Pontifícia Universidade Católica
do Rio de Janeiro

Sergio Ricardo Batuli Maynoldi Ortiga

DCD Tool:

Um conjunto de ferramentas para descoberta e triplificação de cubos de dados estatísticos

Dissertação de Mestrado

Dissertação apresentada como requisito parcial para obtenção do título de Mestre pelo Programa de Pós-Graduação em Informática da PUC-Rio.

Orientador: Prof. Marco Antonio Casanova

Rio de Janeiro

Setembro de 2013

Sergio Ricardo Batuli Maynoldi Ortiga

DCD Tool:
Um conjunto de ferramentas para descoberta e triplificação de cubos de dados estatísticos

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre pelo Programa de Pós-graduação em Informática do Departamento de Informática do Centro Técnico e Científico da PUC-Rio. Aprovada pela Comissão Examinadora abaixo assinada.

Prof. Marco Antonio Casanova
Orientador
Departamento de Informática – PUC-Rio

Prof. Antonio Luz Furtado
Departamento de Informática – PUC-Rio

Prof. Giseli Rabello Lopes
Departamento de Informática – PUC-Rio

Prof. Luiz André P. Paes Leme
Departamento de Informática – UFF

Prof. José Eugenio Leal
Coordenador Setorial do Centro
Técnico Científico – PUC-Rio

Rio de Janeiro, 06 de Setembro de 2013

Sergio Ricardo Batuli Maynoldi Ortiga

Sergio Ricardo Batuli Maynoldi Ortiga graduou-se em Informática pela PUC-RIO em 1991. Desde então trabalhou como consultor em empresas como IBM, Bradesco Seguros, Accenture, Embratel, entre outras. Atualmente atua como consultor na área de Banco de Dados na Fundação Getúlio Vargas, estando envolvido em diversos projetos do IBRE (Instituto Brasileiro de Economia). Possui interesse acadêmico e profissional nas áreas ligadas à Web Semântica, Linked Data, Data Warehouse e Big Data.

Ficha Catalográfica

Ortiga, Sergio Ricardo Batuli Maynoldi

DCD Tool: um conjunto de ferramentas para descoberta e triplificação de cubos de dados estatístcos/ Sergio Ricardo Batuli Maynoldi Ortiga; orientador: Marco Antonio Casanova. – Rio de Janeiro PUC, Departamento de Informática, 2013.

225 f. : il. (color.) ; 30 cm

1. Dissertação (mestrado)–Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2013.

Inclui bibliografia

1. Informática – Teses. 2. Web semântica. 3. Linked data. 4. Triplificação. 5. RDF. 6. Data cube vocabulary. 7. Modelos Dimensionais. I. Casanova, Marco Antonio. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD:004

Dedico este trabalho ao meu pai, Osni Ortiga Filho, *in memoriam*, à minha mãe Ana Lucia Batuli Ortiga e ao meu filho Daniel Muniz Maynoldi Ortiga.

Agradecimentos

Agradeço a todas as pessoas que contribuíram de alguma forma para a minha formação, em especial a minha família e amigos. Dentre eles, em particular, gostaria de agradecer ao Lorenzo Ridolfi, Adriano Branco, Márcia Pesce, Sofia Manso, Ximena Cabrera, Lívia Ruback, além da minha irmã Luciana pelo apoio e incentivo que foram fundamentais para atingir este objetivo. Também gostaria de deixar registrado um agradecimento muito especial à Professora Giseli Rabello Lopes.

À PUC-Rio, pelos auxílios concedidos, sem os quais este trabalho não poderia ter sido realizado.

Outra pessoa a quem muito devo e que foi um grande amigo e colega a quem gostaria de dedicar este trabalho é o Percy Sallas. Também serei eternamente grato ao João Luiz, meu chefe na FGV, por todo apoio que me deu.

Por fim, gostaria de fazer uma dedicatória a uma pessoa muito especial. Nos últimos três anos o professor Marco Antonio Casanova foi muito mais do que um orientador, um mentor ou um amigo. Não encontro palavras adequadas para expressar toda a gratidão que tenho por ele e por tudo que ele fez e representou na minha vida neste período. Em particular agradeço a ele por toda a paciência que sempre teve comigo. Por acreditar em mim quando nem mesmo eu acreditava mais. Por todas as oportunidades que me conceceu sem que eu, honestamente, tivesse feito o suficiente para merecê-las. O Casa é um destes seres humanos especiais cujas ações refletem os melhores significados que podemos extrair da palavra “humanidade”. Muito mais do que um pesquisador e um professor brilhante, eu destaco sua humildade e sua humanidade para conosco. Obrigado por tudo Casa.

Resumo

Ortiga, Sergio Ricardo Batuli Maynoldi; Casanova, Marco Antonio **DCD Tool: Um conjunto de ferramentas para descoberta e triplificação de cubos de dados estatísticos.** Rio de Janeiro, 2012. 225p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

A produção de indicadores sociais e sua disponibilização na Web é uma importante iniciativa de democratização e transparência que os governos em todo mundo vêm realizando nas últimas duas décadas. No Brasil diversas instituições governamentais ou ligadas ao governo publicam indicadores relevantes para acompanhamento do desempenho do governo nas áreas de saúde, educação, meio ambiente entre outras. O acesso, a consulta e a correlação destes dados demanda grande esforço, principalmente, em um cenário que envolve diferentes organizações. Assim, o desenvolvimento de ferramentas com foco na integração e disponibilização das informações de tais bases, torna-se um esforço relevante. Outro aspecto que se destaca no caso particular do Brasil é a dificuldade em se identificar dados estatísticos dentre outros tipos de dados armazenados no mesmo banco de dados. Esta dissertação propõe um arcabouço de software que cobre a identificação das bases de dados estatísticas no banco de dados de origem e o enriquecimento de seus metadados utilizando ontologias padronizadas pelo W3C, como base para o processo de triplificação.

Palavras-Chave

Web Semântica; Dados Estatísticos; Linked Data; Triplificação; RDF; Data Cube Vocabulary; R2RML; Modelagem Dimensional.

