a semântica de utilização das funções, e não da estrutura de armazenamento. Por este motivo, o mapeamento não é tão direto. Funções de conversão, por exemplo (**FPSC1**), podem ser mapeadas em quase todas as classes de retorno, dependendo do atributo "**RetType**", definido no descritor. A razão disto é que o descritor se destina a facilitar e flexibilizar a construção do modelo, e não a implementação interna.

Os procedimentos de *data warehousing* possuem um mapeamento único para a classe **ProcNode,** relacionada à classe **XWSQLProcedure** (Figura 33). Procedimentos não possuem retorno, desta forma, não fazem parte do conjunto dos nós já apresentados. Ppor outro lado, eles receberão todo tipo de parâmetro. As árvores construídas para os procedimentos terão sempre um nó **ProcNode** como raiz e possíveis folhas de nós da hierarquia da classe **QueryNode**.

A montagem do módulo de construção é feita através de *listeners* de conexão. O módulo de implementação conhece uma interface para um *listener*, que é implementado pelo módulo de construção. Esta interface possui métodos para criação de botões, indicando seu tipo de retorno, categoria, parâmetros de entrada e atributos extras necessários para criação do nó de consulta.

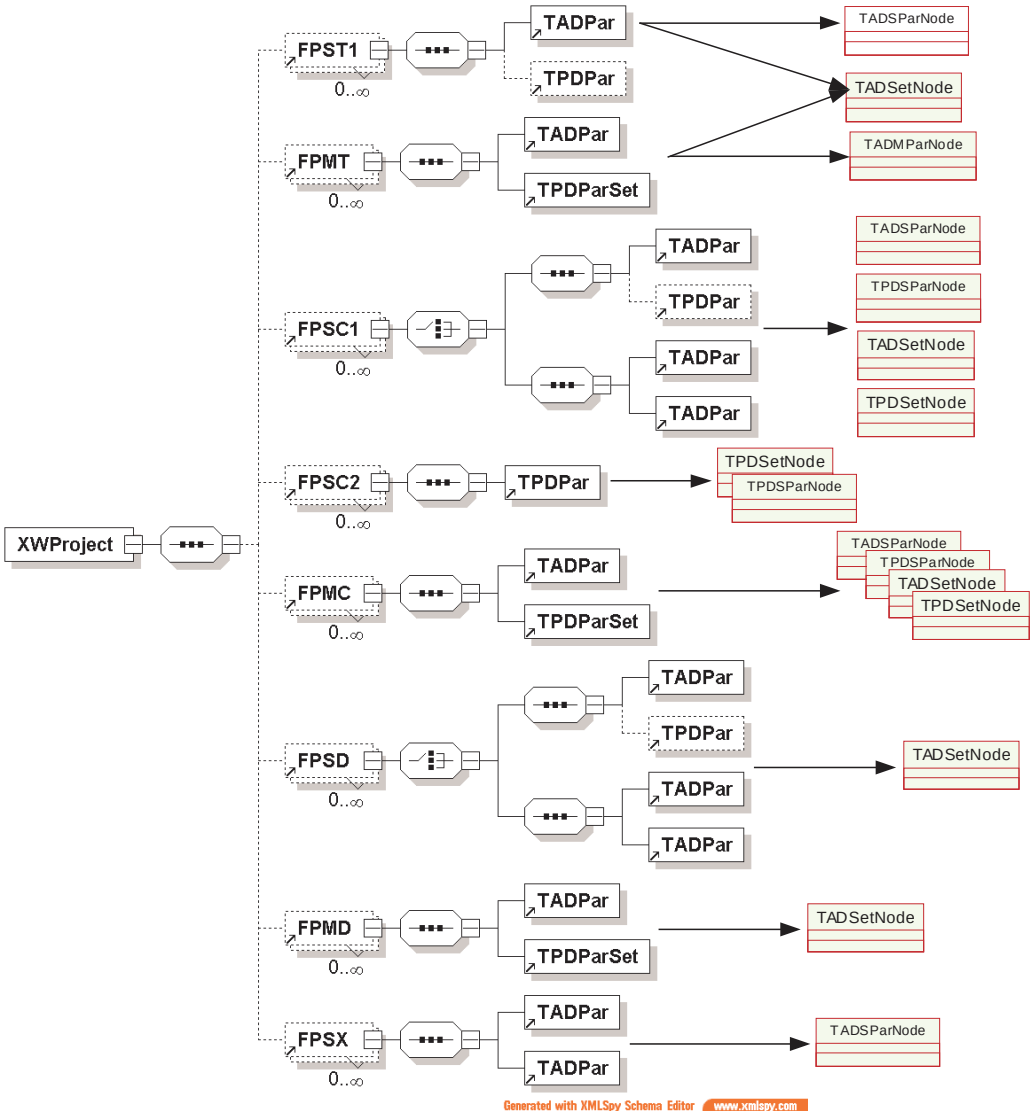

Figura 40 - Exemplo de mapeamento do esquema XML do *framework* - Projeção

6.2.5 – Processador Interativo de consultas

A função principal deste módulo é alterar os parâmetros de uma consulta ou procedimento, e interativamente executá-las e enviá-las para os visualizadores. A cada alteração no valor dos parâmetros variáveis da consulta, esta é automaticamente re-executada.

Este módulo é composto por três painéis principais (Figura 41): ajustes (*Adjust Controls*), controles (botões nos painéis à direita e à esquerda) e visão SQL (*SQL View*). O painel de ajustes lista as caixas de edição correspondentes a cada parâmetro nominal variável da consulta. Além dos valores poderem ser editados diretamente, dois botões, um para incremento (⇑) e outro para decréscimo (⇓), fazem alterações de 10% (quando

os valores são numéricos) e automaticamente enviam para execução e para os visualizadores. O formato completo da consulta pode ser observado no painel de visão SQL (*SQL View*).

Os comandos no painel à direita fornecem controle de armazenamento em disco, execução, remoção e criação de botões associados à consulta. O botão "*Store*" armazena e converte uma consulta sendo manipulada em um botão que é inserido do lado esquerdo do painel. Se desejarmos recarregar a consulta ou executá-la, basta acionar o botão e a mesma passa para a manipulação principal do módulo. Podemos ter um infinito número de botões no lado esquerdo, cada um correspondendo a uma consulta. Todas as consultas carregadas no módulo são armazenadas em disco, em um pacote único, em XML(Figura 42).

Este módulo permite ainda a criação de procedimentos armazenados (**Gen.Proc**) ou visões (**Gen.View**) a partir da consulta corrente.

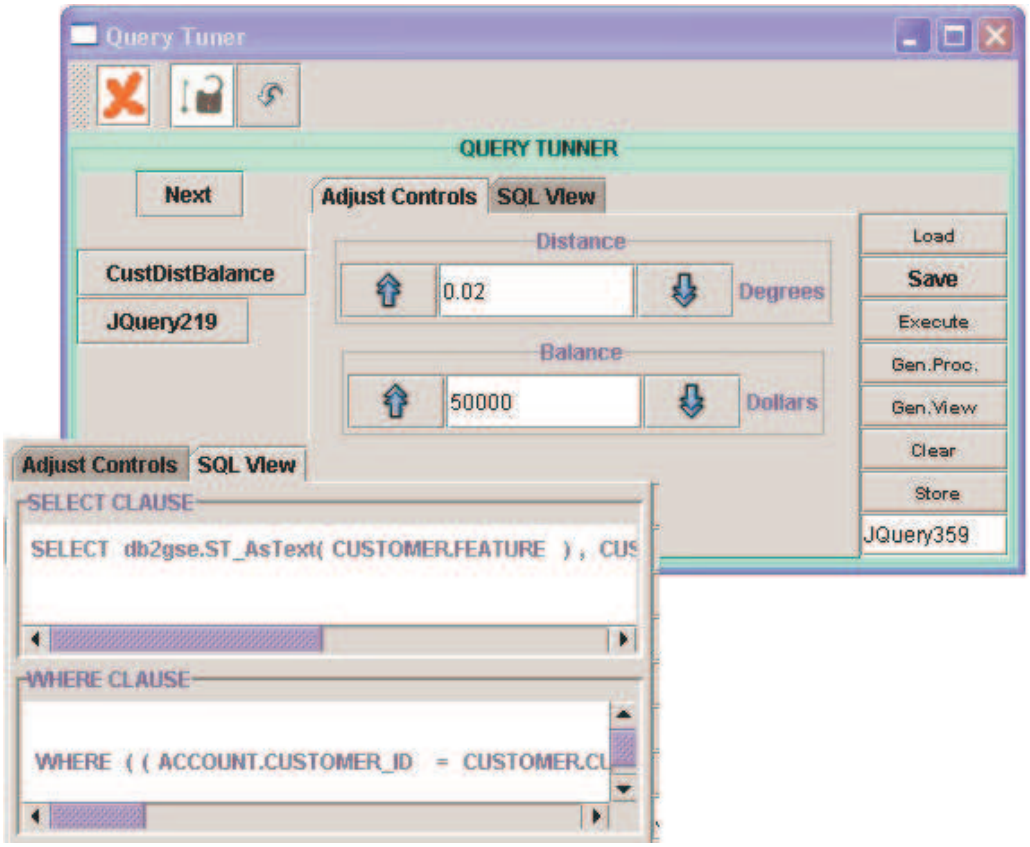

Figura 41 - Sintonizador de consultas

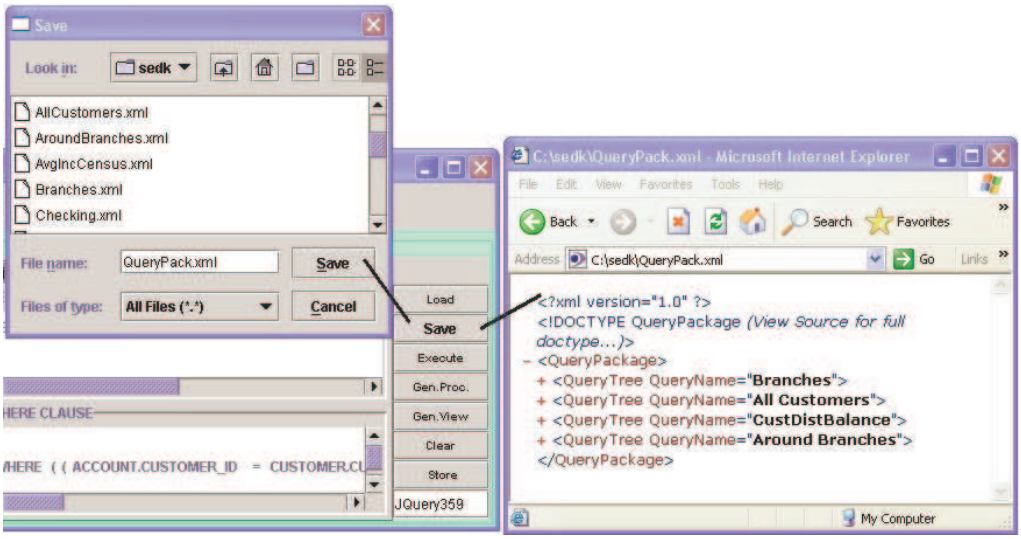

Figura 42 - Recursos do sintonizador de consultas

6.2.6 – Visualizadores

Os módulos de visualização, ou visualizadores, implementam uma classe abstrata geral que é conhecida dos *listeners* de conexão. Todos os visualizadores possuem o mesmo comportamento: receber a consulta, executá-la e mostrar os resultados. A estrutura de consulta **XWQuery** é enviada com informações sobre os TADs retornados, e poderá ser usada pelo visualizador para decidir a melhor forma de apresentar as consultas.

O visualizador de tabelas é utilizado para qualquer TAD, sendo este complexo ou não. Os valores dos TADs são, em última instância, representados com um texto indicativo ou pelo identificador único do objeto. Assim, qualquer consulta envolvendo qualquer TAD pode ser visualizada por este módulo.

6.3 – Conclusão

Mostramos um sistema capaz de gerar uma ferramenta de *data warehousing* para qualquer TAD. O recurso principal que possibilita esta geração é o descritor XML, criado a partir do *framework* **5-Alpha,** apresentado no Capítulo 4 e Capítulo 5. O descritor contém todas as informações necessárias para criação de uma interface inteligente e flexível, capaz de guiar a construção de consultas e procedimentos de forma consistente e livre de erros. O sistema gerador apresentado contém todos os fundamentos necessários para geração de ferramentas com ampla capacidade de análise de *data warehousing*. A sua arquitetura aberta permite ainda um crescimento capaz de atender futuras necessidades e requisitos dos setores de negócios.

Capítulo 7 – Data Warehouse *Espacial*

7.1 – Arquitetura de Suporte

A arquitetura para implementar um sistema de *data warehousing* espacial de categoria C^3 (veja seção 3.3.6) requer vários suportes dentro da linguagem SQL, bem como funcionalidades dentro do banco de dados e assistentes externos ao banco para criação e visualização das consultas. Os trabalhos existentes (Bedard 1999,Chorel 2001,Helmer 2001) não apresentam uma metodologia consistente para definir tal arquitetura. Iremos agora utilizar a metodologia e o *framework* apresentados no Capítulo 4 para mostrar a construção do sistema. Os passos a serem seguidos serão os apresentados na seção 4.2.6 e podemos ainda discutir brevemente os seguintes elementos adicionais de construção, referentes às particularidades dos sistemas de bancos de dados espaciais:

1 - **Tipos definidos pelo usuário** – A construção dos TADs, neste caso, tipos de dados espaciais, dentro da linguagem SQL utilizará o **DB2 Spatial Extender**. Ele contém um conjunto completo de tipos de dados espaciais que seguem o padrão internacional da OGC (Open GIS Consortium e Ryden 1999)

2 - **Funções definidas pelo usuário (***UDFs***) e Procedimentos** (*Stored Procedures*) – Cada tipo estará associado a um conjunto de funções e procedimentos para os diversos tipos de manipulação espacial, que irão compor a extensão espacial do SQL. O **DB2 Spatial Extender** já possui um conjunto razoavelmente completo de funções e procedimentos compatíveis com o padrão da OGC. Utilizaremos o *framework* **5-Alpha** para classificarmos as funções e os procedimentos existentes, e também para mostrar um conjunto de interesse ainda não existente no produto.

3 - **Tabelas de metadados** – O **Data Warehouse Center**, uma ferramenta de *data warehousing* do DB2, possui suporte a metadados, mas não para TADs. Esta é uma das razões de termos optado pelo armazenamento dos metadados em tabelas à parte. Por outro lado, o **DB2 Spatial Extender** já possui um catálogo próprio para armazenamento das informações sobre as entidades espaciais do banco de dados. Nosso trabalho mostrará algumas extensões que também permitem acrescentarinformações sobre o modelo, funcionalidades adicionadas dinamicamente, transformadores e relacionamentos hierárquicos entre as entidades espaciais.

4 - **Módulos de carga dinâmica (*DLLs*)** – Os únicos módulos de carga dinâmica atualmente suportados pelo **DB2 SE** são para geocodificação. De todo modo, já existe um geocodificador padrão incluído no produto.

5 - **Ferramenta de *Data Warehousing*** – O **Data Warehouse Center** do **DB2** (IBM Corporation 2002b) possui um recurso para invocar programas definidos pelo usuário, contudo, ele não possui suporte de baixo nível para operar com TADs. Por isso, a ferramenta de *data warehousing* foi desenvolvida como um pacote independente a partir do sistema apresentado no Capítulo 6.

6 - **Visualizadores** – A ferramenta de *data warehousing* será responsável por enviar o resultado da consulta para os sistemas de visualização capazes de apresentar mapas, levando em conta sistemas de referência, escalas etc.

A Figura 43 mostra a inclusão dos elementos acima citados, usando a metodologia do *framework* **5-Alpha**, dentro do processo de *data warehousing*. Podemos verificar que o suporte para tipos de dados espaciais não só estende, como também se integra ao longo de todos os componentes do processo. Na seção seguinte, veremos a utilização do *framework* **5-Alpha** para definição de todos estes componentes.

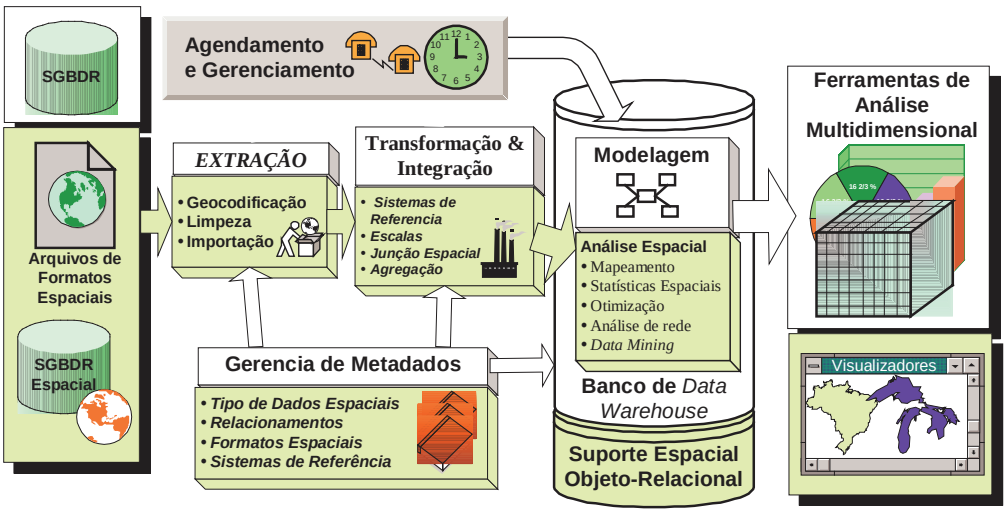

Figura 43 – Arquitetura de *data warehouse* espacial

7.2 – A instância 5-Alpha-*S* do *framework* 5-Alpha

O primeiro passo para construção é classificar os produtos da arquitetura de suporte escolhida (**DB2 Spatial Extender**)de acordo com o *framework* **5-Alpha**. Este será o guia para construção da instância **5-Alpha-S,** que representa o modelo de *Data Warehousing* estendido para tipos de dados espaciais ***S***. Mostraremos aqui como

encaixar o suporte já existente neste *framework* e identificar as áreas que necessitem de expansão, alem disso iremos mostrar exemplos de partes do descritor XML para a instância **5-Alpha-S.**

Existem poucos produtos que oferecem suporte espacial ao banco de dados, entre eles o **DB2 Spatial Extender,** o **Informix Spatial Datablade** e o **Oracle Spatial Cartrige**. Os dois primeiros seguem os padrões do OGC e são, portanto, fundações certificadas para uma arquitetura de *data warehouse* estendido.

A descrição geral da instância **5-Alpha-*S*,** para um conjunto de tipos de dados espaciais ***S*** de um modelo de *data warehouse* estendido ***S***DW, é dada pelas cinco seções **A**, **B**, **C**, **D** e **E**:

[A-0-0] - $\boldsymbol{S}W_{tipos}$ – Conjunto de TADs que representam tipos de dados espaciais. Devemos escrever toda a hierarquia de tipos de dados espaciais, contidos no sistema de suporte, no formato **5-Alpha,** descrito na seção 4.3. É importante observar que estes tipos estarão incluídos em todos os sistemas (B,**X**,**Y**), como explicado na seção 3.3.6.

[B-0-0] - $\boldsymbol{S}W_{transparentes}$ – Os padrões do OGC não sugerem uma semântica específica para tais operadores, porém eles são especificados em um conjunto de funções especializadas, como de igualdade, por exemplo. A sobrecarga destes operadores não é encontrada em nenhum sistema de suporte conhecido e são, de fato, opcionais em sistemas (C,**X**,**Y**).

[C-0-0] - $\boldsymbol{S}W_{MD}$ – **Suporte de metadados** – Deve conter, além de funções para sua manipulação, um catálogo de todos os atributos espaciais do banco de dados, do seu sistema de referência, dos formatos de importação etc.. Devemos classificar neste grupo todas as tabelas do sistema de suporte que contém estas informações, assim como todas as funções de manipulação destas tabelas, como, por exemplo, a de registro e criação de tipos de dados espaciais, ou de qualquer TAD criado para suporte (sistemas de coordenadas etc.).

[D-0-0] - $\boldsymbol{S}W[\boldsymbol{F}_{Suporte\ Relacional}]$ – As funções de suporte relacional já existentes no sistema escolhido para suporte (presentes em todos os sistemas (C,**X**,**Y**)) devem ser devidamente classificadas usando a metodologia explicada na seção 4.3.2. Tomemos como exemplo a função de cálculo de área de Buffer em torno de elementos espaciais:

ST_Geometry ST_Buffer(ST_Geometry g, Double dist)

Primeiramente verificamos o tipo de retorno e parâmetros da função nas áreas da Tabela 12 da seção 4.2.4. O tipo de retorno é um TAD, o que nos leva à primeira coluna da tabela. Temos dois parâmetros, mas o parâmetro de tipo convencional (*Double*) pode ser um atributo de alguma entidade do modelo, ou pode ser desconsiderado se for apenas um valor nominal. Isto nos leva à primeira ou à sexta linha, que indicarão funções de suporte à projeção ou à combinação. Podemos então considerar esta como uma função de conversão, transformação ou manipulação de propriedades de dados. A escolha entre estas funções dependerá da sua semântica. Esta função não extrai nenhuma propriedade específica do parâmetro de entrada (*g*) ou mesmo dá acesso a qualquer de seus atributos. A diferença entre as funções de conversão e transformação se dará se tivermos mudança de tipos ou não, respectivamente. Neste caso, uma avaliação imediata e ingênua dos tipos de parâmetro e retorno indicará uma função de transformação de suporte à projeção. Devemos contudo considerar que, no modelo objeto-relacional, uma referência para um tipo pode, de fato, esconder objetos de classes-filho. Neste último caso temos que analisar as especificações da função.

[E-0-0] - $\boldsymbol{S}$W[$\mathbf{P}_{\text{processamento}}$] = {$\mathbf{PE}_{\text{Extração}}$, $\mathbf{PT}_{\text{Transformação}}$, $\mathbf{PA}_{\text{Análise}}$} – Estas funcionalidades comporão o sistema de *data warehouse* espacial. As arquiteturas necessárias para fornecer suficiente suporte para estas funcionalidades se encontram em sistemas da categoria (C,C,**X**). Por isso, a implementação completa do *framework* **5-alpha-*S*** irá certamente requerer a construção de várias funcionalidades.

7.3 – Descrição Detalhada da instância 5-Alpha-S

A instância do modelo apresentado no Capítulo 4, para *data warehousing* estendido para dados espaciais, está descrita abaixo e na seção seguinte. Alguns dos itens já fazem parte das ferramentas utilizadas neste trabalho, enquanto outros devem ser incorporados. Usaremos a seguinte a notação para representá-los[4]:

dwgse_TAD(*ADT*) – Tipo Abstrato de Dado,

dwgse_Função(*UDF*) – Função definida pelo usuário ou

dwgse_SP(*Stored_Procedure*) – Procedimento Armazenado

[4] Os itens descritos sem procedimento ou funcionalidade são apresentados apenas para formalização do *framework, já que* decidimos não ser necessária a sua implementação neste trabalho.

O conjunto completo de classificação de todo o produto de suporte utilizado (**DB2 SE**) se encontra no Apêndice C. Neste capítulo, incluiremos apenas os itens principais que mostrem a utilização da metodologia do *framework*, a qual poderá ser verificada nas falhas da seqüência de códigos da lista de elementos do *framework*.

7.3.1 – Tipos de Dados Espaciais

Os tipos de dados abaixo estão disponíveis através do **DB2 SE** (IBM Corporation 1998,IBM Corporation 2002a) e dão o suporte básico à manipulação de dados espaciais. Eles refletem as especificações do OGC. A hierarquia entre os tipos de dados espaciais do DB2 pode ser vista na Figura 44 e o Exemplo 15 mostra o descritor XML. Iremos descrever aqui apenas alguns deles, mas a descrição completa pode ser verificada no Apêndice C e Apêndice D. Indicaremos também quando as funções pertencerem à versão 7 ou à versão 8 (Tabela 6 e Tabela 7, na seção 3.2.2).

[A-0-0] - $\boldsymbol{S}W_{tipos}$ – TADs que representam os dados espaciais.

[A-1-0] - *db2gse.ST_Geometry* – Este é o tipo genérico e abstrato de dado espacial. Ele será a interface para todos os tipos especializados e servirá de parâmetro para todas as funções de manipulação.

[A-1-1] - *db2gse.ST_Point* – Tipo básico de ponto bidimensional no espaço

[A-1-2-1] - *db2gse.ST_LineString* – Tipo básico de linha poligonal. Os segmentos de reta devem ser entendidos como um caso particular de linha poligonal.

[A-1-3-1] - *db2gse.ST_Polygon* – Tipo básico de polígono definido por uma linha poligonal fechada, e ainda uma lista de polígonos interiores excluídos.

[A-2-0] - *db2gse.coordref* – Representa um sistema de referência de coordenadas.

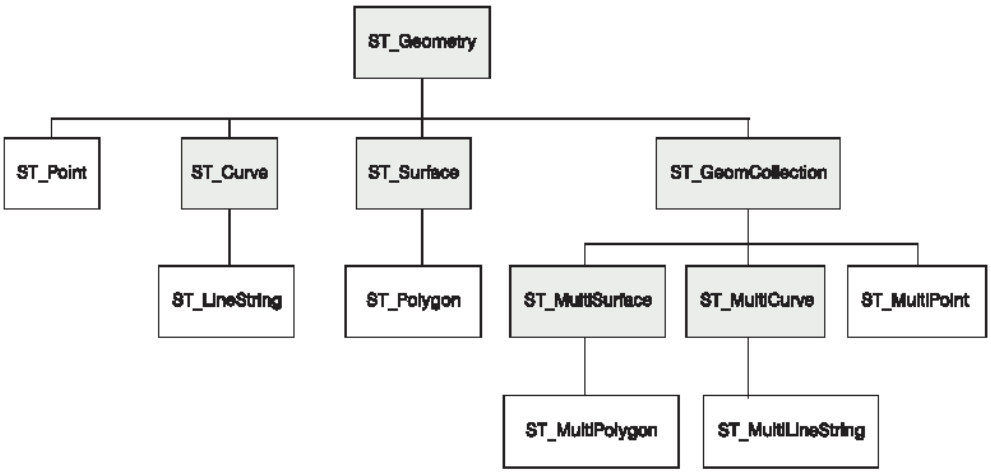

Figura 44 - Hierarquia de tipos do DB2 SE

```
<SectionA>
  <XWTADType SCHID="SCH01" name="ST_Geometry"        typeID="A-1-0">
    <XWTADType SCHID="SCH01" name="ST_Point"          typeID="A-1-1"/>
    <XWTADType SCHID="SCH01" name="ST_Curve"          typeID="A-1-2">
        <XWTADType SCHID="SCH01" name="ST_LineString"    typeID="A-1-2-1"/>
    </XWTADType>

  <XWTPDType name="Integer" typeID="TPDINT"/>
  <XWTPDType name="VARCHAR" typeID="TPDSTR" ></XWTPDType>
  <XWTPDType name="Double" typeID="TPDREAL" ></XWTPDType>
</SectionA>
```

Exemplo 15

7.3.2 – Operadores transparentes

[B-0-0] - ***S***$W_{transparentes}$ – OPERADORES TRANSPARENTES – O produto escolhido para este trabalho (**DB2 Spatial Extender**) não suporta esta categoria de funcionalidades e não a consideramos necessária neste trabalho, contudo achamos interessante definir algumas funções simples para manipular com variáveis lógicas mapeadas através de inteiros:

[B-1-0] - Conectivos Lógicos – Todas estas funções terão a simples finalidade de entender o retorno de funções cujo retorno booleano se faz através de valores inteiros 1 ou 0. O Exemplo 16 mostra um pedaço do descritor XML para esta seção:

[B-1-1] - **Conector AND – ST_AndInt –** Código simplificado: (((**X**)=1)*AND*((**Y**)=1))

[B-1-2] - **Conector OR – ST_OrInt –** Código simplificado: (((**X**)=1)*OR*((**Y**)=1))

[B-1-3] - **Conector AND – ST_NotInt –** Código simplificado: (((**X**)<>1))

```
 <SectionB>
        <XWAnd TrueValue="1" FalseValue="0" FUNCID="B-1-1"
                 RetType="TPDBOOL" BaseName="ST_AndInt">
               <BOOLPar typeID="TPDINT" />
               <BOOLPar typeID="TPDINT" />
        </XWAnd>
…
 </SectionB>
```

Exemplo 16

7.3.3 – Gerência e Suporte de Metadados

[C-0-0] - ***S***W_{MD} – Os metadados atualmente existentes no DB2 SE cobrem apenas detalhes individuais dos elementos (colunas espaciais, sistemas de referência, geocodificadores), e deixam bastante a desejar no que se refere ao processo de ETL espacial ou de criação de modelos.

[C-1-0] - ***S***$W_{CATÁLOGOS}$ – Os tipos de catálogos disponíveis no **DB2 SE** não são completos. Classificaremos os elementos existentes e utilizaremos o modelo de catálogo fornecido pelo *framework* para aqueles não existentes.

[C-1-1] - **Tipos Abstratos de Dados** – Este item consistirá no conjunto de tabelas para armazenamento de todos os Tipos de Dados Espaciais existentes no suporte do **DB2 SE**.

[C-1-1-1] - ***Colunas Espaciais*** (*DB2GSE.GEOMETRY_COLUMNS*) – Contém uma lista com o nome das tabelas e das colunas que possuem tipos de dados espaciais.

[C-1-1-2] - ***Sistemas de Coordenadas*** (*DB2GSE.COORD_REF_SYS*) –– Indicam como cada coordenada do sistema é representada e armazenada.

[C-1-1-3] - ***Sistemas de Referência***(*DB2GSE.SPATIAL_REF_SYS*) – Indicam como os sistemas de coordenadas irão se relacionar com a posição geográfica dos elementos espaciais

[C-1-2] - ***Modelos*** – Será utilizado o suporte do *framework*.

[C-1-3] - **Hierarquias** – Será usado o suporte do *framework*.

[C-1-4] - **Módulos de Processamento** – Iremos registrar aqui os catálogos de registro de módulos existentes. Entretanto, eles não abrangem a possibilidade de extensão flexível de módulos. O suporte fornecido pelo *framework* será necessário para completar os elementos necessários deste catálogo.

[C-1-4-1] - ***Módulos de Geocodificação*** (*DB2GSE.SPATIAL_GEOCODER*) – Já existente no **DB2 SE**. Eles contêm a lista de módulos de geocodificação. Podemos encontrar uma lista detalhada de seus atributos no manual do **DB2 SE** (IBM Corporation 1998,IBM Corporation 2002a)

[C-1-4-2] - ***Módulos de Importação, Limpeza e Validação*** – Estes módulos, que utilizarão as tabelas do modelo de catálogo fornecido pelo *framework*, armazenarão uma lista de módulos de importação de dados espaciais, como conversão de formatos específicos, limpeza, validação ou comunicação com sistemas legados, como citado na seção 3.1.7.

[C-2-0] - *S*W$_{\text{Manipulação de Catálogos}}$ - Procedimentos de Manipulação de Catálogo – O sistema de suporte do **DB2 SE** não possui uma completa funcionalidade para o registro de cada tipo de elemento. Existe ainda uma diferença de recursos entre a versão 7 e a versão 8 do produto. Por este motivo, classificaremos aqueles existentes em ambas as versões do produto, e descreveremos os não existentes, mas necessários. Alguns dos métodos fornecidos pelo produto também criam novos TADs, além de os registrarem no catálogo de metadados. Contudo, julgamos conveniente registrar estes procedimentos nesta seção.

[C-2-1] - **PCSD**$^{n}_{\text{Registro}}$(Parâmetros) – O sistema de suporte de registro de dados. Nenhum dos procedimentos provenientes do suporte do DB2 SE registra os elementos em um modelo, dado que este produto não dá suporte a modelos. Procedimentos adicionais fornecidos pelo suporte do *framework* devem ser utilizados para este registro.

[C-2-1-1] - ***Criação e registro de modelos*** – Utilizará o suporte do *framework*.

[C-2-1-3] - ***Criação e registro de Sistema de Referência –*** *db2gse.gse_enable_sref*(V7) **–** Cria e registra um novo sistema de referência.

[C-2-1-5] - ***Criação e registro de Coluna Espacial –*** *db2gse.gse_register_layer*(V7) **–** Cria ou registra uma coluna de uma tabela para um tipo de dado espacial qualquer. Se a coluna não existir, ela é criada, caso contrário passa a ser associado àquela coluna um tipo de geometria e sistema de referência. Não associa a um modelo.

[C-2-2] - **PCSD**$^{n}_{\text{Remoção}}$(Parâmetros) – O sistema de suporte de remoção de registros dos catálogos de metadados.

[C-2-2-1] - ***Remoção de modelos*** – Utilizará o suporte do *framework*.

[C-2-2-4] - ***Remover Sistema de Referência –*** *db2gse.gse_drop_srs*(V8) **–** Remove, ou desabilita o sistema de referência.

[C-2-2-5] - ***Remover Col. Espacial –*** *db2gse.gse_unregist_layer*(V7) **–** Suporte do **DB2 SE**. Remove uma coluna espacial do catálogo.

[C-2-3] - **PCSM**$^{n}_{\text{Registro}}$(Parâmetros) – Efetua o registro de módulos no sistema, assim como nos catálogos.

[C-2-3-1] - **Registrar Módulo de Geocodificação** - *gse_register_gc*(V7) **–** Suporte do **DB2 SE**.

[C-2-3-3] - **Registrar Módulo de Geocodificação** - *ST_register_geocoder*(V8) **–** Suporte do **DB2 SE**.

[C-2-3-5] - **Criar/Registrar Módulos –** Função geral para registro de módulos. Suporte do *framework*.

[C-2-4] - **PCSH**$^{n}_{\text{Registro}}$ – **Criar/Registrar Hierarquias –** Função geral para registro de hierarquias. Não existe qualquer procedimento a ser registrado do produto de suporte, por isso será usado o suporte do *framework*.

[C-2-5] - **PCSM**$^{n}_{\text{Remoção}}$(Parâmetros) – Efetua a remoção de um módulo no sistema, assim como nos catálogos.

[C-2-5-1] - **Remover Módulo de Geocodificação** - *db2gse.gse_unregist_gc*(V7) **–** Suporte do **DB2 SE**.

[C-2-5-2] - **Desativar Auto-Geocodificação** - *gse_disable_autogc*(V7) **–** Suporte do **DB2 SE**. Desativar o módulo de geocodificação automática.

[C-2-5-3] - **Remover Módulo de Geocodificação** - *ST_unregister_geocoder*(V8) **–** Suporte do **DB2 SE**.

[C-2-5-4] - **Desativar Auto-Geocodificação** - *ST_disable_autogeocoding*(V8) **–** Suporte do **DB2 SE**. Desativar o módulo de geocodificação automática.

[C-2-5-5] - **Remover Módulos –** Função geral para remoção de módulos do catálogo, bem como do sistema. Suporte do *framework*.

[C-2-6] - **PCSH**$^{n}_{\text{Remoção}}$ – **Remover Hierarquias –** Função geral para remoção de hierarquias do catálogo, bem como do sistema. Não existe qualquer procedimento a ser registrado do produto de suporte, por isso será usado o suporte do *framework*.

[C-3-0] - ***S***W$_{\text{Manipulação de Elementos}}$

[C-3-1] - **FCSC**$^{n}_{\text{Criação}}$(Parâmetros) – A lista de funções abaixo será responsável por criar dinamicamente qualquer elemento do modelo espacial, como pontos, linhas etc. Estas funções também fazem parte do suporte relacional. Todas as funções abaixo, descritas em detalhe no manual do **DB2 Spatial Extender** (IBM Corporation 2002a), e nas definições do padrão da OGC, têm por finalidade criar um tipo de dado espacial a partir de uma descrição, em modo texto, do objeto espacial a ser criado.

Método de Classificação: Primeiramente, devemos avaliar, de acordo com a Tabela 11, os tipos de retorno das funções. Uma parte delas recebe como parâmetro um TPD (elementos individuais e não conjuntos) e retorna um TAD. Todas estas estão classificadas na segunda linha da primeira coluna da Tabela 11. O resto das funções recebe como parâmetros um TPD e um TAD (*db2gse.coordref*). Estas condições nos levariam a um classificação imediata na sexta linha da primeira coluna e, como os TADs são diferentes, indicariam uma conversão. O erro neste raciocínio, mencionado na seção 5.2, é que estes dois TADs não pertencem a mesma hierarquia (ver ***S***W$_{\text{tipos}}$ acima), portanto, o TAD (*db2gse.coordref*) está, de fato, funcionando como um parâmetro de TPD para a função.

[C-3-1-1] - *db2gse.ST_Point db2gse.ST_Point (Double,Double,SRID)*

[C-3-1-2] - *db2gse.ST_Geometry db2gse.ST_GeometryFromText(Varchar(4000), db2gse.coordref)*

[C-3-2] - **FCMC**$^{n}_{\text{Criação}}$(Parâmetros) – O **DB2 SE** não possui qualquer função nesta categoria e não julgamos necessária a sua implementação neste trabalho.

[C-3-3] - **FCSA**$^{n}_{\text{Acesso}}$(***S***W$_{\text{tipo}}$) – A lista de funções abaixo será responsável por acessar partes, ou subcomponentes, de um TAD. Por exemplo, um objeto *círculo* seria composto das coordenadas do seu centro, bem como do seu contorno.

Método de Classificação: Primeiramente, devemos avaliar, de acordo com a Tabela 11, na seção 4.2.3, os tipos de parâmetro e retorno das funções. Temos funções unárias e binárias. Todas elas recebem um TAD como parâmetro e algumas recebem ainda um TPD. Isto nos leva à primeira e à sexta linhas da Tabela 11 da seção 4.2.3. Todas as funções unárias que recebem TADs e retornam TPDs poderão ser funções de conversão, ou *accessors*. Para determinar a diferença, não há outra forma senão analisar a semântica da função. As que retornam TADs também podem ser classificadas pela semântica, mas em alguns casos pode haver dubiedade. No item [C-3-3-4] abaixo, por exemplo, que retorna o anel interior de um polígono, poderia-se argumentar que um novo objeto *ST_LineString* está sendo criado a partir do objeto *ST_Polygon*. Temos que perguntar, contudo, se o anel interior já fazia parte do polígono originalmente, o que é verdade neste caso.

A classificação das funções binárias segue o mesmo princípio, sendo que observamos que o parâmetro de TPD extra serve somente para fornecer informações sobre qual parâmetro do TAD queremos obter.

[C-3-3-1] - *Integer* *db2gse.ST_SRID(db2gse.ST_Geometry)*

[C-3-3-2] - *Integer* *db2gse.ST_NumPoints(db2gse.ST_LineString)*

[C-3-3-3] - *db2gse.ST_LineString* *db2gse.ST_ExteriorRing(db2gse.ST_Polygon)*

[C-3-3-4] - *db2gse.ST_LineString db2gse.ST_InteriorRing(db2gse.ST_Polygon,Integer)*

[C-3-4] - **FCSP**$^{n}_{Propriedade}$(***S***W$_{tipo}$) – Estas funções têm por finalidade calcular e retornar propriedades específicas de um tipo de dado espacial.

Método de Classificação: Todos os casos abaixo pertencem à segunda coluna da primeira linha da Tabela 11, na seção 4.2.3. Contudo, a semântica de *accessor* se aplica a este tipo, pois ele não retorna necessariamente um componente interno do dado espacial, mas calcula um valor específico associado a ele. Em alguns casos, estes valores podem até estar armazenados no objeto espacial por motivo de desempenho, mas sua existência não determina a existência do tipo de dado espacial.

[C-3-4-1] - *Double* *db2gse.ST_Area(db2gse.ST_Surface)*

[C-3-4-2] - *Double* *db2gse.ST_Length(db2gse.ST_Curve)*

[C-3-5] - **FCMA**$^{n}_{Acesso}$(***S***W$_{coluna}$) – O **DB2 SE** não possui qualquer função nesta categoria e não julgamos necessária a sua implementação neste trabalho.

[C-3-6] - **FCMP**$^{n}_{Propriedade}$(***S***W$_{coluna}$) – O **DB2 SE** não possui qualquer função nesta categoria e não julgamos necessária a sua implementação neste trabalho.

Todos os elementos acima devem ser inseridos no descritor XML que pode ser visto em detalhe no Apêndice D

7.3.4 – Funções de Suporte Relacional

[D-0-0] - ***S***W[**F**$_{Suporte\ Relacional}$] = {**F**$_{Seleção}$, **F**$_{Projeção}$, **F**$_{Agregação}$ }

[D-1-0] - ***S***W[**F**$_{Seleção}$] – FUNÇÕES DE SUPORTE À SELEÇÃO

[D-1-1] - **FRS1**$^{n}_{Relação}$(***S***W$_{tipo}$) – Esta categoria de funções avalia alguma propriedade das entidades espaciais, retornando verdadeiro ou falso às mesmas. Por razões de implementação, o DB2 SE retorna valores inteiros para representar valores booleanos (0-Falso e 1-Verdadeiro).

Método de Classificação: Primeiramente, devemos avaliar, de acordo com a Tabela 12, na seção 4.2.4, os tipos de parâmetro e retorno das funções. Apesar de retornar um TPD (Inteiro), verificamos, pelo domínio do resultado{0,1}, que o mesmo retorna valores booleanos. Isto classifica estas funções na última coluna da Tabela 12, da seção 4.2.4, e o fato de todos os parâmetros aqui serem tipos de dados espaciais individuais (não de conjunto) nos leva diretamente à primeira linha da última coluna da Tabela 12.

[D-1-1-1] - *Integer db2gse.ST_IsClosed(db2gse.ST_Curve)*

[D-1-2] - **FRS2**$^{n}_{Relação}$(***S***W$_{tipo}^{1}$, ***S***W$_{tipo}^{2}$ [,Parâmetro]) – Estas funções retornam relacionamentos entre tipos de dados espaciais. Por exemplo, determinar se duas linhas se interceptam ou não.

Método de Classificação: Estas funções utilizam exatamente o processo anterior, onde os resultados representam valores booleanos. A diferença aqui é que temos dois parâmetros de tipo de dado espacial, o que nos leva diretamente à quinta linha da última coluna da Tabela 12, na seção 4.2.4.

[D-1-2-1] - *Integer db2gse.ST_Contains(db2gse.ST_Geometry, db2gse.ST_Geometry)*

[D-1-2-2] - *Integer db2gse.ST_Within(db2gse.ST_Geometry, db2gse.ST_Geometry)*

O DB2 SE não possui as outras funções de relacionamento do *framework* **5-Alpha-*S***. De todo modo, apresentaremos algumas funções relevantes para o suporte de um sistema de *data warehouse* espacial que deveriam ser implementadas.

[D-1-3] - **FRS3**$^{n}_{\text{Relação}}$(***S***W$_{\text{tipo}}$, DW$_{\text{parâmetro}}$) – Estas funções testam um elemento de tipo de dado espacial de acordo com o parâmetro fornecido. Um exemplo desta função poderia ser:

[D-1-3-1] - db2gse.dwgse_HasInteriorRings(ST_Curve, NrRings) – Retornar verdadeiro ou falso se a curva tem pelo menos NrRings.

Todas as categorias de funções abaixo requerem a construção de funções agregadoras definidas pelo usuário. A maioria dos sistemas de bancos de dados, incluindo o banco de suporte escolhido (DB2), não possui este suporte. Para resolver este problema, várias funções que se encaixariam nos grupos abaixo serão construídas através de funcionalidades de transformação.

[D-1-4] - **FRM1**$^{n}_{\text{Relação}}$ (***S***W$_{\text{coluna}}$) – Estas funções testam condições de elementos espaciais, analisando todos de uma vez. Por exemplo:

[D-1-4-1] - db2gse.dwgse_IsSmaller(ST_Geometry) – Retorna verdadeiro para o elemento que for o menor de todos os outros. Apesar de poder olhar para todos os elementos do conjunto de uma vez, esta função deve retornar um valor para cada elemento da *tupla* fornecida na sentença de consulta.

[D-1-5] - **FRM2**$^{n}_{\text{Relação}}$ (***S***W$_{\text{coluna}}{}^{1}$, (***S***W$_{\text{coluna}}{}^{2}$ | ***S***W$_{\text{tipo}}$)[,Parâmetro])) – Este outro grupo de funções tem duas versões. A primeira delas testa as condições que envolvem todos os elementos de um conjunto contra todos os elementos de outro conjunto. Este conjunto de funções não é o mais comum, pois cada teste entre dois elementos requer o conhecimento do produto cartesiano entre todos os elementos de ***S***W$_{\text{coluna}}{}^{1}$ e ***S***W$_{\text{coluna}}{}^{2}$. A outra versão é uma categoria simplificada da anterior. Neste caso, a comparação com o conjunto inteiro é feita elemento a elemento. Temos os seguintes exemplos.

[D-1-5-1] - dwgse_IsClosestDistance(ST_GeometryA^{i}, ST_Geometry B^{j}) – Retornará verdadeiro para os dois elementos A^{i} e B^{j} que tiverem a menor distância entre si que todos os outros.

[D-1-5-2] - dwgse_IsClosest(ST_Geometry A^k, ST_Geometry B) – Comparará cada elemento, ou *tupla* do atributo B, com todos os elementos k da coluna A e retornará verdadeiro se o elemento A^k for o mais perto de B. Note-se que para cada elemento individual da coluna B, todos os elementos k da coluna A são testados.

[D-1-6] - **FRM3**$^{n}_{\text{Relação}}$ ((***S***W$_{\text{coluna}}$ | ***S***W$_{\text{tipo}}$),(Parâmetros | Parâmetro$_{\text{Set}}$)) – Cada função irá retornar verdadeiro ou falso de acordo com o relacionamento entre todos os elementos da coluna ***S***W$_{\text{coluna}}$ e um único elemento ou conjunto de TPD. De fato, esta categoria de funções relaciona um dado espacial com um parâmetro convencional, considerando que pelo menos um deles será em forma de conjunto. Não há exemplos no produto de suporte. Propomos o seguinte exemplo:

[D-1-6-1] - dwgse_IsSmaller(ST_Geometry, minSize) – Este exemplo é uma extensão do item [D-1-4-1] acima, mas aqui o menor item retorna verdadeiro, desde que ele seja menor que minSize.

[D-2-0] - ***S***W[**F**$_{\text{Projeção}}$] – FUNÇÕES DE SUPORTE À PROJEÇÃO

[D-2-1] - **FPST**$^{n}_{\text{Transformação}}$(***S***W$_{\text{tipo}}{}^{1}$,[DW$_{\text{parâmetro}}$]) – Estas funções serão responsáveis por receber um tipo de dado espacial como parâmetro e retornar um dado do mesmo tipo com os valores, mas não a estrutura interna, modificados.

Método de Classificação: Cada uma das funções pode ser classificada na primeira ou na sexta linha da primeira coluna da Tabela 12, na seção 4.2.4. A semântica das funções já elimina a possibilidade de *accessors*. Resta-nos escolher entre transformação ou conversão. A diferença aqui irá depender do funcionamento específico de cada função. Se utilizarmos apenas a assinatura das funções como base de classificação, não haverá dúvidas. Porém, se alguma destas funções estiver retornando um tipo diferente de dado espacial, escondido sobre a classe de hierarquia mais alta, temos que classificá-la como conversão. Por conveniência, classificamos todas as funções pela assinatura.

[D-2-1-1] - *db2gse.ST_Geometry db2gse.ST_Boundary(db2gse.ST_Geometry)*

[D-2-2] - **FPSC**$^{n}_{\text{Conversão}}$(***S***W$_{\text{tipo}}{}^{1}$,[***S***W$_{\text{tipo}}{}^{2}$| DW$_{\text{parameter}}$]) – Estas funções retornam um tipo de dado espacial ou primitivo diferente do que foi fornecido.

Método de Classificação: Todas as cinco funções abaixo retornam tipos primitivos, recebendo tipos de dados espaciais, ou então retornam um tipo de dado espacial

diferente do recebido. Esta semântica mostra claramente, na primeira e segunda colunas da primeira linha da Tabela 12, na seção 4.2.4, a classificação destas funções como conversão.

[D-2-2-1] - *db2gse.ST_Polygon* *db2gse.ST_Polygon (db2gse.ST_LineString, db2gse.coordref)*

[D-2-2-2] - *BLOB(1m)* *db2gse.ST_AsBinary(db2gse.ST_Geometry) –*

Todas as funções a seguir também serão classificadas como conversão, dada as suas características de mudança de tipo de dado espacial de parâmetro para o retorno. A diferença aqui é que a semântica delas pode dar margem a dúvidas. De um modo geral, a regra é verificar se não estamos obtendo propriedades e nem atributos de um objeto, e se o tipo de dado espacial retornado está sendo realmente diferente do que foi passado.

[D-2-2-6] - *Double* *db2gse.ST_Distance(db2gse.ST_Geometry, db2gse.ST_Geometry)*

[D-2-2-7] - *db2gse.ST_Point* *db2gse.ST_Centroid(db2gse.ST_Surface)*

[D-2-2-8] - *db2gse.ST_Point* *db2gse.ST_Centroid(db2gse.ST_MultiSurface)*

[D-2-3] - **FPSX**$^{n}_{Combinacao}$(***S***$W_{tipo}{}^{1}$, ***S***$W_{tipo}{}^{2}$,[$DW_{parâmetro}$]) – Estes métodos retornam um tipo de dado espacial produzido através da combinação de outros dois tipos de dados.

Método de Classificação: Pela primeira coluna da quinta linha da Tabela 12,na seção 4.2.4, não há margem para dúvida da classificação destas funções. A conversão ou transformação estão implícitas no processo de combinação. Desta forma, não temos que nos preocupar com o retorno escondido de tipos de dados espaciais diferentes da assinatura da função.

[D-2-3-1] - *db2gse.ST_Geometry* *db2gse.ST_Union(db2gse.ST_Geometry, db2gse.ST_Geometry)*

O resto das classes de funções de suporte à projeção não possui implementação no DB2 SE devido à impossibilidade de construção de funções agregadoras definidas pelo usuário .

[D-2-4] - **FPSD**$^{n}_{DrillDown}$(***S***W_{tipo}, [,$DW_{parâmetro}$]) – Este grupo de funções cria um nível de detalhe a mais a partir de um tipo de dado espacial. Exemplo:

[D-2-4-1] - db2gse.dwgse_Grid(ST_Geometry A,M,N) – Retorna uma grade M×N de objetos espaciais retangulares dentro da área do objeto A.

[D-2-5] - **FPMT**$^{n}_{\text{Transformação}}$(***S***W$_{\text{tipo}}$, DW$_{\text{parâmetro}}$) – As funções de transformação aplicadas ao conjunto total de elementos de uma tabela são bastante genéricas e fornecem inúmeras semânticas. Apenas a título de exemplo, podemos citar:

[D-2-5-1] - dwgse_ScaleUp(ST_Geometry,Rate) – Esta função recebe um objeto espacial e aumenta sua escala e número de pontos de acordo com a maior distância da área coberta por todos os outros objetos dentro do mesmo conjunto. O aumento de escala também é regulado pela taxa Rate.

[D-2-6] - **FPMC**$^{n}_{\text{Conversão}}$(***S***W$_{\text{tipo}}$[,DW$_{\text{parâmetro}}$]) – Este grupo é similar ao anterior, mas realiza conversões de um tipo de dado espacial para outro, como converter um círculo para um quadrado. Vamos ver um exemplo:

[D-2-6-1] - dwgse_FullConvexUnion(ST_Geometry) – Função que cria um grande objeto espacial convexo, que englobe todos os outros objetos do mesmo conjunto.

[D-2-7] - **FPMD**$^{n}_{\text{DrillDown}}$(***S***W$_{\text{tipo}}$[,DW$_{\text{parâmetro}}$]) – Este é, na verdade, um subgrupo do item [D-2-4]. Só que esta função abre a possibilidade de que todos os elementos de um atributo guiem o detalhamento gerado.

[D-3-0] - ***S***W[**F**$_{\text{Agregação}}$] – FUNÇÕES DE SUPORTE À AGREGAÇÃO – O **DB2 SE** não possui nenhuma função de suporte à agregação. A razão disto é que o DB2, até a versão atual (V8.1), não permite a construção de funções agregadoras definidas pelo usuário(ex., AVG(X), COUNT(X) etc.). Vamos sugerir aqui algumas funções de suporte à agregação que seriam de extrema utilidade para um sistema de *data warehousing* espacial.

Os dois tipos de agregação a seguir correspondem a funções que recebem conjuntos de objetos espaciais e retornam elementos isolados ou conjuntos de tipos primitivos de dados, respectivamente.

[D-3-1] - **FA1**$^{n}_{\text{Agregação}}$(***S***W$_{\text{coluna}}{}^{1}$, [***S***W$_{\text{tipo}}$ | ***S***W$_{\text{coluna}}{}^{2}$ | DW$_{\text{parâmetro}}$])

[D-3-1-1] - dwgse_SumArea(ST_Geometry) – Neste exemplo, a função irá retornar a soma das áreas de todas as geometrias da coluna fornecida.

[D-3-2] - **FA2**$^{n}_{Agregação}$(***S***W$_{coluna}$^{1}, [***S***W$_{tipo}$ | ***S***W$_{coluna}$^{2} | DW$_{parâmetro}$])

[D-3-2-1] - dwgse_MinDistances(ST_Geometry gc1, ST_Geometry gc2) - A título de exemplo, esta função retorna as menores distâncias entre todas as geometrias em "gc1" e em "gc2".

As categorias de funções a seguir diferirão novamente no retorno de elementos individuais ou de conjuntos, sendo que, neste caso, retornam elementos objetos espaciais.

[D-3-3] - **FA11**$^{n}_{Agregação}$(***S***W$_{coluna}$,[DW$_{parâmetro}$])

[D-3-3-1] - dwgse_ AggObject(ST_Geometry gc1) - Esta função retorna um objeto espacial, agregando todos os objetos espaciais da coluna de objetos espaciais "gc1".

[D-3-4] - **FA12**$^{n}_{Agregação}$(***S***W$_{coluna}$,[DW$_{parâmetro}$])

[D-3-4-1] - dwgse_ CalculaCentroides(ST_Geometry gc1, MinDist) – Esta função retorna o conjunto centróides dos pontos que estejam dentro da distancia mínima "MinDist" fornecida.

[D-3-5] - **FA21**$^{n}_{Agregação}$(***S***W$_{coluna}$^{1},[***S***W$_{tipo}$ | ***S***W$_{coluna}$^{2}],[Parâmetros])

[D-3-5-1] - dwgse_ Nearest(ST_Geometry gc1, ST_Geometry g2) – Esta função irá retornar o objeto espacial do conjunto "gc2" mais próximo de cada objeto espacial "g2" fornecido.

[D-3-6] - **FA22**$^{n}_{Agregação}$(***S***W$_{coluna}$^{1},[***S***W$_{tipo}$ | ***S***W$_{coluna}$^{2}],[Parâmetros])

[D-3-6-1] - dwgse_NearestSet(ST_Geometry gc1, ST_Geometry g2, N) – Esta função irá retornar o conjunto do "N" objetos espaciais dentro do conjunto gc1 mais próximos de "g2"

O exemplo do descritor para os elementos da seção D, assim como a seção E a seguir se encontra no Apêndice D.

7.3.5 - Funções de processamento de *Data Warehouse* Espacial

[E-0-0] - ***S***W[**P**$_{processamento}$] = {**PE**$_{Extração}$, **PT**$_{Transformação}$, **PA**$_{Análise}$}

Nas categorias A, B, C e D da instância **5-Alpha-S**, nosso trabalho foi em grande parte de classificação das funções de suporte já existentes. Na categoria E, muito pouco existe a ser aproveitado, dado que um produto completo de *data warehouse* espacial (pertencente à classe C3) não existe de fato. Nesta categoria, apenas exemplos significativos de algumas classes serão apresentados como propostas do que deveria ser implementado para dar suporte ao *data warehouse* espacial. Incluímos também algumas implementações que foram feitas no escopo deste trabalho. Os itens de códigos [X-X-X] não implementados não serão apresentados aqui.

[E-1-0] - ***S***W[**PE**$_{\text{Extração}}$]

[E-1-1] - IMPORTAÇÃO

[E-1-1-1] - **PEI**$^{n}_{\text{Importação}}$([***ArqExt*** | ***TabList***$^{B}_{\text{fonte}}$],***TabList***$^{A}_{\text{destino}}$[,Parâmetros) – Procedimentos que converterão os diversos formatos de arquivo para os tipos de dados espaciais do banco (padrão OGC).

[E-1-1-1-1] - **Importar arquivos** *shape* – *gse_import_shape*(V7) – Esta função, implementada pelo **DB2 Spatial Extender**, serve para carregar arquivos no formato padrão de SIGs. Este formato é utilizado pelos SIGs da **ESRI** (**ArcInfo**, **ArcGIS**, **ArcView** etc.), os mais utilizados.

[E-1-1-1-3] - **Importação geral** – db2gse.dwgse_SpatialImport – Esta funcionalidade tem a finalidade de realizar uma importação mais geral, onde o tipo de formato é automaticamente determinado e a função de importação correspondente é invocada.

[E-1-1-2] - **PEA**$^{n}_{\text{Agregação}}$([***ArqExt*** | ***TabList***$^{B}_{\text{fonte}}$], ***TabList***$^{A}_{\text{destino}}$[,Parâmetros]) – A semântica deste grupo de funções é realizar uma pré-agregação nos dados de um arquivo, ou tabela, antes de inseri-los no banco. Apesar da grande variedade de funções, um dos exemplos que poderíamos apresentar seria o de arquivos contendo listas de pontos que precisam ser agregados para produzir um único objeto. Este é o caso de arquivos gerados por GPS de coleta dinâmica, queproduzirão, no final, um contorno, ou linhas de demarcação política. Não implementaremos qualquer procedimento desta categoria nesta instância do *framework*.

[E-1-1-3] - **PES**$^{n}_{\text{Sequencializador}}$([***ArqExt*** | ***TabList***$^{B}_{\text{fonte}}$],***TabList***$^{A}_{\text{destino}}$[,Parâmetros]) – A semântica espacial da importação de dados espaciais em seqüência não difere muito da tradicional. Um exemplo bastante convencional seria similar à agregação, onde temos

que pegar coordenadas de vários arquivos em seqüência para construção de um objeto espacial. Não implementaremos qualquer procedimento desta categoria nesta instância do *framework*.

[E-1-2] - PREPROCESSAMENTO

[E-1-2-1] - $\mathbf{PEP}^{n}_{\text{Preprocessamento}}(\boldsymbol{TabList}^{A}_{\text{fonte}}, \boldsymbol{TabList}^{A}_{\text{destino}}[,\text{Parâmetros}])$

[E-1-2-1-1] - **Geocodificar** – db2gse.dwgse_DataGeocode – Iniciar o módulo de geocodificação e extrair as informações espaciais implícitas nos dados convencionais, convertendo-as em tipos de dados espaciais (seção 3.1.7).

[E-1-2-2] - $\mathbf{PEL}^{n}_{\text{Limpeza}}(\boldsymbol{TabList}^{A}_{\text{fonte}}, \boldsymbol{TabList}^{A}_{\text{destino}}[,\text{DW}_{\text{parâmetro}}])$ – **Limpeza Espacial ou *Spatial Cleansing*** – Processo de conversão de dados aonde "o mesmo pedaço de informação deve estar representado da mesma forma" (Gray e Watson 1998). Por exemplo, Av. ou Avenida (ver capítulo 3). Não implementaremos qualquer procedimento desta categoria nesta instância do *framework*, dado a necessidade de utilização de produtos não acessíveis durante este trabalho.

[E-1-2-3] - $\mathbf{PEV}^{n}_{\text{Validação}}(\boldsymbol{TabList}^{A}_{\text{fonte}}, \boldsymbol{TabList}^{A}_{\text{destino}}[,\text{DW}_{\text{parâmetro}}])$ – A maioria dos exemplos de validação espacial estará bastante relacionado ao processo de geocodificação. Os endereços fornecidos para geocodificação precisam ser validados, mesmo após a limpeza. Em muitos casos, são fornecidos números e nomes de ruas inexistentes.

[E-2-0] - $\boldsymbol{S}\text{W}[\mathbf{PT}_{\text{Transformação}}]$

[E-2-1] - $\mathbf{PTG1}^{n}_{\text{Transformação}}(\boldsymbol{TabList}_{\text{fonte}}, \boldsymbol{TabList}_{\text{destino}}[,\text{DW}_{\text{Lista de Parâmetros}}])$ – Forma geral da funcionalidade de transformação. Vai representar os procedimentos de transformação gerais, que não pertençam às categorias apresentadas nesta seção.

[E-2-2] - $\mathbf{PTG2}^{n}_{\text{Transformação}}(\boldsymbol{TabList}_{\text{fonte}}, \boldsymbol{TabList}_{\text{destino}}, \text{SQL}_{\text{list}}[,\text{DW}_{\text{Lista de Parâmetros}}])$ – Este procedimento implementa a execução de uma sentença SQL qualquer, contendo funções de manipulação espacial, sobre a tabela-fonte e disponibiliza o resultado nas tabelas de saída. O procedimento abaixo, implementado para este fim, recebe as tabelas de entrada, saída, e mapeamento de parâmetros, bem como o SQL que deve ser executado:

[E-2-2-1] - db2gse.dwgse_SpatialSQL

[E-2-3] - SELEÇÃO

[E-2-3-1] - **PTSR**$^{n}_{\text{Seleção}}$(***TabList***$_{\text{fonte}}$,***TabList***$_{\text{destino}}$,**FRS_**$^{k}_{\text{Relação}}$[,DW$_{\text{Parâmetros}}$]) – Esta categoria contém os procedimentos de suporte de seleção espacial, já descritos na **seção D**, para realizar seleções e junções, e inserir o resultado nas tabelas de saída. Podemos citar o seguinte procedimento de exemplo:

[E-2-3-1-1] - dwgse_InsideJoin(TabFonte, TabDestino, ST_Geometry A, ST_Geometry B) – Realiza a junção de duas tabelas com elementos de dados espaciais, que estejam associados pela relação de contenção, ou seja, retornam as *tuplas* cujo elemento A estivesse contido dentro do elemento B.

[E-2-3-2] - **PTMR**$^{n}_{\text{Seleção}}$(***TabList***$_{\text{fonte}}$, ***TabList***$_{\text{destino}}$, **FRM_**$^{k}_{\text{Relação}}$ [,DW$_{\text{Parâmetros}}$]) – Este grupo de procedimentos realiza seleções e junções, que utilizam funções de relacionamento agregado. A maior importância deste grupo de funções é que elas, muito constantemente, substituirão as funções de manipulação de agregação.

[E-2-3-2-1] - dwgse_NearJoin(TabFonte, TabDestino, ST_Geometry A, ST_Geometry B) – Produz a junção de duas tabelas de objetos espaciais, onde os objetos no conjunto A estejam mais próximos de cada elemento no conjunto B. Se tivermos uma tabela com clientes e outra com filiais de um banco, a tabela resultante possuiria os cliente que moram mais perto de cada filial. Este procedimento tem que fazer cálculos de relacionamento entre cada elemento de A contra todos os elementos de B. Isto pode ser feito usando as funções de categoria **FRM2**$^{n}_{\text{Relação}}$(***S***W$_{\text{coluna}}$, ***S***W$_{\text{tipo}}$), se o sistema possuir este suporte, caso contrário, como no sistema de suporte escolhido (**DB2 Spatial Extender**), a operação de junção terá que ser feita diretamente na implementação da função de transformação.

[E-2-4] - SEPARAÇÃO

[E-2-4-2] - **PTSS2**$^{n}_{\text{Separação}}$(***TabList***$_{\text{fonte}}$, ***TabList***$_{\text{destino}}$, **FCSA**$^{k}_{\text{Acesso}}$ [,DW$_{\text{Parâmetros}}$])

[E-2-4-2-1] - dwgse_Medidas(TabFonte,TabDestino,ST_Geometry) – Este procedimento de exemplo produzirá uma tabela com as áreas e perímetros de todos os objetos espaciais.

[E-2-5] - CONCATENAÇÃO

[E-2-5-1] - **PTSU**$^{n}_{\text{Concatenação}}$(***TabList***$_{\text{fonte}}$, ***TabList***$_{\text{destino}}$, **FPSX**$^{k}_{\text{Combinação}}$[,DW$_{\text{Parâmetros}}$])

[E-2-5-1-1] - dwgse_Integrar(TabFonte ta, TabFonte tb, TabDestino, ST_Geometry a, ST_Geometry b) – Este procedimento de exemplo produzirá uma tabela com a união das geometrias **a** e **b** das respectivas tabelas **ta** e **tb**.

[E-2-6] - SUMARIZAÇÃO

[E-2-6-1] - $\mathbf{PTSA1}^{n}{}_{\text{Sumarização}}(\boldsymbol{TabList}_{\text{fonte}},\boldsymbol{TabList}_{\text{destino}},\mathbf{FA11}^{k}{}_{\text{Agregação}}[,\text{DW}_{\text{Parâms.}}])$

[E-2-6-1-1] - dwgse_AggObject(TabFonte, TabDestino, ST_Geometry A, ST_Geometry B) – Produzem uma tabela de valores agregados dos elementos de A que estiverem contidos dentro do elemento B. Neste caso, a agregação não se dá nos objetos espaciais em si, mas nos atributos (colunas) associados, que serão agregados. Este é um processo de seleção seguido de uma agregação.

[E-2-6-2] - $\mathbf{PTMA1}^{n}{}_{\text{Sumarização}}(\boldsymbol{TabList}_{\text{fonte}},\boldsymbol{TabList}_{\text{destino}},\mathbf{FA12}^{k}{}_{\text{Agregação}}[,\text{DW}_{\text{Parâmetros.}}])$

[E-2-6-2-1] - dwgse_ZoomOut(TabFonte, TabDestino, ST_Geometry, Scale) – Traz todos os dados espaciais para uma escala menor, onde alguns grupos de objetos espaciais serão agregados, se transformando em um só, e alguns poderão ser convertidos de tipo. Suponhamos, por exemplo, que um contorno pode ser transformado em um ponto, ou um conjunto de pontos pode ser transformado em um ponto. Esta função está classificada nesta categoria, porque pode ser necessário utilizar todos os objetos espaciais do conjunto para determinar um objeto agregado que seja produzido.

[E-2-7] - TRANSFORMAÇÃO

[E-2-7-1] - $\mathbf{PTST1}^{n}{}_{\text{Transformação}}(\boldsymbol{TabList}_{\text{fonte}},\boldsymbol{TabList}_{\text{destino}},\mathbf{FPST}^{k}{}_{\text{Transformação}}[,\text{DW}_{\text{Parâmetros.}}])$ – Veja o seguinte exemplo:

[E-2-7-1-1] - dwgse_Converte(TabFonte, TabDestino, newRef) – Faz conversão de sistema de referência e traz todos os dados espaciais para a mesma escala. O sistema de coordenadas já associa os dados espaciais a uma certa escala, mas isto pode ser útil tanto para ajustar os valores para comparação quanto para junção de objetos em diferentes escalas

[E-2-8] - ENRIQUECIMENTO

[E-2-8-1] - $\mathbf{PTSE1}^{n}{}_{\text{Enriquecimento}}(\boldsymbol{TabList}_{\text{fonte}},\ \boldsymbol{TabList}_{\text{destino}},\ \mathbf{FPSC}^{k}{}_{\text{Conversão}}\ [,\text{DW}_{\text{Parâmetros}}])$ – Procedimentos que processam cada elemento da *tupla*, aumentando a complexidade

dos objetos espaciais através da conversão dos mesmos para outros tipos de objetos espaciais.

[E-2-8-1-1] - dwgse_ZoomIn(TabFonte, TabDestino, ST_Geometry, Scale) – Diminui a escala do conjunto de objetos espaciais para o atributo Scale, de maneira que alguns objetos, como pontos, podem se converter em polígonos (contornos).

[E-2-8-2] - $\mathbf{PTSE2}^{n}_{\text{Enriquecimento}}(\boldsymbol{TabList}_{\text{fonte}}, \boldsymbol{TabList}_{\text{destino}}, \mathbf{FCSC}^{n}_{\text{Criação}}\ [,DW_{\text{Parâmetros}}])$ – Podemos denominar este grupo de procedimentos como "enriquecimento por construção". Este grupo tem por função converter tipos primitivos de dados, mais usualmente coordenadas, em objetos espaciais.

[E-2-8-2-1] - dwgse_GeocodeLatLong(TabFonte,TabDestino,Lat,Long) – Este seria o exemplo mais clássico, apesar de raramente implementado desta forma, de um enriquecimento por construção. A função faz a leitura de valores de latitude e longitude, e os converte em objetos espaciais.

[E-2-8-4] - $\mathbf{PTME2}^{n}_{\text{Enriquecimento}}(\boldsymbol{TabList}_{\text{fonte}}, \boldsymbol{TabList}_{\text{destino}}, \mathbf{FCMC}^{n}_{\text{Criação}}[,DW_{\text{Parâmetros}}])$ – Esta é uma variação do processo de enriquecimento por construção do item anterior qui é necessário computar todos os elementos do conjunto para poder produzir os objetos espaciais. Um exemplo bastante comum é um conjunto de coordenadas de latitude e longitude de uma coluna se transformar em um único objeto de linha poligonal.

[E-2-8-5] - $\mathbf{PTME3}^{n}_{\text{Enriquecimento}}(\boldsymbol{TabList}_{\text{fonte}}, \boldsymbol{TabList}_{\text{destino}}, \mathbf{FPMC}^{k}_{\text{Conversão}}\ [,DW_{\text{Parâmetros}}])$ – A variação deste grupo de funções em relação as $\mathbf{PTS1}^{n}$ é que aqui temos que considerar todo o conjunto de elementos para poder construir cada novo objeto espacial "enriquecido".

[E-2-8-5-1] - dwgse_CreateStoreArea(TabFonte, TabDestino, ST_Geometry, WeightCol, minSize) – Cria um *buffer* em torno de objetos espaciais, ponderando o tamanho dele (*minSize*) de acordo com os valores fornecidos. Este exemplo de enriquecimento poderia ser melhor categorizado como funcionalidade de análise de negócios, contudo expressa adequadamente um processo de enriquecimento envolvendo análise de conjunto.

[E-2-9] - CONVERSÃO

[E-2-9-1] - $\mathbf{PTSC1}^{n}_{\text{Conversão}}(\boldsymbol{TabList}_{\text{fonte}}, \boldsymbol{TabList}_{\text{destino}}, \mathbf{FPSC1}^{k}_{\text{Conversão}}\ [,DW_{\text{Parâmetros}}])$

[E-2-9-1-1] - dwgse_ConvertToXML(TabFonte, TabDestino, ST_Geometry a) – Este exemplo de procedimento de conversão converte todos os objetos da coluna "a" da tabela–fonte, em formato texto, para o XML da tabela de destino.

[E-2-10] - *DRILLDOWN*

[E-2-10-2] - $\mathbf{PTMD}^{n}_{DrillDown}$ (***TabList***$_{fonte}$, ***TabList***$_{destino}$, $\mathbf{FPMD}^{n}_{DrillDown}$ [,$DW_{Parâmetros}$])

[E-2-10-2-1] - dwgse_UniformGrid(TabFonte, TabDestino, ST_Geometry a, nrCel) – Este procedimento vai produzir uma grade retangular, de "nrCel" células, dentro da área coberta por cada objeto espacial "a" na tabela-fonte.

[E-3-0] - ***S***W[$\mathbf{PA}_{Análise}$]

O próximo conjunto representa as funções usadas pelo sistema de análise multidimensional espacial (MOLAP). As tarefas básicas definem os predicados espaciais, que realizam a junção entre as dimensões espaciais e pré-computam as agregações. Este conjunto de funções é parte de todos os sistemas de categoria (C,**X**,C). Estamos apresentando estas funções por formalidade, dado que estas funcionalidades serão automaticamente geradas pela ferramenta de *data warehousing* para TADs, apresentada no Capítulo 6, através da definição de hierarquia descrita no arquivo XML na seção 4.3.2

[E-3-1] - ***S***$HW_{hierarquia}$

[E-3-2] - $\mathbf{PAHS1}^{n}_{Hierarquia}$(***S***$W_{coluna}{}^{1}$, ***S***$W_{coluna}{}^{2}$ [,$\mathbf{FRS2}^{n}$[,$\mathbf{FPSD}^{n}$]])

[E-3-3] - $\mathbf{PAHS2}^{n}_{Hierarquia}$(***S***$HW_{hierarquia}$, ***S***$W_{coluna}$[, $\mathbf{FRS2}^{n}$ [,$\mathbf{FPSD}^{n}$]])

[E-3-4] - $\mathbf{PAHM1}^{n}_{Hierarquia}$(***S***$W_{coluna}{}^{1}$, ***S***$W_{coluna}{}^{2}$[,$\mathbf{FRM2}^{n}$[,$\mathbf{FPMD}^{n}$]])

[E-3-5] - $\mathbf{PAHM2}^{n}_{Hierarquia}$(***S***$HW_{hierarquia}$, ***S***$W_{coluna}$[,$\mathbf{FRM2}^{n}$[,$\mathbf{FPMD}^{n}$]])

[E-3-6] - $\mathbf{PAHM3}^{n}_{Hierarquia}$(***S***$W_{coluna}{}^{1}$, ***S***$W_{coluna}{}^{2}$[,$\mathbf{FRM3}^{n}$[,$\mathbf{FPMD}^{n}$]])

[E-3-7] - $\mathbf{PAHM4}^{n}_{Hierarquia}$(***S***$HW_{hierarquia}$, ***S***$W_{coluna}$[,$\mathbf{FRM3}^{n}$[,$\mathbf{FPMD}^{n}$]])

[E-3-8] - $\mathbf{PA}^{n}_{AddData}$(***S***$HW_{hierarquia}$, ***S***$W_{coluna}$, *FunçãoAgg*[,*FunçãoDrillDown*])

[E-3-9] - $\mathbf{PA}^{n}_{Agregação}$(***S***$HW_{hierarquia}$, Nível, $DW_{parâmetro}$)

[E-3-10] - $\mathbf{PA}^{n}_{DrillDown}$(***S***$HW_{hierarquia}$, *Nível*)

[E-3-11] - [**PAG**$_{\text{Análise}}$] – Descreve os procedimentos de análise espacial não convencionais, bem como outras que não as de análise multidimensional (OLAP). Podemos encontrar estas funcionalidades em sistemas (**X**,**Y**,B), como descrito na seção 3.3.6. Os exemplos abaixo ilustram algumas destas funcionalidades desenvolvidas no protótipo:

[E-3-11-1] - **Mapeamento** – Este grupo de procedimentos é utilizado para o desenho e manipulação de mapas. Elas fogem ao escopo do banco de dados por dependerem do cliente ou do sistema de visualização. A ferramenta gerada no Capítulo 6 possui estes recursos independentemente do banco, apenas obtendo os valores dos atributos e os convertendo nos objetos visuais. Na seção 7.4, a seguir, mostraremos o protótipo de ferramenta gerada especialmente para manipulação de dados espaciais.

[E-3-11-2] - **Estatísticas Espaciais** – Procedimentos que avaliam padrões espaciais. Por exemplo, "determinar uma área em torno das minhas filiais proporcional ao volume de recursos nas contas de poupança".

[E-3-11-2-1] - dwgse_CreateStoreArea – Este procedimento, desenvolvido para o protótipo da seção 7.4, reproduz uma das funcionalidades do **ArcView Business Analyst** (ESRI 2001). Ele permite a criação de áreas em torno de objetos proporcionais a algum atributo relacionado da *tupla*, como no exemplo acima citado.

[E-3-11-3] - **Otimização de Negócios** – Estes procedimentos maximizam critérios especiais ou não espaciais, como obter os hospitais mais próximos que retêm algum recurso necessário ao paciente.

[E-3-11-3-1] - dwgse_MarketArea – Este procedimento, também baseado em uma das funcionalidades do **ArcView Business Analyst** (ESRI 2001), é capaz de criar uma área com um percentual mínimo de elementos espaciais em torno de elementos espaciais-pai.

[E-3-11-4] - **Análise de Rede** – Estes procedimentos estão relacionados à topologia espacial e ao relacionamentos entre elementos. A consulta mais comum neste caso é achar o menor trajeto dentro de uma rede de caminhos possíveis.

Finalizamos aqui todo o processo de classificação e análise fornecido pela metodologia de construção e análise da instância **5-Alpha-S** do *framework* **5-Alpha**. O descritor construído, que pode ser encontrado no Apêndice D, será usado como base para a geração da ferramenta de *data warehousing* espacial, mostrada na seção seguinte.

7.4 – Protótipo de *Data Warehousing* Espacial

7.4.1 – Descrição Geral

O sistema SEDK foi um protótipo desenvolvido para demonstrar a implementação do *framework* **5-Alpha** para tipos de dados espaciais utilizando o **DB2 Spatial Extender** como suporte de funções. Ele contém várias ferramentas, painéis e interfaces especializadas para atender requisitos de análise espacial de negócios (Figura 45).

Somente uma parte do *framework* **5-Alpha** foi implementada. O protótipo não incorpora as funcionalidades de *data warehousing* (seção E) na interface com o usuário, contudo os procedimentos armazenados foram implementados. A seção A está incorporada diretamente no módulo de administração e nos metadados. A seção B não foi implementada diretamente, apesar de algumas funções de igualdade entre tipos espaciais ter sido implementada. As funções de manipulação de registro de módulos de catálogo (seção C) não têm suporte no **DB2 Spatial Extender**, e a parte de manipulação de TADs no catálogo está implementada diretamente no módulo de administração. O resto da seção C, no que se refere a construtores e acesso a propriedades, está implementada, mas foi inserida diretamente nos painéis de funções, de acordo com o seu tipo de retorno. A seção D como um todo está suportada neste protótipo, porém as funções de agregação não são suportadas pelo DB2. Portanto, estas funções ou foram removidas do protótipo ou implementadas como funções “virtuais” (veja seção 6.2.3 adiante).