Abstract

Ortiga, Sergio Ricardo Batuli Maynoldi; Casanova, Marco Antonio (Advisor). **DCD Tool: A toolkit for the discovery and triplification of statistical data cubes.** Rio de Janeiro, 2013. 225p. MSc. Dissertation – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

The production of social indicators and their availability on the Web is an important initiative for the democratization and transparency that governments have been doing in the last two decades. In Brazil, several government or government-linked institutions publish relevant indicators to help assess the government performance in the areas of health, education, environment and others. The access, query and correlation of these data demand substantial effort, especially in a scenario involving different organizations. Thus, the development of tools, with a focus on the integration and availability of information stored in such bases, becomes a significant effort. Another aspect that requires attention, in the case of Brazil, is the difficulty in identifying statistical databases among others type of data that share the same database. This dissertation proposes a software framework which covers the identification of statistical data in the database of origin and the enrichment of their metadata using W3C standardized ontologies, as a basis for the triplification process.

Keywords

Semantic Web; Statistical Data; Linked Data; Triplification; RDF; Data Cube Vocabulary; R2RML; Dimensional Modeling.

Sumário

Lista de Figuras

Lista de Quadros

Lista de Tabelas

1.
Introdução

1.1
Linked Data

De 01 de Maio de 2007 quando sua primeira versão apareceu contendo apenas 12 datasets, até hoje, a nuvem LOD (Linked Open Data [1]) mantida por Richard Cyganiak[2] observou um significativo crescimento, tendo atingido a marca de 295 datasets em 19 de setembro de 2011.

O Linked Data[3] estabeleceu-se como o padrão da Web de dados. Empresas, governos e sociedade publicam dados nesse padrão, a fim de permitir a sua reutilização posterior por quaisquer interessados. No Brasil, estimulados por iniciativas de open-government[4] e investimentos em eGovernment[5] representantes das diversas esferas de poder do Brasil, federal, estaduais e municipais, vêm publicando uma grande quantidade de dados, com o objetivo de assegurar acesso, transparência e participação social no governo, bem como estimular a luta contra a corrupção e o desenvolvimento de novas tecnologias. Entretanto, a maioria destes dados estão publicados ainda no padrão da Web de documentos.

Um expressivo volume de indicadores econômicos, sociais e ambientais produzidos por órgãos públicos ou instituições ligadas ao governo, residem hoje em Data Warehouses ou banco de dados estatísticos como é o caso do

[1] http://richard.cyganiak.de/2007/10/lod

[2] http://richard.cyganiak.de/#/me

[3] http://linkeddata.org

[4] http://www.acessoainformacao.gov.br/acessoainformacaogov/acesso-informacao-mundo/governo-aberto.asp

[5] http://www.governoeletronico.gov.br/

FGVDADOS[6], produzido pela FGV[7], o SIDRA[8], o PAM[9] e o PPM[10], produzidos pelo IBGE, os indicadores estaduais de ciência e tecnologia do MCT[11] (Ministério da Ciência e Tecnologia), o ALADI[12], o CEPAL[13], o DIEESE[14], o IPEADATA[15], entre outros[16]

Assim, tornar acessíveis esses diversos indicadores armazenados em bancos de dados mantidos por instituições como a FGV, o IBGE, o Ministério da Ciência e Tecnologia, o Instituto de Economia Agrícola, a Confederação Nacional do Comércio, o Ministério do Desenvolvimento, Indústria e Comércio Exterior, o Ministério da Agricultura, Pecuária e Abastecimento, a Receita Federal e muitos outros, seria uma inestimável contribuição não apenas com o governo mas com toda a sociedade brasileira.

1.2
Motivação

Informações estatísticas constituem-se em uma das mais importantes fontes de informações utilizadas por pessoas e instituições no exercício, planejamento ou aferição e controle de suas atividades (Pesce, 2011).

Seja no plano empresarial, científico ou governamental, os dados estatísticos representam observações ou medições utilizadas para identificar desempenho, aferir sucesso ou fracasso ou identificar pontos fortes e fracos das iniciativas ou pesquisas realizadas em cada um desses planos. Particularmente, no domínio empresarial, dados estatísticos sobre as vendas de seus produtos,

[6] http://portalibre.fgv.br/main.jsp?lumChannelId=402880811D8E34B9011D92C493F131B2

[7] http://portal.fgv.br/

[8] http://www.sidra.ibge.gov.br/

[9] http://www.ibge.gov.br/home/estatistica/economia/pam/

[10] http://www.ibge.gov.br/home/estatistica/economia/ppm/2010/

[11] http://www.mcti.gov.br/

[12] http://www.aladi.org/nsfWeb/sitioport/

[13] http://www.eclac.org/

[14] http://www.dieese.org.br/

[15] http://www.ipeadata.gov.br/

[16] http://www2.camara.leg.br/documentos-e-pesquisa/biblarq/indicadores-economicos-e-sociais

sobre a evolução do mercado ou sobre indicadores econômicos são uma contribuição crucial para decisões estratégicas de gestão (Salas, et al., 2012). No plano governamental, temos uma farta oferta de bases de dados disponibilizadas por entidades ligadas ao governo brasileiro contendo relevantes informações sobre a evolução dos indicadores sociais brasileiros que ainda não estão integradas ao LOD.

Um desses projetos de pesquisa culminou, em 2011, na publicação da dissertação de mestrado de Percy Salas (Salas, 2011). A ferramenta desenvolvida, a StdTrip, inspirou a elaboração de uma segunda ferramenta, a OLAP2DataCube (Salas, et al., 2012), e, em seguida, a proposta do desenvolvimento de um framework, o OLAP2DataCube Catalog On Demand (Ruback, et al., 2013), cujo objetivo seria oferecer uma solução para integrar à nuvem LOD, os dados residentes em Data Warehouses localizados em bancos de dados governamentais.

O OLAP2DataCube Catalog On Demand propõe-se a oferecer uma solução baseada na abordagem das consultas federadas. Diversos elementos compõem a solução. O framework é estruturado em três camadas, descritas em (Ruback, et al., 2013):

1. Camada de Aplicação (*Client Application*), camada voltada para suportar a interação entre usuários (ou aplicações usuárias) e o framework;
2. Camada de Mediação, que é responsável por realizar as operações de mediação entre a camada de aplicação e o Catálogo (*Catalog*), e da camada de aplicação com o wrapper. Nesta camada são processadas as solicitações enviadas pela camada de aplicação, realizam-se consultas ao catálogo e encaminham-se solicitações ao wrapper para gerar respostas que atendam às solicitações realizadas pela aplicação cliente;
3. Camada do Wrapper, é a camada responsável pelo acesso "físico" aos banco de dados onde as informações estão armazenadas e gerar as triplas RDF em resposta às solicitações encaminhadas pela camada de mediação.