O conjunto completo de módulos do sistema de consulta se compõe de:

- Uma versão modificada do administrador do banco
- Uma implementação do construtor de consultas com nós feitos especificamente para as funções de suporte relacional para os tipos de dados espaciais.
- O processador interativo de consultas.
- O visualizador de tabelas
- O visualizador de Mapas

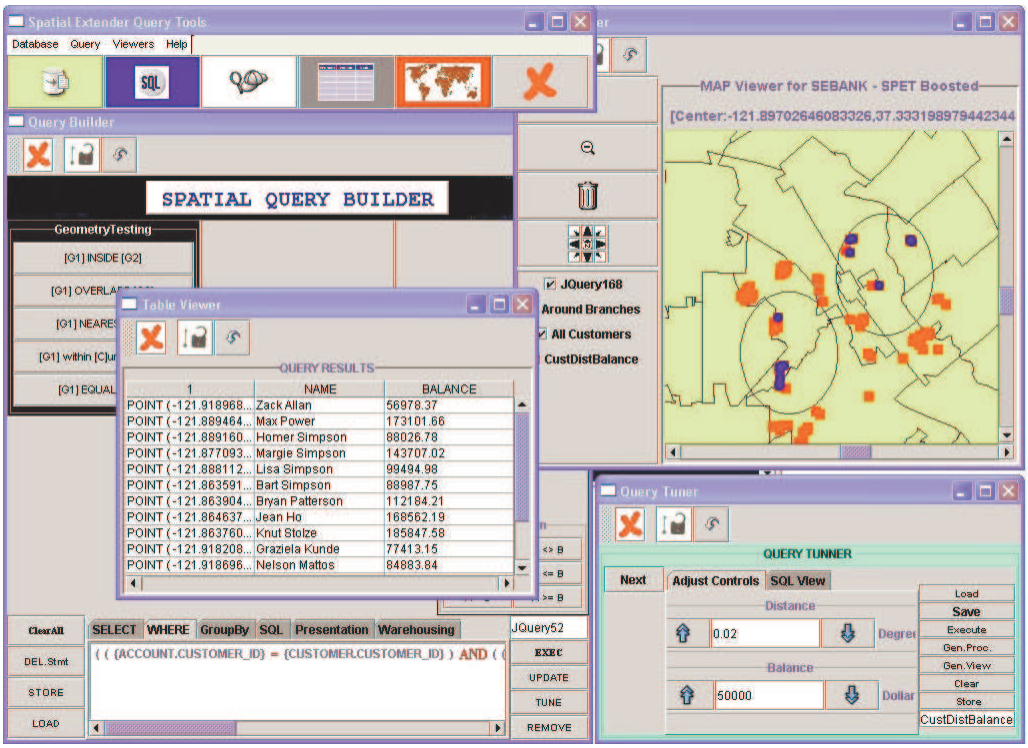

Figura 45 – Pacote completo de gerenciamento de consultas espaciais

7.4.2 – Módulo de Administração do Banco de Dados Espacial

Este módulo foi implementado a partir do módulo de administração básico já definido (seção 6.2.2). Ele possui algumas funcionalidades extras, para ativar e desativar o suporte de dados espaciais do DB2, e ainda criar colunas de dados espaciais e registrá-las no catálogo. Estas funcionalidades fazem parte do *framework* que estariam no módulo de construção, mas por razões de utilização do produto final, foi requerido que estas funcionalidades fossem diretamente implementadas no módulo de administração.

Os tipos de serviços disponíveis na interface com usuário do módulo são (Figura 46):

1) Administração do Banco (**Database Administration)** – Serviços básicos de conexão com o banco.

   a) Ativação (**Enable DB**) - Utiliza o usuário (USER) e a senha (PASSWORD) fornecidos para ativar o **DB2 Spatial Extender** no banco. O usuário deve ter privilégios administrativos.

b) Desativação (**Disable DB**) - Utiliza o usuário (USER) e a senha (PASSWORD) fornecidos para desativar completamente o **DB2 Spatial Extender**. O usuário deve ter privilégios administrativos.

2) Administração de Tabelas (**Table Administration)** – Fornece serviços para manipular tabelas individuais do banco e os tipos de dados espaciais contidos nela. Permite a visualização da lista de esquemas, tabelas, colunas e tipos de dados.

a) Registrar coluna (**RegisterCol**) – Cria ou registra uma nova coluna de um tipo de dado espacial em uma coluna, tabela ou esquema fornecidos. A funcionalidade também permite que um geocodificador seja automaticamente associado a esta coluna.

b) Remover coluna (***UnregisterCol***) – Remove uma coluna de tipo de dado espacial do catálogo.

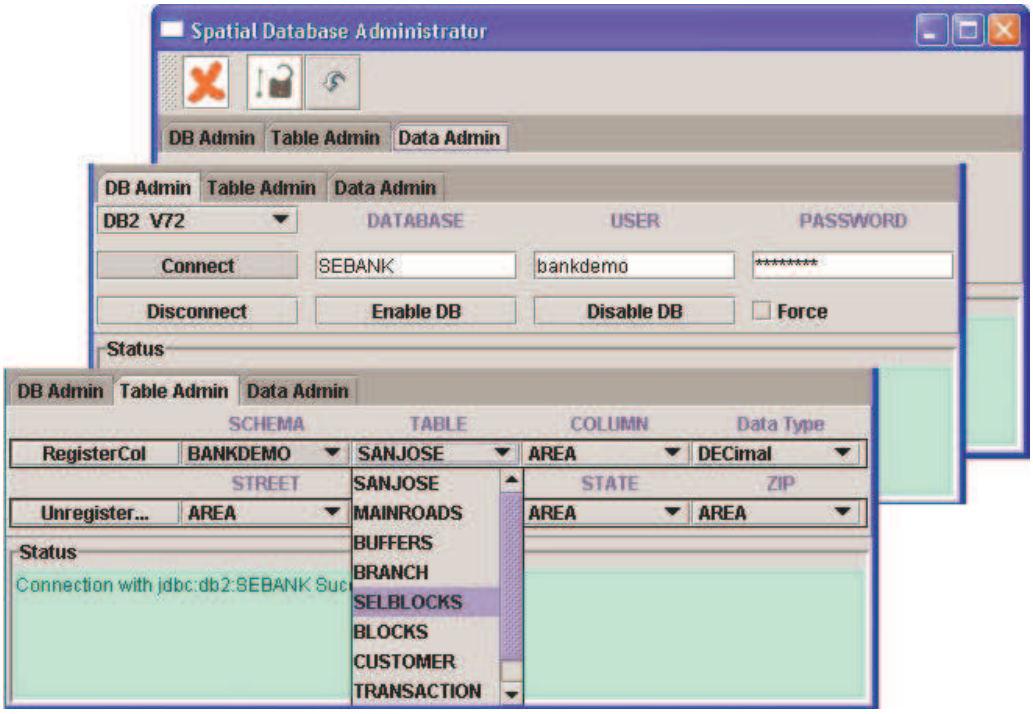

Figura 46 – Sistema de administração

7.4.3 – Query Builder

O construtor de consultas deste protótipo foi implementado diretamente para atender as consultas espaciais. Por requisitos de desempenho e flexibilidade, utilizamos a opção de implementar nós especializados para as funções de consulta (Figura 47).

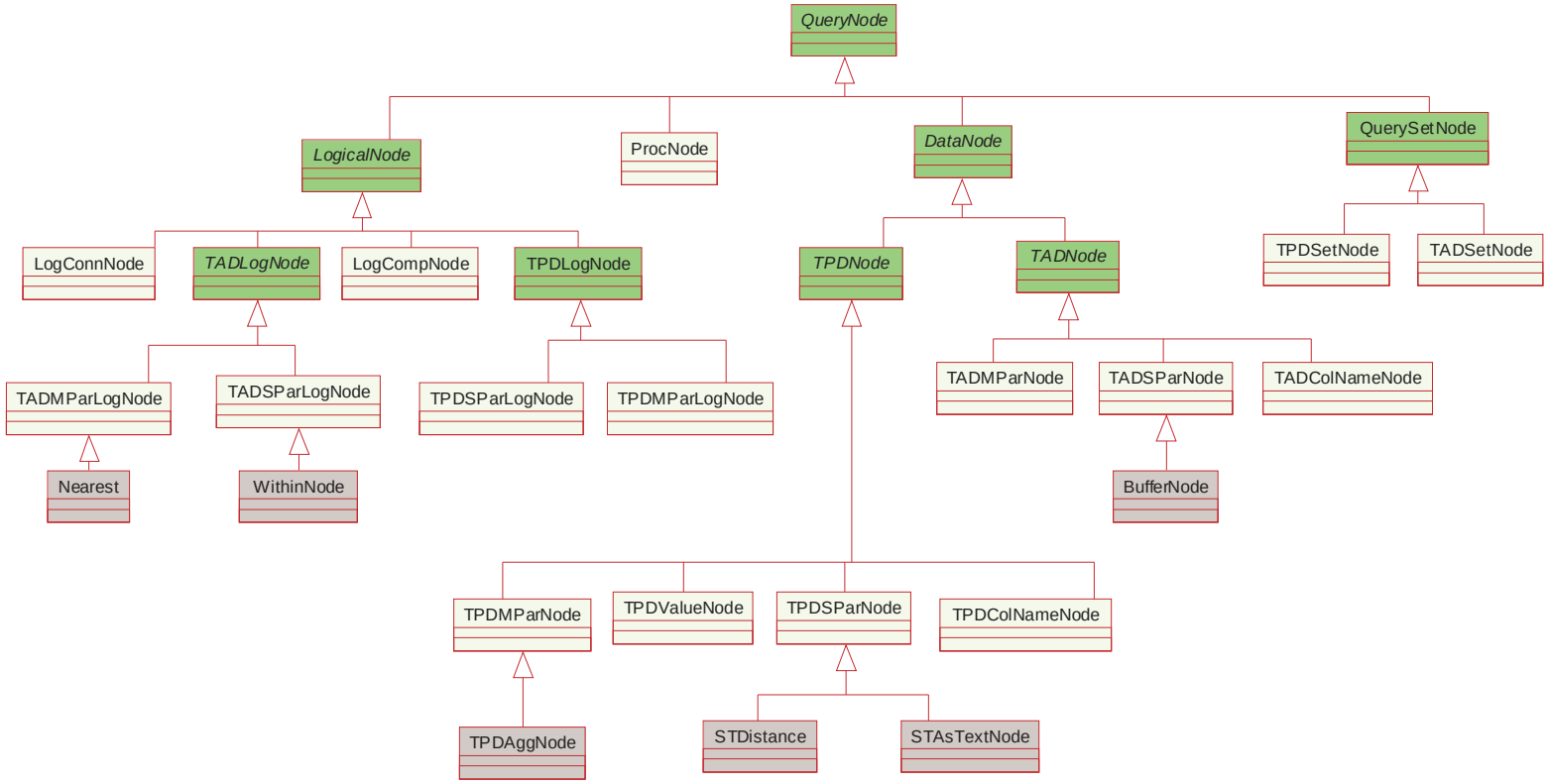

Figura 47 - Estrutura dos nós de consulta espacial

O painel principal de consultas, implementado para a IBM no **Silicon Valley Lab**, pode ser visto na Figura 48. Seu *layout* é ligeiramente diferente em relação ao modelo projetado neste trabalho. Essa configuração visava atender a requisitos da empresa. A quantidade de funções é bem mais simplificada que o modelo do Capítulo 7. Este módulo se destinava a ser um demonstrativo para análise espacial de negócios.

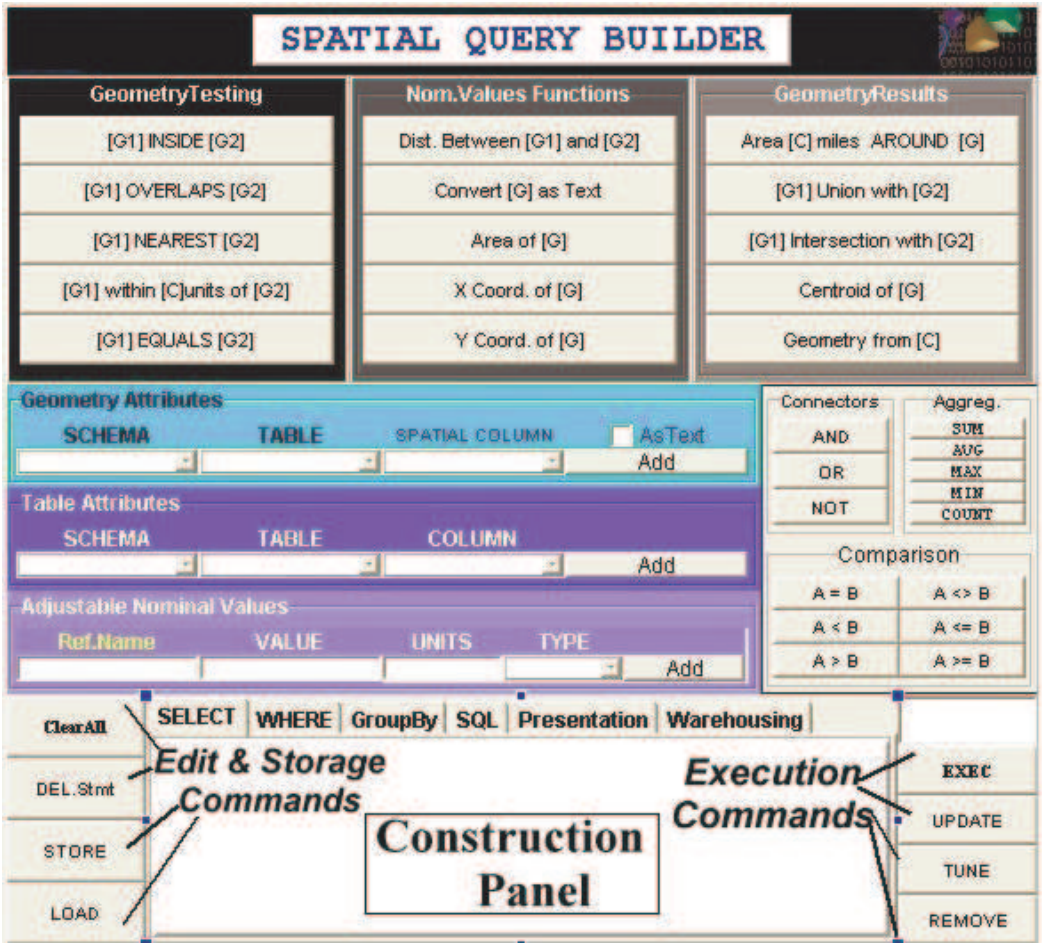

Figura 48 – Construtor de consultas

A Figura 49 ilustra o processo de inserção de uma função dentro do painel de construção. Já a Figura 50 mostra todos os "*tabs*" do painel de construção. No "*tab*" SQL, pode ser vista a consulta SQL em modo texto.

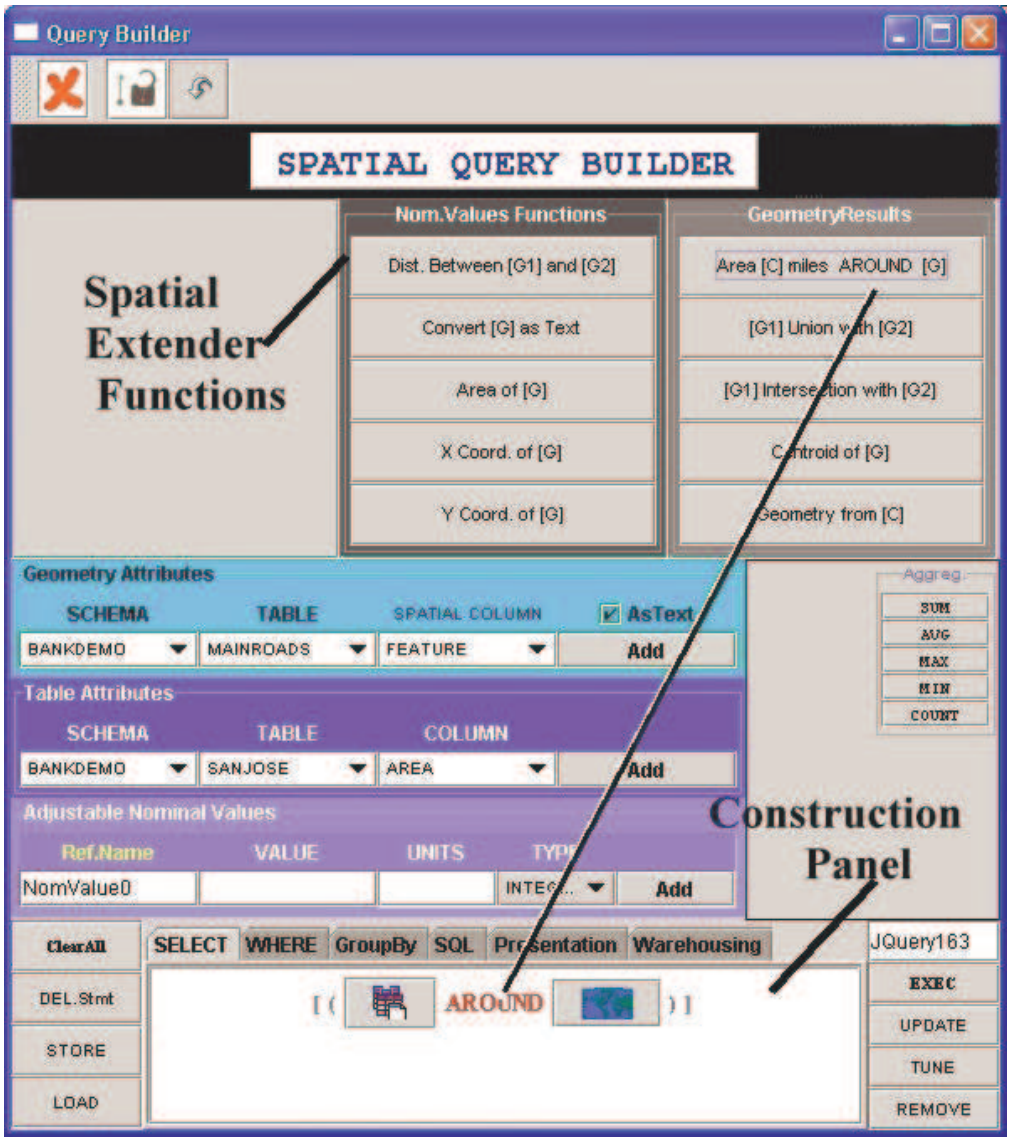

Figura 49 – Painel do construtor de consultas

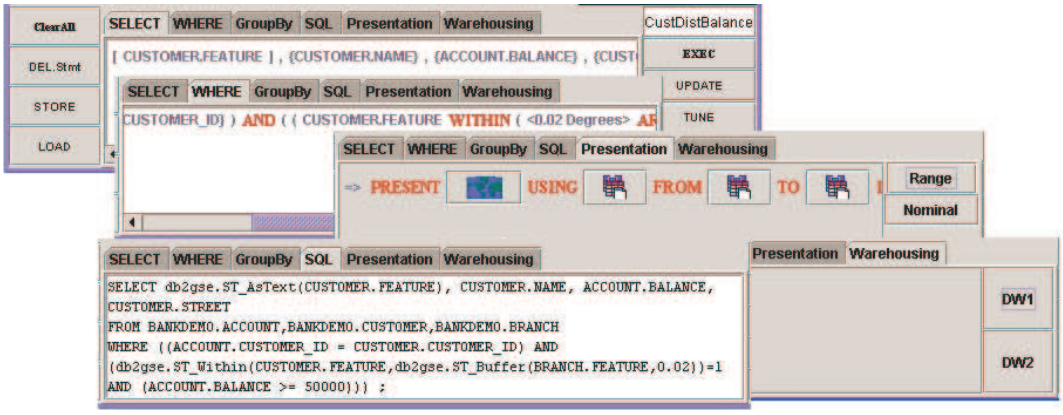

Figura 50 - *Query builder tabs*

A criação especializada de classes para os nós de funções atende a uma outra flexibilidade desta implementação: os botões de funções não têm que corresponder a uma única função, mas podem representar uma cadeia de cláusulas inseridas por toda a sentença SQL. Isto permite a criação de funções “virtuais”, que não poderiam ser criadas diretamente no banco de dados. A Figura 51 ilustra uma função “Próximo de” (NEAREST), que não pode ser criada diretamente, pois necessita do cálculo de todos os elementos da tabela para o cálculo de cada um.

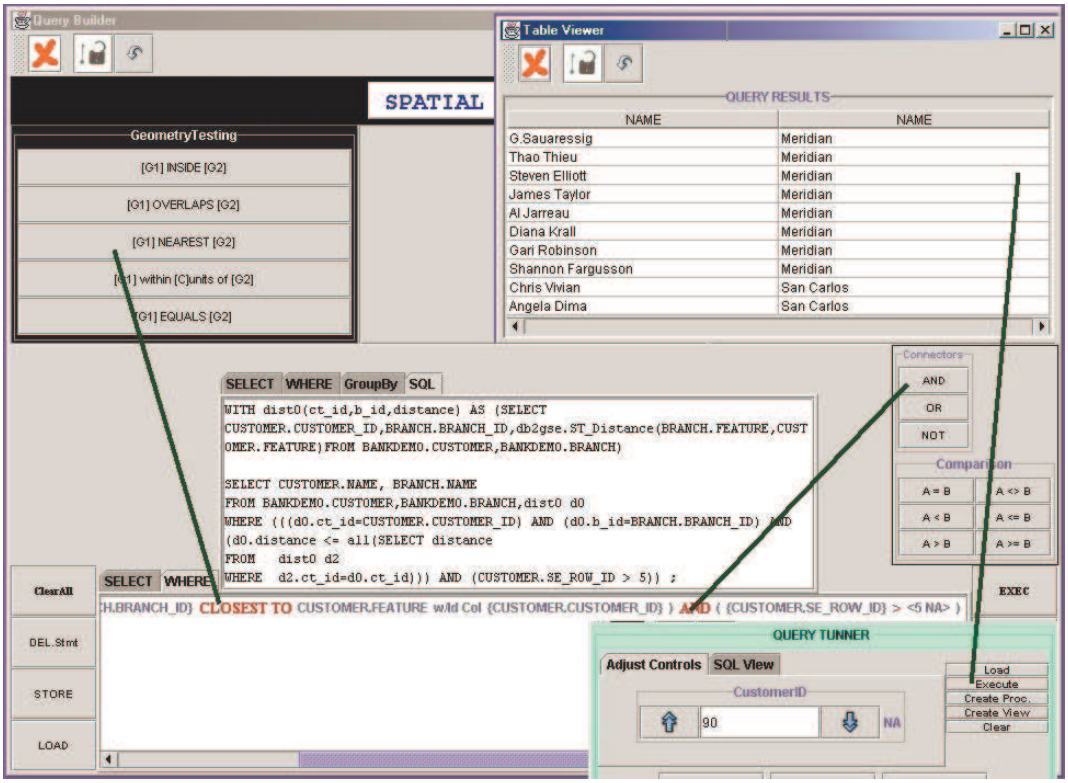

Figura 51 – Função "próximo de" virtual

Todas as consultas podem ser armazenadas e recarregadas (Figura 52) em um formato XML. Ele espelha toda a árvore de consulta da estrutura **XWQuery**.

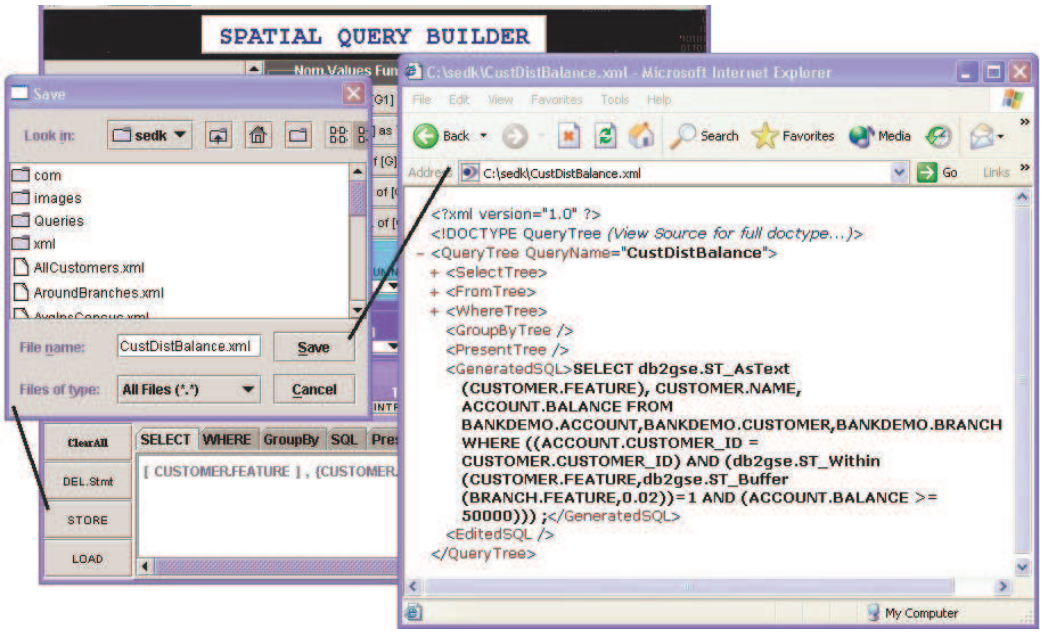

Figura 52 - Comandos de edição e armazenamento

7.4.4 – Visualizador de Mapas

Este visualizador é modificado para trabalhar com um módulo de processamento de mapas chamado SPET, desenvolvido pela (IBM© - China). Ele permite a visualização espacial e o manuseio dos resultados de consultas espaciais. O projeto original não tinha capacidade dinâmica, sendo que cada consulta tinha que ser configurada em arquivos de conteúdo fixo. A modificação feita neste trabalho incorporou apenas o manuseio visual, acrescentando a consulta e os dados vindos do **DB2 Spatial Extender**.

As facilidades incorporadas do SPET incluem o *Zoom* e o *Pan* (deslocamento do mapa), além do gerenciamento de camadas. Cada consulta enviada ao visualizador fica armazenada em uma camada independente. Podemos desativar uma camada, ou trocar a sua posição, permitindo que camadas de base, como polígonos, fiquem mais abaixo e outras, que retornem linhas ou pontos, fiquem desenhadas por último.

O visualizador tem ainda como receber informações codificadas sobre o modo de representação, mostrando, por exemplo, vários graus de coloração de um ponto de acordo com o valor de alguma variável, junto com a sua legenda.

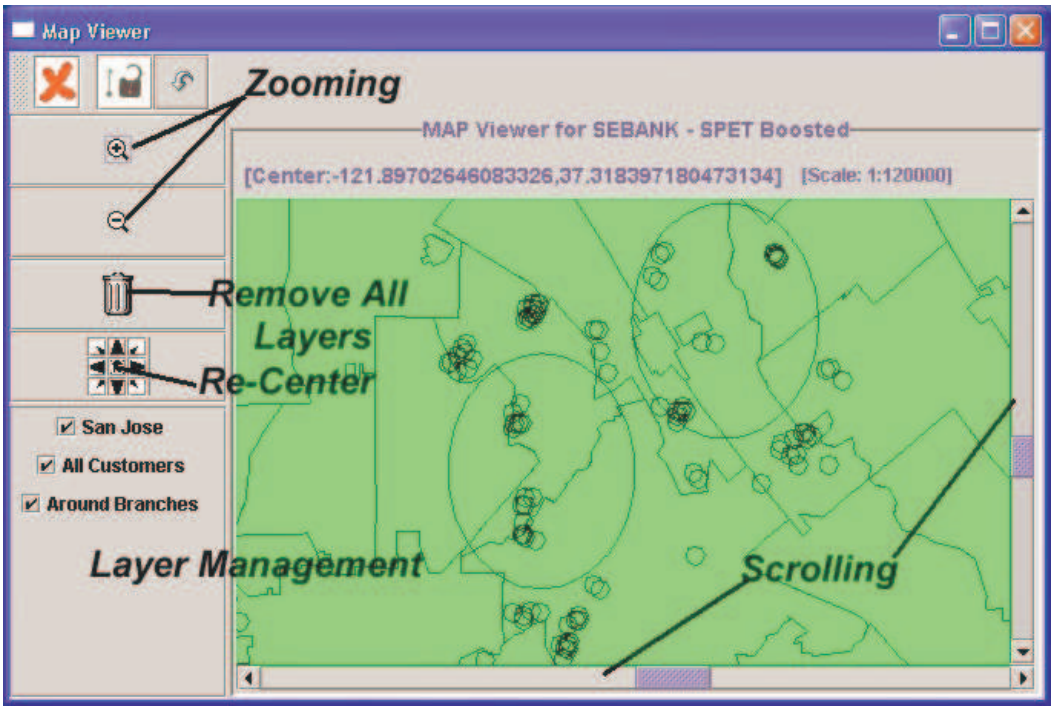

Figura 53 - Visualizador de mapas

Capítulo 8 – Conclusões

8.1 – O Trabalho

A principal contribuição deste trabalho é um modelo geral para inclusão da manipulação de TADs no processo de *data warehousing*, no qual incluímos o processo de extração e transformação (*ETL*), assim como a análise multidimensional (*OLAP*). Este modelo, apresentado pelo *framework* **5-Alpha**, permite a organização de qualquer suporte de funções de suporte para os TADs já existentes, ou pode guiar a construção das mesmas.

O *framework* foi aplicado para tipos de dados espaciais na construção de um sistema de *data warehousing* espacial, onde verificamos que este modelo é adequado para inclusão do suporte já existente. Ele também revelou quais as funções necessárias que ainda não existem no suporte.

Mostramos, no Capítulo 6, um sistema capaz de gerar uma ferramenta especializada para *data warehousing* através de um descritor do *framework* **5-Alpha,** escrito em formato XML. Este sistema se mostrou capaz de construir consultas e procedimentos de *data warehousing* de forma guiada e minimizando a possibilidade de erros sintáticos, facilitando o trabalho dos analistas de negócios em conhecer toda a gama de funcionalidades específicas para os TADs. O protótipo da ferramenta de consulta para dados espaciais foi desenvolvido para demonstrações de análises espaciais de negócios na IBM (**Sillicon Valley Lab**), e produziu excelente resposta e interesse por parte dos gerentes da área de análise de negócios. Por fim, este trabalho ainda apresenta o estado da arte na utilização de TADs em sistemas de *data warehousing*, mostrando uma solução geral para sua manipulação.

O estudo do processo de *data warehousing* espacial teve como finalidade a validação da aplicação do nosso modelo. Contudo, apresentamos também várias contribuições relativas à modelagem e às vantagens do processo de *data warehousing* espacial (Seção 3.3). Apresentamos aqui uma taxonomia de *data warehousing* espacial, mostrando as várias possíveis evoluções da relação entre SIGs e *data warehousing* (Seção 3.3.6). A ferramenta gerada por este trabalho foi validada no exemplo de sistema de *data warehousing* espacial de máxima evolução ($\mathbf{C^3}$), mostrado no protótipo da seção 7.4.

Acreditamos também que oferecemos uma contribuição ao mostrar um método exaustivo de classificação e organização de processos de *Data Warehousing*, que contêm um grande número de possíveis funções e módulos.

O sistema apresentado pode ser utilizado para qualquer TAD, independente dele possuir características físicas ou temporais, como tipos de dados espaciais, ou ser meramente um dado estruturado cuja manipulação e inter-relacionamento requeiram funções especializadas. O *framework* **5-Alpha** poderá ser usado para guiar a arquitetura e o desenvolvimento de soluções de *data warehousing*, permitindo atingir a análise multidimensional dos dados.

8.2 – Prosseguimento

O *framework* proposto, assim como a ferramenta apresentada neste trabalho são grandes contribuições mas podemos ainda explorar as seguintes linhas de trabalho:

1 - Completar o protótipo para a geração de todas as categorias de funções. O protótipo apresentado contém uma arquitetura expansível que permite acomodar todas as categorias de funções descritas aqui. Uma contribuição futura será implementar todas estas categorias.

2 - Explorar a incorporação de outros Tipos Abstratos de Dados. Acreditamos que a aplicação do framework e da ferramenta com outros Tipos Abstratos de Dados, como por exemplo, Tipos de Dados Nebulosos vai demostrar e validar as contribuições deste trabalho.

3 - Uma ferramenta "guiada" (*Whizards*) para construção do descritor XML do *Framework*. O descritor apresentado é muito extenso devido a abranger todas as possíveis categorias semânticas tas funções a serem criadas. Em vista disto, uma ferramenta para sua construção poderia flexibilizar a utilização do *framework*.

4 - Geração de módulos que possam ser incorporados em ferramentas comerciais como o DB2 *Data Warehouse Center*. A ferramenta e o protótipo apresentado são sistemas independentes. Nos acreditamos que uma integração desta ferramenta com as atuais ferramentas de *data warehousing* poderia trazer um grande poder e flexibilidade de manipulação com Tipos Abstratos de Dados.

Apêndice A – Simbologia

A.1 – Simbologia de funções e Procedimentos

O formato geral para todas funções e procedimentos utilizados neste trabalho segue a seguinte simbologia:

<Tipo de ação**><**Código da Classe**>**$_{\text{<sub-categoria>}}$(Parâmetros ...)

- <Tipo de ação>**:**
    - **F** – Especifica funções, ou seja, métodos ou funções no SQL que retornam valores e são chamados dentro do contexto de uma consulta ou sentença em SQL.
    - **P** – Especificação de procedimentos, ou seja métodos que não retornam valores, apenas executam ações e são executados fora do contexto de uma consulta ou sentença em SQL.

O código da classe é formado da seguinte forma:

<Código da Classe**>=<**Categoria**><**Categoria de parâmetros**><**Sub-Categoria**>**:

- **<**Categoria**>:**
    - **B** – Funções de comparação e conectivos lógicos
    - **C** – Funções de manipulação de elementos do catálogo
    - **P** – Funções de suporte a projeção
    - **R** – Funções de suporte a seleção
    - **A** – Funções de suporte a agregação, para tipo de ação "**F**" – Esta categoria já estabelece implicitamente a categoria de parâmetro(conjuntos) e a subcategoria portanto, é determinada na Tabela 13
    - **E** – Procedimentos de Extração
    - **T** – Procedimentos de Transformação
    - **A** – Procedimentos de Análise, para tipo de ação "**P**"

- **<**Categoria de parâmetros**>**(Omitidos para as categorias de agregação(**A**), extração(**E**) e comparação e conectivos lógicos(**B**))**:**
    - **S** – Recebem elementos de *tuplas* como parâmetro, *default* para categoria "**C**" e "**A**" com tipo de ação "**P**"(**PA**__)
    - **M** – Recebem conjuntos de *tuplas* como parâmetro
- **<**Sub-Categoria**>:**

Usadas somente com a <Categoria> de comparadores e conectivos lógicos(**B**)	o **EQ** – Função de igualdade o **NE** – Função de diferença o **LS** – Função de "menor" o **LE** – Função de "menor ou igual" o **GT** – Função de "maior" o **GE** – Função de "maior ou igual" o **AN** – Função de conectivo "E"(AND) o **OR** – Função de conectivo "OU"(OR) o **NO** – Função de conectivo "NÃO"(NOT)
Usadas somente com a <Categoria> de manipulação de metadados(**C**)	o **C** – Funções de Criação o **A** – Funções de Acesso o **P** – Funções de Propriedades o **D** – Procedimentos de registro de dados o **M** – Procedimentos de registro de módulos o **H** – Procedimentos de registro de hierarquias
Usadas somente com a <Categoria> de suporte a projeção(**P**)	o **T** – Funções de Transformação o **C** – Funções de Conversão o **D** – Funções de *DrillDown* o **X** – Funções de Combinação

Usadas somente com a <Categoria> de suporte a seleção(**R**)	o **1** – Funções de propriedade booleana o **2** – Funções de relacionamento binário entre TADs o **3** – Funções de propriedade TADs com parâmetros o **STD** – Funções de relacionamento entre TPDs
Usadas somente com a <Categoria> de suporte a agregação(**A**)	o **1** – Funções de Conversão com retorno de TPD o **2** – Funções de Conversão com retorno de TPD_{set} o **1**<***Y***> – Funções de Conversão ou Transformação o **2**<***Y***> – Funções tipo **1*Y*** com Combinação o <***X***>**1** – Funções com retorno de TAD o <***X***>**2** – Funções com retorno de TAD_{set} o **STD1** – Funções de Conversão ou Transformação de TPDs com retorno de TPD o **STD2** – Funções de Conversão ou Transformação de TPDs com retorno TPD_{set}
Usadas somente com a <Categoria> de Extração(**E**)	o **I** – Procedimentos de Importação o **A** – Procedimentos de Agregação o **S** – Procedimentos de Sequenciamento o **P** – Procedimentos de Preprocessamento o **L** – Procedimentos de Limpeza o **V** – Procedimentos de Validação

Usadas somente com a <Categoria> de procedimentos de Transformação(**T**)	o **R** – Procedimentos de Seleção o **S** – Procedimentos de Separação. o **U** – Procedimentos de Concatenação. o **A** – Procedimentos de Sumarização. o **T** – Procedimentos de Transformação. o **E** – Procedimentos de Enriquecimento. o **C** – Procedimentos de Conversão o **D** – Procedimentos de *DrillDown*.

Usadas somente com a <Categoria> de procedimentos de Análise(**A**)	o **A** – Procedimento para adicionar variáveis a hierarquia o **S** – Procedimentos para agregação de hierarquias o **D** – Procedimentos para nível de detalhe de hierarquias. o **H1** – Procedimentos de criação de hierarquias simples o **H2** – Procedimentos de criação de hierarquias recursivas o **G** – Procedimentos de análise Ad Hoc

A.2 – Simbologia do texto

Utilizaremos os seguintes tipos de formatos de caracteres para os vários tipos de elementos do texto:

- *Extrangerismo Técnico* – Usado para representar palavras estrangeiras mas que precisam ser usadas em sua língua original, por exemplo, *Data Warehousing*.
- *Função Nativa* – Usado para representar funções originais de produtos.
- Função Nativa Implementada – Usado para representar funções não existentes que deverão ser implementadas.
- Função OGC – Usado para representar funções propostas pelo OGC.
- **CODIGO DE FUNÇÃO** – Usado para representar códigos de funções classificadas pelo *framework* deste trabalho.
- ***X*** – Usado para representar conjuntos de Tipos Abstratos de Dados
- **Nome Próprio –** Usado para representar nomes próprios de produtos, algoritmos, fabricantes, etc.
- Listagem XML – Usado para representar trechos de código em XML
- **XMLElement** – Usado para representar um elemento no código XML.
- **XMLAtribute –** Usado para representar um atributo de um elemento no código XML.
- ***Atributo*** – Usado para indicar atributos de elementos de tabelas.
- Referencia – Usado para indicar uma referencia bibliográfica no texto.
- Referencias Finais – Usado para representar as referencias no fim do texto.

Apêndice B – Descritor XML do Framework 5-Alpha

B.1 – Estruturas básicas

O *framework* contém uma lista de códigos para guiar a classificação de seus elementos e um *template* para descrição do modelo. Este *template*, ou descritor é escrito utilizando a linguagem XML A codificação, denominada de **5-Alpha**(ABCDE) representa os cinco requisitos do modelo como: Tipos (**A**)bstratos de Dados, Operadores (**B**)ásicos, (**C**)atálogo de metadados, Suporte relacional no Banco de (**D**)ados e Processo de *Data Warehouse* (**E**)xtendido.

A lista abaixo mostra todas as funções já especificadas do modelo de *Data Warehouse* estendido dentro a codificação do *framework* **5-Alpha**, bem como a estrutura do descritor em XML.

Na Figura 12 do Capítulo 4 mostramos o diagrama geral do descritor em XML do *framework*. Ele contém uma seção para cada um dos cinco requisitos do *fremework*(A,B,C,D e E) ainda uma seção extra para definição de recursos(**<Resources>**) de interação com o sistema externo(Exemplo 17). A raiz principal do modelo é o elemento **<Model>** e os dois principais itens de recursos são esquema(**<BaseSchema>**) e arquivo externo (**<XWExtFile>**). Os elementos de esquema possuem um atributo nome(**name**) e um identificador (**SCHID**) que é usado para referenciar o esquema de objetos no banco de dados. Isto permite que possamos criar e alterar o esquema de localização dos objetos sem precisar alterar todo o descritor.

```
<Model name="">
        <Resources>
                <BaseSchema SCHID="idvalue0" name="" />
                <XWExtFile ARQID="idvalue1" ExtDir="" ExtName="" />
        </Resources>
…
</Model>
```

Exemplo 17

Todos os parâmetros de funções ou procedimentos estão mapeados através de cinco classes de elementos(Exemplo 19):

- (**<TADPar>**) para parâmetros de TADs
- (**<TPDPar>**) para parâmetros de TPDs
- (**<TPDParSet>**) para parâmetros de conjuntos de TADs
- (**<TADParSet**>) para parâmetros de conjuntos de TPDs

- (**<BOOLPar>**) para parâmetros de retorno booleano – Este tipo de parâmetro específico tem por finalidade permitir outros tipos de dados para indicar valores booleanos(ex. Inteiros[0-1], caracteres[‘S’,‘N’]) e até mesmo TADs que possam representar valores lógicos, como por exemplo tipos de dados nebulosos.

Os elementos de parâmetros ficam aninhados dentro dos elementos de definição de funções, como no Exemplo 8[5]. Todos estes elementos possuem um atributo para indicar:

- O nome da coluna correspondente ao parâmetro(**parName**)
- A descrição do parâmetro(**parDesc**)
- Um identificador único do tipo do parâmetro correspondente aos tipos de dados definidos na seção A(**typeID**)
- Dois rótulos de fantasia para serem exibidos nos controles da interface como o usuário. Estes rótulos opcionais estão destinados a serem colocados antes(**FakeNameBefore**) e depois(**FakeNameAfter**) do parâmetro. No caso, por exemplo de uma função “Distância” com dois parâmetros P1 e P1, poderia ser exibida das seguintes formas(Exemplo 3):

Rótulo Anterior	**P1**	Rótulo Posterior	Rótulo Anterior	**P2**	Rótulo Posterior
“Distância entre(“	A	**“)**	**E “**	**“(”**	B **“)”**
“(“	A	**“)**	**Distante “**	**“de (”**	B **“)”**

Exemplo 18

```
…
        <TADPar FakeNameAfter="" FakeNameBefore="" parDesc="" parName=""
                typeID="idvalue0" />
        <TPDPar FakeNameAfter="" FakeNameBefore="" parDesc="" parName=""
                typeID="idvalue0" />

        <TPDParSet FakeNameAfter="" FakeNameBefore="" parDesc="" parName=""
                   typeID="idvalue0" />
        <TADParSet FakeNameAfter="" FakeNameBefore="" parDesc="" parName=""
                   typeID="idvalue0" />

        <BOOLPar FakeNameAfter="" FakeNameBefore="" parDesc="" parName=""
                  typeID="idvalue0" />
```

Exemplo 19

[5] Apesar de termos especificado o **framework 5-Alpha** para trabalhar com parâmetros binários, esta padronização do descritor permite descrevermos funções com qualquer número de parâmetros.

B.2 – Seção A

[A-0-0] - ***X***W$_{tipos}$ – Esta seção contém todos os tipos de dados utilizados no sistema. O elemento (**SectionA**) inicializa esta seção. A definição dos TADs, através do elemento (**XWTADType**), pode ser recursiva(Exemplo 20). Os TPDs(**XWTADType**), por outro lado não permitem hierarquização. Cada TAD possui um atributo de nome(**name**) que é o nome real no banco, o código do esquema onde ele está definido(**SCHID**) e um identificador único do tipo(**typeID**).

```
<SectionA>
        <XWTADType SCHID="idvalue0" name="" typeID="idvalue2">
                <XWTADType SCHID="idvalue0" name="" typeID="idvalue3" />
        </XWTADType>
        <XWTPDType name="" typeID="idvalue4" />
</SectionA>
```

Exemplo 20

B.3 – Seção B

[B-0-0] - ***X***W$_{transparentes}$ –Esta seção define funções que vão oferecer operadores de comparação(Igual, Maior, Menor e Diferente) e lógicos(“E”, “OU” e “NÃO”). Os elementos do descritor que definem estes dois grupos de operadores(Figura 54 e Exemplo 21) são respectivamente(**<XWEqual>**, **<XWGreat>**, **<XWLess>**, **<XWDiff>**) e (**<XWAnd>**, **<XWOr>**, **<XWNot>**). Cada função possui um atributo de nome(**BaseName**) que é o nome real no banco, o código do esquema onde ele está definido(**SCHID**) e um identificador único para de função dentro do modelo(**FUNCID**).

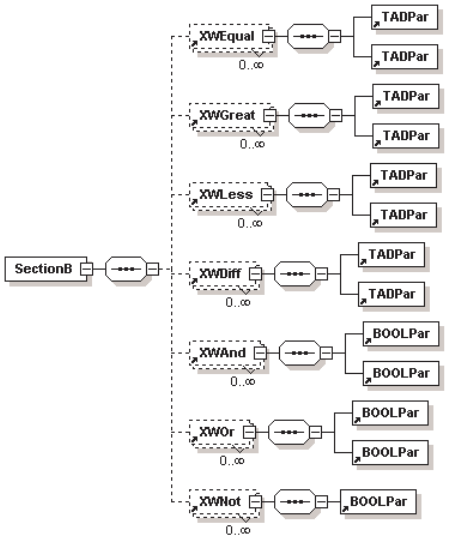

Figura 54 - Esquema da Seção B

```
<SectionB>
        <XWEqual BaseName="" SCHID="idvalue0" FUNCID="idvalue5" >
                <TADPar FakeNameAfter="" FakeNameBefore="" parDesc="" parName=""
                        typeID="idvalue0" />
                <TADPar FakeNameAfter="" FakeNameBefore="" parDesc="" parName=""
                        typeID="idvalue0" />
        </XWEqual>
        <XWGreat BaseName="" SCHID="idvalue0" FUNCID="idvalue5" >
                <TADPar FakeNameAfter="" FakeNameBefore="" parDesc="" parName=""
                        typeID="idvalue0" />
                <TADPar FakeNameAfter="" FakeNameBefore="" parDesc="" parName=""
                        typeID="idvalue0" />
        </XWGreat>
        <XWLess BaseName="" SCHID="idvalue0" FUNCID="idvalue5" >
                <TADPar FakeNameAfter="" FakeNameBefore="" parDesc="" parName=""
                        typeID="idvalue0" />
                <TADPar FakeNameAfter="" FakeNameBefore="" parDesc="" parName=""
                        typeID="idvalue0" />
        </XWLess>
        <XWDiff BaseName="" SCHID="idvalue0"  FUNCID="idvalue5" >
                <TADPar FakeNameAfter="" FakeNameBefore="" parDesc="" parName=""
                        typeID="idvalue0" />
                <TADPar FakeNameAfter="" FakeNameBefore="" parDesc="" parName=""
                        typeID="idvalue0" />
        </XWDiff>
        <XWAnd BaseName="" FUNCID="idvalue5" FalseValue="" RetType="idvalue0"
                SCHID="idvalue0" TrueValue="">
                <BOOLPar FakeNameAfter="" FakeNameBefore="" parDesc="" parName=""
                        typeID="idvalue0" />
                <BOOLPar FakeNameAfter="" FakeNameBefore="" parDesc="" parName=""
                        typeID="idvalue0" />
        </XWAnd>
        <XWOr BaseName="" FUNCID="idvalue6" FalseValue="" RetType="idvalue0"
                SCHID="idvalue0" TrueValue="">
                <BOOLPar FakeNameAfter="" FakeNameBefore="" parDesc="" parName=""
                        typeID="idvalue0" />
                <BOOLPar FakeNameAfter="" FakeNameBefore="" parDesc="" parName=""
                        typeID="idvalue0" />
        </XWOr>
        <XWNot BaseName="" FUNCID="idvalue7" FalseValue="" RetType="idvalue0"
                SCHID="idvalue0" TrueValue="">
                <BOOLPar FakeNameAfter="" FakeNameBefore="" parDesc="" parName=""
                        typeID="idvalue0" />
        </XWNot>
</SectionB>
```

Exemplo 21

B.4 – Seção C

[C-0-0] - ***X***W$_{MD}$ – O suporte de metadados(Figura 55) está dividido em três seções especificadas pelos elementos **<XWCatalogs>**, **<XWCatHandle>** e **<XWDataHandle>** que definem os catálogos e seus procedimentos de manipulação alem de uma lista de funções de manipulação de TADs dentro do modelo.

[C-1-0] - ***X***W$_{\text{CATÁLOGOS}}$ – Contém a especificação de catálogos para TADs(**<XWTypeCatalog>**), módulos de procedimentos (**<XWModuleCatalog>**) e parâmetros para procedimentos (**<XWModuleParCatalog>**), como pode ser visto no Exemplo 6. A tabela que especifica o catálogo de tipos possui o código do esquema onde ela está definida (**SCHID**) e atributos para indicar as colunas da tabela que definem o nome do esquema (**XWSchemaName**), tabela (**TableName**) e coluna(**XWSourceCol**) de um tipo que estará na coluna(**XWTypeName**). A especificação do catálogo de módulos é similar com a diferença que ao invés do atributo de nome de coluna e tipo temos um atributo para definir o nome do módulo(**XWModuleName**), nome do arquivo no sistema operacional para executar o módulo(**XWFileName**), nome do procedimento no banco de dados(**XWSProcName**) e identificador do módulo(**NameID**). Os parâmetros de procedimentos apenas definem uma lista de parâmetros (**<XWModulePar>**) e tipos (**parType**) para cada módulo (**<CatalogModuleName>**).

```
<SectionC>
        <XWCatalogs>
                <XWTypeCatalog SCHID="idvalue0" TableName="" XWSchemaName=""
                                XWSourceCol="" XWTableName="" XWTypeName="" />
                <XWModuleCatalog NameID="idvalue8" SCHID="idvalue0" TableName=""
                                XWFileName="" XWModuleName="" XWSProcName=""
                                XWSchemaName="" />
                <XWModuleParCatalog   CatalogModuleName="">
                        <XWModulePar name="" parType="" />                    …
                </XWModuleParCatalog>
        </XWCatalogs>
…
</SectionC>
```

Exemplo 22

[C-2-0] - ***X***W$_{\text{Manipulação de Catálogos}}$ – Os elementos do descritor(Exemplo 7) para os procedimentos que cadastram os TADs(**<PCSD>**), os módulos(**<PCSM>**) e as hierarquias(**<PCSH>**) nos catálogos possuem apenas um atributo para definir o nome no banco(**BaseName**) e o código do esquema(**SCHID**). Os parâmetros de TPDs, descritos no inicio desta seção, são acrescentados como nós filhos para inserir nome do tipo, módulo, coluna, tabela, etc.

```
<SectionC>
        …
    <XWCatHandle>
        <PCD BaseName="" SCHID="idvalue0">
            <TPDPar …/>
                …
        </PCD>
        <PCM BaseName="" SCHID="idvalue0">
            <TPDPar …/>
            …
        </PCM>
        …
    </XWCatHandle>
…
</SectionC>
```

Exemplo 23

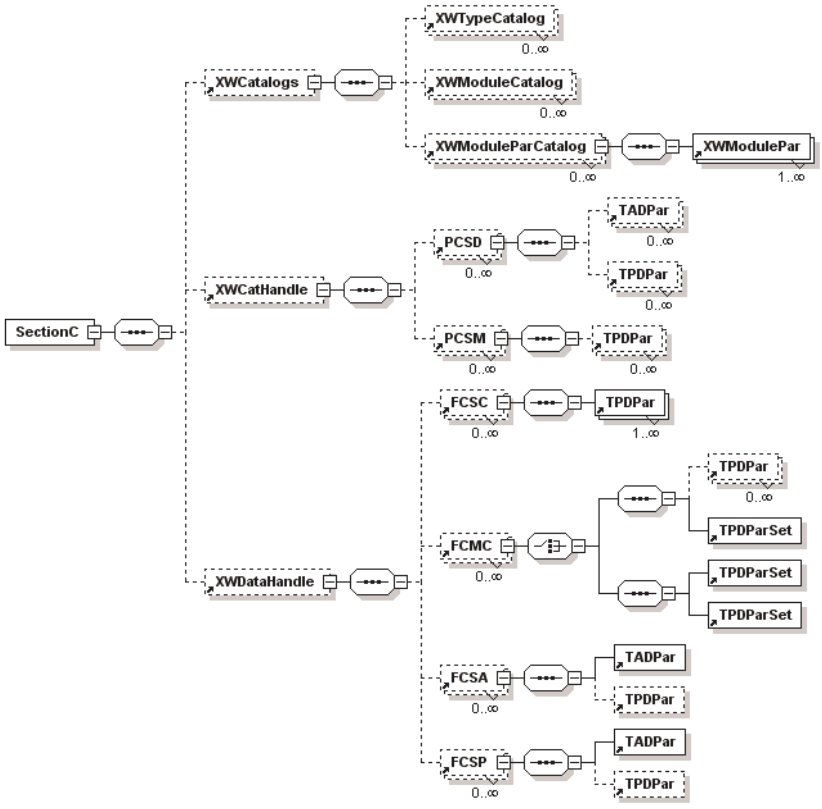

Figura 55 - Esquema da Seção C

[C-3-0] - ***X***W$_{\text{Manipulação de Elementos}}$ – As funções de manipulação de elementos (Exemplo 8), cuja seção é definida pelo elemento(**<XWDataHandle>**), seguem o mesmo padrão da Seção B. Cada função possui um atributo de nome(**BaseName**) que é o nome real no banco, o código do esquema onde ele está definido(**SCHID**) e um identificador único para de função dentro do modelo(**FUNCID**). Os três parâmetros extras aqui são para determinar o retorno e

a forma de chamada das funções. O atributo **ISMETHOD**(S/N) indica se a função será chamada como método(ex. OBJ..método(parâmetro)). O atributo (**RetType**) referencia o identificador de um tipo de dado definido na Seção A e o atributo (**retMode**) indica se o retorno será de uma única *tupla*(**"ONE"**) ou de conjunto de *tuplas*(**"SET"**). A lista abaixo mostra os elementos do descritor para cada função:

[C-3-1] - **FCSC**$^{n}_{Criação}$(Parâmetros) – Elementos (**<FCSC>**)

[C-3-2] - **FCMC**$^{n}_{Criação}$(Parâmetros) – Elementos (**<FCMC>**)

[C-3-3] - **FCSA**$^{n}_{Acesso}$(***X***W$_{tipo}$) – Elementos (**<FCSA>**)

[C-3-4] - **FCSP**$^{n}_{Propriedade}$(***X***W$_{tipo}$) – Elementos (**<FCSP>**)

[C-3-5] - **FCMA**$^{n}_{Acesso}$(***X***W$_{coluna}$) – Elementos (**<FCMA>**)

[C-3-6] - **FCMP**$^{n}_{Propriedade}$(***X***W$_{coluna}$) – Elementos (**<FCMP>**)

```
<SectionC>
                …

        <XWDataHandle>
                <FCSC BaseName="" FUNCID="idvalue9" ISMETHOD="S"
                        RetType="idvalue0" SCHID="idvalue0" retMode="ONE">
                        <TPDPar FakeNameAfter="" FakeNameBefore="" parDesc=""
                                parName="" typeID="idvalue0" />
                </FCSC>
                <FCMC BaseName="" FUNCID="idvalue10" ISMETHOD="S"
                        RetType="idvalue0" SCHID="idvalue0" retMode="SET">
                        <TPDPar FakeNameAfter="" FakeNameBefore="" parDesc=""
                                parName="" typeID="idvalue0" />
                        <TPDParSet FakeNameAfter="" FakeNameBefore="" parDesc=""
                                parName="" typeID="idvalue0" />
                </FCMC>
                <FCSA BaseName="" FUNCID="idvalue11" ISMETHOD="S"
                        RetType="idvalue0" SCHID="idvalue0" retMode="ONE">
                        <TADPar FakeNameAfter="" FakeNameBefore="" parDesc=""
                                parName="" typeID="idvalue0" />
                        <TPDPar FakeNameAfter="" FakeNameBefore="" parDesc=""
                                parName="" typeID="idvalue0" />
                </FCSA>
                <FCSP BaseName="" FUNCID="idvalue12" ISMETHOD="S"
                        RetType="idvalue0" SCHID="idvalue0" retMode="ONE">
                        <TADPar FakeNameAfter="" FakeNameBefore="" parDesc=""
                                parName="" typeID="idvalue0" />
                        <TPDPar FakeNameAfter="" FakeNameBefore="" parDesc=""
                                parName="" typeID="idvalue0" />
                </FCSP>
        </XWDataHandle>
…
</SectionC>
```

Exemplo 24

B.5 – Seção D

[D-0-0] - $\boldsymbol{X}$W[$\mathbf{F}_{\text{Suporte Relacional}}$] = {$\mathbf{F}_{\text{Seleção}}$, $\mathbf{F}_{\text{Projeção}}$, $\mathbf{F}_{\text{Agregação}}$ } – A seção de suporte relacional é determinada pelo elemento (**<SecionD>**) do descritor. Ela as sub-seções de Seleção, Projeção e Agregação são definidas respectivamente pelos elementos **<XWSelect>**(Exemplo 25), **<XWProject>**(Exemplo 26) e **<XWAgregate>**(Exemplo 27) do descritor.

[D-1-0] - $\boldsymbol{X}$W[$\mathbf{F}_{\text{Seleção}}$] – A seção de funções de suporte a seleção do descritor(Figura 56) é determinada pelo elemento **<XWSelect>**. Podemos verificar pelo Exemplo 25 e Exemplo 26 que elas diferem em relação as funções anteriores por não possuírem modo de retorno(**ONE** ou **SET**). Alem disso existem dois parâmetros para indicar os valores do tipo de dado de retorno(**RetType**) que serão usados como verdadeiro(**TrueValue**) e falso(**FalseValue**). A lista abaixo vai mostrar os elementos do descritor para especificar cada uma delas.

[D-1-1] - **FRS1**$^{n}_{\text{Relação}}$($\boldsymbol{X}$W$_{\text{tipo}}$) – Elementos (**<FRS1>**)

[D-1-2] - **FRS2**$^{n}_{\text{Relação}}$($\boldsymbol{X}$W$_{\text{tipo}}{}^{1}$, $\boldsymbol{X}$W$_{\text{tipo}}{}^{2}$, [DW$_{\text{parâmetro}}$])) – Elementos (**<FRS2>**)

[D-1-3] - **FRS3**$^{n}_{\text{Relação}}$($\boldsymbol{X}$W$_{\text{tipo}}$, DW$_{\text{parâmetro}}$) – Elementos (**<FRS3>**)

```
<SectionD>
        <XWSelect>
                <FRS1 BaseName="" FUNCID="idvalue13" FalseValue="" ISMETHOD="S"
                        RetType="idvalue0" SCHID="idvalue0" TrueValue="">
                        <TADPar FakeNameAfter="" FakeNameBefore="" parDesc=""
                                parName="" typeID="idvalue0" />
                </FRS1>
                <FRS2 BaseName="" FUNCID="idvalue14" FalseValue="" ISMETHOD="S"
                        RetType="idvalue0" SCHID="idvalue0" TrueValue="">
                        <TADPar ... />
                        <TADPar ... />
                </FRS2>
                <FRS3 BaseName="" FUNCID="idvalue15" FalseValue="" ISMETHOD="S"
                        RetType="idvalue0" SCHID="idvalue0" TrueValue="">
                        <TADPar ... />
                        <TPDPar FakeNameAfter="" FakeNameBefore="" parDesc=""
                                parName="" typeID="idvalue0" />
                </FRS3>
                <FRSSTD BaseName="" FUNCID="idvalue16" FalseValue="" ISMETHOD="S"
                        RetType="idvalue0" SCHID="idvalue0" TrueValue="">
                        <TPDPar FakeNameAfter="" FakeNameBefore="" parDesc=""
                                parName="" typeID="idvalue0" />
                </FRSSTD>
        ..
        </XWSelect>
</SectionD>
```

Exemplo 25

[D-1-4] - **FRM1**$^{n}_{\text{Relação}}$ (***X***W$_{\text{coluna}}$) – Elementos (**<FRM1>**)

[D-1-5] - **FRM2**$^{n}_{\text{Relação}}$ (***X***W$_{\text{coluna}}{}^{1}$, (***X***W$_{\text{coluna}}{}^{2}$ | ***X***W$_{\text{tipo}}$), [DW$_{\text{parâmetro}}$]) – Elementos (**<FRM2>**).

[D-1-6] - **FRM3**$^{n}_{\text{Relação}}$ (***X***W$_{\text{coluna}}$, ***X***W$_{\text{tipo}}$) – Elementos (**<FRM3>**)

```
<SectionD>
        <XWSelect>
        …
                <FRM1 BaseName="" FUNCID="idvalue17" FalseValue="" ISMETHOD="S"
                        RetType="idvalue0" SCHID="idvalue0" TrueValue="">
                        <TADParSet FakeNameAfter="" FakeNameBefore=""
                                typeID="idvalue0" />
                </FRM1>
                <FRM2 BaseName="" FUNCID="idvalue18" FalseValue="" ISMETHOD="S"
                        RetType="idvalue0" SCHID="idvalue0" TrueValue="">
                        <TADParSet FakeNameAfter="" FakeNameBefore=""
                                typeID="idvalue0" />
                        <TADParSet FakeNameAfter="" FakeNameBefore=""
                                typeID="idvalue0" />
                </FRM2>
                <FRM3 BaseName="" FUNCID="idvalue19" FalseValue="" ISMETHOD="S"
                        RetType="idvalue0" SCHID="idvalue0" TrueValue="">
                        <TADParSet FakeNameAfter="" FakeNameBefore=""
                                typeID="idvalue0" />
                        <TADPar FakeNameAfter="" FakeNameBefore="" parDesc=""
                                parName="" typeID="idvalue0" />
                </FRM3>
                <FRMSTD BaseName="" FUNCID="idvalue21" FalseValue="" ISMETHOD="S"
                        RetType="idvalue0" SCHID="idvalue0" TrueValue="">
                        <TPDParSet FakeNameAfter="" FakeNameBefore="" parDesc=""
                                parName="" typeID="idvalue0" />
                        <TPDPar FakeNameAfter="" FakeNameBefore="" parDesc=""
                                parName="" typeID="idvalue0" />
                </FRMSTD>
        ..
        </XWSelect>
</SectionD>
```

Exemplo 26

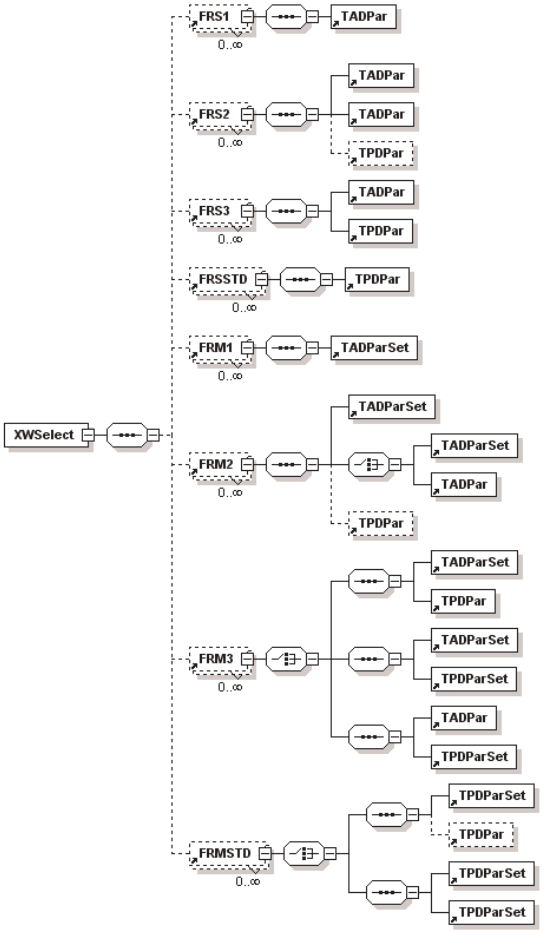

Figura 56 - Esquema da Seção D - Seleção

[D-2-0] - $\boldsymbol{X}$W[$\mathbf{F}_{\text{Projeção}}$] – O descritor de funções de projeção(Figura 57) usa o elemento **< XWProject >**(Exemplo 26) e cada elemento de descrição de função usa os mesmos atributos da seção **[C-3-0]** onde podemos ter retorno individual ou de conjunto. Aqui está a lista de elementos do descritor para cada função:

[D-2-1] - $\mathbf{FPST}^{n}_{\text{Transformação}}(\boldsymbol{X}W_{tipo},[DW_{parâmetro}])$ – Elementos (**<FPST1>**) ou (**<FPST1>**)

[D-2-2] - $\mathbf{FPSC}^{n}_{\text{Conversão}}(\boldsymbol{X}W_{tipo}^{1},[\ \boldsymbol{X}W_{tipo}^{2}|\ DW_{parameter}])$ – Elementos (**<FPSC>**)

[D-2-3] - $\mathbf{FPSX}^{n}_{\text{Combinacao}}(\boldsymbol{X}W_{tipo}^{1},\ \boldsymbol{X}W_{tipo}^{2},[DW_{parâmetro}])$ – Elementos (**<FPSX>**)

[D-2-4] - $\mathbf{FPSD}^{n}_{\text{DrillDown}}(\boldsymbol{X}W_{tipo}\ [,DW_{parâmetro}])$ – Elementos (**<FPSD>**)

[D-2-5] - $\mathbf{FPMT}^{n}_{\text{Transformação}}(\boldsymbol{X}W_{tipo},\ DW_{parâmetro})$ – Elementos (**<FPMT>**)

[D-2-6] - $\mathbf{FPMC}^{n}_{\text{Conversão}}(\boldsymbol{X}W_{tipo},\ DW_{parâmetro})$ – Elementos (**<FPMC>**)

[D-2-7] - $\mathbf{FPMD}^{n}_{\text{DrillDown}}(\boldsymbol{X}W_{tipo}\ [,DW_{parâmetro}])$ – Elementos (**<FPMD>**)

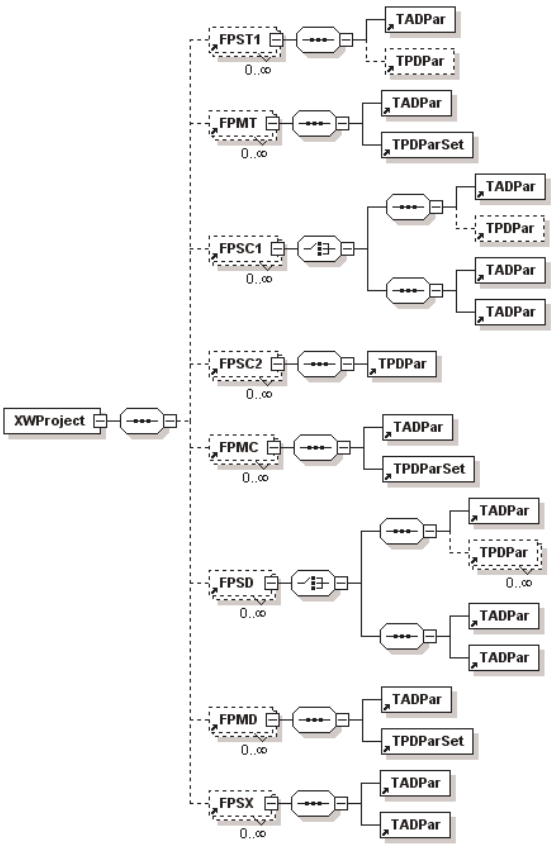

Figura 57 - Esquema da Seção D – Projeção

```
<SectionD>  …
        <XWProject>
                <FPST1 BaseName="" FUNCID="idvalue22" ISMETHOD="S"
                        RetType="idvalue0" SCHID="idvalue0" retMode="SET">
                        <TADPar FakeNameAfter="" FakeNameBefore="" parDesc=""
                                parName="" typeID="idvalue0" />
                        <TPDPar FakeNameAfter="" FakeNameBefore="" parDesc=""
                                parName="" typeID="idvalue0" />
                </FPST1>
                <FPMT BaseName="" FUNCID="idvalue23" ISMETHOD="S"
                        RetType="idvalue0" SCHID="idvalue0" retMode="SET">
                        <TADPar FakeNameAfter="" FakeNameBefore="" parDesc=""
                                parName="" typeID="idvalue0" />
                        <TPDParSet FakeNameAfter="" FakeNameBefore="" parDesc=""
                                parName="" typeID="idvalue0" />
                </FPMT>
                <FPSC1 BaseName="" FUNCID="idvalue24" ISMETHOD="S"
                        RetType="idvalue0" SCHID="idvalue0" retMode="SET">
                        <TADPar FakeNameAfter="" FakeNameBefore="" parDesc=""
                                parName="" typeID="idvalue0" />
                        <TPDPar FakeNameAfter="" FakeNameBefore="" parDesc=""
                                parName="" typeID="idvalue0" />
                </FPSC1>
```

```
                <FPSC2 BaseName="" FUNCID="idvalue25" ISMETHOD="S"
                        RetType="idvalue0" SCHID="idvalue0" retMode="SET">
                        <TPDPar FakeNameAfter="" FakeNameBefore="" parDesc=""
                                parName="" typeID="idvalue0" />
                </FPSC2>
                <FPMC BaseName="" FUNCID="idvalue26" ISMETHOD="S"
                        RetType="idvalue0" SCHID="idvalue0" retMode="SET">
                        <TADPar FakeNameAfter="" FakeNameBefore="" parDesc=""
                                parName="" typeID="idvalue0" />
                        <TPDParSet FakeNameAfter="" FakeNameBefore="" parDesc=""
                                parName="" typeID="idvalue0" />
                </FPMC>
                <FPSD BaseName="" FUNCID="idvalue27" ISMETHOD="S"
                        RetType="idvalue0" SCHID="idvalue0" retMode="SET">
                        <TADPar FakeNameAfter="" FakeNameBefore="" parDesc=""
                                parName="" typeID="idvalue0" />
                        <TPDPar FakeNameAfter="" FakeNameBefore="" parDesc=""
                                parName="" typeID="idvalue0" />
                </FPSD>
                <FPMD BaseName="" FUNCID="idvalue28" ISMETHOD="S"
                        RetType="idvalue0" SCHID="idvalue0" retMode="SET">
                        <TADPar FakeNameAfter="" FakeNameBefore="" parDesc=""
                        parName="" typeID="idvalue0" />
                        <TPDParSet FakeNameAfter="" FakeNameBefore="" parDesc=""
                        parName="" typeID="idvalue0" />
                </FPMD>
                <FPSX BaseName="" FUNCID="idvalue29" ISMETHOD="S"
                        RetType="idvalue0" SCHID="idvalue0" retMode="SET">
                        <TADPar FakeNameAfter="" FakeNameBefore="" parDesc=""
                                parName="" typeID="idvalue0" />
                        <TADPar FakeNameAfter="" FakeNameBefore="" parDesc=""
                                parName="" typeID="idvalue0" />
                </FPSX>
        </XWProject>
</SectionD>
```

Exemplo 27

[D-3-0] - $\boldsymbol{X}$W[$\mathbf{F}_{\text{Agregação}}$] – O elemento do descritor que define a seção de funções de agregação é **<XWAgregate>**. A descrição individual de cada função de agregação(**<FA1>**, **<FA2>**, etc.), como pode ser visto no Exemplo 28, Exemplo 29 e Exemplo 30, é a mesma no que ser refere aos atributos, contudo devido a grande possibilidade de combinações de categorias de parâmetros nós dividimos o descritor em seis seções(Figura 58).

[D-3-1] - $\mathbf{FA1}^{n}_{\text{Agregação}}(\boldsymbol{X}W_{\text{coluna}}{}^{1}, [\boldsymbol{X}W_{\text{tipo}} \mid \boldsymbol{X}W_{\text{coluna}}{}^{2} \mid DW_{\text{parâmetro}}])$ – Elementos (**<FA1>**) da seção (**<FA1Section>**)

[D-3-2] - $\mathbf{FA2}^{n}_{\text{Agregação}}(\boldsymbol{X}W_{\text{coluna}}{}^{1}, [\boldsymbol{X}W_{\text{tipo}} \mid \boldsymbol{X}W_{\text{coluna}}{}^{2} \mid DW_{\text{parâmetro}}])$ – Elementos (**<FA2>**) da seção (**<FA2Section>**)

```
<SectionD>
        …
        <XWAgregate>
                <FA1Section>
                        <FA1 BaseName="" FUNCID="idvalue30" ISMETHOD="S"
                                RetType="idvalue0" SCHID="idvalue0" retMode="SET">
                                <TADParSet FakeNameAfter="" FakeNameBefore=""
                                        typeID="idvalue0" />
                                <TADPar FakeNameAfter="" FakeNameBefore="" parDesc=""
                                        parName="" typeID="idvalue0" />
                        </FA1>
                </FA1Section>
                <FA2Section>
                        <FA2 BaseName="" FUNCID="idvalue31" ISMETHOD="S"
                                RetType="idvalue0" SCHID="idvalue0" retMode="SET">
                                <TADParSet FakeNameAfter="" FakeNameBefore=""
                                        typeID="idvalue0" />
                                <TADPar FakeNameAfter="" FakeNameBefore="" parDesc=""
                                        parName="" typeID="idvalue0" />
                        </FA2>
                </FA2Section>
…
```

Exemplo 28

[D-3-3] - **FA11**$^{n}_{\text{Agregação}}$(***X***W$_{\text{coluna}}$,[DW$_{\text{parâmetro}}$]) – Elementos (**<FA11>**) da seção (**<FA11Section>**)

[D-3-4] - **FA12**$^{n}_{\text{Agregação}}$(***X***W$_{\text{coluna}}$,[DW$_{\text{parâmetro}}$]) – Elementos (**<FA12>**) da seção (**<FA12Section>**)

```
<SectionD>
        …
        <XWAgregate>
…
                <FA11Section>
                        <FA11 BaseName="" FUNCID="idvalue32" ISMETHOD="S"
                                RetType="idvalue0" SCHID="idvalue0" retMode="SET">
                                <TADParSet FakeNameAfter="" FakeNameBefore=""
                                        typeID="idvalue0" />
                                <TPDPar FakeNameAfter="" FakeNameBefore="" parDesc=""
                                        parName="" typeID="idvalue0" />
                        </FA11>
                </FA11Section>
                <FA12Section>
                        <FA12 BaseName="" FUNCID="idvalue33" ISMETHOD="S"
                                RetType="idvalue0" SCHID="idvalue0" retMode="SET">
                                <TADParSet FakeNameAfter="" FakeNameBefore=""
                                        typeID="idvalue0" />
                                <TPDPar FakeNameAfter="" FakeNameBefore="" parDesc=""
                                        parName="" typeID="idvalue0" />
                        </FA12>
                </FA12Section>
```

Exemplo 29

[D-3-5] - **FA21**$^{n}{}_{Agregação}$(***X***W$_{coluna}{}^{1}$, [***X***W$_{tipo}$ | ***X***W$_{coluna}{}^{2}$]) – Elementos (**<FA21>**) da seção (**<FA21Section>**)

[D-3-6] - **FA22**$^{n}{}_{Agregação}$(***X***W$_{coluna}{}^{1}$, [***X***W$_{tipo}$ | ***X***W$_{coluna}{}^{2}$]) – Elementos (**<FA22>**) da seção (**<FA22Section>**)

```
<XWAgregate>
        …
        <FA21Section>
                <FA21 BaseName="" FUNCID="idvalue34" ISMETHOD="S"
                        RetType="idvalue0" SCHID="idvalue0" retMode="SET">
                        <TADParSet FakeNameAfter="" FakeNameBefore=""
                                typeID="idvalue0" />
                        <TADPar FakeNameAfter="" FakeNameBefore="" parDesc=""
                                parName="" typeID="idvalue0" />
                </FA21>
        </FA21Section>
        <FA22Section>
                <FA22 BaseName="" FUNCID="idvalue35" ISMETHOD="S"
                        RetType="idvalue0" SCHID="idvalue0" retMode="SET">
                        <TADParSet FakeNameAfter="" FakeNameBefore=""
                                typeID="idvalue0" />
                        <TADPar FakeNameAfter="" FakeNameBefore="" parDesc=""
                                parName="" typeID="idvalue0" />
                </FA22>
        </FA22Section>
        <FASTD1Section>
                <FASTD1 BaseName="" FUNCID="idvalue36" ISMETHOD="S"
                          RetType="idvalue0" SCHID="idvalue0" retMode="SET">
                        <TPDParSet FakeNameAfter="" FakeNameBefore="" parDesc=""
                                parName="" typeID="idvalue0" />
                        <TPDPar FakeNameAfter="" FakeNameBefore="" parDesc=""
                                parName="" typeID="idvalue0" />
                </FASTD1>
        </FASTD1Section>
        <FASTD2Section>
                <FASTD2 BaseName="" FUNCID="idvalue37" ISMETHOD="S"
                          RetType="idvalue0" SCHID="idvalue0" retMode="SET">
                        <TPDParSet FakeNameAfter="" FakeNameBefore="" parDesc=""
                                parName="" typeID="idvalue0" />
                        <TPDPar FakeNameAfter="" FakeNameBefore="" parDesc=""
                                parName="" typeID="idvalue0" />
                </FASTD2>
        </FASTD2Section>
        ...
</XWAgregate>
```

Exemplo 30

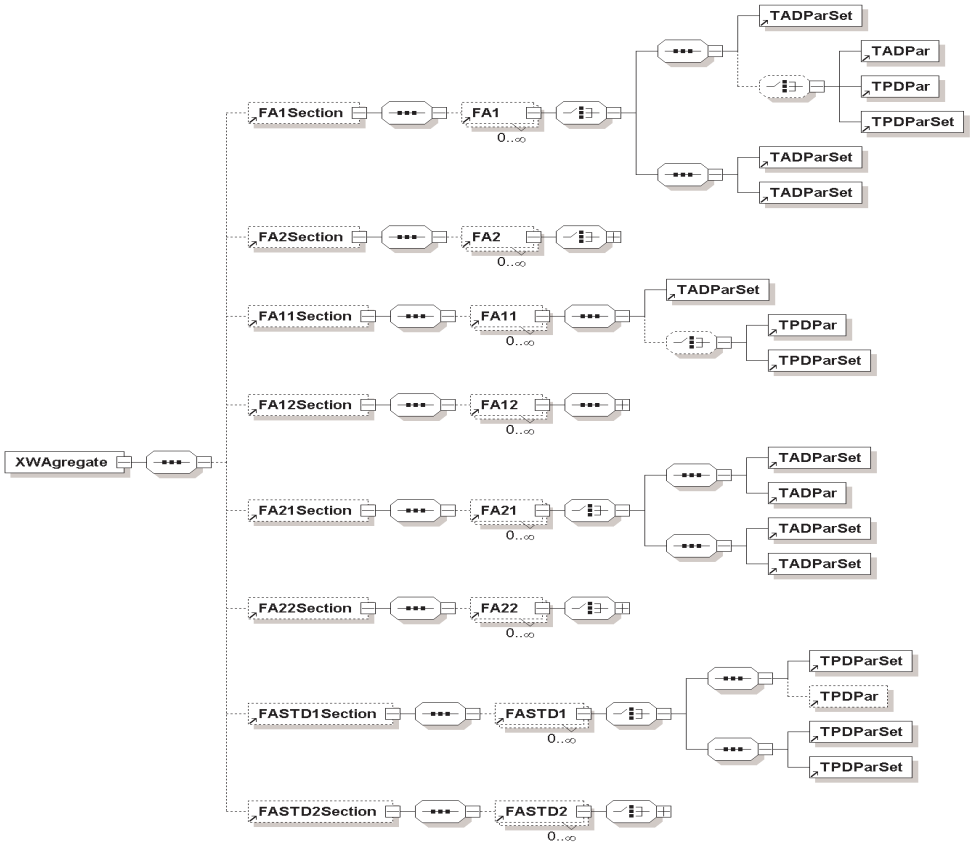

Figura 58 - Esquema da Seção D – Agregação

B.6 – Seção E

[E-0-0] - $\boldsymbol{X}$W[$\mathbf{P}_{\text{processamento}}$] = {$\mathbf{PE}_{\text{Extração}}$, $\mathbf{PT}_{\text{Transformação}}$, $\mathbf{PA}_{\text{Análise}}$}

[E-1-0] - $\boldsymbol{X}$W[$\mathbf{PE}_{\text{Extração}}$] – Os procedimentos de extração são registrados dentro do elemento **<XWExtract>** do descritor e esta seção pode ser vista na Figura 59.