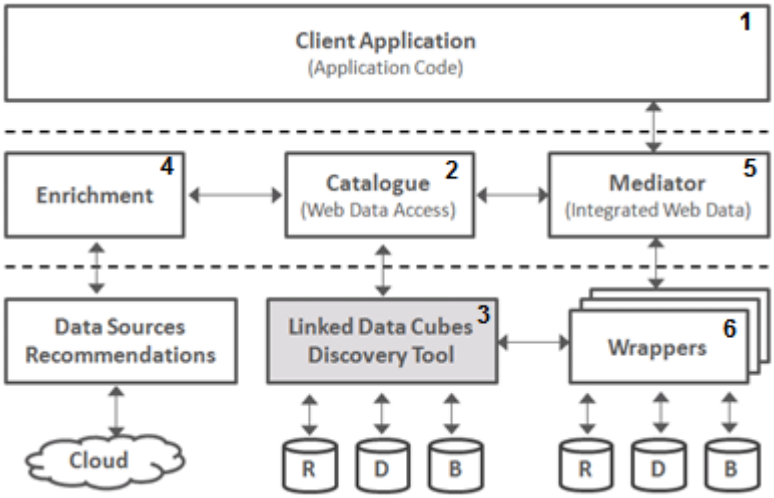


Figura 1:OLAP2DataCube Catalog on Demand.

A camada superior do framework representa onde estão as aplicações cliente (elemento 1 na figura 1) que irão enviar as solicitações de consulta através dos vocabulários padrões utilizados para construção do catálogo RDF (elemento 2) apresentado na camada intermediária. O catálogo é construído a partir da ferramenta tema deste trabalho, a Linked Data Cube Discovery Tool (elemento 3), representada na camada inferior da figura e enriquecido com informações externas fornecidas pela ferramenta de enriquecimento (elemento 4) apresentada na camada intermediária. Todo o acesso ao catálogo, bem como a recuperação dos dados triplificados a partir das bases de origem será realizada pelo mediador (elemento 5), o único dos elementos que interage diretamente com as aplicações cliente. O mediador, ao receber uma solicitação de consulta, busca no catálogo as informações que lhe permitirão construir as consultas que serão submetidas às bases de dados de origem através dos wrappers (elemento 6) apresentados na camada inferior da figura.

O enfoque do presente trabalho concentra-se sobre o desenvolvimento de uma ferramenta focada na descoberta de *data cubes* e na produção e publicação de suas estruturas em triplas.

1.3 Objetivo

Nesta dissertação propomos um arcabouço de software que possibilita a identificação de modelos dimensionais, onde informações estatísticas são armazenadas, através de heurísticas baseadas em “melhores práticas” de modelagem dimensional. O arcabouço permite que os metadados destes modelos sejam recuperados a partir dos catálogos dos bancos de dados onde

eles estão armazenados sejam enriquecidos com informações fornecidas por um usuário e, finalmente, triplas sejam geradas e carregadas no componente *Catalog* do OLAP2DataCube Catalog on Demand.

1.4 Contribuições

O presenta trabalho pretende oferecer como contribuições resultantes do seu desenvolvimento os seguintes itens:

- Uma ferramenta capaz de identificar modelos dimensionais com seus diversos componentes. Tanto modelos "estrela[17]" como "em floco de neve[18]" são igualmente tratados na abordagem proposta.
- Um conjunto de heurísticas baseadas em "melhores práticas" de modelagem dimensional e um algoritmo através dos quais modelos dimensionais podem ser identificados; as heurísticas servem de base para o desenvolvimento da ferramenta supracitada.
- Um catálogo genérico de metadados de bancos relacionais que pode ser utilizado para migrações de modelos ou que pode servir de base para geração de triplas RDF para criação de catálogos de metadados de bases de dados relacionais em gerenciadores de dados RDF, como por exemplo o Virtuoso.
- Uma interface que permite o enriquecimento de metadados de esquemas multidimensionais recuperados de bases relacionais.
- Uma ferramenta para geração de triplas utilizando vocabulários padronizados como R2RML, Data Cube Vocabulary e SKOS.

1.5 Trabalhos Relacionados

O presente trabalho pode ser classificado dentre aqueles que propõem ferramentas voltadas para realizar mapeamento entre RDB e RDF.

[17] http://pic.dhe.ibm.com/infocenter/dataarch/v8r1/topic/com.ibm.datatools.dimensional.ui.doc/topics/c_dm_star_schemas.html

[18] http://pic.dhe.ibm.com/infocenter/dataarch/v8r1/index.jsp?topic=%2Fcom.ibm.datatools.dimensional.ui.doc%2Ftopics%2Fc_dm_snowflake_schemas.html

Com o crescimento da nuvem LOD, o mapeamento de dados relacionais para RDF tornou-se cada vez mais uma necessidade. Além disso, RDF tem sido usado também para integração de dados.

Basicamente temos duas maneiras de implementar RDB2RDF. Da primeira maneira, convertem-se os dados relacionais em triplas RDFs que ficam fisicamente armazenadas em algum repositório. Estas triplas, por sua vez, podem conter outras triplas oriundas de diferentes fontes, de forma que temos todo esse conjunto de triplas devidamente integradas e vinculadas através do RDF. Existem algumas desvantagens com esta abordagem e as principais são o espaço de armazenamento necessário para guardar todas as triplas geradas a partir dos dados originais, o tempo necessário para a realização deste processo de ETL e a dificuldade de manter a integridade e o sincronismo entre os dados armazenados no banco de dados de origem e as suas correspondentes triplas RDF.

A segunda alternativa é não materializar os dados relacionais em triplas RDF, deixando-os em seu lugar de origem. Neste caso, cria-se um mapeamento, usando triplas RDF baseadas em uma ontologia de domínio orientada para este fim (mapear bancos relacionais em RDF), de maneira que, através de consultas on-the-fly SPARQL, seja possível recuperar os dados em seu banco de origem e responder à solicitação que foi realizada. A consulta SPARQL é traduzida para uma SQL, que, por sua vez, é realizada nos bancos de dados onde as informações procuradas estão fisicamente armazenadas. Esta abordagem resolve parte do problema. Porque, na verdade, quando pensamos em Linked Data, os dados conectados podem estar fisicamente distribuídos por diferentes bancos de dados espalhados pela Web. Desta forma, uma abordagem adequada deveria permitir consultas federadas SPARQL sobre diferentes SPARQL endpoints.