[E-1-1] - **Importação** – Os procedimentos de importação, registrados na seção correspondente ao elemento **<XWImport>**(Exemplo 31). Eles podem conter qualquer número de parâmetros de tipos primitivos de dados e seguem o padrão de conter nome no banco(**BaseName**), identificador único de procedimento(**PEID**) e de esquema(**SCHID**). Alem disso, todos eles possuem duas versões, uma contendo as informações de arquivos de entrada e outro contendo informações de uma tabela de um banco de dados externo. As informações de arquivos de entrada são fornecidas pelo elemento **<XWFile>** que contém um identificador (**ARQID**) para um arquivo de sistema definido na seção de recursos externos. Os dados das tabelas externas contém atributos

para indicar banco (**Database**), esquema (**SourceSchema**), usuário (**UserName**) e senha (**Password**) de qualquer tabela externa (**SourceTable**). A tabela de destino dos dados importados contém atributos equivalentes as tabelas de entrada, com exceção de usuário e senha e todos os procedimentos podem conter um número qualquer de tabelas de destino, bem como tabelas ou arquivos de entrada. A lista abaixo mostra os elementos do descritor para cada procedimento.

[E-1-1-1] - **PEI**$^{n}_{\text{Importação}}$([***ArqExt*** | ***TabList***$^{B}_{\text{fonte}}$],***TabList***$^{A}_{\text{destino}}$[,$DW_{\text{parâmetro}}$]) – Elementos (**<PEII1>**) e (**<PEII2>**), de acordo com entrada de arquivo ou tabela.

[E-1-1-2] - **PEA**$^{n}_{\text{Agregação}}$([***ArqExt*** | ***TabList***$^{B}_{\text{fonte}}$], ***TabList***$^{A}_{\text{destino}}$[,$DW_{\text{parâmetro}}$]) – Elementos (**<PEIA1>**) e (**<PEIA2>**), de acordo com entrada de arquivo ou tabela.

[E-1-1-3] - **PES**$^{n}_{\text{Sequencializador}}$([***ArqExt***|***TabList***$^{B}_{\text{fonte}}$],***TabList***$^{A}_{\text{destino}}$[,$DW_{\text{parâm.}}$]) – Elementos (**<PEIS1>**) e (**<PEIS2>**), de acordo com entrada de arquivo ou tabela.

```
<SectionE>
        <XWExtract>
                <XWImport>
                        <PEII1 BaseName="" PEID="idvalue38" SCHID="idvalue0">
                                <XWFile ARQID="idvalue0" Seq="" />
                                <TgtTable TargetSchema="" TargetTable="" />
                                <TPDPar FakeNameAfter="" FakeNameBefore="" parDesc=""
                                        parName="" typeID="idvalue0" />
                        </PEII1>
                        <PEII2 BaseName="" PEID="idvalue39" SCHID="idvalue0">
                                <SrcTable Database="" Password="" SourceSchema=""
                                        SourceTable="" UserName="" />
                                <TgtTable TargetSchema="" TargetTable="" />
                                <TPDPar FakeNameAfter="" FakeNameBefore="" parDesc=""
                                        parName="" typeID="idvalue0" />
                        </PEII2>
                        <PEIA1 BaseName="" PEID="idvalue40" SCHID="idvalue0">
                                <XWFile ARQID="idvalue0" Seq="" />
                                <TgtTable TargetSchema="" TargetTable="" />
                                <TPDPar FakeNameAfter="" FakeNameBefore="" parDesc=""
                                        parName="" typeID="idvalue0" />
                        </PEIA1>
                        <PEIA2 BaseName="" PEID="idvalue41" SCHID="idvalue0">
                                <SrcTable Database="" Password="" SourceSchema=""
                                        SourceTable="" UserName="" />
                                <TgtTable TargetSchema="" TargetTable="" />
                                <TPDPar FakeNameAfter="" FakeNameBefore="" parDesc=""
                                        parName="" typeID="idvalue0" />
                        </PEIA2>
```

```
                    <PEIS1 BaseName="" PEID="idvalue42" SCHID="idvalue0">
                        <XWFile ARQID="idvalue0" Seq="" />
                        <TgtTable TargetSchema="" TargetTable="" />
                        <TPDPar FakeNameAfter="" FakeNameBefore="" parDesc=""
                                parName="" typeID="idvalue0" />
                    </PEIS1>
                    <PEIS2 BaseName="" PEID="idvalue43" SCHID="idvalue0">
                        <SrcTable Database="" Password="" SourceSchema=""
                                SourceTable="" UserName="" />
                        <TgtTable TargetSchema="" TargetTable="" />
                        <TPDPar FakeNameAfter="" FakeNameBefore="" parDesc=""
                                parName="" typeID="idvalue0" />
                    </PEIS2>
                </XWImport>
```

Exemplo 31

[E-1-2] - **Preprocessamento** – O descritor da seção de funções de pré-processamento utiliza o elemento **<XWPreprocess>** e a lista de elementos para cada função encontra-se abaixo:

[E-1-2-1] - $\mathbf{PEP}^{n}_{\text{Preprocessamento}}(\boldsymbol{TabList}^{A}_{\text{fonte}}, \boldsymbol{TabList}^{A}_{\text{destino}}[,DW_{\text{parâmetro}}])$ – Elementos (**<PEPP>**)

[E-1-2-2] - $\mathbf{PEL}^{n}_{\text{Limpeza}}(\boldsymbol{TabList}^{A}_{\text{fonte}}, \boldsymbol{TabList}^{A}_{\text{destino}}[,DW_{\text{parâmetro}}])$ – Elementos (**<PEPL>**)

[E-1-2-3] - $\mathbf{PEV}^{n}_{\text{Validação}}(\boldsymbol{TabList}^{A}_{\text{fonte}}, \boldsymbol{TabList}^{A}_{\text{destino}}[,DW_{\text{parâmetro}}])$ – Elementos (**<PEPV>**)

```
<SectionE>
        <XWExtract>
                <XWPreprocess>
                        <PEPP BaseName="" PEID="idvalue44" SCHID="idvalue0">
                            <SrcTable  SourceSchema="" SourceTable=""/>
                            <TgtTable TargetSchema="" TargetTable="" />
                            <TPDPar FakeNameAfter="" FakeNameBefore="" parDesc=""
                                    parName="" typeID="idvalue0" />
                        </PEPP>
                        <PEPL BaseName="" PEID="idvalue45" SCHID="idvalue0">
                            <SrcTable SourceSchema="" SourceTable="" />
                            <TgtTable TargetSchema="" TargetTable="" />
                            <TPDPar FakeNameAfter="" FakeNameBefore="" parDesc=""
                                    parName="" typeID="idvalue0" />
                        </PEPL>
                        <PEPV BaseName="" PEID="idvalue46" SCHID="idvalue0">
                            <SrcTable SourceSchema="" SourceTable="" />
                            <TgtTable TargetSchema="" TargetTable="" />
                            <TPDPar FakeNameAfter="" FakeNameBefore="" parDesc=""
                                    parName="" typeID="idvalue0" />
                        </PEPV>
                </XWPreprocess>
        </XWExtract>
...
```

Exemplo 32

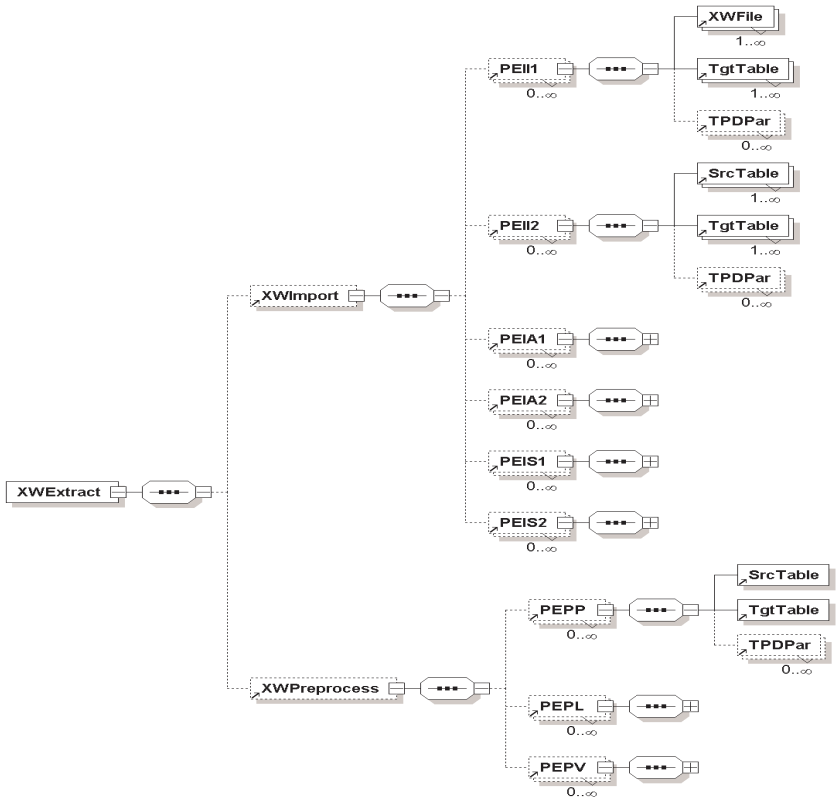

Figura 59 - Esqema da Seção E – Extração

[E-2-0] - ***X***W[**PT**$_{\text{Transformação}}$] – Esta seção (**<XWTranform>**) contém um elemento para cada categoria de transformação(Figura 60). O primeiro elemento (**<XWGeneral>**) define funções genéricas para transformação (Exemplo 33). Estas funções podem conter qualquer número de tabelas de entrada(**<SrcTable>**) e de destino(**<TgtTable>**) assim como parâmetros(**<TPDPar>**). Os outros atributos do procedimento são iguais aos de extração com exceção do mapeamento de colunas e de SQL configurável. O primeiro elemento(**<ColMapping>**) permite relacionar quais as colunas das tabelas fonte que irão se transferir para as tabelas destino e o segundo(**<SQLMapping>**) permite que a tabela destino seja produzida a partir do resultado de uma consulta em SQL. A lista abaixo contém a descrição de cada elemento do descritor:

[E-2-1] - **PTG1**$^{n}_{\text{Transformação}}$(***TabList***$_{\text{fonte}}$,***TabList***$_{\text{destino}}$[,Parâmetros]) – Elementos (**<PTG>**)

[E-2-2] - **PTG2**$^{n}_{\text{Transformação}}$(***TabList***$_{\text{fonte}}$, ***TabList***$_{\text{destino}}$, SQL$_{\text{list}}$[,Parâmetros]) – Elementos (**<PTG>**)

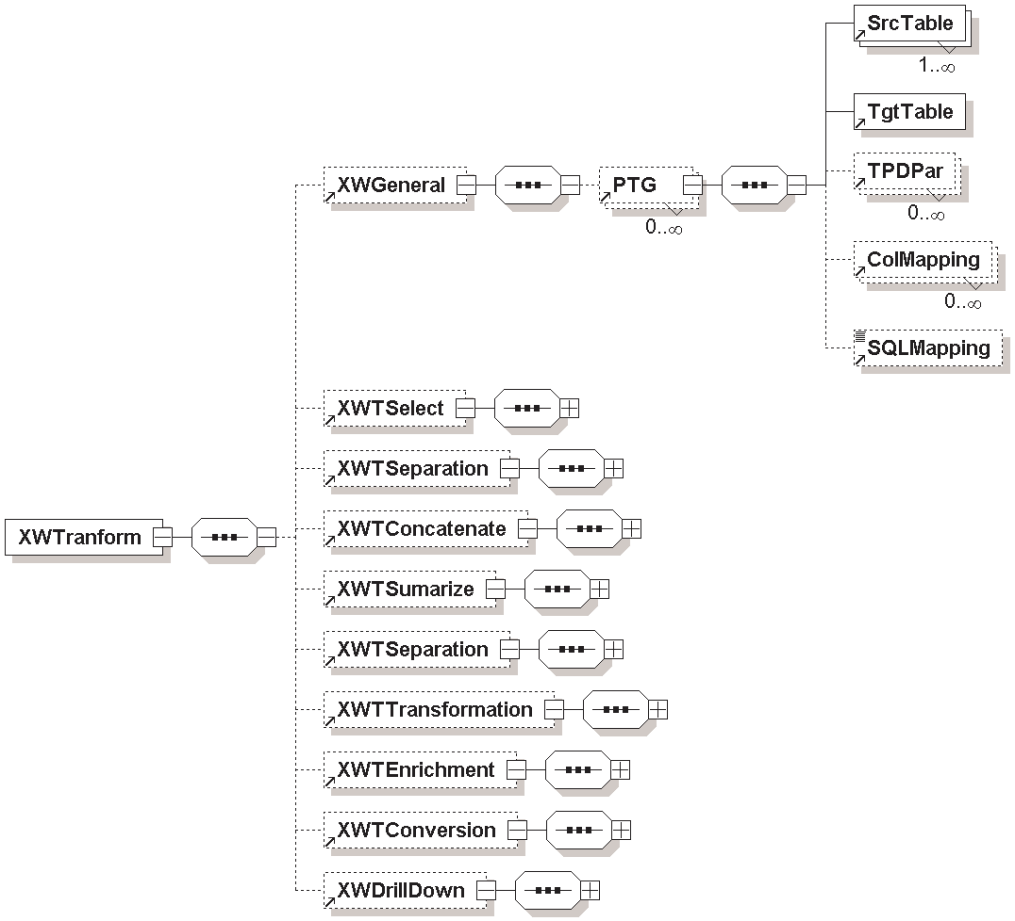

Figura 60 - Esquema da Seção E – Transformação

```
<SectionE>
        <XWTranform>
                <XWGeneral>
                        <PTG BaseName="" SCHID="idvalue0">
                                <SrcTable   SourceSchema=""  SourceTable="" />
                                <TgtTable TargetSchema="" TargetTable="" />
                                <TPDPar FakeNameAfter="" FakeNameBefore="" parDesc=""
                                                parName="" typeID="idvalue0" />
                                <ColMapping SourceCol="" TargetCol="" />
                                <SQLMapping>SQLMapping</SQLMapping>
                        </PTG>
                </XWGeneral>
…
```

Exemplo 33

[E-2-3] - **Seleção** – Os procedimentos específicos para transformação(Figura 61 e Exemplo 34) por seleção vão seguir a padronização dos procedimentos gerais de transformação com a exceção de permitir até duas tabelas de entrada(para possibilita a realização de junções) e um identificador da função

de suporte relacional que será usada para realizar a seleção(**FRSID** e **FRMID**). A lista abaixo contém a descrição de cada elemento do descritor:

[E-2-3-1] - **PTSR**$^{n}_{\text{Seleção}}$(***TabList***$_{\text{fonte}}$, ***TabList***$_{\text{destino}}$, **FRS_**^{k} [,Parâmetros) – Elementos (**<PTSR>**)

[E-2-3-2] - **PTMR**$^{n}_{\text{Seleção}}$(***TabList***$_{\text{fonte}}$, ***TabList***$_{\text{destino}}$, **FRM_**^{k} [,Parâmetros]) – Elementos (**<PTMR>**)

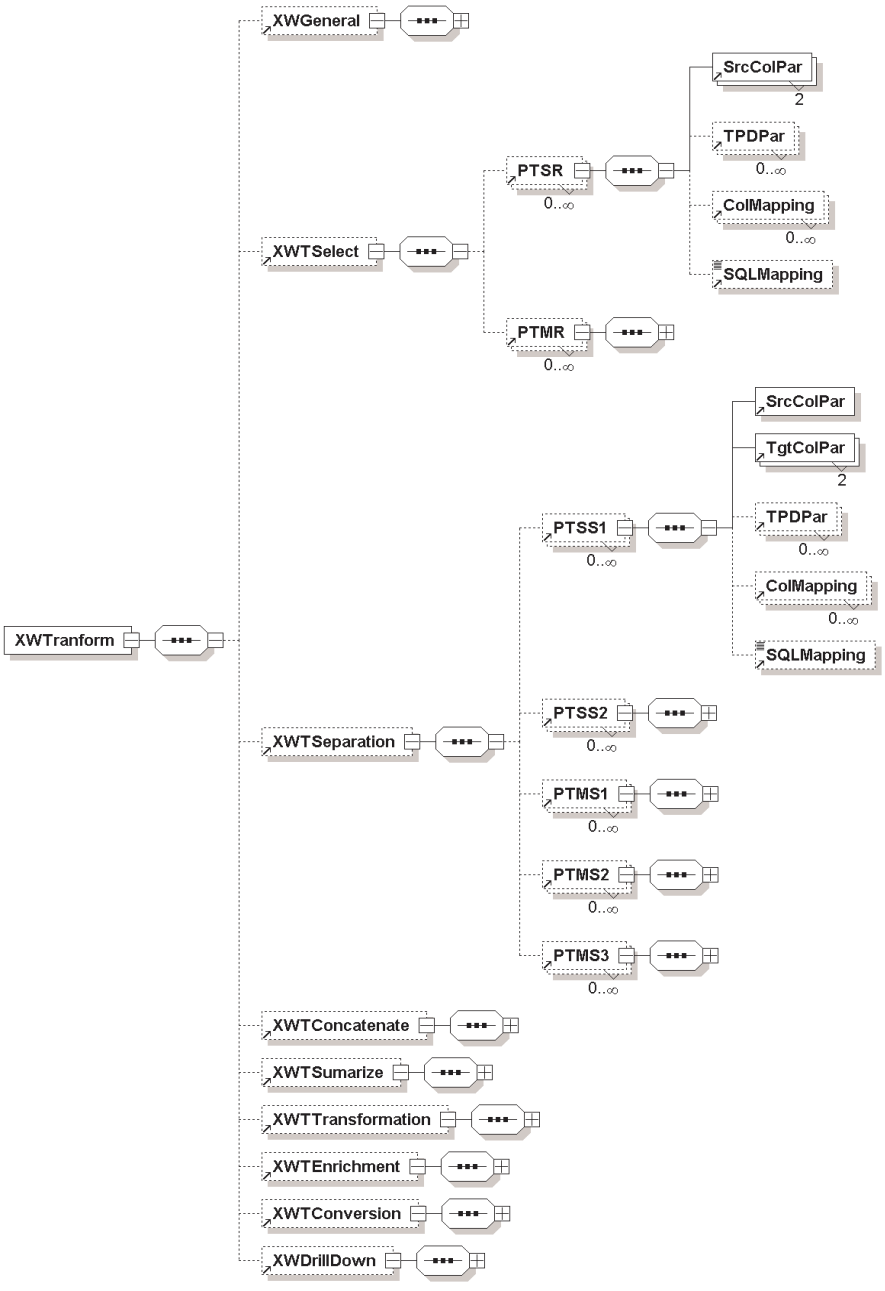

Figura 61 - Esquema da Seção E – Transformação – Seleção e Separação

```
<XWTSelect>
        <PTSR BaseName="" FRSID="idvalue0" SCHID="idvalue0" SourceSchema1=""
                SourceSchema2="" SourceTable1="" SourceTable2="" TargetSchema=""
                TargetTable="">
                <SrcColPar name="" />
                <TPDPar FakeNameAfter="" FakeNameBefore="" parDesc="" parName=""
                        typeID="idvalue0" />
                <ColMapping AggFunc="" SourceCol="" TargetCol="" />
                <SQLMapping>SQLMapping</SQLMapping>
        </PTSR>
        <PTMR BaseName="" FRMID="idvalue0" SCHID="idvalue0" SourceSchema1=""
                SourceSchema2="" SourceTable1="" SourceTable2="" TargetSchema=""
                TargetTable="">
                <SrcColPar name="" />
                <TPDPar FakeNameAfter="" FakeNameBefore="" parDesc="" parName=""
                        typeID="idvalue0" />
                <ColMapping AggFunc="" SourceCol="" TargetCol="" />
                <SQLMapping>SQLMapping</SQLMapping>
        </PTMR>
</XWTSelect>
```

Exemplo 34

[E-2-4] - Separação – As funções de separação(**<XWTSeparation>**), contrárias as de seleção, recebem como parâmetro uma única tabela de entrada e duas tabelas de destino(Exemplo 35). A lista abaixo contém a descrição de cada elemento do descritor:

[E-2-4-1] - $\mathbf{PTSS1}^{n}_{\text{Separação}}(\boldsymbol{TabList}_{\text{fonte}}, \boldsymbol{TabList}_{\text{destino}}, \mathbf{FPSC}^{k}$ [,Parâmetros]) – Elementos (**<PTSS1>**)

[E-2-4-2] - $\mathbf{PTSS2}^{n}_{\text{Separação}}(\boldsymbol{TabList}_{\text{fonte}}, \boldsymbol{TabList}_{\text{destino}}, \mathbf{FCA}^{k}$ [,Parâmetros]) – Elementos (**<PTSSS2>**)

[E-2-4-3] - $\mathbf{PTMS1}^{n}_{\text{Separação}}(\boldsymbol{TabList}_{\text{fonte}}, \boldsymbol{TabList}_{\text{destino}}, \mathbf{FPMC}^{k}$ [,Parâmetros]) – Elementos (**<PTMS1>**)

[E-2-4-4] - $\mathbf{PTMS2}^{n}_{\text{Separação}}(\boldsymbol{TbList}_{\text{fonte}}, \boldsymbol{TbList}_{\text{destino}}, \mathbf{FA12}^{k}$ [,Parâmetros]) – Elementos (**<PTMS2>**)

```
<XWTSeparation>
        <PTSS1 BaseName="" FPSCID1="idvalue0" FPSCID2="idvalue0" SCHID="idvalue0"
                        SourceSchema="" SourceTable="" TargetSchema1=""
                        TargetSchema2="" TargetTable1="" TargetTable2="">
                <SrcColPar name="" />
                <TgtColPar name="" />
                <TPDPar FakeNameAfter="" FakeNameBefore="" parDesc="" parName=""
                                typeID="idvalue0" />
                <ColMapping AggFunc="" SourceCol="" TargetCol="" />
                <SQLMapping>SQLMapping</SQLMapping>
        </PTSS1>
        <PTSS2 BaseName="" FCAID1="idvalue0" FCAID2="idvalue0" SCHID="idvalue0"
                        SourceSchema="" SourceTable="" TargetSchema1=""
                        TargetSchema2="" TargetTable1="" TargetTable2="">
                <SrcColPar name="" />
                <TgtColPar name="" />
                <TPDPar FakeNameAfter="" FakeNameBefore="" parDesc="" parName=""
                                typeID="idvalue0" />
                <ColMapping AggFunc="" SourceCol="" TargetCol="" />
                <SQLMapping>SQLMapping</SQLMapping>
        </PTSS2>
        <PTMS1 BaseName="" FPMCID1="idvalue0" FPMCID2="idvalue0"
                        SCHID="idvalue0" SourceSchema="" SourceTable=""
                        TargetSchema1="" TargetSchema2="" TargetTable1=""
                        TargetTable2="">
                <SrcColPar name="" />
                <TgtColPar name="" />
                <TPDPar FakeNameAfter="" FakeNameBefore="" parDesc="" parName=""
                        typeID="idvalue0" />
                <ColMapping AggFunc="" SourceCol="" TargetCol="" />
                <SQLMapping>SQLMapping</SQLMapping>
        </PTMS1>
        <PTMS2 BaseName="" FA12ID1="idvalue0" FA12ID2="idvalue0" SCHID="idvalue0"
                        SourceSchema="" SourceTable="" TargetSchema1=""
                        TargetSchema2="" TargetTable1="" TargetTable2="">
                <SrcColPar name="" />
                <TgtColPar name="" />
                <TPDPar FakeNameAfter="" FakeNameBefore="" parDesc="" parName=""
                                typeID="idvalue0" />
                <ColMapping AggFunc="" SourceCol="" TargetCol="" />
                <SQLMapping>SQLMapping</SQLMapping>
        </PTMS2>
        <PTMS3 BaseName="" FA2ID1="idvalue0" FA2ID2="idvalue0" SCHID="idvalue0"
                        SourceSchema="" SourceTable="" TargetSchema1=""
                        TargetSchema2="" TargetTable1="" TargetTable2="">
                <SrcColPar name="" />
                <TgtColPar name="" />
                <TPDPar FakeNameAfter="" FakeNameBefore="" parDesc="" parName=""
                                typeID="idvalue0" />
                <ColMapping AggFunc="" SourceCol="" TargetCol="" />
                <SQLMapping>SQLMapping</SQLMapping>
        </PTMS3>
</XWTSeparation>
```

Exemplo 35

[E-2-5] - **Concatenação** – O formato dos procedimentos de concatenação(Figura 62) são iguais aos de seleção, recebem duas tabelas de entrada e uma de destino. Os outros parâmetros são iguais. Os identificadores de funções(**FPSXID** e **FA21ID**) apontam para as funções de suporte de projeção que serão usadas para realizar a transformação(Exemplo 36). A lista abaixo contém a descrição de cada elemento do descritor:

[E-2-5-1] - $\mathbf{PTSU}^{n}_{\text{Concatenação}}(\boldsymbol{TbList}_{\text{fonte}}, \boldsymbol{TbList}_{\text{destino}}, \mathbf{FPSX}^{k}$ [,Parâmetros]) – Elementos (**<PTSU>**)

[E-2-5-2] - $\mathbf{PTMU}^{n}_{\text{Concatenação}}(\boldsymbol{TabList}_{\text{fonte}}, \boldsymbol{TabList}_{\text{destino}}, \mathbf{FA21}^{k}$ [,Parâmetros]) – Elementos (**<PTMU>**)

```
<XWTConcatenate>
        <PTSU BaseName="" FPSXID="idvalue0" SCHID="idvalue0" SourceSchema1=""
                SourceSchema2="" SourceTable1="" SourceTable2="" TargetSchema=""
                TargetTable="">
                <SrcColPar name="" />
                <TgtColPar name="" />
                <TPDPar FakeNameAfter="" FakeNameBefore="" parDesc="" parName=""
                        typeID="idvalue0" />
                <ColMapping AggFunc="" SourceCol="" TargetCol="" />
                <SQLMapping>SQLMapping</SQLMapping>
        </PTSU>
        <PTMU BaseName="" FA21ID="" SCHID="idvalue0" SourceSchema1=""
                SourceSchema2="" SourceTable1="" SourceTable2="" TargetSchema=""
                TargetTable="">
                <SrcColPar name="" />
                <TgtColPar name="" />
                <TPDPar FakeNameAfter="" FakeNameBefore="" parDesc="" parName=""
                        typeID="idvalue0" />
                <ColMapping AggFunc="" SourceCol="" TargetCol="" />
                <SQLMapping>SQLMapping</SQLMapping>
        </PTMU>
</XWTConcatenate>
```

Exemplo 36

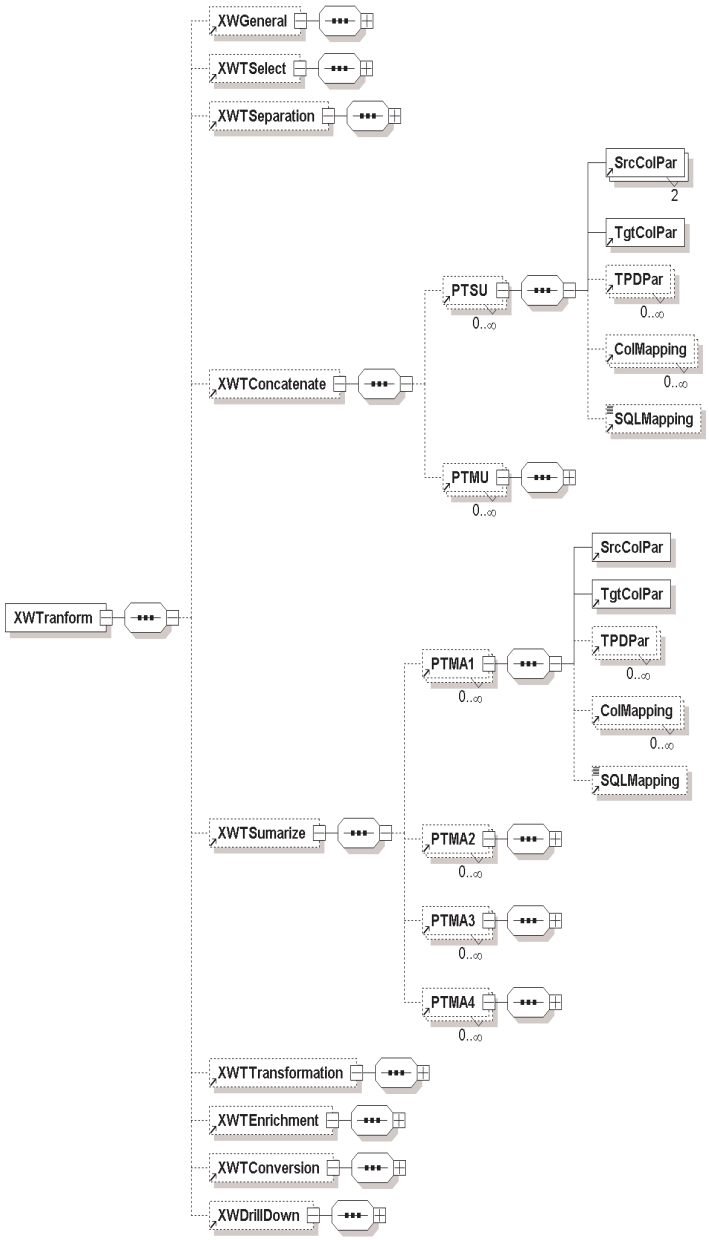

Figura 62 - Esquma da Seção E – Transformação – Concatenação e sumarização

[E-2-6] - **Sumarização** – Os procedimentos de sumarização(**<XWTSumarize>**) recebem somente uma tabela de entrada e uma de destino(Exemplo 37). Seu objetivo vai ser a agregação através das funções de suporte relacional indicadas(**FA11ID**, **FA12ID**, **FA21ID** e **FA22ID**). A lista abaixo contém a descrição de cada elemento do descritor:

[E-2-6-1] - $\mathbf{PTSA1}^{n}_{\text{Sumarização}}(\boldsymbol{TbList}_{\text{fonte}},\boldsymbol{TbList}_{\text{destino}},\mathbf{FA11}^{k}$ [,Parâmetros]) – Elementos (**<PTSA1>**)

[E-2-6-2] - **PTMA1**$^{n}_{\text{Sumarização}}$(***TbList***$_{\text{fonte}}$,***TbList***$_{\text{destino}}$,**FA12**^{k}[,Parâmetros]) – Elementos (**<PTMA1>**)

[E-2-6-3] - **PTSA2**$^{n}_{\text{Sumarização}}$(***TbList***$_{\text{fonte}}$,***TbList***$_{\text{destino}}$,**FA21**^{k}[,Parâmetros]) – Elementos (**<PTSA2>**)

[E-2-6-4] - **PTMA2**$^{n}_{\text{Sumarização}}$(***TbList***$_{\text{fonte}}$,***TbList***$_{\text{destino}}$,**FA22**^{k}[,Parâmetros]) – Elementos (**<PTMA2>**)

```
<XWTSumarize>
        <PTMA1 BaseName="" FA11ID="idvalue0" SCHID="idvalue0" SourceSchema=""
                        SourceTable="" TargetSchema="" TargetTable="">
                <SrcColPar name="" />
                <TgtColPar name="" />
                <TPDPar FakeNameAfter="" FakeNameBefore="" parDesc="" parName=""
                                typeID="idvalue0" />
                <ColMapping AggFunc="" SourceCol="" TargetCol="" />
                <SQLMapping>SQLMapping</SQLMapping>
        </PTMA1>
        <PTMA2 BaseName="" FA12ID="idvalue0" SCHID="idvalue0" SourceSchema=""
                        SourceTable="" TargetSchema="" TargetTable="">
                <SrcColPar name="" />
                <TgtColPar name="" />
                <TPDPar FakeNameAfter="" FakeNameBefore="" parDesc="" parName=""
                                typeID="idvalue0" />
                <ColMapping AggFunc="" SourceCol="" TargetCol="" />
                <SQLMapping>SQLMapping</SQLMapping>
        </PTMA2>
        <PTMA3 BaseName="" FA21ID="idvalue0" SCHID="idvalue0" SourceSchema=""
                        SourceTable="" TargetSchema="" TargetTable="">
                <SrcColPar name="" />
                <TgtColPar name="" />
                <TPDPar FakeNameAfter="" FakeNameBefore="" parDesc="" parName=""
                                typeID="idvalue0" />
                <ColMapping AggFunc="" SourceCol="" TargetCol="" />
                <SQLMapping>SQLMapping</SQLMapping>
        </PTMA3>
        <PTMA4 BaseName="" FA22ID="idvalue0" SCHID="idvalue0" SourceSchema=""
                        SourceTable="" TargetSchema="" TargetTable="">
                <SrcColPar name="" />
                <TgtColPar name="" />
                <TPDPar FakeNameAfter="" FakeNameBefore="" parDesc="" parName=""
                        typeID="idvalue0" />
                <ColMapping AggFunc="" SourceCol="" TargetCol="" />
                <SQLMapping>SQLMapping</SQLMapping>
        </PTMA4>
</XWTSumarize>
```

Exemplo 37

[E-2-7] - Transformação – Segue o mesmo padrão do descritor de procedimentos de sumarização e utiliza os identificadores de função

especificados no *framework*(Figura 63 e Exemplo 38). A lista abaixo contém a descrição de cada elemento do descritor:

[E-2-7-1] - $\mathbf{PTST1}^{n}_{\text{Transformaçao}}(\boldsymbol{TbList}_{\text{fonte}}, \boldsymbol{TbList}_{\text{destino}}, \mathbf{FPST}^{k}$[,Parâmetros]) – Elementos (**<PTST1>**)

[E-2-7-2] - $\mathbf{PTMT1}^{n}_{\text{Transformaçao}}(\boldsymbol{TbList}_{\text{fonte}}, \boldsymbol{TbList}_{\text{destino}}, \mathbf{FPMT}^{k}$[,Parâmetros])]) – Elementos (**<PTMT1>**)

[E-2-7-3] - $\mathbf{PTMT2}^{n}_{\text{Transformaçao}}(\boldsymbol{TbList}_{\text{fonte}}, \boldsymbol{TbList}_{\text{destino}}, \mathbf{FA12}^{k}$[,Parâmetros]) – Elementos (**<PTMT1>**)

```
<XWTTransformation>
        <PTST1 BaseName="" FPSTID="idvalue0" SCHID="idvalue0" SourceSchema=""
                SourceTable="" TargetSchema="" TargetTable="">
                <SrcColPar name="" />
                <TgtColPar name="" />
                <TPDPar FakeNameAfter="" FakeNameBefore="" parDesc="" parName=""
                        typeID="idvalue0" />
                <ColMapping AggFunc="" SourceCol="" TargetCol="" />
                <SQLMapping>SQLMapping</SQLMapping>
        </PTST1>
        <PTMT1 BaseName="" FPMTID="idvalue0" SCHID="idvalue0" SourceSchema=""
                SourceTable="" TargetSchema="" TargetTable="">
                <SrcColPar name="" />
                <TgtColPar name="" />
                <TPDPar FakeNameAfter="" FakeNameBefore="" parDesc="" parName=""
                        typeID="idvalue0" />
                <ColMapping AggFunc="" SourceCol="" TargetCol="" />
                <SQLMapping>SQLMapping</SQLMapping>
        </PTMT1>
        <PTMT2 BaseName="" FA12ID="idvalue0" SCHID="idvalue0" SourceSchema=""
                SourceTable="" TargetSchema="" TargetTable="">
                <SrcColPar name="" />
                <TgtColPar name="" />
                <TPDPar FakeNameAfter="" FakeNameBefore="" parDesc="" parName=""
                        typeID="idvalue0" />
                <ColMapping AggFunc="" SourceCol="" TargetCol="" />
                <SQLMapping>SQLMapping</SQLMapping>
        </PTMT2>
</XWTTransformation>
```

Exemplo 38

[E-2-8] - **Enriquecimento** – Segue o padrão anterior(Exemplo 39). A lista abaixo contém a descrição de cada elemento do descritor:

[E-2-8-1] - $\mathbf{PTSE1}^{n}_{\text{Enriquecimento}}(\boldsymbol{TbList}_{\text{fonte}}, \boldsymbol{TabList}_{\text{destino}}, \mathbf{FPSC}^{k}$ [,Parâmetros]) – Elementos (**<PTSE1>**)

[E-2-8-2] - **PTSE2**$^{n}_{\text{Enriquecimento}}$(***TbList***$_{\text{fonte}}$,***TbList***$_{\text{destino}}$,**FCSC**^{n} [,Parâmetros]) – Elementos (**<PTSE2>**)

[E-2-8-3] - **PTME1**$^{n}_{\text{Enriquecimento}}$(***TbList***$_{\text{fonte}}$,***TbList***$_{\text{destino}}$,**FA12**^{k} [,Parâmetros]) – Elementos (**<PTME1>**)

[E-2-8-4] - **PTME2**$^{n}_{\text{Enriquecimento}}$(***TabList***$_{\text{fonte}}$,***TabList***$_{\text{destino}}$,**FCMC**^{n} [,Parâmetros]) – Elementos (**<PTME2>**)

[E-2-8-5] - **PTME3**$^{n}_{\text{Enriquecimento}}$(***TbList***$_{\text{fonte}}$, ***TbList***$_{\text{destino}}$, **FPMC**^{k} [,Parâmetros]) – Elementos (**<PTME3>**)

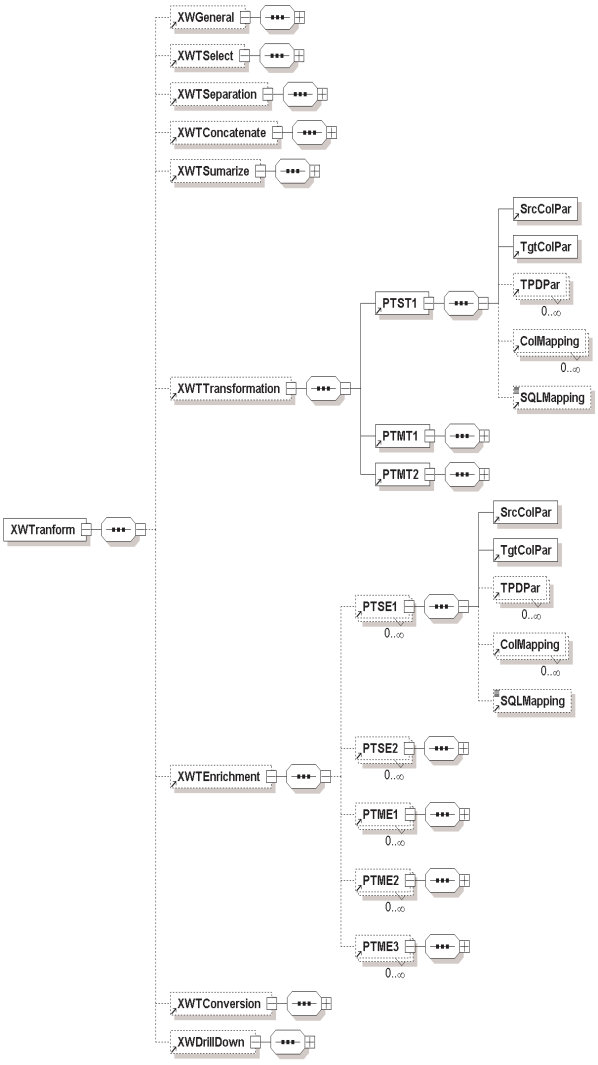

Figura 63 - Esquema da Seção E – Transformação – Enriquecimento

```
<XWTEnrichment>
        <PTSE1 BaseName="" FPSCID="idvalue0" SCHID="idvalue0" SourceSchema=""
                SourceTable="" TargetSchema="" TargetTable="">
                <SrcColPar name="" />
                <TgtColPar name="" />
                <TPDPar FakeNameAfter="" FakeNameBefore="" parDesc="" parName=""
                        typeID="idvalue0" />
                <ColMapping AggFunc="" SourceCol="" TargetCol="" />
                <SQLMapping>SQLMapping</SQLMapping>
        </PTSE1>
        <PTSE2 BaseName="" FCSCID="idvalue0" SCHID="idvalue0" SourceSchema=""
                SourceTable="" TargetSchema="" TargetTable="">
                <SrcColPar name="" />
                <TgtColPar name="" />
                <TPDPar FakeNameAfter="" FakeNameBefore="" parDesc="" parName=""
                        typeID="idvalue0" />
                <ColMapping AggFunc="" SourceCol="" TargetCol="" />
                <SQLMapping>SQLMapping</SQLMapping>
        </PTSE2>
        <PTME1 BaseName="" FA12ID="idvalue0" SCHID="idvalue0" SourceSchema=""
                SourceTable="" TargetSchema="" TargetTable="">
                <SrcColPar name="" />
                <TgtColPar name="" />
                <TPDPar FakeNameAfter="" FakeNameBefore="" parDesc="" parName=""
                        typeID="idvalue0" />
                <ColMapping AggFunc="" SourceCol="" TargetCol="" />
                <SQLMapping>SQLMapping</SQLMapping>
        </PTME1>
        <PTME2 BaseName="" FCMCID="idvalue0" SCHID="idvalue0" SourceSchema=""
                SourceTable="" TargetSchema="" TargetTable="">
                <SrcColPar name="" />
                <TgtColPar name="" />
                <TPDPar FakeNameAfter="" FakeNameBefore="" parDesc="" parName=""
                        typeID="idvalue0" />
                <ColMapping AggFunc="" SourceCol="" TargetCol="" />
                <SQLMapping>SQLMapping</SQLMapping>
        </PTME2>
        <PTME3 BaseName="" FPMCID="idvalue0" SCHID="idvalue0" SourceSchema=""
                SourceTable="" TargetSchema="" TargetTable="">
                <SrcColPar name="" />
                <TgtColPar name="" />
                <TPDPar FakeNameAfter="" FakeNameBefore="" parDesc="" parName=""
                        typeID="idvalue0" />
                <ColMapping AggFunc="" SourceCol="" TargetCol="" />
                <SQLMapping>SQLMapping</SQLMapping>
        </PTME3>
</XWTEnrichment>
```

Exemplo 39

[E-2-9] - **Conversão** – Também seguem o padrão anterior(Exemplo 40 e Figura 64). A lista abaixo contém a descrição de cada elemento do descritor:

[E-2-9-1] - $\mathbf{PTSC1}^{n}{}_{\text{Conversão}}(\boldsymbol{TabList}_{\text{fonte}}, \boldsymbol{TabList}_{\text{destino}}, \mathbf{FPSC}^{k}$ [,Parâmetros]) – Elementos (**<PTSC1>**)

[E-2-9-2] - **PTSC2**$^{n}_{Conversão}$(***TabList***$_{fonte}$, ***TabList***$_{destino}$, **FA1**^{n} [,Parâmetros]) – Elementos (**<PTSC2>**)

[E-2-9-3] - **PTMC1**$^{n}_{Conversão}$ (***TabList***$_{fonte}$,***TabList***$_{destino}$,**FPMC**$^{k}_{Conversão}$ [,Parâmetros]) – Elementos (**<PTMC1>**)

[E-2-9-4] - **PTMC2**$^{n}_{Conversão}$ (***TabList***$_{fonte}$,***TabList***$_{destino}$,**FA2**^{n} [,Parâmetros]) – Elementos (**<PTMC2>**)

```
<XWTConversion>
        <PTSC1 BaseName="" FPSCID="idvalue0" SCHID="idvalue0" SourceSchema=""
                SourceTable="" TargetSchema="" TargetTable="">
                <SrcColPar name="" />
                <TgtColPar name="" />
                <TPDPar FakeNameAfter="" FakeNameBefore="" parDesc="" parName=""
                        typeID="idvalue0" />
                <ColMapping AggFunc="" SourceCol="" TargetCol="" />
                <SQLMapping>SQLMapping</SQLMapping>
        </PTSC1>
        <PTMC1 BaseName="" FA1ID="idvalue0" SCHID="idvalue0" SourceSchema=""
                SourceTable="" TargetSchema="" TargetTable="">
                <SrcColPar name="" />
                <TgtColPar name="" />
                <TPDPar FakeNameAfter="" FakeNameBefore="" parDesc="" parName=""
                        typeID="idvalue0" />
                <ColMapping AggFunc="" SourceCol="" TargetCol="" />
                <SQLMapping>SQLMapping</SQLMapping>
        </PTMC1>
        <PTMC2 BaseName="" FCMCID="idvalue0" SCHID="idvalue0" SourceSchema=""
                SourceTable="" TargetSchema="" TargetTable="">
                <SrcColPar name="" />
                <TgtColPar name="" />
                <TPDPar FakeNameAfter="" FakeNameBefore="" parDesc="" parName=""
                        typeID="idvalue0" />
                <ColMapping AggFunc="" SourceCol="" TargetCol="" />
                <SQLMapping>SQLMapping</SQLMapping>
        </PTMC2>
</XWTConversion>
```

Exemplo 40

[E-2-10] - ***DrillDown*** – Adota todo o padrão anterior e de acordo com o Exemplo 41. A lista abaixo contém a descrição de cada elemento do descritor:

[E-2-10-1] - **PTSD**$^{n}_{DrillDown}$(***TabList***$_{fonte}$, ***TabList***$_{destino}$, **FPSD**^{k} [,Parâmetros]) – Elementos (**<PTSD>**)

[E-2-10-2] - **PTMD**$^{n}_{DrillDown}$ (***TabList***$_{fonte}$, ***TabList***$_{destino}$, **FPMD**^{n} [,Parâmetros]) – Elementos (**<PTSD>**)

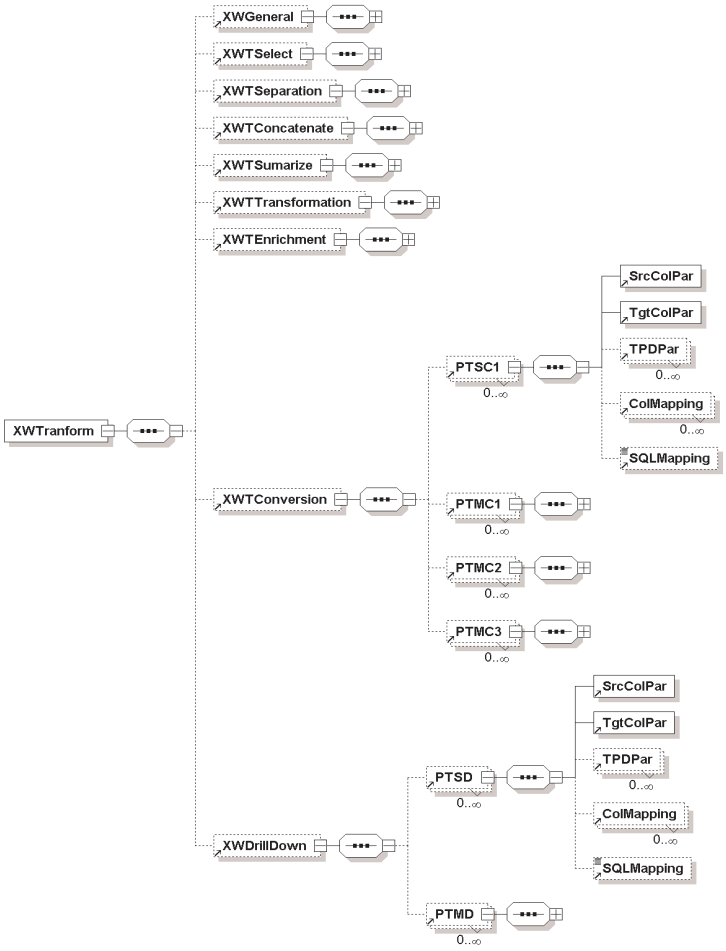

Figura 64 - Esquema da Seção E – Transformação – Conversão e *DrillDown*

```
<XWDrillDown>
        <PTSD BaseName="" FPSDID="idvalue0" SCHID="idvalue0" SourceSchema=""
                SourceTable="" TargetSchema="" TargetTable="">
                <SrcColPar name="" />
                <TgtColPar name="" />
                <TPDPar FakeNameAfter="" FakeNameBefore="" parDesc="" parName=""
                        typeID="idvalue0" />
                <ColMapping AggFunc="" SourceCol="" TargetCol="" />
                <SQLMapping>SQLMapping</SQLMapping>
        </PTSD>
        <PTMD BaseName="" FPMDID="idvalue0" SCHID="idvalue0" SourceSchema=""
                SourceTable="" TargetSchema="" TargetTable="">
                <SrcColPar name="" />
                <TgtColPar name="" />
                <TPDPar FakeNameAfter="" FakeNameBefore="" parDesc="" parName=""
                        typeID="idvalue0" />
                <ColMapping AggFunc="" SourceCol="" TargetCol="" />
                <SQLMapping>SQLMapping</SQLMapping>
        </PTMD>
</XWDrillDown>
```

Exemplo 41

[E-3-0] - ***X***W[**PA**$_{\text{Análise}}$] – As funções de análise(**<XWAnalysis>**) foram definidas no descritor de forma diferente(Figura 65). Foi mais conveniente especificar

diretamente as hierarquias. O elemento **<XWHLeaf>** define a hierarquia entre dois elementos de TAD. Os atributos principais da hierarquia definem uma coluna(**ChildColumn**), tabela(**ChildTable**) e esquema(**ChildSchema**) filhos e os correspondentes da coluna pai(**FatherColumn**, **FatherTable** e **FatherSchema**). A hierarquia pode conter qualquer número de colunas agregadas da tabela filha(**<XWHCol>**) assim como funções para agregação(**AggFunc**) ou *DrillDown*(**DrillFunc**). O processo de agregação pode conter vários elemento(**<TPDPar>**) de parâmetro para as funções de relacionamento(**FRS2ID**) que serão base da hierarquia.

Uma hierarquia pode ser encapsulada dentro de uma outra hierarquia de nível superior usando um elemento **<XWHBranch>**. Este elemento contém os mesmos dados da hierarquia básica com exceção dos valores da coluna filha que é substituída por um elemento básico de hierarquia(**<XWHLeaf>**). O Exemplo 42 mostra duas hierarquias, uma simples e outra de três níveis, onde um elemento **<XWHBranch>** contém um **<XWHLeaf>**.

O último elemento da seção de análise são procedimentos gerais de análise(**<XWAdHoc>**). Esta seção tem por finalidade especificar qualquer procedimento que não possa ter sido classificado nas categorias anteriores(Exemplo 43).

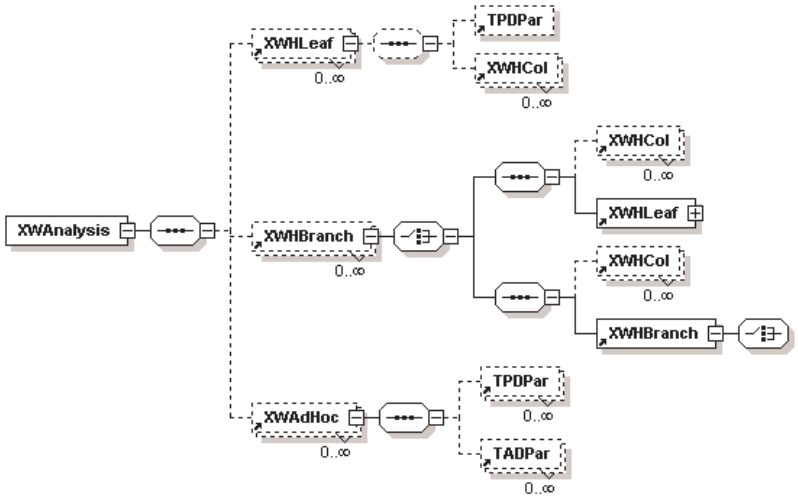

Figura 65 - Esquema da Seção E – Análise

```
<XWAnalysis>
        <XWHLeaf ChildColID="" ChildColumn="" ChildSchema="" ChildTable=""
                        FPSDID="idvalue0" FRS2ID="idvalue0" FatherColID=""
                        FatherColumn="" FatherSchema="" FatherTable="" HID="idvalue47"
                        ResultColID="" ResultColumn="" ResultSchema="" ResultTable="">
                <TPDPar FakeNameAfter="" FakeNameBefore="" parDesc="" parName=""
                                typeID="idvalue0" />
                <XWHCol AggFunc="" ChildColumn="" ChildSchema="" ChildTable=""
                                DrillFunc="" HID="idvalue0" ResultColIDMapping=""
                                ResultColMapping="" />
        </XWHLeaf>
        <XWHBranch    FPSDID="idvalue0" FRS2ID="idvalue0" FatherColID=""
                        FatherColumn="" FatherSchema="" FatherTable="" HID="idvalue48"
                        ResultColID="" ResultSchema="" ResultTable="">
                <XWHCol AggFunc="" ChildColumn="" ChildSchema="" ChildTable=""
                                DrillFunc="" HID="idvalue0" ResultColIDMapping=""
                                ResultColMapping="" />
                <XWHLeaf ChildColID="" ChildColumn="" ChildSchema="" ChildTable=""
                                FPSDID="idvalue0" FRS2ID="idvalue0" FatherColID=""
                                FatherColumn="" FatherSchema="" FatherTable=""
                                HID="idvalue49" ResultColID="" ResultColumn=""
                                ResultSchema="" ResultTable="">
                        <TPDPar FakeNameAfter="" FakeNameBefore="" parDesc=""
                                        parName="" typeID="idvalue0" />
                        <XWHCol AggFunc="" ChildColumn="" ChildSchema="" ChildTable=""
                                        DrillFunc="" HID="idvalue0" ResultColIDMapping=""
                                        ResultColMapping="" />
                </XWHLeaf>
        </XWHBranch>
..
</XWAnalysis>
```

Exemplo 42

```
        <XWAdHoc BaseName="" SCHID="idvalue0">
                <TPDPar FakeNameAfter="" FakeNameBefore="" parDesc="" parName=""
                                typeID="idvalue0" />
                <TADPar FakeNameAfter="" FakeNameBefore="" parDesc="" parName=""
                                typeID="idvalue0" />
        </XWAdHoc>
</XWAnalysis>
```

Exemplo 43

*Apêndice C – Listagem completa da instância 5-Alpha-**S**.*

C.1 – Tipos de Dados Espaciais

[A-0-0] - ***S***W_{tipos} – Tipos Abstratos de Dados(TADs) representando os dados espaciais.

[A-1-0] - *db2gse.ST_Geometry* – Este é o tipo genérico e abstrato de dado espacial. Ele será a interface para todos tipos especializados e servir de parâmetro para todas as funções de manipulação.

Os tipos a seguir representam a especialização de dado espacial para geometrias simples, ou seja que não são compostas de outras geometrias.

[A-1-1] - *db2gse.ST_Point* – Tipo básico de ponto bi-dimensional no espaço

[A-1-2] - *db2gse.ST_Curve* – Tipo abstrato representando todos tipos de curvas.

[A-1-2-1] - *db2gse.ST_LineString* – Tipo básico de linha poligonal. Os segmentos de reta devem ser entendidos como um caso particular de linha poligonal.

[A-1-3] - *db2gse.ST_Surface* – Representa uma superfície e não somente uma linha poligonal fechada. Usado para armazenar estruturas de dados "*raster*"(seção 3.1.2)

[A-1-3-1] - *db2gse.ST_Polygon* – Tipo básico de polígono e é definido por uma linha poligonal fechada e ainda uma lista de polígonos interiores excluídos.

Os tipos abaixo representam a especialização de um dado espacial para coleções de elementos espaciais.

[A-1-4] - *db2gse.ST_GeomCollection* **–** Utilizada para representar geometrias que são compostas de conjuntos de geometrias.

[A-1-4-1] - *db2gse.ST_MultiPoint* – Representa conjuntos de pontos.

[A-1-4-2] - *db2gse.ST_MultiSurface* – Representa conjuntos de superfícies, contíguas ou não.

[A-1-4-2-1] - *db2gse.ST_MultiPolygon* – Representa conjuntos de polígonos.

[A-1-4-3] - *db2gse.ST_MultiCurve* – Tipo abstrato representando todos os tipos de conjuntos de curvas.

[A-1-4-3-1] - *db2gse.ST_MultiLineString* – Representa conjuntos de linhas poligonais.

[A-2-0] - *db2gse.coordref* – Representa um sistema de referencia de coordenadas.

C.2 – Operadores transparêntes

[B-0-0] - ***S***W$_{transparentes}$ – OPERADORES TRANSPARÊNTES

[B-1-0] - Conectivos Lógicos – Todas estas funções terão a simples finalidade de entender o retorno de funções cujo retorno booleano se faz através de valores inteiros 1 ou 0:

[B-1-1] - **Conector AND – ST_AndInt –** Código simplificado: (((**X**)=1)*AND*((**Y**)=1))

[B-1-2] - **Conector OR – ST_OrInt –** Código simplificado: (((**X**)=1)*OR*((**Y**)=1))

[B-1-3] - **Conector AND – ST_NotInt –** Código simplificado: (((**X**)<>1))

C.3 – Gerência e Suporte de Metadados

[C-0-0] - ***S***W$_{MD}$.

[C-1-0] - ***S***W$_{CATÁLOGOS}$ – A lista abaixo contém a classificação dos catálogos dismponíveis no **DB2 SE** alem de alguns provenientes do modelo de catálogo fornecido pelo *framework*.

[C-1-1] - **Tipos Abstratos de Dados** – Este item constituirá o conjunto de tabelas para armazenamento de todos os Tipos de Dados Espaciais existentes no suporte do **DB2 SE.**

[C-1-1-1] - ***Colunas Espaciais***(*DB2GSE.GEOMETRY_COLUMNS*) – Contém uma lista de com no nome das tabelas e colunas com dados espaciais e seus tipos.

[C-1-1-2] - ***Sistemas de Coordenadas***(*DB2GSE.COORD_REF_SYS*) –– Indica como cada coordenada do sistema é representada e armazenada.

[C-1-1-3] - ***Sistemas de Referência***(*DB2GSE.SPATIAL_REF_SYS*) – Indicam como os sistemas de coordenadas irão se relacionar com a posição geográfica dos elementos espaciais

[C-1-2] - ***Modelos*** – Será utilizado o suporte do *framework*.

[C-1-3] - **Hierarquias** – Será usado o suporte do *framework*.

[C-1-4] - **Módulos de Processamento** – O suporte fornecido pelo *framework* será necessário para completar os elementos necessários deste catálogo.

[C-1-4-1] - ***Módulos de Geocodificação***(*DB2GSE.SPATIAL_GEOCODER*) – Existente no **DB2 SE**.

[C-1-4-2] - ***Módulos de Importação, Limpeza e Validação*** – Utilizarão as tabelas do modelo de catálogo fornecido pelo *framework*..

[C-2-0] - *S*W$_{\text{Manipulação de Catálogos}}$ - Procedimentos de Manipulação de Catálogo.

[C-2-1] - **PCSD**$^{n}_{\text{Registro}}$(Parâmetros) – O sistema de suporte de registro de dados. Procedimentos adicionais fornecidos pelo suporte do *framework* devem ser utilizados para este registro.

[C-2-1-1] - ***Criação e registro de modelos*** – Utilizará o suporte do *framework*.

[C-2-1-2] - ***Criação e registro de sistema de coordenadas*** – *ST_create_coordsys*(V8) – Suporte do **DB2 SE**. Cria e registra um novo sistema de coordenadas mas não associa a um modelo.

[C-2-1-3] - ***Criação e registro de Sistema de Referência –*** *db2gse.gse_enable_sref*(V7) **–** Cria e registra um novo sistema de referencia.

[C-2-1-4] - ***Criação e registro de Sistema de Referência –*** *db2gse.gse_create_srs*(V8) **–** Cria e registra um novo sistema de referencia.

[C-2-1-5] - ***Criação e registro de Coluna Espacial –*** *db2gse.gse_register_layer*(V7) **–** Cria ou registra uma coluna de uma tabela para um tipo de dados espacial qualquer. Se a coluna não existir ela é criada, caso contrário passa a ser associada àquela coluna um tipo de geometria e sistema de referência. Não associa a um modelo.

[C-2-1-6] - ***Criação e registro de Coluna Espacial –*** *ST_register_spatial_column*(V8) **–** Cria ou registra uma coluna de uma tabela para um tipo de dados espacial qualquer. Se a coluna não existir ela é criada, caso contrário passa a ser associada àquela coluna um tipo de geometria e sistema de referência. Não associa a um modelo.

[C-2-2] - **PCSD**$^{n}_{\text{Remoção}}$(Parâmetros) – O sistema de suporte de remoção de registros dos catálogo de metadados.

[C-2-2-1] - ***Remoção de modelos*** – Utilizará o suporte do *framework*.

[C-2-2-2] - ***Remoção de sistema de coordenadas*** – *ST_drop_coordsys*(V8) – Suporte do **DB2 SE**. Remove um sistema de coordenadas.

[C-2-2-3] - ***Remover Sistema de Referência –*** *db2gse.gse_disable_sref*(V7) **–** Remove, ou desabilita o sistema de referencia.

[C-2-2-4] - ***Remover Sistema de Referência –*** *db2gse.gse_drop_srs*(V8) **–** Remove, ou desabilita o sistema de referencia.

[C-2-2-5] - ***Remover Col. Espacial –*** *db2gse.gse_unregist_layer*(V7) **–** Suporte do **DB2 SE.** Remove uma coluna espacial do catálogo.

[C-2-2-6] - ***Remover Col .Espacial*** - *ST_unregister_spatial_column*(V8) **–** Suporte do **DB2 SE**. Remove uma coluna espacial do catálogo.

[C-2-3] - **PCSM**$^{n}_{\text{Registro}}$(Parâmetros) – Efetua o registro de módulos no sistema, assim como nos catálogos.

[C-2-3-1] - **Registrar Módulo de Geocodificação** - *gse_register_gc*(V7) **–** Suporte do **DB2 SE**.

[C-2-3-2] - **Ativar Auto-Geocodificação** - *db2gse.gse_enable_autogc*(V7) **–** Suporte do **DB2 SE**.

[C-2-3-3] - **Registrar Módulo de Geocodificação** - *ST_register_geocoder*(V8) **–** Suporte do **DB2 SE**.

[C-2-3-4] - **Ativar Auto-Geocodificação** - *ST_enable_autogeocoding*(V8) **–** Suporte do **DB2 SE**.

[C-2-3-5] - **Criar/Registrar Módulos –** Função geral para registro de módulos. Suporte do *framework*.

[C-2-4] - **PCSH**$^{n}_{\text{Registro}}$ – **Criar/Registrar Hierarquias –** Função geral para registro de hierarquias. Não existe nenhum procedimento a ser registrado do produto de suporte, por isso será usado o suporte do *framework*.

[C-2-5] - **PCSM**$^{n}_{\text{Remoção}}$(Parâmetros) – Efetua a remoção de um módulo no sistema, assim como nos catálogos.

[C-2-5-1] - **Remover Módulo de Geocodificação -** *db2gse.gse_unregist_gc*(V7) **–** Suporte do **DB2 SE**.

[C-2-5-2] - **Desativar Auto-Geocodificação -** *gse_disable_autogc*(V7) **–** Suporte do **DB2 SE**. Desativar o módulo de geocodificação automática.

[C-2-5-3] - **Remover Módulo de Geocodificação -** *ST_unregister_geocoder*(V8) **–** Suporte do **DB2 SE**.

[C-2-5-4] - **Desativar Auto-Geocodificação -** *ST_disable_autogeocoding*(V8) **–** Suporte do **DB2 SE**. Desativar o módulo de geocodificação automática.

[C-2-5-5] - **Remover Módulos –** Função geral para remoção de módulos do catálogo, bem como do sistema. Suporte do *framework*.

[C-2-6] - **PCSH**$^{n}_{\text{Remoção}}$ – **Remover Hierarquias –** Função geral para remoção de hierarquias do catálogo, bem como do sistema. Não existe nenhum procedimento a ser registrado do produto de suporte, por isso será usado o suporte do *framework*.

[C-3-0] - **S**W$_{\text{Manipulação de Elementos}}$

[C-3-1] - **FCSC**$^{n}_{\text{Criação}}$(Parâmetros).

[C-3-1-1] - *db2gse.ST_Point db2gse.ST_Point (Double,Double,SRID)*

[C-3-1-2] - *db2gse.ST_Geometry db2gse.ST_GeometryFromText(Varchar(4000), db2gse.coordref)*

[C-3-1-3] - *db2gse.ST_Point db2gse.ST_PointFromText(Varchar(4000), db2gse.coordref)*

[C-3-1-4] - *db2gse.ST_LineString db2gse.ST_LineFromText(Varchar(4000), db2gse.coordref)*

[C-3-1-5] - *db2gse.ST_Polygon db2gse.ST_PolyFromText(Varchar(4000), db2gse.coordref)*

[C-3-1-6] - *db2gse.ST_MultiPoint db2gse.ST_MPointFromText(Varchar(4000), db2gse.coordref)*

[C-3-1-7] - *db2gse.ST_MultiLineString db2gse.ST_MultiLineFromText(String, db2gse.coordref)*

[C-3-1-8] - *db2gse.ST_MultiPolygon db2gse.ST_MPolyFromText(Varchar(4000), db2gse.coordref)*

[C-3-1-9] - *db2gse.ST_Geometry db2gse.ST_GeomFromWKB(Blob(1M), db2gse.coordref)*

[C-3-1-10] - *db2gse.ST_Point db2gse.ST_PointFromWKB(Blob(1M),SRID)*

[C-3-1-11] - *db2gse.ST_LineString db2gse.ST_LineFromWKB(Blob(1M), db2gse.coordref)*

[C-3-1-12] - *db2gse.ST_Polygon db2gse.ST_PolyFromWKB(Blob(1M),SRID)*

[C-3-1-13] - *db2gse.ST_MultiPoint db2gse.ST_MPointFromWKB(Blob(1M), db2gse.coordref)*

[C-3-1-14] - *db2gse.ST_MultiLineString db2gse.ST_MLineFromWKB(Blob(1M) db2gse.coordref)*

[C-3-1-15] - *db2gse.ST_MultiPolygon db2gse.ST_MPolyFromWKB(Blob(1M) db2gse.coordref)*

[C-3-1-16] - *db2gse.ST_Geometry db2gse.ST_WKBToSQL(Blob(1M))*

[C-3-1-17] - *db2gse.ST_Geometry df db2gse.ST_WKTToSQL(Varchar(4000))*

[C-3-1-18] - *db2gse.ST_Geometry db2gse.ST_GeometryFromShape (Blob(1M), db2gse.coordref)*(V8.1)

[C-3-1-19] - *db2gse.ST_Geometry db2gse.GeometryFromShape (Blob(1M), db2gse.coordref)*(V7)

[C-3-1-20] - *db2gse.ST_Point db2gse.ST_PointFromShape (Blob(1M), db2gse.coordref)*(V8.1)

[C-3-1-21] - *db2gse.ST_Point db2gse.PointFromShape (Blob(1M), db2gse.coordref)*(V7)

[C-3-1-22] - *db2gse.ST_LineString db2gse.ST_LineFromShape (Blob(1M), db2gse.coordref)*(V8.1)

[C-3-1-23] - *db2gse.ST_LineString db2gse.LineFromShape (Blob(1M), db2gse.coordref)*(V7)

[C-3-1-24] - *db2gse.ST_Polygon db2gse.ST_PolyFromShape (Blob(1M), db2gse.coordref)*(V8.1)

[C-3-1-25] - *db2gse.ST_Polygon db2gse.PolyFromShape (Blob(1M), db2gse.coordref)*(V7)

[C-3-1-26] - *db2gse.ST_MultiPoint db2gse.ST_MPointFromShape (Blob(1M), db2gse.coordref)*(V8.1)

[C-3-1-27] - *db2gse.ST_MultiPoint db2gse.MPointFromShape (Blob(1M), db2gse.coordref)*(V7)

[C-3-1-28] - *db2gse.ST_MultiLineString db2gse.ST_MLineFromShape (Blob(1M), db2gse.coordref)*(V8.1)

[C-3-1-29] - *db2gse.ST_MultiLineString db2gse.MLineFromShape(Blob(1M), db2gse.coordref)*(V7)

[C-3-1-30] - *db2gse.ST_MultiPolygon db2gse.ST_MPolyFromShape(Blob(1M), db2gse.coordref)*(V8.1)

[C-3-1-31] - *db2gse.ST_MultiPolygon db2gse.MPolyFromShape(Blob(1M), db2gse.coordref)*(V7)

[C-3-1-32] - *db2gse.ST_Geometry db2gse.ST_ShapeToSQL (Blob(1M)* (V8.1)

[C-3-1-33] - *db2gse.ST_Geometry db2gse. ShapeToSQL(Blob(1M))*(V7)

[C-3-2] - **FCMC**$^{n}_{Criação}$(Parâmetros) – O **DB2 SE** não possui nenhuma função nesta categoria e não julgamos necessário a sua implementação nesse trabalho.

[C-3-3] - **FCSA**$^{n}_{Acesso}$(***S***W$_{tipo}$)

[C-3-3-1] - *Integer db2gse.ST_SRID(db2gse.ST_Geometry)*

[C-3-3-2] - *Integer db2gse.ST_NumPoints(db2gse.ST_LineString)*

[C-3-3-3] - *db2gse.ST_LineString db2gse.ST_ExteriorRing(db2gse.ST_Polygon)*

[C-3-3-4] - *db2gse.ST_LineString db2gse.ST_InteriorRing(db2gse.ST_Polygon,Integer)*

[C-3-3-5] - *Integer db2gse.NumInteriorRing(db2gse.ST_Polygon)*

[C-3-3-6] - *db2gse.ST_Geometry db2gse.ST_GeometryN(db2gse.ST_GeomCollection, Integer)*

[C-3-3-7] - *Integer db2gse.ST_NumGeometries(db2gse.ST_GeomCollection)*

[C-3-3-8] - *db2gse.ST_Point db2gse.ST_PointN(db2gse.ST_Curve, Integer)*

[C-3-3-9] - *db2gse.ST_Point db2gse.ST_Endpoint(db2gse.ST_Curve)*

[C-3-3-10] - *db2gse.ST_Point db2gse.ST_StartPoint(db2gse.ST_Curve)*

[C-3-3-11] - *db2gse.ST_Point db2gse.ST_Endpoint(db2gse.ST_Curve)*

[C-3-3-12] - *Integer db2gse.ST_CoordDim(db2gse.ST_Geometry)*

[C-3-3-13] - *Integer db2gse.ST_Dimension(db2gse.ST_Geometry)*

[C-3-3-14] - *Varchar(4000) db2gse.ST_GeometryType (db2gse.ST_Geometry)*

[C-3-3-15] - *Double db2gse.ST_X(ST_Point)*

[C-3-3-16] - *Double db2gse.ST_Y(ST_Point)*

[C-3-3-17] - *Double db2gse.Z(ST_Point)(V7)*

[C-3-3-18] - *Double db2gse.ST_Z(ST_Point)(V8.1)*

[C-3-3-19] - *Double db2gse.M(ST_Point)(V7)*

[C-3-3-20] - *Double db2gse.ST_M(ST_Point)(V8.1)*

[C-3-4] - **FCSP**$^{n}_{\text{Propriedade}}$(***S***W$_{\text{tipo}}$)

[C-3-4-1] - *Double db2gse.ST_Area(db2gse.ST_Surface)*

[C-3-4-2] - *Double db2gse.ST_Length(db2gse.ST_Curve)*

[C-3-4-3] - *Double db2gse.ST_Length(db2gse.ST_MultiCurve)*

[C-3-4-4] - *Double db2gse.ST_Perimeter(db2gse.ST_Surface)*

[C-3-4-5] - *Double db2gse.ST_Perimeter(db2gse.ST_MultiSurface)*

[C-3-5] - **FCMA**$^{n}_{\text{Acesso}}$(***S***W$_{\text{coluna}}$) – O **DB2 SE** não possui nenhuma função nesta categoria e não julgamos necessário a sua implementação nesse trabalho.

[C-3-6] - **FCMP**$^{n}_{Propriedade}$(***S***W$_{coluna}$) – O **DB2 SE** não possui nenhuma função nesta categoria e não julgamos necessário a sua implementação nesse trabalho.

C.4 – Funções de Suporte Relacional

[D-0-0] - ***S***W[**F**$_{Suporte\ Relacional}$] = {**F**$_{Seleção}$, **F**$_{Projeção}$, **F**$_{Agregação}$ }

[D-1-0] - ***S***W[**F**$_{Seleção}$] – FUNÇÕES DE SUPORTE A SELEÇÃO

[D-1-1] - **FRS1**$^{n}_{Relação}$(***S***W$_{tipo}$)

[D-1-1-1] - *Integer* *db2gse.ST_IsClosed(db2gse.ST_Curve)*

[D-1-1-2] - *Integer* *db2gse.ST_IsClosed(db2gse.ST_MultiCurve)*

[D-1-1-3] - *Integer* *db2gse.ST_IsRing(db2gse.ST_Curve)*

[D-1-1-4] - *Integer* *db2gse.ST_IsSimple(db2gse.ST_Geometry)*

[D-1-1-5] - *Integer* *db2gse.ST_IsValid(db2gse.ST_Geometry)*

[D-1-1-6] - *Integer* *db2gse.Is3d(db2gse.ST_Geometry)*(V7)

[D-1-1-7] - *Integer* *db2gse.ST_Is3d(db2gse.ST_Geometry)*(V8.1)

[D-1-1-8] - *Integer* *db2gse.IsMeasured(db2gse.ST_Geometry)*(V7)

[D-1-1-9] - *Integer* *db2gse.ST_IsMeasured(db2gse.ST_Geometry)*(V8.1)

[D-1-2] - **FRS2**$^{n}_{Relação}$(***S***W$_{tipo}{}^{1}$, ***S***W$_{tipo}{}^{2}$ [,Parâmetro])

[D-1-2-1] - *Integer* *db2gse.ST_Contains(db2gse.ST_Geometry, db2gse.ST_Geometry)*

[D-1-2-2] - *Integer* *db2gse.ST_Within(db2gse.ST_Geometry, db2gse.ST_Geometry)*

[D-1-2-3] - *Integer* *db2gse.ST_Crosses(db2gse.ST_Geometry,db2gse.ST_Geometry)*

[D-1-2-4] - *Integer* *db2gse.ST_Touches(db2gse.ST_Geometry,db2gse.ST_Geometry)*

[D-1-2-5] - *Integer* *db2gse.ST_Intersects(db2gse.ST_Geometry, db2gse.ST_Geometry)*

[D-1-2-6] - *Integer* *db2gse.EnvelopesIntersect(db2gse.ST_Geometry, db2gse.ST_Geometry)*(V7)

[D-1-2-7] - *Integer db2gse.ST_EnvelopesIntersect(db2gse.ST_Geometry, db2gse.ST_Geometry)*(V8.1)

[D-1-2-8] - *Integer db2gse.ST_Overlaps(db2gse.ST_Geometry, db2gse.ST_Geometry)*

[D-1-2-9] - *Integer db2gse.ST_Disjoint(db2gse.ST_Geometry, db2gse.ST_Geometry)*

[D-1-2-10] - *Integer db2gse.ST_Equals(db2gse.ST_Geometry, db2gse.ST_Geometry)*

[D-1-2-11] - *Integer db2gse.ST_OrderingEquals(db2gse.ST_Geometry, db2gse.ST_Geometry)*

[D-1-3] - **FRS3**$^{n}_{\text{Relação}}$(***S***W$_{\text{tipo}}$, DW$_{\text{parâmetro}}$):

[D-1-3-1] - db2gse.dwgse_HasInteriorRings(ST_Curve, NrRings) – Retornar verdadeiro ou falso se a curva tem pelo menos NrRings.

[D-1-4] - **FRM1**$^{n}_{\text{Relação}}$ (***S***W$_{\text{coluna}}$) – Estas funções testam condições de elementos espaciais todos entre si. Por exemplo:

[D-1-4-1] - db2gse.dwgse_IsSmaller(ST_Geometry) – Retorna verdadeiro para o elemento que for o menor de todos os outros. Apesar de poder olhar para todos os elementos do conjunto de uma vez, esta função deve retornar um valor para cada elemento da *tupla* fornecido na sentença de consulta.

[D-1-5] - **FRM2**$^{n}_{\text{Relação}}$ (***S***W$_{\text{coluna}}{}^{1}$, (***S***W$_{\text{coluna}}{}^{2}$ | ***S***W$_{\text{tipo}}$)[,Parâmetro]))

[D-1-5-1] - dwgse_IsClosestDistance(ST_Geometry A^i, ST_Geometry B^j) – Irá retornar verdadeiro para os dois elementos A^i e B^j que estiverem na menor distância que todos os outros.

[D-1-5-2] - dwgse_IsClosest(ST_Geometry A^k, ST_Geometry B) – Irá comparar cada elemento, ou *tupla* do atributo B com todos os elementos k da coluna A e retornar verdadeiro se o elemento A^k é o mais perto de B. Note-se que para cada elemento individual da coluna B, todos os elementos k da coluna A são testados.

[D-1-6] - **FRM3**$^{n}_{\text{Relação}}$ ((***S***W$_{\text{coluna}}$ | ***S***W$_{\text{tipo}}$),(Parâmetros | Parâmetro$_{\text{Set}}$)):

[D-1-6-1] - dwgse_IsSmaller(ST_Geometry, minSize) – Este exemplo é uma extensão do item [D-1-4-1] acima onde aqui o menor item retorna verdadeiro desde que ele seja menor que minSize.