1.5.1
Ferramentas de conversão de Bancos de Dados relacionais para RDF

Uma abordagem experimental que atinge certo nível de integração de dados é apresentada no trabalho de (Hartig, et al., 2009) e em SQUIN[19] que permitem a realização de consultas sobre a Web de Dados como se fosse um único bando de dados. Outras ferramentas produzidas com este objetivo são:

D2RQ[20] (Seaborne, et al., 2004) consiste em uma linguagem de mapeamento entre esquemas de bancos relacionais e ontologias RDFS/OWL. A plataforma D2RQ cria uma visão RDF do banco relacional que pode ser acessado através de Jena (Carrol, et al., 2004) e da linguagem de consulta SPARQL. Além disso, usando o D2R Server o banco relacional pode ser acessado via Web via protocolo SPARQL e Linked Data. A primeira versão do DBpedia em 2007 foi feita usando o D2R Server.

RDBtoOnto[21] (Cerbah, 2008): agindo como uma ferramenta automática de ETL, a partir de configurações previamente realizadas, o RDBtoOnto gera, a partir de um banco de dado relacional uma ontologia povoada e finamente ajustada em RDFS/OWL Seu diferencial é justamente este "ajuste fino" obtido a partir da exploração do esquema do banco e da identificação de padrões estruturados escondidos nos dados.

Triplify[22] (Auer, et al., 2009): trata-se de um plugin leve que supre a necessidade de uma solução de mapeamento simples, usando SQL como linguagem de mapeamento, transformando o resultado da consulta em triplas RDF e Linked Data. Não há suporte para SPARQL. Foi codificado em PHP puro mas foi adaptado para várias aplicações populares da Web como Wordpress, Joomla, osCommerce e etc.

Virtuoso RDF View[23] (Erling, et al., 2009): mapeia dados relacionais em RDF e permite a execução de consultas SPARQL sobre o banco relacional

[19] http://squin.sourceforge.net/

[20] http://d2rq.org/

[21] http://www.tao-project.eu/researchanddevelopment/demosanddownloads/RDBToOnto.html

[22] http://triplify.org/

[23] http://docs.openlinksw.com/virtuoso/rdfsparqlintegrationmiddleware.html#rdfviews

simultaneamente com um repositório RDF local, viabilizando assim uma integração entre dados relacionais e RDF.

1.5.2 StdTrip

A StdTrip (Salas, 2011) foi construída para suportar o processo de representação de esquemas de bancos de dados relacionais em classes e propriedades do vocabulário RDF.

Um dos pontos de destaque dessa ferramenta é o reuso de vocabulários padronizados a fim de garantir a interoperabilidade dentro do espaço da *Linked Open Data (*LOD).

O processo de triplificação da ferramenta, ilustrado visto na figura 2, segue a seguinte ordem: conversão, alinhamento, seleção, inclusão, conclusão e saída.

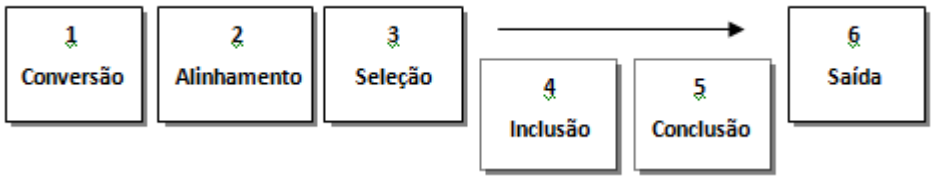


Figura 2: Arquitetura da Ferramenta StdTrid (Salas, 2011)

1.5.3 Olap2DataCube

O OLAP2DataCube (Salas, et al., 2012) é uma ferramenta que traduz para triplas RDF cubos de dados e o seu respectivo conteúdo, a partir de bancos de dados relacionais.

A ferramenta foi desenvolvida como um plug-in para o OntoWiki (Auer, et al., 2006), suportando a criação colaborativa, manutenção e publicação de bases de conhecimento no formato RDF. A ferramenta também oferece várias interfaces para publicar e consultar dados.

A ferramenta trabalha exclusivamente com bancos relacionais e, em particular, com modelos em estrela (Thomsen, 1997). O OLAP2DataCube utiliza estes modelos como entrada do seu processo de conversão para o

formato de triplas RDF. O vocabulário utilizado para a geração das triplas é o DataCubeVocabulary.

O processo é realizado em três fases:

1. Extração dos metadados do banco de dados e categorização das tabelas;
2. Definição do cubo
3. Mapeamento para RDF.

No próximo capítulo apresentaremos o *framework OLAP2DataCube Catalog On Demand*, detalhando sua arquitetura, camadas e módulos, além da forma como estes se comunicam. Também serão detalhados os processo de conversão de dados por meio do *framework OLAP2DataCube Catalog On Demand*, bem como os processos de geração do catálogo e de enriquecimento dos dados das dimensões.

1.6 Organização do trabalho

O restante da dissertação foi estruturado conforme descrito a seguir.

O Capítulos 2 e 3 apresentam conceitos básicos sobre os quais o trabalho está fundamentado. O capítulo 2 foca o modelo dimensional e sua materialização em cubo de dados, e o capítulo 3 foca na triplificação destes Cubos.

O Capítulo 4 descreve o *framework OLAP2DataCube Catalog On Demand*, detalhando sua arquitetura, camadas e processo.

O Capítulo 5 descreve a ferramenta DCD Tool, abordando os três módulos que a compõem de forma sucinta.

O Capítulo 6 detalha o processo de investigação e descoberta de modelos dimensionais nos bancos de dados relacionais.

O Capítulo 7 apresenta as conclusões finais e propõe trabalhos futuros.

1.7 Resumo

Neste capítulo introduzimos os temas que serão abordados nesta dissertação, descrevendo os principais motivos e objetivos que nos levaram a

desenvolvê-la. Listamos as contribuições esperadas, bem como apresentamos a forma como o trabalho está organizado.