[D-2-0] - ***S***W[$\mathbf{F}_{\text{Projeção}}$] – FUNÇÕES DE SUPORTE A PROJEÇÃO

[D-2-1] - **FPST**$^{n}_{\text{Transformação}}$(***S***$W_{tipo}^{1}$,[$DW_{parâmetro}$])

[D-2-1-1] - *db2gse.ST_Geometry db2gse.ST_Boundary(db2gse.ST_Geometry)*

[D-2-1-2] - *db2gse.ST_Geometry db2gse.ST_Envelope(db2gse.ST_Geometry)*

[D-2-1-3] - *db2gse.ST_Geometry db2gse.ST_ConvexHull(db2gse.ST_Geometry)*

[D-2-1-4] - *db2gse.ST_Geometry db2gse.ST_Buffer(db2gse.ST_Geometry, Double)* –

[D-2-1-5] - *db2gse.ST_Geometry db2gse.ST_Transform(db2gse.ST_Geometry, db2gse.coordref)*

[D-2-2] - **FPSC**$^{n}_{\text{Conversão}}$(***S***W_{tipo}^{1},[***S***W_{tipo}^{2}| $DW_{parameter}$])

[D-2-2-1] - *db2gse.ST_Polygon db2gse.ST_Polygon (db2gse.ST_LineString, db2gse.coordref)*

[D-2-2-2] - *BLOB(1m) db2gse.ST_AsBinary(db2gse.ST_Geometry)* –

[D-2-2-3] - *BLOB(1m) db2gse.AsBinaryShape(db2gse.ST_Geometry)*(V7) -

[D-2-2-4] - *BLOB(1m) db2gse.ST_AsBinaryShape(db2gse.ST_Geometry)*(V8.1) -

[D-2-2-5] - *Varchar(4000) db2gse.ST_AsText(db2gse.ST_Geometry)* –

[D-2-2-6] - *Double db2gse.ST_Distance(db2gse.ST_Geometry, db2gse.ST_Geometry)*

[D-2-2-7] - *db2gse.ST_Point db2gse.ST_Centroid(db2gse.ST_Surface)*

[D-2-2-8] - *db2gse.ST_Point db2gse.ST_Centroid(db2gse.ST_MultiSurface)*

[D-2-2-9] - *db2gse.ST_Point db2gse.ST_PointOnSurface(db2gse.ST_Surface)*

[D-2-2-10] - *db2gse.ST_Point db2gse.ST_PointOnSurface(db2gse.ST_MultiSurface)*

[D-2-2-11] - *db2gse.ST_Geometry db2gse.LocateAlong (db2gse.ST_Geometry, Double)*(V7)

[D-2-2-12] - *db2gse.ST_Geometry db2gse.ST_ LocateAlong (db2gse.ST_Geometry, Double)*(V8.1)

[D-2-2-13] - *db2gse.ST_Geometry db2gse.LocateBetween (db2gse.ST_Geometry, Double,Double)*(V7)

[D-2-2-14] - *db2gse.ST_Geometry db2gse.ST_ LocateBetween (db2gse.ST_Geometry, Double,Double)*(V8.1)

[D-2-3] - **FPSX**$^{n}_{Combinacao}$(***S***W$_{tipo}^{1}$, ***S***W$_{tipo}^{2}$,[DW$_{parâmetro}$])

[D-2-3-1] - *db2gse.ST_Geometry db2gse.ST_Union(db2gse.ST_Geometry, db2gse.ST_Geometry)*

[D-2-3-2] - *db2gse.ST_Geometry db2gse.ST_Difference(db2gse.ST_Geometry, db2gse.ST_Geometry)*

[D-2-3-3] - *db2gse.ST_Geometry db2gse.ST_SymmetricDiff(db2gse.ST_Geometry, db2gse.ST_Geometry)*

[D-2-3-4] - *db2gse.ST_Geometry db2gse.ST_Intersection(db2gse.ST_Geometry, db2gse.ST_Geometry)*

[D-2-4] - **FPSD**$^{n}_{DrillDown}$(***S***W$_{tipo}$, [,DW$_{parâmetro}$]):

[D-2-4-1] - db2gse.dwgse_Grid(ST_Geometry A,M,N) – Retorna uma grade M×N de objetos espaciais retangulares dentro da área do objeto A.

[D-2-5] - **FPMT**$^{n}_{Transformação}$(***S***W$_{tipo}$, DW$_{parâmetro}$):

[D-2-5-1] - dwgse_ScaleUp(ST_Geometry,Rate) – Esta função recebe um objeto espacial e aumenta sua escala e número de pontos de acordo com a maior distância da área coberta por todos os outros objetos dentro do mesmo conjunto. O aumento de escala também é regulado pela taxa Rate.

[D-2-6] - **FPMC**$^{n}_{Conversão}$(***S***W$_{tipo}$[,DW$_{parâmetro}$]):

[D-2-6-1] - dwgse_FullConvexUnion(ST_Geometry) – Função que cria um grande objeto espacial convexo que englobe todos os outros objetos do mesmo conjunto.

[D-2-7] - **FPMD**$^{n}_{DrillDown}$(***S***W$_{tipo}$[,DW$_{parâmetro}$]) – Este é na verdade é um subgrupo do item [D-2-4].

[D-3-0] - ***S***W[**F**$_{Agregação}$] – FUNÇÕES DE SUPORTE À AGREGAÇÃO.

[D-3-1] - **FA1**$^{n}_{Agregação}$(***S***W$_{coluna}$^{1}, [***S***W$_{tipo}$ | ***S***W$_{coluna}$^{2} | DW$_{parâmetro}$])

[D-3-1-1] - dwgse_SumArea(ST_Geometry) – Neste exemplo a função irá retornar a soma das áreas de todas as geometrias da coluna fornecida.

[D-3-2] - **FA2**$^{n}_{Agregação}$(***S***W$_{coluna}$^{1}, [***S***W$_{tipo}$ | ***S***W$_{coluna}$^{2} | DW$_{parâmetro}$])

[D-3-2-1] - dwgse_MinDistances(ST_Geometry g1, ST_Geometry g2) - A título de exemplo, esta função retorna as menores distâncias para cada geometria em "g1" para a geometria "g2".

[D-3-3] - **FA11**$^{n}_{Agregação}$(***S***W$_{coluna}$,[DW$_{parâmetro}$])

[D-3-3-1] - dwgse_ AggObject(ST_Geometry gc1) - Esta função retorna um objeto espacial agregando todos os objetos espaciais da coluna de objetos espaciais "gc1".

[D-3-4] - **FA12**$^{n}_{Agregação}$(***S***W$_{coluna}$,[DW$_{parâmetro}$])

[D-3-4-1] - dwgse_ CalculaCentroides(ST_Geometry gc1, MinDist) – Esta função retorna o conjunto centróides dos pontos que estejam dentro da distancia mínima "MinDist" fornecida.

[D-3-5] - **FA21**$^{n}_{Agregação}$(***S***W$_{coluna}$^{1},[***S***W$_{tipo}$ | ***S***W$_{coluna}$^{2}],[Parâmetros])

[D-3-5-1] - dwgse_ Nearest(ST_Geometry gc1, ST_Geometry g2) – Esta função irá retornar o objeto espacial do conjunto "gc2" mais próximo de cada objeto espacial "g2" fornecido.

[D-3-6] - **FA22**$^{n}_{Agregação}$(***S***W$_{coluna}$^{1},[***S***W$_{tipo}$ | ***S***W$_{coluna}$^{2}],[Parâmetros])

[D-3-6-1] - dwgse_NearestSet(ST_Geometry gc1, ST_Geometry g2, N) – Esta função irá retornar o conjunto do "N" objetos espaciais dentro do conjunto gc1 mais próximos de "g2"/

C.5 – Funções de processamento de *Data Warehouse* Espacial

[E-0-0] - ***S***W[$\mathbf{P}_{processamento}$] = {$\mathbf{PE}_{Extração}$, $\mathbf{PT}_{Transformação}$, $\mathbf{PA}_{Análise}$}

.

[E-1-0] - ***S***W[$\mathbf{PE}_{Extração}$]

[E-1-1] - IMPORTAÇÃO

[E-1-1-1] - $\mathbf{PEI}^{n}_{Importação}$([***ArqExt*** | $\boldsymbol{TabList}^{B}_{fonte}$],$\boldsymbol{TabList}^{A}_{destino}$[,Parâmetros).

[E-1-1-1-1] - **Importar arquivos** *shape* **–** *gse_import_shape*(V7) **–** Esta função, implementada pelo **DB2 Spatial Extender**, serve para carregar arquivos no formato padrão de Sistemas de Informação Geográfica(SIG). Este formato é utilizado pelos SIGs da **ESRI**(**ArcInfo**, **ArcGIS**, **ArcView**, etc.) que são os mais utilizados.

[E-1-1-1-2] - **Importar arquivos SDE** – *gse_import_sde* **–** Este outro método de importação é específico para alguns produtos da **ESRI**(**ArcSDE**)

[E-1-1-1-3] - **Importação geral** – db2gse.dwgse_SpatialImport – Esta funcionalidade, implementada neste projeto, tem a finalidade de realizar uma importação mais geral onde o tipo de formato é automaticamente determinado a função de importação correspondente é invocada.

[E-1-1-2] - $\mathbf{PEA}^{n}_{Agregação}$([***ArqExt*** | $\boldsymbol{TabList}^{B}_{fonte}$], $\boldsymbol{TabList}^{A}_{destino}$[,Parâmetros]) – Não implementados.

[E-1-1-3] - $\mathbf{PES}^{n}_{Sequencializador}$([***ArqExt*** | $\boldsymbol{TabList}^{B}_{fonte}$],$\boldsymbol{TabList}^{A}_{destino}$[,Parâmetros]) – Não implementados.

[E-1-2] - PREPROCESSAMENTO

[E-1-2-1] - $\mathbf{PEP}^{n}_{Preprocessamento}$($\boldsymbol{TabList}^{A}_{fonte}$, $\boldsymbol{TabList}^{A}_{destino}$[,Parâmetros])

[E-1-2-1-1] - **Geocodificar** – db2gse.dwgse_DataGeocode – Iniciar o módulo de geocodificação e extrair as informações espaciais implícitas nos dados convencionais convertendo-as em tipos de dados espaciais(seção 3.1.7).

[E-1-2-2] - $\mathbf{PEL}^{n}_{Limpeza}$($\boldsymbol{TabList}^{A}_{fonte}$, $\boldsymbol{TabList}^{A}_{destino}$[,$DW_{parâmetro}$]) – **Limpeza Espacial ou *Spatial Cleansing*** – Não implementados.

[E-1-2-3] - **PEV**$^{n}_{\text{Validação}}$(***TabList***$^{A}_{\text{fonte}}$, ***TabList***$^{A}_{\text{destino}}$[,$DW_{\text{parâmetro}}$]) – Não implementados.

[E-2-0] - ***S***W[**PT**$_{\text{Transformação}}$]

[E-2-1] - **PTG1**$^{n}_{\text{Transformação}}$(***TabList***$_{\text{fonte}}$, ***TabList***$_{\text{destino}}$[,$DW_{\text{Lista de Parâmetros}}$]) – Forma geral da funcionalidade de transformação. Vai representar os procedimentos de transformação gerais que não pertençam as categorias apresentadas nesta seção.

[E-2-2] - **PTG2**$^{n}_{\text{Transformação}}$(***TabList***$_{\text{fonte}}$, ***TabList***$_{\text{destino}}$, SQL_{list}[,$DW_{\text{Lista de Parâmetros}}$]) – Este procedimento implementa a execução de uma sentença SQL qualquer contendo funções de manipulação espacial sobre a tabela fonte e depondo o resultado nas tabelas de saída. O procedimento abaixo, implementado para este fim, recebe as tabelas de entrada, saída, mapeamento de parâmetros e o SQL que deve ser executado:

[E-2-2-1] - db2gse.dwgse_SpatialSQL

[E-2-3] - SELEÇÃO

[E-2-3-1] - **PTSR**$^{n}_{\text{Seleção}}$(***TabList***$_{\text{fonte}}$,***TabList***$_{\text{destino}}$,**FRS_**$^{k}_{\text{Relação}}$[,$DW_{\text{Parâmetros}}$]):

[E-2-3-1-1] - dwgse_InsideJoin(TabFonte, TabDestino, ST_Geometry A, ST_Geometry B) – Realiza a junção de duas tabelas com elementos de dados espaciais que estejam associados pela relação de contenção, ou seja, retornam as *tuplas* cujo elemento A estivesse contido dentro do elemento B.

[E-2-3-2] - **PTMR**$^{n}_{\text{Seleção}}$(***TabList***$_{\text{fonte}}$, ***TabList***$_{\text{destino}}$, **FRM_**$^{k}_{\text{Relação}}$ [,$DW_{\text{Parâmetros}}$]):

[E-2-3-2-1] - dwgse_NearJoin(TabFonte, TabDestino, ST_Geometry A, ST_Geometry B) – Produz a junção de duas tabelas objetos espaciais onde os objetos no conjunto A estejam mais próximos de cada elemento no conjunto B. Se tivermos uma tabela com clientes e outra com filiais de um banco a tabela resultante possuiria os cliente que moram mais perto de cada filial. Este procedimento necessita fazer cálculos de relacionamento entre cada elemento de A contra todos os elementos de B. Isto pode ser feito usando as funções de categoria **FRM2**$^{n}_{\text{Relação}}$(***S***W_{coluna}, ***S***W_{tipo}) se o sistema possuir este suporte, caso contrário, como é o caso do sistema de suporte escolhido(**DB2 Spatial Extender**), a operação de junção terá que ser feita diretamente na implementação da função de transformação.

[E-2-4] - SEPARAÇÃO

[E-2-4-2] - $\mathbf{PTSS2}^{n}_{\text{Separação}}(\boldsymbol{TabList}_{\text{fonte}}, \boldsymbol{TabList}_{\text{destino}}, \mathbf{FCSA}^{k}_{\text{Acesso}}\ [,DW_{\text{Parâmetros}}])$

[E-2-4-2-1] - dwgse_Medidas(TabFonte,TabDestino,ST_Geometry) – Este procedimento de exemplo irá produzir uma tabela com as áreas e perímetros de todos os objetos espaciais.

[E-2-5] - CONCATENAÇÃO

[E-2-5-1] - $\mathbf{PTSU}^{n}_{\text{Concatenação}}(\boldsymbol{TabList}_{\text{fonte}}, \boldsymbol{TabList}_{\text{destino}}, \mathbf{FPSX}^{k}_{\text{Combinacao}}[,DW_{\text{Parâmetros}}])$

[E-2-5-1-1] - dwgse_Integrar(TabFonte ta, TabFonte tb, TabDestino, ST_Geometry a, ST_Geometry b) – Este procedimento de exemplo irá produzir uma tabela com a união das geometrias **a** e **b** das respectivas tabelas **ta** e **tb**.

[E-2-6] - SUMARIZAÇÃO

[E-2-6-1] - $\mathbf{PTSA1}^{n}_{\text{Sumarização}}(\boldsymbol{TabList}_{\text{fonte}},\boldsymbol{TabList}_{\text{destino}},\mathbf{FA11}^{k}_{\text{Agregação}}[,DW_{\text{Parâms.}}])$

[E-2-6-1-1] - dwgse_AggObject(TabFonte, TabDestino, ST_Geometry A, ST_Geometry B) – Produzem uma tabela de valores agregados dos elementos de A que estiverem contidos dentro do elemento B. De fato, neste caso a agregação não se dá nos objetos espaciais em si mas nos atributos(colunas) associadas que serão agregadas. Este é um processo de seleção seguido de uma agregação.

[E-2-6-2] - $\mathbf{PTMA1}^{n}_{\text{Sumarização}}(\boldsymbol{TabList}_{\text{fonte}},\boldsymbol{TabList}_{\text{destino}},\mathbf{FA12}^{k}_{\text{Agregação}}[,DW_{\text{Parâms.}}])$

[E-2-6-2-1] - dwgse_ZoomOut(TabFonte, TabDestino, ST_Geometry, Scale) – Traz todos os dados espaciais para uma escala mais menor onde alguns grupos de objetos espaciais serão agregados se transformando num só, alem de alguns poderem ser convertidos de tipo. Suponhamos por exemplo, que um contorno pode ser transformado num ponto ou um conjunto de pontos também pode ser transformado num ponto. Esta função está classificada nesta categoria por que pode ser necessário utilizar todos os objetos espaciais do conjunto para determinar um objeto agregado que seja produzido.

[E-2-7] - TRANSFORMAÇÃO

[E-2-7-1] - $\mathbf{PTST1}^{n}_{\text{Transformaçao}}(\boldsymbol{TabList}_{\text{fonte}},\boldsymbol{TabList}_{\text{destino}},\mathbf{FPST}^{k}_{\text{Transformação}}[,DW_{\text{Parâms.}}])$ – Veja o seguinte exemplo:

[E-2-7-1-1] - dwgse_Converte(TabFonte, TabDestino, newRef) – Faz conversão de sistema de referência e traz todos os dados espaciais para a mesma escala. O sistema de coordenadas já associa os dados espaciais a uma certa escala, mas isto pode ser útil para ajustar os valores para comparação e junção de objetos em diferentes escalas

[E-2-8] - ENRIQUECIMENTO

[E-2-8-1] - $\mathbf{PTSE1}^{n}_{\text{Enriquecimento}}(\boldsymbol{TabList}_{\text{fonte}}, \boldsymbol{TabList}_{\text{destino}}, \mathbf{FPSC}^{k}_{\text{Conversão}}\ [,DW_{\text{Parâms}}])$:

[E-2-8-1-1] - dwgse_ZoomIn(TabFonte, TabDestino, ST_Geometry, Scale) – Diminui a escala do conjunto de objetos espaciais para o atributo de Scale de forma que alguns objetos, como pontos, por exemplo, podem se converter em polígonos(contornos).

[E-2-8-2] - $\mathbf{PTSE2}^{n}_{\text{Enriquecimento}}(\boldsymbol{TabList}_{\text{fonte}},\boldsymbol{TabList}_{\text{destino}},\mathbf{FCSC}^{n}_{\text{Criação}}\ [,DW_{\text{Parâms}}])$:

[E-2-8-2-1] - dwgse_GeocodeLatLong(TabFonte,TabDestino,Lat,Long) – Este seria o exemplo mais clássico, apesar de raramente implementado desta forma, de um enriquecimento por construção. A função faz a leitura de valores de Latitude e Longitude e converte em objetos espaciais de acordo.

[E-2-8-4] - $\mathbf{PTME2}^{n}_{\text{Enriquecimento}}(\boldsymbol{TabList}_{\text{fonte}},\boldsymbol{TabList}_{\text{destino}},\mathbf{FCMC}^{n}_{\text{Criação}}[,DW_{\text{Parâms}}])$ - Não implementados.

[E-2-8-5] - $\mathbf{PTME3}^{n}_{\text{Enriquecimento}}(\boldsymbol{TabList}_{\text{fonte}}, \boldsymbol{TabList}_{\text{destino}}, \mathbf{FPMC}^{k}_{\text{Conversão}}\ [,DW_{\text{Parâms}}])$:

[E-2-8-5-1] - dwgse_CreateStoreArea(TabFonte, TabDestino, ST_Geometry, WeightCol, minSize) – Cria um *buffer* em torno de objetos espaciais ponderando o tamanho do *buffer* (*minSize*) de acordo com os valores fornecidos. Este exemplo de enriquecimento poderia ser melhor categorizado como funcionalidade de análise de negócios contudo expressa adequadamente um processo de enriquecimento envolvendo análise de conjunto.

[E-2-9] - CONVERSÃO

[E-2-9-1] - $\mathbf{PTSC1}^{n}_{\text{Conversão}}(\boldsymbol{TabList}_{\text{fonte}}, \boldsymbol{TabList}_{\text{destino}}, \mathbf{FPSC1}^{k}_{\text{Conversão}}\ [,DW_{\text{Parâms}}])$

[E-2-9-1-1] - dwgse_ConvertToXML(TabFonte, TabDestino, ST_Geometry a) – Este exemplo de procedimento de conversão vai converter todos os objetos da coluna "a" da tabela fonte em formato texto XML na tabela de destino.

[E-2-10] - *DRILLDOWN*

[E-2-10-2] - $\mathbf{PTMD}^{n}_{DrillDown}$ (***TabList***$_{fonte}$, ***TabList***$_{destino}$, $\mathbf{FPMD}^{n}_{DrillDown}$ [,$DW_{Parâms}$])

[E-2-10-2-1] - dwgse_UniformGrid(TabFonte, TabDestino, ST_Geometry a, nrCel) – Este procedimento vai produzir uma grade retangular de "nrCel" células dentro da área coberta por cada objeto espacial "a" na tabela fonte.

[E-3-0] - ***S***W[$\mathbf{PA}_{Análise}$]

[E-3-11] - [$\mathbf{PAG}_{Análise}$] :

[E-3-11-1] - **Mapeamento** – Implementado no protótipo do Capítulo 5..

[E-3-11-2] - **Estatísticas Espaciais:**

[E-3-11-2-1] - dwgse_CreateStoreArea – Este procedimento, desenvolvido para o protótipo da seção 7.4, reproduz uma das funcionalidades do **ArcView Business Analyst**(ESRI 2001). Ele permite a criação de áreas em torno de objetos proporcionais a algum atributo relacionado da *tupla*, como no exemplo acima citado.

[E-3-11-3] - **Otimização de Negócios**:

[E-3-11-3-1] - dwgse_MarketArea – Este procedimento, também baseado numa das funcionalidades do **ArcView Business Analyst**(ESRI 2001), é capaz de criar uma área como um percentual mínimo de elementos espaciais em torno de elementos espaciais pai.

[E-3-11-4] - **Análise de Rede** – Não implementados.

*Apêndice D – Descritor XML da instância 5-Alpha-**S***

O documento XML abaixo apresenta o descritor do *framework* **5-Alpha** para a implementação de *data warehousing* espacial mostrada no Capítulo 7. Apenas alguns trechos foram colocados aqui a título de exemplo.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Model SYSTEM "Fw5ALPHASource.dtd" >
<Model name="Espacial">
 <Resources>
  <BaseSchema SCHID="SCH01" name="db2gse"/>
  <XWExtFile ARQID="ARQ01" ExtDir="'C:\Diretorio"' ExtName="'Arquivo.txt"'/>
 </Resources>
 <SectionA>
  <XWTADType SCHID="SCH01" name="ST_Geometry"        typeID="A-1-0">
   <XWTADType SCHID="SCH01" name="ST_Point"          typeID="A-1-1"/>
   <XWTADType SCHID="SCH01" name="ST_Curve"          typeID="A-1-2">
        <XWTADType SCHID="SCH01" name="ST_LineString"    typeID="A-1-2-1"/>
   </XWTADType>
   <XWTADType SCHID="SCH01" name="ST_Surface"        typeID="A-1-3">
        <XWTADType SCHID="SCH01" name="ST_Polygon" typeID="A-1-3-1"/>
   </XWTADType>
   <XWTADType SCHID="SCH01" name="ST_GeomCollection"  typeID="A-1-4"/>
  </XWTADType>
  <XWTADType SCHID="SCH01" name="coordref" typeID="A-2-0"/>

  <XWTPDType name="Integer" typeID="TPDINT"/>
  <XWTPDType name="VARCHAR" typeID="TPDSTR" ></XWTPDType>
  <XWTPDType name="Double" typeID="TPDREAL" ></XWTPDType>
  <XWTPDType name="Boolean" typeID="TPDBOOL" />
  <XWTPDType name="BLOB(1m)" typeID="TPDBLOB" />
 </SectionA>

 <SectionB>
        <XWAnd TrueValue="1" FalseValue="0" FUNCID="B-1-1" RetType="TPDBOOL" BaseName="ST_AndInt">
                <BOOLPar typeID="TPDINT" />
                <BOOLPar typeID="TPDINT" />
        </XWAnd>
        <XWOr TrueValue="1" FalseValue="0" FUNCID="B-1-2" RetType="TPDBOOL" BaseName="ST_OrInt">
                <BOOLPar typeID="TPDINT" />
                <BOOLPar typeID="TPDINT" />
        </XWOr>
        <XWNot TrueValue="1" FalseValue="0" FUNCID="B-1-3" RetType="TPDBOOL" BaseName="ST_NotInt">
                <BOOLPar typeID="TPDINT" />
        </XWNot>
 </SectionB>
 <SectionC>
  <XWCatalogs>
   <XWTypeCatalog SCHID="SCH01" TableName="GEOMETRY_COLUMNS" XWSchemaName="LAYER_SCHEMA"
XWSourceCol="LAYER_COLUMN" XWTableName="LAYER_NAME" XWTypeName="GEOMETRY_TYPE"/>
   <XWModuleCatalog NameID="C-1-4-1" SCHID="SCH01" TableName="SPATIAL_GEOCODER"
XWFileName="VENDOR_SPECIFIC" XWModuleName="GC_NAME" XWSProcName="PRIMARY_UDF"/>
   <XWModuleParCatalog CatalogModuleName="C-1-4-1">
    <XWModulePar name="GCID" parType="TPDINT"/>
    <XWModulePar name="VENDOR_NAME" parType="TPDSTR" />
    <XWModulePar name="PRECISION_LEVEL" parType="TPDINT" />
    <XWModulePar name="GEO_AREA" parType="TPDSTR" />
    <XWModulePar name="DESCRIPTION" parType="TPDSTR" />
   </XWModuleParCatalog>
  </XWCatalogs>
```

```
   <XWCatHandle>
     <PCSD BaseName="gse_register_layer" SCHID="SCH01">
       <TPDPar parName="layerSchema" typeID="TPDINT"/>
       <TPDPar typeID="TPDSTR" parName="layerTable" ></TPDPar>
       <TPDPar typeID="TPDSTR" parName="layerColumn" ></TPDPar>
       <TPDPar typeID="TPDSTR" parName="layerTypeName" />
       <TPDPar typeID="TPDINT" parName="srId" />
       <TPDPar typeID="TPDSTR" parName="geoSchema"></TPDPar>
       <TPDPar typeID="TPDSTR" parName="geoTable" ></TPDPar>
       <TPDPar typeID="TPDSTR" parName="geoColumn" />
       <TPDPar typeID="TPDINT" parName="nAttributes" />
       <TPDPar typeID="TPDSTR" parName="attr1Name" />
       <TPDPar typeID="TPDSTR" parName="attr2Name" />
       <TPDPar typeID="TPDSTR" parName="attr3Name" />
       <TPDPar typeID="TPDSTR" parName="attr4Name" />
       <TPDPar typeID="TPDSTR" parName="attr5Name" />
       <TPDPar typeID="TPDSTR" parName="attr6Name" />
       <TPDPar typeID="TPDSTR" parName="attr7Name" />
       <TPDPar typeID="TPDSTR" parName="attr8Name" />
       <TPDPar typeID="TPDSTR" parName="attr9Name" />
       <TPDPar typeID="TPDSTR" parName="attr10Name" />

     </PCSD>
     <PCSM BaseName="gse_register_gc" SCHID="SCH01">
       <TPDPar parName="gcId"   typeID="TPDINT"/>
       <TPDPar parName="gcName" typeID="TPDSTR"/>
       <TPDPar parName="vendorName" typeID="TPDSTR"/>
       <TPDPar parName="primaryUDF" typeID="TPDSTR"/>
       <TPDPar parName="precisionLevel" typeID="TPDINT"/>
       <TPDPar parName="vendorSpecific" typeID="TPDSTR"/>
       <TPDPar parName="geoArea" typeID="TPDSTR"/>
       <TPDPar parName="description" typeID="TPDSTR"/>
     </PCSM>
   </XWCatHandle>
   <XWDataHandle>
     <FCSC BaseName="ST_Point" FUNCID="C-3-1-1" ISMETHOD="S" RetType="A-1-1" SCHID="SCH01"
retMode="ONE">
       <TPDPar FakeNameBefore="Ponto com Coordenada X=" parDesc="Coordenada X" parName="X"
typeID="TPDREAL"/>
       <TPDPar typeID="TPDREAL" FakeNameBefore="e Coordenada Y=" parName="Y" parDesc="Coordenada
Y"></TPDPar>
       <TPDPar typeID="TPDINT" parDesc="Identificador do sistema de coordenadas" parName="coordRef"
FakeNameBefore="e Sistema de Coordenada=" />
     </FCSC>
     <FCSA BaseName="ST_NumPoints" FUNCID="C-3-3" ISMETHOD="N" RetType="TPDINT" SCHID="SCH01"
retMode="ONE">
       <TADPar FakeNameBefore="Numero de Pontos da linha" parDesc="Linha a qual desejamos obter o número de
pontos" parName="lineString" typeID="A-1-2-1"></TADPar>
     </FCSA>
     <FCSP BaseName="ST_Area" FUNCID="C-3-4-1" ISMETHOD="N" RetType="TPDREAL" SCHID="SCH01"
retMode="ONE">
       <TADPar FakeNameBefore="A Área do objeto" parDesc="Objeto espacial bidimensional(supoerficie)"
parName="superficie" typeID="A-1-3"/>
     </FCSP>
   </XWDataHandle>
 </SectionC>
 <SectionD>
   <XWSelect>
     <FRS1 BaseName="ST_IsClosed" FUNCID="D-1-1-1" FalseValue="0" ISMETHOD="N" RetType="TPDINT"
SCHID="SCH01" TrueValue="1">
       <TADPar FakeNameAfter=") está fechada" FakeNameBefore="A curva (" parDesc="Uma linha linha poligonal"
parName="curva" typeID="A-1-2"/>
     </FRS1>
     <FRS2 BaseName="ST_Contains" FUNCID="D-1-2-1" FalseValue="0" ISMETHOD="N" RetType="TPDINT"
SCHID="SCH01" TrueValue="1">
       <TADPar FakeNameAfter=")" FakeNameBefore="O objeto (" parDesc="Objeto que contem" parName="contem"
typeID="A-1-0"/>
       <TADPar FakeNameAfter=")" FakeNameBefore="contem o objeto (" parDesc="objeto que estara contido"
parName="estaContido" typeID="A-1-0"/>
     </FRS2>
     <FRS3 BaseName="dwgse_HasInteriorRings" FUNCID="D-1-3-1" FalseValue="0" ISMETHOD="N"
RetType="TPDINT" SCHID="SCH01" TrueValue="1">
       <TADPar FakeNameAfter=") tem pelo menos " FakeNameBefore="A curva (" parDesc="Linha poligonal fechada"
parName="curva" typeID="A-1-2"></TADPar>
       <TPDPar FakeNameAfter=") aneis interiores" FakeNameBefore="(" parDesc="Numero minimo de aneis interiores"
parName="nrAneis" typeID="TPDINT"/>
```

```
    </FRS3>
    <FRM1 BaseName="dwgse_IsSmaller" FUNCID="D-1-4-1" FalseValue="0" ISMETHOD="S" RetType="TPDINT"
SCHID="SCH01" TrueValue="1">
      <TADParSet FakeNameAfter=") é o menor" FakeNameBefore="O objeto (" typeID="A-1-0" parDesc="Objeto
espacial" parName="geometria" />
    </FRM1>

    <FRM2 BaseName="IsClosestDistance" FUNCID="D-1-5-1" FalseValue="0" ISMETHOD="N" RetType="TPDINT"
SCHID="SCH01" TrueValue="1">
      <TADParSet FakeNameAfter=") está mais proximo de (" FakeNameBefore="O objeto (" typeID="A-1-0"
parDesc="Objeto espacial" parName="g1"></TADParSet>
      <TADParSet FakeNameAfter=")" typeID="A-1-0" parDesc="Objeto Espacial" parName="g2" />
    </FRM2>
    <FRM3 BaseName="dwgse_IsSmaller(" FUNCID="D-1-6-1" FalseValue="0" ISMETHOD="S" RetType="TPDINT"
SCHID="SCH01" TrueValue="1">
      <TADParSet FakeNameAfter="é o menor que (" FakeNameBefore="O objeto (" typeID="A-1-0" parDesc="Objeto
espacial" parName="g" />
      <TPDPar FakeNameAfter=")" parDesc="Menor tamanho" parName="minSize" typeID="TPDINT"/>
    </FRM3>
  </XWSelect>
  <XWProject>
    <FPST1 BaseName="ST_Boundary" FUNCID="D-2-1-1" ISMETHOD="N" RetType="A-1-0" SCHID="SCH01"
retMode="ONE">
      <TADPar FakeNameAfter=")" FakeNameBefore="O contorno da objeto (" parDesc="objeto espacial origem do
contorno" parName="g" typeID="A-1-0"/>
    </FPST1>
    <FPMT BaseName="dwgse_ScaleUp" FUNCID="D-2-5-1" ISMETHOD="N" RetType="A-1-0" SCHID="SCH01"
retMode="SET">
      <TADPar FakeNameAfter=")" FakeNameBefore="aumento de escala do objeto (" parDesc="objeto espacial"
parName="g" typeID="A-1-0"></TADPar>
      <TPDParSet FakeNameAfter=")" FakeNameBefore="de (" parDesc="taxa de aumento de escala"
parName="escala" typeID="TPDINT"></TPDParSet>
    </FPMT>
    <FPSC1 retMode="ONE" FUNCID="D-2-2-2" RetType="TPDBLOB" BaseName="ST_AsBinary" SCHID="SCH01">
        <TADPar typeID="A-1-0" FakeNameBefore="O valor binario do objeto (" FakeNameAfter=")" parName="g"
parDesc="Objeto espacial a ser convertido" />
    </FPSC1>
    <FPSD BaseName="dwgse_Grid" FUNCID="D-2-4-1" ISMETHOD="N" RetType="A-1-0" SCHID="SCH01"
retMode="SET">
      <TADPar FakeNameAfter=")" FakeNameBefore="grade regular sobre (" parDesc="objeto espacial" parName="g"
typeID="A-1-0"></TADPar>
      <TPDPar FakeNameAfter="X" FakeNameBefore="com tamanho " parDesc="Numero de linhas da grade"
parName="m" typeID="TPDINT"/>
      <TPDPar typeID="TPDINT" parDesc="Numero de colunas da grade" parName="n" />
    </FPSD>
    <FPSX BaseName="ST_Union" FUNCID="D-2-3-1" ISMETHOD="N" RetType="A-1-0" SCHID="SCH01"
retMode="ONE">
      <TADPar FakeNameAfter=")" FakeNameBefore="A uniao de (" parDesc="objeto espacial" parName="g1"
typeID="A-1-0"></TADPar>
      <TADPar FakeNameAfter=")" FakeNameBefore="com (" parDesc="objeto espacial" parName="g2" typeID="A-1-
0"/>
    </FPSX>
  </XWProject>
  <XWAgregate>
    <FA1Section>
      <FA1 BaseName="dwgse_SumArea" FUNCID="D-3-1-1" ISMETHOD="N" RetType="TPDREAL" SCHID="SCH01"
retMode="ONE">
        <TADParSet FakeNameBefore="A soma total das areas de" typeID="A-1-0"></TADParSet>
      </FA1>
    </FA1Section>
    <FA2Section>
      <FA2 BaseName="dwgse_MinDistances" FUNCID="D-3-2-1" ISMETHOD="N" RetType="TPDREAL"
SCHID="SCH01" retMode="SET">
        <TADParSet FakeNameAfter=")" FakeNameBefore="A menor distância dentro de (" typeID="A-1-0"
parName="g1" parDesc="Conjunto de onde selecionar o de menor distancia"></TADParSet>
        <TADPar FakeNameAfter=")" FakeNameBefore="até o objeto (" parDesc="objeto alvo da menor distância"
parName="g2" typeID="A-1-0"></TADPar>
      </FA2>
    </FA2Section>
    <FA11Section>
      <FA11 BaseName="dwgse_AggObject" FUNCID="D-3-3-1" ISMETHOD="N" RetType="A-1-0" SCHID="SCH01"
retMode="ONE">
        <TADParSet FakeNameAfter=")" FakeNameBefore="A agregação de todos os objetos dentro de (" typeID="A-1-
0" parDesc="Nome da coluna de objetos que deve ser agregada" parName="gc1" />
      </FA11>
    </FA11Section>
```