2.
Modelos Dimensionais e Cubos de Dados

2.1
Modelos Dimensionais

Desde a década de 1960, empresas e universidades já pensavam em criar grandes bancos de dados para suportar operações de negócios. Nesta década, um projeto de pesquisa realizado em conjunto pela *General Mills* e pela *Darmouth University* introduziu os termos "Fato" e "Dimensão" que atualmente são utilizados para designar os dois elementos centrais de um modelo dimensional.

Modelos dimensionais são construídos para permitir a um especialista em determinada área de negócio, avaliar o registro histórico de eventos relevantes para aquele negócio a fim de identificar padrões que permitam diagnosticar tendências de consumo, comportamento tipificado, fraudes e etc.

A estrutura de um modelo dimensional representa um relacionamento *n*-ário entre *n* entidades. As tabelas que representam os domínios deste relacionamento são chamadas "Dimensões" e o relacionamento entre elas é chamado "Fato". Doravante iremos utilizar o termo "F-TAB" para designar as tabelas onde estão armazenados os "fatos" e "D-TAB" para aquelas que representam as dimensões.

O conteúdo de uma F-TAB representa o registro histórico (temporal) de eventos (fatos) ocorridos no passado e sobre os quais desejamos realizar nossas análises. Desta forma, este conjunto de eventos registrados nas F-TABs estão, todos, relacionados com determinado "assunto". Portanto, um modelo dimensional é orientado a assuntos. A nomenclatura padrão usada é "Tabela Fato" embora existam menções à "Tabela de Fato". Em português, no entanto, "Tabela de Fato" soa estranho, parece que estamos falando que aquela tabela específica, é uma tabela "de verdade" (de fato) ao invés de falarmos que ela é uma tabela cujo conteúdo representam fatos, isto é, literalmente a "Tabela dos

Fatos", aquela onde estão armazenados os fatos que serão objeto de análise em um particular modelo.

A estrutura de uma F-TAB é definida por colunas que representam as chaves estrangeiras das suas dimensões (e constituem a sua chave primária composta) e colunas que representam as observações nela registradas. Nem todas F-TABs possuem observações, essas são conhecidas como "*Factless Fact Table*". Nessas tabelas as observações não estão declaradas mas são obtidas a partir de contagens realizadas sobre as suas linhas. Cada linha em uma F-TAB representa individualmente o registro de um evento ocorrido no tempo, ou seja, de um fato. Usamos as observações explicitamente declaradas através de colunas ou aquelas que são inerentes (contagens das linhas, por exemplo) à tabela para quantificar estes fatos.

Os eventos registrados em uma F-TAB possuem um conjunto de características que servem para qualificá-los: são as "dimensões" do modelo.

Cada D-TAB representa um objeto de negócio relevante ao contexto que está sendo analisado. Mas como podemos identificar os candidatos à D-TAB em um modelo? Temos que ter em mente que o foco do modelo dimensional são os fatos que ele registra. Assim, uma vez que estamos tratando com fatos e que sabemos que o fato representa o registro histórico de determinado evento, existem algumas informações que naturalmente são necessárias sabermos para que o registro daquele evento seja acurado: O QUE ocorreu, com QUEM ocorreu, QUANDO ocorreu, ONDE ocorreu, PORQUE ocorreu e COMO ocorreu (Machado, 2006). Então, numa visão bastante prática, um modelo dimensional genérico provavelmente terá uma estrutura do seguinte tipo:

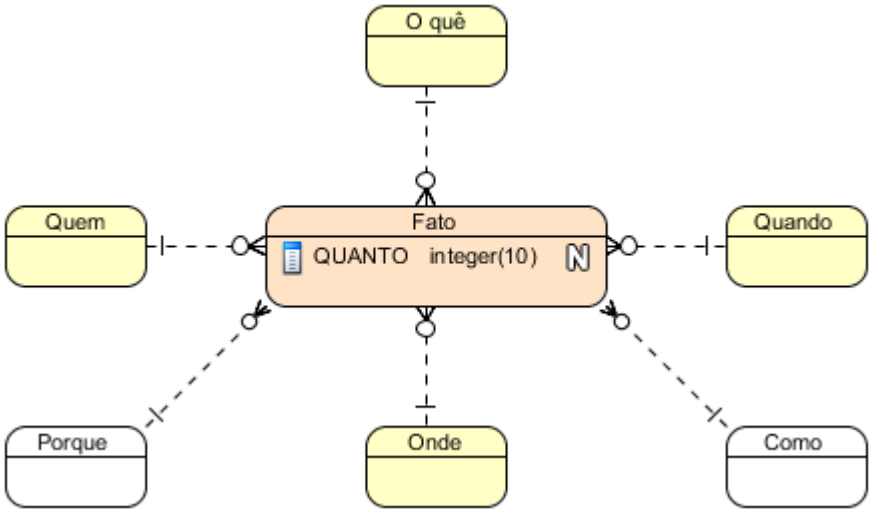


Figura 3: Modelo dimensional "genérico"

Na figura 3 destacamos em amarelo as principais dimensões que normalmente aparecem em qualquer modelo dimensional (Machado, 2006). Suponha um modelo dimensional que tenha por objetivo oferecer análises para a área de Marketing de uma empresa, a D-TAB “cliente” certamente se fará presente (Quem), bem como alguma D-TAB relacionada com os produtos ou serviços oferecidos pela empresa (O quê), a D-TAB tempo é obrigatória (Quando). Em geral, existem inúmeros qualificadores candidatos a dimensões quando analisamos determinado contexto de negócio. No entanto, um modelo dimensional que tivesse algumas dezenas ou mesmo centenas de dimensões, tornar-se-ia ilegível e difícil de ser compreendido. Por esta razão costumamos agrupar essas características “qualificadoras” em entidades que chamamos “Dimensões”. Portanto, as dimensões de um modelo dimensional são entidades ou tabelas, que reúnem um conjunto de características afins. Uma outra interpretação que podemos ter é que as Dimensões possuem um conjunto de propriedades que as define ou caracteriza e que são usadas para realizar a qualificação dos fatos registrados no modelo. Vejamos outro exemplo disso.

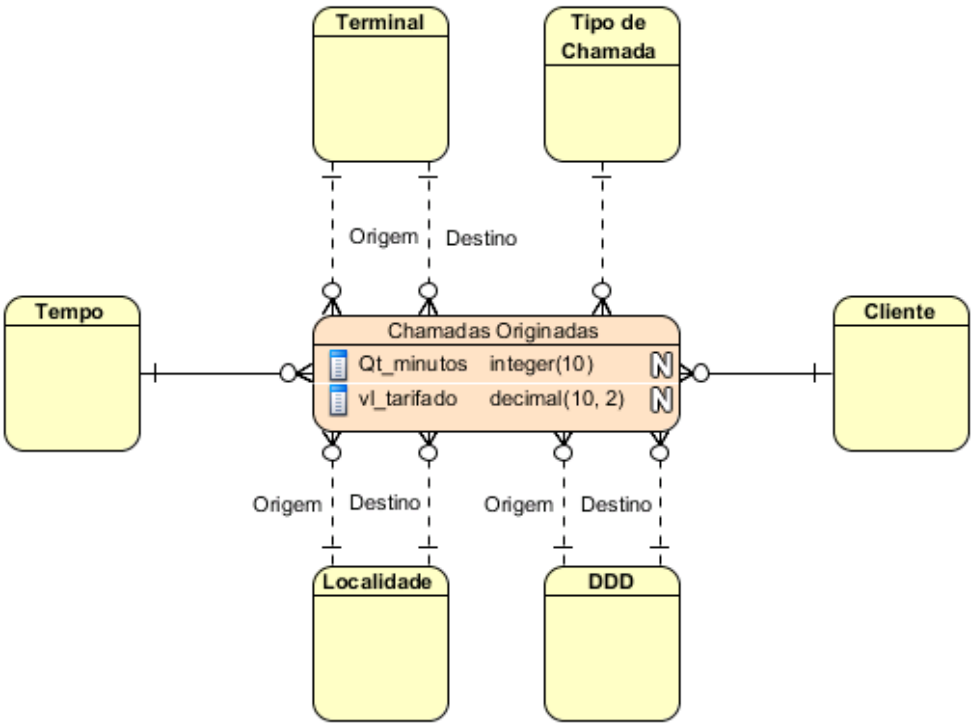


Figura 4: Exemplo de Modelo dimensional para Telecom

Analisar o tráfego de chamadas originadas (ou saintes), ou seja, as chamadas que são realizadas a partir dos terminais (telefones) que pertencem à certa operadora de telecomunicações, é uma necessidade de toda empresa de Telecom. O modelo acima é uma versão bastante enxuta e simplificada de um modelo que atende às necessidades básicas de análise de chamadas originadas.

O fato registrado neste modelo é uma chamada telefônica originada em algum terminal pertencente àquela operadora. Uma chamada telefônica possui como atributos quantitativos, por exemplo, a sua duração (quantidade de minutos) e o seu custo (valor tarifado). Neste caso, são registradas duas medidas explícitas: duração e custo e ainda podemos deduzir uma outra medida que é a quantidade de chamadas realizadas, simplesmente realizando uma contagem sobre as linhas da tabela, uma vez que cada uma delas representa uma chamada.

Com relação aos qualificadores deste modelo, temos: o número do terminal que originou a chamada, o número do terminal para o qual a chamada foi realizada, o DDD do terminal de origem e o DDD do terminal de destino, a localidade onde a chamada foi originada e a localidade de destino da chamada, o cliente que realizou a chamada e o tipo de chamada (local, DDD ou DDI) realizada.

Através deste modelo podemos avaliar o volume de chamadas realizadas por determinado cliente, podemos identificar se existe alguma localidade para a qual ele realiza chamadas preferencialmente e oferecer algum desconto ou vantagem quando ele realizar chamadas especificamente para esta localidade, podemos ainda, verificar para que regiões as chamadas originadas na rede da operadora têm maior destinação e elaborar planos que sejam vantajosos para os nossos clientes quando realizarem chamadas para tais regiões.

Mas podemos ir muito além disso. Suponha que a empresa em questão mude a sua política de tarifas e, em seguida, perceba um aumento ou uma redução significativa no volume do tipo de chamada cuja tarifa foi alterada, isso pode evidenciar que o ajuste de tarifas teve influência no comportamento do cliente. Se, no entanto, não houver mudança significativa no volume, isso poderá indicar que a questão da tarifa ou não foi percebida adequadamente ou então não foi significativa a ponto de alterar o comportamento dos clientes. No caso de uma redução de tarifa, para estimular, por exemplo, o uso dos serviços

de DDD, talvez a redução não tenha sido significativa ou, ao menos, percebida desta maneira pelos clientes. Por outro lado, se houve aumento da tarifa e ainda assim os clientes permaneceram fiéis à empresa, isso pode estar relacionado, por exemplo, com a qualidade superior do serviço de DDD oferecido pela empresa.

Convém salientar que o modelo aqui ilustrado foi proposto com base na experiência de campo do autor desta dissertação. Mas, ainda assim, é bem modesto em relação a uma situação real. No entanto, como pudemos perceber, mesmo um modelo tímido como este é capaz de alimentar um grande número de análises e projeções significantes não apenas para tomada de decisões, como também para avaliação de decisões tomadas previamente.

2.1.1
Características dos Modelos Dimensionais

Segundo Ralph Kimball, ao se projetar um modelo dimensional, algumas regras devem ser observadas (Ross, 2009), enquanto algumas práticas devem ser evitadas (Kimball, 2001). Abaixo apresentamos cada um destes conjuntos de regras:

REGRAS A SEREM OBSERVADAS
1.1 Carregar nas estrutruras dimensionais os dados no nível mais granular (ou detalhado) possível.
1.2 Construir as estruturas dimensionais em torno de processos de negócio.
1.3 Garantir que cada F-TAB tenha uma D-TAB tempo associada.
1.4 Garantir que todos os registros (linhas) numa F-TAB tenha o mesmo grão (nível de detalhe).
1.5 Solucionar os relacionamentos N:M na F-TAB. *
1.6 Solucionar os relacionamentos 1:N nas dimensões. **
1.7 Manter nas dimensões informações usadas como rótulos de relatórios ou domínios de valores utilizados para aplicar filtros.
1.8 Assegurar-se de que a chave primária das dimensões será uma chave artificial (Kimball, 1998).

1.9 Criar dimensões em conformidade *** para integrar os dados em toda a empresa (grandes corporações tendem a ter uma grande quantidade de data marts e um gigantesco data warehouse) (Kimball, 2011).
1.10 Continuamente deve-se balancear os requisitos e os objetivos para garantir a entrega de uma solução de DW/BI que seja bem recebida pelos usuários de negócio, capaz de suportar suas tomadas de decisão.