```
      <FA12Section>
        <FA12 BaseName="dwgse_CalculaCentroides" FUNCID="D-3-4-1" ISMETHOD="N" RetType="A-1-1"
SCHID="SCH01" retMode="SET">
          <TADParSet FakeNameAfter=")" FakeNameBefore="Lista de Centroides de (" typeID="A-1-0"
parDesc="Conjunto de pontos a calcular centroide" parName="gc1" />
          <TPDPar FakeNameBefore="a uma distancia minima de " parDesc="distancia minima entre pontos a calcular
centroide" parName="MinDist" typeID="TPDREAL"/>
        </FA12>
      </FA12Section>
      <FA21Section>
        <FA21 BaseName="dwgse_Nearest" FUNCID="D-3-5-1" ISMETHOD="N" RetType="A-1-0" SCHID="SCH01"
retMode="ONE">
          <TADParSet FakeNameAfter=")" FakeNameBefore="O objeto de (" typeID="A-1-0" parDesc="Conjunto de
objetos a escolher o mais proximo" parName="gc1" />
          <TADPar FakeNameAfter=")" FakeNameBefore="mais proximo de (" parDesc="Objeto alvo a calcular a distancia"
parName="g2" typeID="A-1-0"/>
        </FA21>
      </FA21Section>
      <FA22Section>
        <FA22 BaseName="dwgse_NearestSet" FUNCID="D-3-6-1" ISMETHOD="N" RetType="A-1-0" SCHID="SCH01"
retMode="SET">
          <TADParSet FakeNameAfter=")" FakeNameBefore="O conjunto de objetos de (" typeID="A-1-0"
parDesc="Conjunto de objetos a selecionar o mais proximo" parName="gc1"></TADParSet>
          <TADPar FakeNameAfter=")" FakeNameBefore="mais proximos de (" parDesc="Objeto alvo da proximidade"
parName="g2" typeID="A-1-0"/>
          <TPDPar typeID="TPDINT" parDesc="Numero de objetos" parName="N" FakeNameBefore="de quantidade não
maior do que (" FakeNameAfter=")" />
        </FA22>
      </FA22Section>
    </XWAgregate>
  </SectionD>
  <SectionE>
    <XWExtract>
      <XWImport>
        <PEII1 BaseName="gse_import_shape" PEID="E-1-1-1-1" SCHID="SCH01">
          <XWFile ARQID="ARQ01"/>
          <TgtTable TargetSchema="" TargetTable=""/>
          <TPDPar parName="layerSchema" typeID="TPDSTR"/>
          <TPDPar parName="layerTable" typeID="TPDSTR"/>
          <TPDPar parName="layerColumn" typeID="TPDSTR"/>
          <TPDPar parName="filemName" typeID="TPDSTR"/>
          <TPDPar parName="exceptionFile" typeID="TPDSTR"/>
          <TPDPar parName="srId" typeID="TPDINT"/>
          <TPDPar parName="commitScope" typeID="TPDINT"/>
        </PEII1>
      </XWImport>
    </XWExtract>
    <XWTranform>
      <XWTSelect>
        <PTSR BaseName="dwgse_InsideJoin" FRSID="D-1-2-1" SCHID="SCH01" SourceTable1="TabFonte1"
SourceTable2="TabFonte2" TargetTable="TabDestino">
          <SrcColPar name="FEATURE"></SrcColPar>
          <TPDPar parName="A" typeID="A-1-0"/>
          <TPDPar typeID="A-1-0" parName="B" ></TPDPar>
        </PTSR>
      </XWTSelect>
    </XWTranform>
  </SectionE>
</Model>
```

Apêndice E – Glossário de Termos

Achamos relevante destacar os seguintes termos e significados usados no texto:

1 - **Tipos Primitivos de Dados (TPDs)** – Tipos de dados nativos da linguagem de consulta, como Inteiros, Reais, Caracteres, etc.

2 - **Tipos Abstratos de Dados (TADs)** – Usaremos este termo para especificar as várias classes de tipos definidos pelo usuário no modelo de dados objeto-relacional. Utilizamos também os seguintes termos derivados ou relacionados:

2.1 - **Um Tipo Abstrato de Dado** – Define um tipo específico, como "Imagem" ou "Geometria", mas sem especificar um objeto ou instância específica deste tipo.

2.2 - **Atributo de um TAD** – São as variáveis que compõem a estrutura do TAD.

2.3 - **Estrutura de um TAD** – É o conjunto de variáveis, de TPDs ou TADs, que compõem um TAD.

2.4 - **Valor, elemento ou objeto de um TAD** – São os vários objetos ou instâncias da classe.

2.5 - **Valor de um objeto de um TAD** – São os valores dos objetos, mais especificamente, o valor dos atributos dos objetos de um TAD.

2.6 - **Propriedades de um TAD** – São as características da classe, não necessariamente seus atributos. Pode estar relacionado ao seu comportamento.

2.7 - **Propriedade de um objeto de um TAD** – Valores calculados sobre os valores dos atributos do TAD. Por exemplo, a área de um objeto "círculo" seria calculado sobre as dimensões de cada objeto círculo.

3 - **Banco de** *Data Warehouse* – Especifica o banco de dados físico, junto com seus sistemas especializados, aonde estarão depositados os dados extraídos acumulados para futura análise.

4 - **Processo de** *Data Warehousing* – Constitui todo o processo de Extração, Transformação e Carga (ETL) dos dados no Banco de *Data Warehouse* (3), mais a análise, multidimensional ou não, dos dados.

5 - **Sistema de** *Data Warehousing* – Especifica um sistema completo, incluindo o Banco de *Data Warehouse*, para realizar o Processo de *Data Warehousing* (4).

6 - *Data Warehousing* Estendido para TADs – É um dos produtos deste trabalho, define todo o Processo de *Data Warehousing* (4) quando incluídos os TADs.

7 - **Função** – Especifica as funções com parâmetro e retorno definido que farão parte do suporte para TADs dentro do banco de dados (na linguagem de consulta)

8 - **Procedimento** – Especifica uma função de parâmetro definido e sem tipo de retorno, que deve ser invocada diretamente e não faz parte do escopo de consulta da linguagem.

Apêndice F – Referencias

F.1 – Referencias pesquisadas

A lista seguinte mostra todas os trabalhos, produtos e estudos pesquisados. Nem todos podem ter sido referenciados no texto mas forma base para formação das idéias conceitos:

- **1974 – 1997** – Al-Taha, Snodgrass e Soo 1993,Bailey 1982,Barquin e Edelstein 1997,Berson e Smith 1997,Braga at al. 1997,Braga, Nobre e Pinheiro 1994,Candy 1995,Cavalcanti e Salgado 1994,Clementini e Di Felice 1994,Clementini e Di Felice 1996,Clementini, Di Felice e Egenhofer 1994,Codd 1993,Codd, Codd e & Salley 1993,Computerwire 1996,Coppock e Rhind 1996,Devlin 1997b,Devlin 1997a,Edelstein 1996,Edelstein 1997,Egenhofer e Herring 1991b,Fisch 1992,Grelot 1994,Gupta 1997,Guttag 1977,Günther e Sheck 1991,Güting 1989,Güting 1994,Han 1997a,Han 1997b,Han e Koperski 1995,Han, Koperski e Adhikary 1996,Han, Xu e Lu 1990,Hufford 1997,Imhoff 1995,Inmon 1992,Kaplan 1996,Kemp e Kowalczyk 1994,Kimball 1996,Klug 1982,Langran 1992,Lee e Zhang 1989,Liskov e Zilles 1974,Lorie e Meier 1984,Mimno 1997,Navathe e Elmasri 1994,Newell, Theriault e Easterfield 1992,Nobre e Carvalho 1996,Open GIS Consortium 1997,Orman 1991,Orr 1997,Parent e Church 1987,Pequet 1996,Pequet e Wentz 1994,Renolen 1996,Sakaguchi 1996,Seibold 1997,Seshadri, Livny e Ramakrishna 1997,Stefanovic 1997b,Stefanovic 1997a,Stonebraker 1986,Tomlinson 1987,Walchowicz e Healey 1994,Wu e Buchmann 1997
- **1998 – 2000** – Adelman e Moss 2000,Altoe e Marino 1998,Ballard at al. 1998,Barclay, Gray e Slutz 2000,Bauman 1999,Berkel 2000,Braga 1998a,Cabibbo e Torlone 1999,Castro e Machado 1998,Cavalcanti e Cerqueira 1998,Chen at al. 1999,Clarke 2000,CUBEWERX 1999,Davis 1998,ESRI 1998,Gallas 1999,GarterGroup 1998,Gray e Watson 1998,Güting at al. 1999,Güting e Dieker 2000,Han 1998,Han, Chee e Chiang 1998,Han, Stefanovic e Koperski 1998b,Han, Stefanovic e Koperski 1998a,Han, Truffet e Zhou 1999,Hill 1998,IBM Corporation 1998,Kemp e Lee 2000,Kimball 1998,Kimball at al. 1998,Kimball 2000,King County GIS Center 2000,Koperski 1999,Koperski, Han e Adhikary 1998,Lowe 2000,Maruyama, Tamura e Uramoto 1999,Open GIS Consortium e Ryden 1999,Oracle Corporation 2000,Papadias, Kalnis e Mamoulis 1999,Pedersen e Jensen 1998,Perez at al. 1998,Power 1999,Rauber, Tomsich e Riedel 2000,Rodrigues 1998,Romero, Bernardino e Martinho 1998,Santos 1998,Santos, Telles e Santos 1998,Sarawagi, Agrawal e Gupta 1999b,Sarawagi, Agrawal e Gupta 1999a,Silva e Campos 1998,Skejellaug 1999,Soares 1998,Stefanovic, Han e Koperski 2000,Trillium e Harte-Hanks Data Technologies 1999,Vassiliadis at al. 2000,Wash 2000,Zornes 2000
- **2001 – 2004** – Abelló, Samos e Saltor 2003,Adler 2001,Adler e Higgs 2003,Carvalho 2004,Chawla at al. 2001,Chorel 2001,Coss, Zeidenstein e Gonzales 2001,CUBEWERX

2004a,CUBEWERX 2004b,de Macedo e Fernandes 2003,DeBloch at al. 2001,Denis e François 2001,DIGIBASE 2004,Dorsey 2001,ESRI 2001,ESRI 2003b,ESRI 2003a,ESRI 2004,Fernandes 2002,Ferreira e Tanaka 2001,Freitas 2001,Grey 2001,Han, Kamber e Tung 2001,Helmer 2001,IBM Corporation 2001,IBM Corporation 2002d,IBM Corporation 2002c,IBM Corporation 2002a,IBM Corporation 2002b,IBM Corporation 2002e,Jones 2001,Kache 2001,Kimball 2001b,Kimball 2001a,Lema e Güting 2002,Lutz 2004,Maniatis at al. 2003,MapInfo 2001,MAPTEL 2004,Marchand, Bedard e Edwards 2001,Meador 2001,Morimoto 2001,Morimoto at al. 2003,Morimoto, Kubo e Kanda 2001,North Texas GIS Consortium 2001,Open GIS Consortium 2001,Oracle Corporation 2001,Papadias at al. 2001,Papadias at al. 2002,Pedersen, Riis e Pedersen 2002,Pendse 2001,Rao at al. 2003,Rifaieh e Benharkat 2002,Rivest, Bedard e Marchand 2001,Rizzi e Golfarelli 2001,Shekhar, Lu e Zhang 2001,Thomas 2001,Vassiliadis at al. 2003,Vassiliadis, Simitsis e Skiadopoulos 2002,Wang at al. 2003,Zhang.L. at al. 2004,Zhou e Prasher 2004

F.2 - Referencias do texto

1. Abelló A., Samos J. & Saltor F. "Implementing Operations to Navigate Semantic Star Schemas". *In:6th International Workshop on Data Warehousing and OLAP (DOLAP 2003)* pp.56-62. New Orleans(USA). 2003
2. Adelman S. & Moss L. "Data Warehouse Failures".Anonymous Data Warehouse Risks in Data Warehouse Project Management. Addison Wesley Longman. 2000;
3. Adler D.W. "DB2 Spatial Extender - Spatial data within the RDBMS". *In:Proceedings of 27th International Conference on Very Large Data Bases* pp.687-690. 2001
4. Adler D.W. & Higgs J. "IBM DB2 and Microsoft SQL/Server Integration at Metro Nashville". *In:23nd Annual ESRI International User Conference* San Diego, CA, USA. 2003
5. Al-Taha K.K., Snodgrass R.T. & Soo M.D. "Bibliography on Spatio-Temporal databases" *SIGMOD Record* 22(I):59-67. 1993;
6. Data Warehouse - Estudo de Caso - BancosAltoe F. & Marino M.T. 1998;
7. Bailey N.T.J. "Biomathematics of Malaria" 1 ed. Taylor and Francis. 1982;
8. Ballard C. et al. "Data Modeling Techniques for Data Warehousing" San Jose,CA: AnonymousInternational Technical Support Organization; 1998;

9. Barclay T., Gray J. & Slutz D. "Microsoft TerraServer: A Spatial Data Warehouse". *In:Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data* pp.307-318. Austin,TX,USA. 2000

10. Barquin R.C. & Edelstein H. "Planning and Designing the Data Warehouse" New Jersey: Barquin RC and Edelstein H.Prentice Hall PTR; 1997;

11. Batty P. "Exploiting relational database technology in a GIS" *Computers and Geosciences* 18(4):453-62. 1992;

12. Bauman P. "A Database Array Algebra for Spatio-Temporal Data and Beyond".Anonymous Next Generation Information Technologies and Systems. 1999;

13. Beckmann N. et al. "The R^* -Tree: An Efficient and Robust Method for Points and Rectangles$^+$".Proceedings of the 1990 ACM SIGMOD - International Coference on Management of Data. pp.322-331. Atlantic City,NJ. 1990

14. Bedard Y. "Geospatial Data and Data Warehousing". *In:Discovering Geographic Knowledge in Data-Rich Environments* Kirkland,WA,USA. 1999

15. Berkel J.v. Data Warehouse; where to locate GIS.[Unpublished] 2000;

16. Berson A. & Smith S.J. "Data Warehousing, Data Mining, and OLAP" Wyzalek J.McGraw-Hill; 1997;

17. Bezerra E. SQL - Principais características do SQL:1999. SQL Magazine. 1(4). 32-38. 2003;

18. Braga A.L., Ferramentas de Manipulação Nebulosa de Dados com Aplicação em Sistemas de Informação Geográfica, M.Sc., Engenharia de Sistemas e Computação - Universidade Federal do Rio de Janeiro - COPPE, 1998a;

19. Braga A.L., Fuzzy Manipulation Tools of Data with Application in Geographic Information Systems, M.Sc., Computing Engineering Department - Rio de Janeiro Federal University - COPPE, 1998b;

20. Braga A.L. et al. "GISEpi: a simple geographical information system to support public health surveillance and epidemiological investigations" *Computer Methods and Programs in Biomedicine* 53:33-45. 1997;

21. Braga A.L., Nobre F.F. & Pinheiro R.S. "SIGEPI - A Geográphical Information System for Application in Epidemiology and Public Health". *In:World Congress on Medical Phisics and Biomedical Egineering* 39a - Part2. p.541 1994

22. Burrough P.A. "Principles of Geografical Information Systems for Land Resource Assesment" Clarenden Press.Oxford; 1986;

23. Burrough P.A. "Development of inteligent geographical information systems" *International Journal of Geographical Information Systems* 6(1):1-11. 1992;

24. Cabibbo L. & Torlone R. "A Framework for the Investigation of Aggregate Functions in Database Queries". *In:Seventh International Conference on Database Theory (ICDT-99)* pp.383-397. 1999

25. Camara G. et al. "SPRING: Integrating remote sensing and GIS by object-oriented data modelling" *Computer & Graphics* 3(20) 1996;

26. Candy J.T., Development of a Prototype Temporal Geographic Information System, Master of Science, Simon Fraser University, 1995;

27. Carvalho B.F. Arquiteturas de Ferramentas OLAP. SQL Magazine. 1(9). 12-16. 2004;

28. Data Warehouse - Estudo de Caso - Cadeia de Suprimento e DemandaCastro A. & Machado L.F.D.C. 1998;

29. Cavalcanti A.E.C. & Salgado A.C. "Um Estudo para Tratar a Dimenso de Tempo em Siatemas de Banco de Dados".IX Simposio Brasileiro de Baco de Dados. I. pp.357-381. Sao Carlos. 1994

30. Cavalcanti M.C.Y. & Cerqueira A.A.C. Data Warehouses Orientados a Objetos.[Unpublished] 1998;

31. Chawla S. et al. "Modeling spatial dependencies for mining geospatial data: An introduction".Miller HJ and Han J. Geographic data mining and Knowledge Discovery. Taylor and Francis. 2001;

32. Chen W. et al. "High Level Indexing of User-Defined Types". *In:Proceedings of 25th International Conference on Very Large Data Bases* pp.554-564. 1999

33. Chorel J. "A new approach to creating & managing a spatial data warehouse consortium". *In:GITA's 24th Annual Conference and Exhibition* San Diego, CA, USA. 2001

34. Clarke K.C. "Getting Started with Geographic Information Systems" 3rd ed. AnonymousPrentice Hall College Division; 2000;

35. Clementini E. & Di Felice P. A Comparison of Methods for RepresentingTopological Relationships. Information Sciences. 80 1-34. 1994;

36. Clementini E. & Di Felice P. A Model for Representing TopologicalRelationships Between Complex Geometric Features in Spatial Databases,Information Sciences. Information Sciences. 90(1–4). 121-136. 1996;

37. Clementini E., Di Felice P. & Egenhofer M.J. "Topological relations between regions with holes" *International Journal of Geographical Information Systems* 8(2):129-42. 1994;

38. Codd E.F. "A Relational Model of Data for Large Shared Data Banks" *Communications of ACM* 6:377-87. 1970;

39. Codd E.F. Providing OLAP (On-Line Analytical Processing) to useranalysts: An IT mandate. 1993;

40. Codd E.F., Codd S.B. & & Salley. Beyond decision support. Computerworld. 27(30). 1993;

41. Computerwire. What is Metadata. Data Warehousing Tools Bulletin. 1996;

42. Coppock J.T. & Rhind D.W. The History of GIS. xxxx. 1996;

43. Coss R., Zeidenstein K. & Gonzales M. "Picture This! Making Your Data Warehouse Spatially Aware" *Journal of Data Warehousing* 6(3) 2001;

44. CUBEWERX. CUBEWERX Spatial Data Warehouse Architecture.[Unpublished] 1999;

45. CUBEWERX. CubeSTOR - Spatial Data Warehouse.[Unpublished] 2004a;

46. CUBEWERX. CubeWerx Suite V.3.2 Overview.[Unpublished] 2004b;

47. Davis Jr C.A. & Borges K.A.d.V. "Object-Oriented GIS in Pratice" *URISA Journal* :786-95. 1994;

48. Davis J.R. IBM's DB2 Spatial Extender: Managing Geo-Spatial Information Within The DBMS.[Unpublished] 1998;

49. de Macedo K.A.C. & Fernandes D. "Metodologia para Avaliação de Geocodificadro de Imagens SAR". *In:XI Simpósio Brasileiro de Sensoriamento Remoto* Belo Horizonte. 2003

50. DeBloch S. et al. "Indexing Support in DB2 Universal Database". *In:Component Database Systems* pp.105-138. 2001

51. Denis B. & François M. "The New Age of Topographic Data - Management and Access". *In:GEOTec Media's, GIS 2001 Conference* Vancouver, CANADA. 2001

52. Devlin B. "Data Warehouse: from architecture to implementation" Cote LD.Addison Wesley Longman,Inc.; 1997b;

53. Devlin B. Managing Time in the Data Warehouse. InfoDB. 11(1). 7-11. 1997a;

54. DIGIBASE. 2004;

55. Dorsey P. Data Warehouses,Ad Hoc Query Tools and Other Ways to Destroy Your Company.[Unpublished] 2001;

56. Dunn R. Statistical Mapping. The American Statistician. 41(2). 153-156. 1987;

57. Edelstein H. Technology How-To -- Mining Data Warehouses -- New software helps discover information within databases that queries and reports can't reveal. Techsearch Information Week.(561). 1996;

58. Edelstein H. "An Introdution to Data Warehousing".Barquin RC and Edelstein H. Planning and Designing the Data Warehouse. New Jersey. Prentice Hall PTR. 1997;

59. Egenhofer J.J. "Why Not SQL!" *International Journal of Geographical Information Systems* 6(2):71-85. 1992;

60. Egenhofer M.J. & Herring J.R. Categorizing binary topological relationships between regions, lines and points in geographic databases. Department of Surveying Engineering, University of Maine, Orono, ME. 1991b;

61. Egenhofer M.J. & Herring J.R. "High-Level Spatial Data Structures for GIS".Maguire DJ, Goodchild MF, and Rhind DW. Geographic Information

Systems: Principles and Aplications. 16. Ny,USA. Longman Scientific & Techinical. 1991a;

62. ESRI. Spatial Data Warehousing for Hospital Organizations. 1998;

63. ESRI. ArcView Business Analyst.[Unpublished] 2001;

64. Interoperability Extension for ArcExplorer—Java Edition Now Available. 2003b;

65. ESRI. Spatial Data Standards and GIS Interoperability.[Unpublished] 2003a;

66. Environmental Systems Research Institute(ESRI). 2004;

67. Fernandes L. "ORACLE 9i para Desenvolvedores - Oracle Developer 6i" Narcisi G and Reinprecht R.Axcel Books do Brasil Editora Ltda.; 2002;

68. Ferreira A.C. & Tanaka A.K. "An Architecture for Dimensional and Spatial Analyses Integration". *In:World Multiconference on Systemics, Cybernetics and Informatics(SCI 2001/ISAS)* 2001

69. Fisch S. A History of GIS. Design Management. 16(1). 1992;

70. Frank A.U. "Spatial concepts, geometric data models, and geometric data structures" *Computers and Geosciences* 18(4):409-17. 1992;

71. Franklin Wm.R. "Computer Systems and Low-Level Data Structures for GIS".Maguire DJ, Goodchild MF, and Rhind DW. Geogrphic Information Systems: Principles and Aplications. 15. Ny,USA. Longman Scientific & Techinical. 1991;

72. Freitas G.M., Uma Ferramenta de Apoio à Modelagem de Dados Dimensional, MSc, Escola de Governo Fundação João Pinheiro, 2001;

73. Gallas S. Kimball Vs. Inmon. DM Direct Newsletter. 1999;

74. DW Data PreparationGarterGroup. 1998;

75. Goodchild M.F. "Geographical data modeling" *Computers and Geosciences* 18(4):401-8. 1992;

76. Gray P. & Watson H.J. "Decision Support in the Data Warehouse" 1 ed. New Jersey,NY: AnonymousPrentice Hall; 1998;

77. Grelot J. "Cartography in the GIS Age" *The Cartographic Journal* 31:56-60. 1994;

78. Grey B.O. Spatially Enabling an incident Data Warehouse.[Unpublished] 2001;

79. Gupta V.R. An Introdution to Data Warehousing.[Unpublished] 1997;

80. Guttag J. "Abstract Data Types and the Development of Data Structures" *Communications of ACM* 1977;

81. Guttman A. "The R-Trees:A dynamic index structure for spatial searching".Proceedings of the 1990 ACM SIGMOD - International Coference on Management of Data. pp.47-57. 1984

82. Günther O. & Sheck H.J. "Extending a DBMS with Spatial Operations".Anonymous Advances in Spatial Databases. Berlin. Springer-Verlag. 1991;

83. Güting R.H. "An Extensible Relational System for Geometric Applications".Proceedings of the 15th International Conference on Very Large Databases(VLDB). pp.33-44. Amsterdam. 1989

84. Güting R.H. "An Introduction to Spatial Database Systems" *VLDB Journal* 3(4):1-32. 1994;

85. Güting R.H. & Dieker S. "Plug and Play with Query Algebras: SECONDO. A Generic DBMS Development Environment". *In:Procedings. of the International. Database Engineering and Applications Symposium* pp.380-390. 2000

86. Güting R.H. et al. "Secondo/QP: Implementation of a Generic Query Processor". *In:10th Int. Conf. on Database and Expert System Applications (DEXA'99)* pp.66-87. 1999

87. Han J. "Conference Tutorial Notes: Spatial Data Mining and Spatial Data Warehousing". *In:5th Int'l Symposium on Spatial Databases(SSD'97)* Βερλιν,Γερμανψ. 1997b

88. Han J. "Spatial Data Mining and Spatial Data Warehousing - Tutorial". *In:SSD'97 Conference* 1997a

89. Han J. "Towards On-Line Analytical Mining in Large Databases" *SIGMOD Record (ACM Special Interest Group on Management of Data)* 27(1) 1998;

90. Han J., Chee S. & Chiang J.Y. "Issues for On-Line Analytical Mining of Data Warehouses". *In:Proc. of 1998 SIGMOD'96 Workshop on Research Issues on Data Mining and Knowledge Discovery (DMKD'98)* pp.21-25. 1998

91. Han J., Kamber M. & Tung A.K.H. "Spatial Clustering Methods in Data Mining: A Survey" Miller HJ and Han J.Taylor and Francis; 2001;

92. Han J. & Koperski K. "Discovery of Spatial Association Rules in Geographic Information Databases". *In:Proc. 4th Int'l Symp. on Large Spatial Databases (SSD95)* pp.47-66. Maine. 1995

93. Han J., Koperski K. & Adhikary J. "Spatial Data Mining: Progress and Challenges - Survey Paper". *In:1996 SIGMOD'96 Workshop. on Research Issues on Data Mining and Knowledge Discovery (DMKD'96)* Montreal,Canada. 1996

94. Han J., Stefanovic N. & Koperski K. " An Efficient Two-Step Method for Classification of Spatial Data". *In:Proc. 1998 International Symposium on Spatial Data Handling SDH'98* Vancouver,Canada. 1998a

95. Han J., Stefanovic N. & Koperski K. "Selective Materialization: An Efficient Method for Spatial Data Cube Construction". *In:Pacific-Asia Conference on Knowledge Discovery and Data Mining(PAKDD'98) [Lecture Notes in Artificial Intelligence, 1394, Springer Verlag, 1998]* pp.144-158. Melbourne, AU. 1998b

96. Han J., Truffet D. & Zhou X. "Efficient Polygon Amalgamation Methods for Spatial OLAP and Spatial Data Mining "". *In:Proc. 6th Int. Symp. on Large Spatial Databases (SSD'99)* pp.167-187. Hong Kong. 1999

97. Han J., Xu X. & Lu W. "RT-Tree: An Improved R-Tree Indexing Structure for Temporal Spatial Databases". *In:Proc. 4th Int. Symp. on Spatial Data Handling (SSDH'90)* pp.1040-1049. Zurich, Switzerland. 1990

98. Helmer T. "Using object Relational Database Management Systems to enable enterprise GIS". *In:GITA's 24th Annual Conference and Exhibition* San Diego, CA, USA. 2001

99. Herring J.R. "TIGRIS: A data model for an object-oriented Geographic Information System" *Computers and Geosciences* 18(4):443-52. 1992;

100. Hill J. "Challenges of Data Preparation for Data Warehouse". *In:U.S. Symposium/ITxpo 98* 1998

101. Hufford D. "Metadata Repositories: The key to Unlocking Information in the Data Warehouses".Barquin RC and Edelstein H. Planning and Designing the Data Warehouse. 11. New Jersey. Prentice Hall PTR. 1997;

102. IBM Corporation. "IBM DB2 Spatial Extender User's Guide and Reference - Version 7" IBM Corporation. IBM Χορπορατιον; 1998;

103. IBM Corporation. "IBM Informix Spatial DataBlade Module User's Guide - Version 8.11" IBM Corporation.IBM Χορπορατιον; 2001;

104. IBM Corporation. "Business Intelligence Tutorial: Introdution to the Data Warehouse Center - Version 8 " IBM Corporation.IBM Corporation; 2002d;

105. IBM Corporation. Data Warehouse Center Administration Guide - Versão 8. 2002b;

106. IBM Corporation. Data Warehouse Center Application Integration Guide - Versão 8. 2002e;

107. IBM Corporation. Information Catalog Manager Administration Guide - Versão 8. 2002c;

108. IBM Corporation. "IBM DB2 Spatial Extender User's Guide and Reference - Version 8" IBM Corporation. IBM Χορπορατιον; 2002a;

109. Imhoff C. "New York".Anonymous Seminar on Data Modeling for the Data Warehouse. New York. New York:DAMA. 1995;

110. Inmon W.H. "Building the Data Warehouse" New York: AnonymousJohn Wiley & Sons; 1992;

111. Jones M.A. "Why GPS is in your future". *In:GITA's 24th Annual Conference and Exhibition* San Diego, CA, USA. 2001

112. Kache H. Spatial Support for DB2 Federated Databases. 2001;

113. Kaplan E.D. "Understanding GPS: Principles and Applications" AnonymousArtech House; 1996;

114. Kemp Z. & Kowalczyk A. "Incorporating the temporal dimension en a GIS".Inovations in GIS - selected papers from the First National Conference on GIS Research UK. I. pp.89-103. UK. 1994

115. Kemp Z. & Lee H. "A Multidimensional Model for Exploratory Spatiotemporal Analysis". *In:Proceedings of the 5th International Conference on GeoComputation* University of Greenwich, UK. 2000

116. Kimball R. "Data Warehouse Toolkit " AnonymousWiley, John & Sons, Incorporated; 1996;

117. Kimball R. Meta Meta Data Data. DBMS Magazine. March 1998;

118. Kimball R. Many Alternate Realities. Intelligent Enterprise. 3(3). 2000;

119. Kimball R. Address Space. Intelligent Enterprise. 4(2). 2001b;

120. Kimball R. Spatial Enabling Your Data Warehouse. Intelligent Enterprise. 4(1). 2001a;

121. Kimball R. et al. "The Chess Pieces".Anonymous The Data Warehouse Lifecycle Toolkit: Expert Methods for Designing, Developing and Deploying Data Warehouse. 1. 1998;

122. King County GIS Center. Data Warehouse overview.[Unpublished] 2000;

123. Klug A.C. "Equivalence of Relational Algebra and Relational Calculus Query Languages Having Aggregate Functions" *Journal of the ACM(JACM)* 29(3):699-717. 1982;

124. Kolias V.J., Kolias J.G. & Yassoglou N.J. "A query language for retrieving information from a soil data bank" *Computers and Geosciences* 7(4):393-400. 1981;

125. Koperski K., A Progressive Refinement Approach to Spatial Data Mining, Ph.D., Simon Fraser University, 1999;

126. Koperski K., Han J. & Adhikary J. "Mining Knowledge in Geographical Data" *Communications of ACM* 1998;

127. Langran G. "Time in Geographic Information Systems" London: Taylor and Francis.Taylor & Francis; 1992;

128. Lee Y.C. & Zhang G. "Developments in Geographic Information System Technology" *ASCE Journal of Surveying Engineering* 15:304-23. 1989;

129. Lema J.A.C. & Güting R.H. "Dual Grid: A New Approach for Robust Spatial Algebra Implementation" *GeoInformatica* 6(1):57-76. 2002;

130. Light R. "Iniciando em XML" 1 ed. Assumpção Filho MM.MAKRON Books do Brasil Editora Ltda.; 1997;

131. Liskov B. & Zilles S. "Programming with Abstract Data Types". *In:SIGPLAN Notices* 1974

132. Lorie R. & Meier A. "Using a Relational DBMS for Geographical Databases".Anonymous GeoProcessing. 1984;

133. Lowe J.W. Spinning Straw into Gold: Data Warehouses and Spatial Web Sites . Geospatial Solutions Magazine. 2000;

134. Lutz D. Spatial ETL Tools Improve Geospatial Intelligence. GEO World.(February 2004). 2004;

135. Maguire D.J. & Dangermond J. "The Functionality of GIS".Maguire DJ, Goodchild MF, and Rhind DW. Geogrphic Information Systems: Principles and Aplications. 21. Ny,USA. Longman Scientific & Techinical. 1991;

136. Maguire D.J. & Raper J.F. "Design models and functionality in GIS" *Computers and Geosciences* 18(4):387-94. 1992;

137. Maniatis S.A. et al. "Advanced visualization for OLAP". *In:DOLAP 2003, ACM Sixth International Workshop on Data Warehousing and OLAP* pp.9-16. New Orleans, Louisiana, USA. 2003

138. MapInfo I. Managing Spatial Data in a Relational Database Management System - A white paper.[Unpublished] 2001;

139. MAPTEL NETWORKS S.A.U. 2004;

140. Marchand P., Bedard Y. & Edwards G. "A hypercube-based method for spatio-temporal exploration and analysis" *GeoInformatica* 2001;

141. Maruyama H., Tamura K. & Uramoto N. "XML and JAVA: developing WEB applications" 1st ed. O'Brien M.Addison Wesley Longman, Inc.; 1999;

142. Meador R.W. "Geocoding Technology A Useful Tool For Enterprise Applications". *In:GITA's 24th Annual Conference and Exhibition, "2001: A Geospatial Odyssey"* San Francisco-CA. 2001

143. Mediano M.R., Casanova M.A. & Dreux M. "A Family of storage Methods for Geographic Data".9no Simpósio Brasileiro de Banco de Dados. pp.310-323. 1995

144. Mimno P.R. "Data Warehousing Architectures".Barquin RC and Edelstein H. Planning and Designing the Data Warehouse. 8. New Jersey. Prentice Hall PTR. 1997;

145. Morehouse S. "The ARC/INFO Geographic Information System" *Computers and Geosciences* 18(4):435-41. 1992;

146. Morimoto Y. "Mining Frequent Neighboring Class Sets in Spatial Databases". *In:SIGKDD* San Francisco,CA. 2001

147. Morimoto Y. et al. "Extracting Spatial Knowledge from the Web". *In:Symposium on Applications and the Internet (SAINT 2003)* pp.326-333. Orlando,FL, USA. 2003

148. Morimoto Y., Kubo H. & Kanda T. Mining Optimized Distance and/or Orientation Rules in Spatial Databases.[Unpublished] 2001;

149. Murai T., Miyakoshi M. & Shimbo M. "A Fuzzy Document Retrieval Method Based on Two-Valued Indexing" *Fuzzy Sets and Systems* 30(2):103-20. 1989;

150. Navathe S.B. & Elmasri R. "Fundamental of Database Systems" Joraanstad D.Addison-Wesley Publishing Company; 1994;

151. Neteler M. & Mitasova H. "Open Source GIS A GRASS GIS Approach" Boston, Dordrecht, London: AnonymousKluwer Academic Publishers; 2002;

152. Newell R.G., Theriault D. & Easterfield M. "Temporal GIS-Modeling the evolution of spatial data in time" *Computers and Geosciences* 18(4):427-33. 1992;

153. Nobre F.F. & Carvalho M.S. Spatial and Temporal Analysis of Epidemiological Data.[Unpublished] 1996;

154. North Texas GIS Consortium. Spatial Data Warehouse.[Unpublished] 2001;

155. Open GIS Consortium, INC. 1997;

156. Open GIS Consortium I. Geocoder Service Specification. 2001; OGC-01-026r1.

157. Open GIS Consortium I. & Ryden K. OpenGIS Simple Features Specification For SQL. 1999; 99-049.

158. Openshaw S. "Developing Appropriate Spatial Analysis Methods for GIS".Maguire DJ, Goodchild MF, and Rhind DW. Geogrphic Information Systems: Principles and Aplications. 25. Ny,USA. Longman Scientific & Techinical. 1991;

159. Oracle Corporation. ORACLE Spatial - Java Library User's Guide. 2000;

160. Oracle Corporation. "ORACLE Spatial User's Guide - Release 9.0.1 " Oracle Corporation.Oracle Corporation; 2001;

161. Orman L.V. "Elements in information systems: components and architecture" New York: Stewart CJr.Macmillan Publishing Company; 1991;

162. Orr K. Data Warehouse Technology.[Unpublished] 1997;

163. Papadias D., Kalnis P. & Mamoulis N. "Hierarchical Constraint Satisfaction in Spatial Databases". *In:Proceedings of the Sixteenth National Conference on Artificial Intelligence and Eleventh Conference on Innovative Applications of Artificial Intelligence* pp.142-147. 1999

164. Papadias D. et al. "Efficient {OLAP} Operations in Spatial Data Warehouses". *In:Proc. of the 7th Intl. Symp. on Spatial and Temporal Databases* pp.443-459. Redondo Beach, Italy. 2001

165. Papadias D. et al. "Indexing Spatio-Temporal Data Warehouses". *In:IEEE Int. Conference on Data Enginnering (ICDE)* 2002

166. Parent P. & Church R. "Evolution of Geographic Information Systems as Decision Making Tools". *In:2nd Annual Conference of the American Society for Photogrammetry and Remote Sensing and the American Congress on Surveying and Mapping* San Francisco,CA. 1987

167. Pedersen D., Riis K. & Pedersen T.B. "A powerful and SQL-compatible data model and query language for OLAP". *In:Proceedings of the thirteenth Australasian conference on Database technologies* 5. 2002

168. Pedersen T.B. & Jensen C.S. "Multidimensional Data Modeling for Complex Data". *In:Proceeding of the ICDE' 99* 1998

169. The origins of today's OLAP products. 2001;

170. Pequet D. TEMPEST: A Spatial-Temporal GIS.[Unpublished] 1996;

171. Pequet D. & Wentz E.A. "An Approach for Time-Based Spatial Analysis of Spatio-Temporal Data" *Advances in GIS Research* (Proceedings 1):489-504. 1994;

172. Data Warehouse - Estudo de Caso - SeguradoraPerez H.A.d.M. et al. 1998;

173. Portier M.M.M. Definition of CIGALES: A geographical Information System Query Language.[Unpublished] 1996;

174. A Brief History of Decision Support Systems. 1999;

175. Rao F. et al. "Spatial hierarchy and OLAP-favored search in spatial data warehouse". *In:Proceedings of the 6th ACM international workshop on Data warehousing and OLAP* pp.48-55. New Orleans, Louisiana, USA. 2003

176. Rauber A., Tomsich P. & Riedel H. "Integrating geo-spatial data into OLAP systems using a set-based quad-tree representation". *In:4th IEEE / IFIP International Conference on Information Technology for BALANCED AUTOMATION SYSTEMS in Production and Transportation* Berlin,Germany. 2000

177. Renolen A. Temporal Maps and Temporal Geographical Information Systems(Review of Rsearch).[Unpublished] 1996;

178. Rifaieh R. & Benharkat N.A. "Query-based data warehousing tool". *In:Proceedings of the 5th ACM international workshop on Data Warehousing and OLAP* 2002

179. Rivest S., Bedard Y. & Marchand P. "Towards better support for spatial decision-making: Defining the characteristics of Spatial On-Line Analytical Processing (SOLAP)" *GEOMATICA* (December) 2001;

180. Rizzi S. & Golfarelli M. "WanD: A CASE Tool for Data Warehouse Design". *In:Proceedings 17th International Conference on Data Engineering (ICDE 2001)* pp.7-9. Heidelberg, Germany. 2001

181. Rodrigues L.H. "The Awesome Power of JAVA Beans" Greenwich, CT: Mulvaney C.Manning Publications Co.; 1998;

182. Data Warehouse - Estudo de Caso - SupermercadoRomero A., Bernardino F. & Martinho L. 1998;

183. Sakaguchi T. A Review of the Data Warehousing Literature.[Unpublished] 1996;

184. Samet H. "The Design and Analysis of Spatial Data Structures" New York: Harrison MA and Addison-Wesley.Addison-Wesley; 1990;

185. Santos R.J.L. DATA WAREHOUSE: ABORDANDO O USO DO PARADIGMA ORIENTADO A OBJETO NO SEU DESENVOLVIMENTO.[Unpublished] 1998;

186. Data Warehouse - Estudo de Caso - Agencia de ViagensSantos R.R.C., Telles R.F.M. & Santos R.J.L. 1998;

187. Sarawagi S., Agrawal R. & Gupta A. Modeling Multidimensional Databases. 1999b;

188. Sarawagi S., Agrawal R. & Gupta A. On Computing the Data Cube. 1999a;

189. Seibold B. Why Data Warehouse Projects Fail. INSIGHT: PHARMACEUTICAL INFORMATION TECHNOLOGY. 1997;

190. Seshadri P., Livny M. & Ramakrishna R.S. "The Case for Enhanced Abstract Data Types". *In:VLDB'97, Proceedings of 23rd International Conference on Very Large Data Bases* pp.66-75. Athens, Greece. 1997

191. Shekhar S., Lu C.T. & Zhang P. "Map Cube: A Visualization Tool for Spatial Data Warehouse".Miller HJ and Han J. Geographic Data Mining and Knowledge Discovery. 2001;

192. Silva E.C.S.d. & Campos M.L.M. "Integracao de Sistemas de Informacao Geográfica e Ferramentas OLAP". *In:GIS Brasil 98* 1998

193. Skejellaug B. A Formal Comparison of ADT-based Dimensional Query Languages. University of OSLO - Department of Informatics. 1999; **276**.

194. Soares V.J.d.A., Modelagem Incremental no Ambiente de Data Warehouse, MSc, UNIVERSIDADE FEDERAL DO RIO DE JANEIRO - INFORMÁTICA, Rio de Janeiro - RJ - Brazil, 1998;

195. Stefanovic N. Design and implementation of on-line analytical processing.[Unpublished] 1997b;

196. Stefanovic N., Design and Implementation of On-Line Analytical Processing (OLAP) of Spatial Data, M.Sc., Simon Fraser University, 1997a;

197. Stefanovic N., Han J. & Koperski K. "Object-Based Selective Materialization for EfficientImplementation of Spatial Data Cubes '" *IEEE Transactions on Knowledge and Data Engineering* 12(6) 2000;

198. Stonebraker M. "Inclusion of New Types in Relational Data Base Systems". *In:Proceedings of the Second /EEE Conference on Data Engineering* pp.262-269. 1986

199. Dimensional Data Modeling. 2001;

200. Tomlinson R.F. "Current and potential uses of geographical information systems" *International Journal of Geographical Information Systems* 1:203-18. 1987;

201. Trillium & Harte-Hanks Data Technologies. The Trillium Software System.[Unpublished] 1999;

202. Vassiliadis P. et al. "A Framework for the Design of ETL Scenarios". *In:Advanced Information Systems Engineering, 15th International Conference, CAiSE 2003* 2681. pp.520-535. Klagenfurt, Austria. 2003

203. Vassiliadis P., Simitsis A. & Skiadopoulos S. "Conceptual modeling for ETL processes". *In:Proceedings of the 5th ACM international workshop on Data Warehousing and OLAP* pp.14-21. McLean, Virginia, USA. 2002

204. Vassiliadis P. et al. "ARKTOS: A Tool For Data Cleaning and Transformation in Data Warehouse Environments" *IEEE Data Engineering Bulletin* 23(4):42-7. 2000;

205. Walchowicz M. & Healey R.G. "Towards temporality in GIS".Inovations in GIS - selected papers from the First National Conference on GIS Research UK. I. pp.105-115. UK. 1994

206. Wang B. et al. "DB integration: Efficient OLAP operations for spatial data using peano trees". *In:Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery* pp.28-34. San Diego, CA, USA. 2003

207. Wash R. New Version of Microsoft MapPoint Revolutionizes Business Analysis. PressPass - Microsoft.(March,22). 2000;

208. Worboys M.F. "Object-Oriented approaches to geo-referenced information" *International Journal of Geographical Information Systems* 8(4):385-99. 1994;

209. Worboys M.F. et al. "Computational support for spatial information handling: models and algorithms".Worboys MF. Innovations in GIS. 6. London. Taylor & Francis. 1994;

210. Wu M.-C. & Buchmann A.P. "Research Issues in Data Warehousing". *In:BTW 1997* pp.61-82. 1997

211. Yen J. "Fuzzy Logic-A Modern Perspective" *IEEE Transactions on Knowledge and Data Engineering* 11(1):165 1999;

212. Zadeh L. "Fuzzy Logic" *IEEE Computer* 1988;

213. Zadeh L.A. "Fuzzy Sets" *Information and Control* 8(8):338-53. 1965;

214. Zhang.L., Li Y., Rao F., Yu X.L., Chen Y. & Liu D. Enabling Spatial in Business Intelligence System. 2004;

215. Zhou X. & Prasher S. "Multiresolution Amalgamation: Dynamic Spatial Data Cube Generation". *In:The Fifteenth Australasian Database Conference* 27. Dunedin, New Zealand. 2004

216. Zimmermann H.C. "Fuzzy Set Theory and its Applications" 2nd ed. Massachusetts: AnonymousAcademic Publishers; 1990;

217. Zornes A. A Taxonomy of Corporate Data Warehouses. I. T. News & Features. 2000;