Tabela 1: Melhores Práticas em Modelagem Dimensional (Ross, 2009)

* existem casos especiais em que esta regra pode não ser aplicada. Maiores detalhes podem ser obtidos no artigo que detalha a construção de tabelas "ponte" (Kimball, 2012) como solução de determinadas situações especiais, que podemos encontrar em modelagem dimensional[24].

** outra regra que pode vir a ser "burlada" em casos específicos, como veremos mais adiante neste trabalho.

*** dimensões em conformidade são dimensões que são compartilhadas por mais de uma F-TAB.

REGRAS QUE DEVEM SER EVITADAS
2.1 Colocar atributos texto em uma F-TAB com a intenção de usá-los para realizar operações de agrupamento e filtragem
2.2 Limitar o uso de atributos descritivos detalhados em dimensões para economizar espaço
2.3 Quebrar em múltiplas dimensões as hierarquias e níveis hierárquicos.
2.4 Adiar ou "fugir" de soluções que tratem e lidem com dimensões que mudam vagarosa ou rapidamente (*Slowly Changing Dimensions e Rapidly Changing Dimensions*) (Kimball, 2013) (Kimball, 1996) (Kimball, 2005).
2.5 Usar "chaves inteligentes[25]" para estabelecer junções entre dimensões e F-TABs.

[24] http://www.kimballgroup.com/2012/02/01/design-tip-142-building-bridges/

[25] Chave é uma coluna usada para identificar univocamente um registro em uma tabela. Chaves "inteligentes" são colunas que além de atuarem como chave de identificação do

2.6 Adicionar uma D-TAB a uma F-TAB que possua grão diferente do seu.
2.7 Construir modelos dimensionais orientados a atender ou gerar um relatório específico.
2.8 Misturar registros de fatos com diferentes grãos na mesma F-TAB.
2.9 Manter o nível mais detalhado do modelo do Data Warehouse em um formato clássico de E-R.
2.10 Agregar F-TABs ou encolher as dimensões por questões de desempenho. A falta de performance pode ser resolvida com hardware e paralelismo.
2.11 Não garantir a conformidade das medidas em diferentes F-TABs. Ou seja, garantir a unicidade semântica de determinada medida, independente dela existir em mais de uma F-TAB. Assim, se tivermos uma medida chamada "receita" em diferentes F-TABs, seu significado deve ser o mesmo independente da sua localização física.
2.12 Não garantir a conformidade das dimensões no modelo. Uma D-TAB deve ser única no modelo. Assim não teremos diferentes versões da D-TAB cliente ainda que a empresa para a qual estejamos modelando tenha clientes pessoa física e pessoa jurídica e, entre esses, empresas de portes diferenciados ou que atuem em diferentes áreas ou segmentos de negócio. A D-TAB cliente será sempre única e será compartilhadas pelas diferentes F-TABs existentes no modelo.

Tabela 2: Práticas que devem ser evitadas em Modelo Dimensionais (Kimball, 2001)

Para identificarmos um modelo dimensional em um catálogo de banco de dados, necessitamos identificar quais as características que estes modelos possuem que nos permitam distingui-lo dos demais. A partir das recomendações acima, formulamos as seguintes assertivas, que tem por objetivo nos permitir identificar um modelo dimensional:

---

registro contém alguma informação sobre a instância que ela identifica, como por exemplo, o CPF ou a matrícula de um funcionário. Também são conhecidas como chaves “naturais”.

A. De 1.8 e 2.5, assumimos que as chaves artificiais a serem utilizadas nas dimensões deverão ser sequências numéricas simples;
B. De 2.11, assumimos que, se duas medidas coexistem em um mesmo ambiente dimensional, e possuem o mesmo nome, ainda que pertençam à diferentes F-TABs, elas representam a mesma informação e, portanto, tem o mesmo significado;
C. De 1.9 e 2.12 , assumimos a unicidade de uma D-TAB em um certo ambiente dimensional. Desta forma, equivalências entre dimensões só existirão quando estivermos comparando dimensões de diferentes ambientes dimensionais;
D. De 2.9, assumimos que apenas modelos organizados de forma dimensional serão tratados no presente trabalho, sendo excluído do nosso escopo todo e qualquer modelo E-R clássico, normalizado ou não;
E. As recomendações 1.5, 1.6 e 2.3 reforçam a construção de modelos dimensionais em formato estrela, ao mesmo tempo que desaconselham fortemente a utilização de modelos em floco de neve. Assim, deveríamos restringir o escopo do presente trabalho ao tratamento exclusivo de modelos no formato "estrela" (o qual será explicado a seguir). No entanto não o faremos pois modelos estrela são largamente utilizados na construção de Data Marts. No entanto para construção da primeira camada de dados de um Data Warehouse, é muito comum encontrarmos uma implementação em "floco de neve". Como estaremos lidando justamente com a implementação da primeira camada de um Data Warehouse, normalmente construída em um banco relacional, não poderemos excluir do nosso escopo o tratamento de modelos no formato "floco de neve";
F. Finalmente, sabendo-se que a F-TAB herda as chaves-primárias das suas dimensões, e, por ser uma prática consagrada em modelagem dimensional, assumimos que a chave primária de uma F-TAB é definida pela concatenação das chaves estrangeiras nela declaradas, oriundas das chaves primárias de suas dimensões;

Um ponto importante e fundamental que podemos observar a partir, principalmente, da assertiva F enunciada anteriormente é que uma F-TAB representa a implementação de um relacionamento n-ário entre as suas dimensões. Ressaltar este ponto é de suma importância para que se entenda o porquê de desprezarmos os relacionamentos 1:N e 1:1 na heurística proposta para identificação de modelos dimensionais.

2.1.2 Tipos de Modelos Dimensionais

Existem dois tipos de modelos dimensionais, os modelos em formato estrela, que correspondem aos modelos tratados até agora e defendidos por Ralph Kimball (Kimball, 2002) e os modelos em "floco de neve" (Moody, et al., 2000). Estes últimos podem ser resumidos como sendo um modelo estrela cujas hierarquias das dimensões foi normalizada, criando, desta forma, um outro conjunto de tabelas que se relacionam com as dimensões e que são chamadas de "Outrigger" ou dimensões secundárias (Machado, 2006).

O modelo estrela é assim chamado porque uma certa configuração dele com cinco tabelas dimensões e uma F-TAB lembra o formato de uma estrela de cinco pontas:

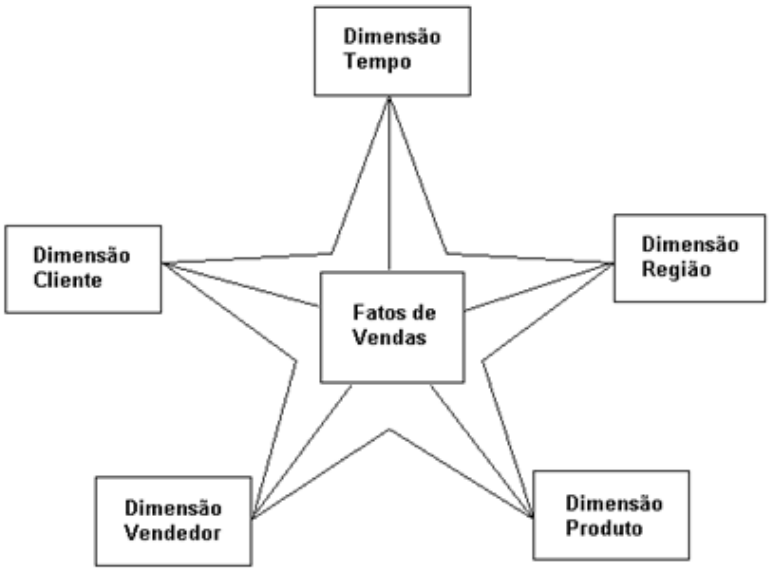


Figura 5: Modelo em formato "estrela"

No modelo estrela, todas as dimensões ocupam a mesma "órbita" em torno da F-TAB. Na figura 6, o círculo ali desenhado representa a órbita

ocupada pelas dimensões e as únicas entidades ou tabelas que existem são a Fato e as dimensões que compartilham esta única órbita.

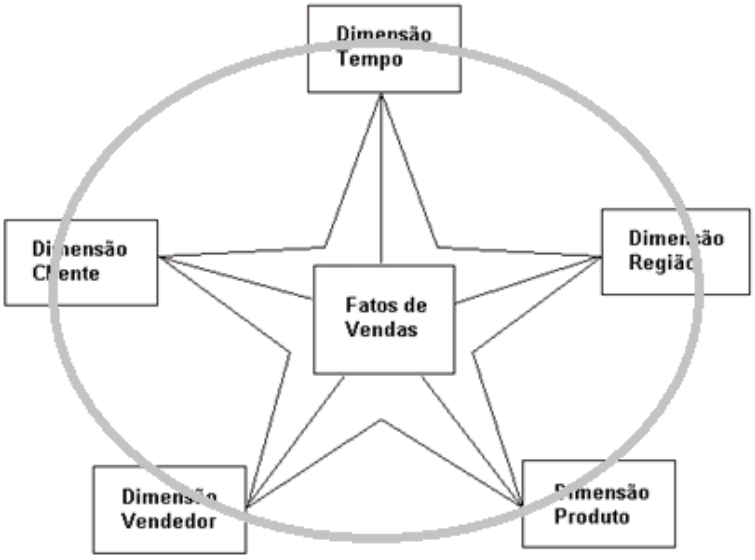


Figura 6: Ilustração das Dimensões orbitando uma Fato

À primeira vista isso pode não parecer nada demais. No entanto um observador mais atento certamente irá perceber que a D-TAB Região, por exemplo, pressupõe uma formação hierárquica, uma vez que as informações nela contidas de fato estão organizadas como uma hierarquia: país, região, unidade federativa, município, etc. A forma "canônica" de se retratar uma estrutura dessas seria:

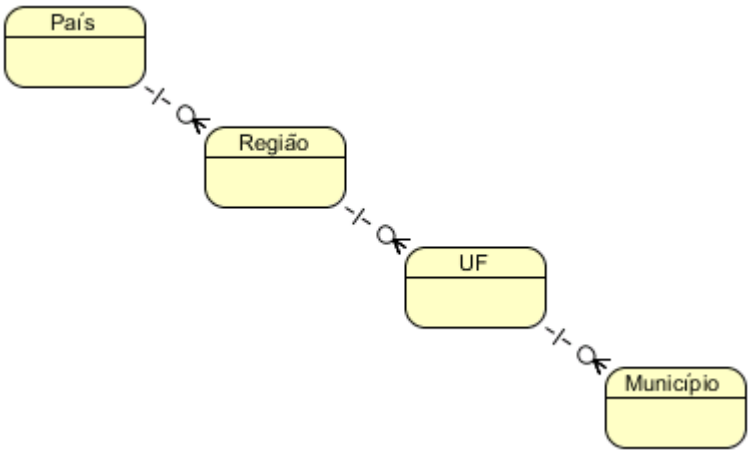


Figura 7: Exemplo de hierarquia normalizada

No entanto, modelos estrela não comportam esse tipo de estrutura. A hierarquia acima representada seria encapsulada e colapsada dentro da D-TAB Região. Assim o conteúdo da D-TAB Região seria algo do tipo:

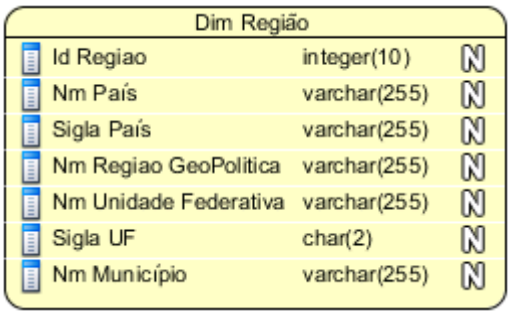


Figura 8: Exemplo da estrutura de uma D-TAB desnormalizada

Segundo dados do IBGE, o Brasil tem 5.564 municípios. Assim, a D-TAB região teria 5.564 linhas, sendo que as colunas Nm País e Sigla País teriam o seu conteúdo repetido: “Brasil” e “BR” para cada uma delas.

Esta tabela não se encontra normalizada, uma característica comum a todo modelo estrela. As dimensões estão, todas, desnormalizadas. Num modelo em “floco de neve”, por outro lado, as hierarquias que estavam contidas nas dimensões transformam-se em tabelas independentes vinculadas à D-TAB da qual foram extraídas através de relacionamentos 1:N.

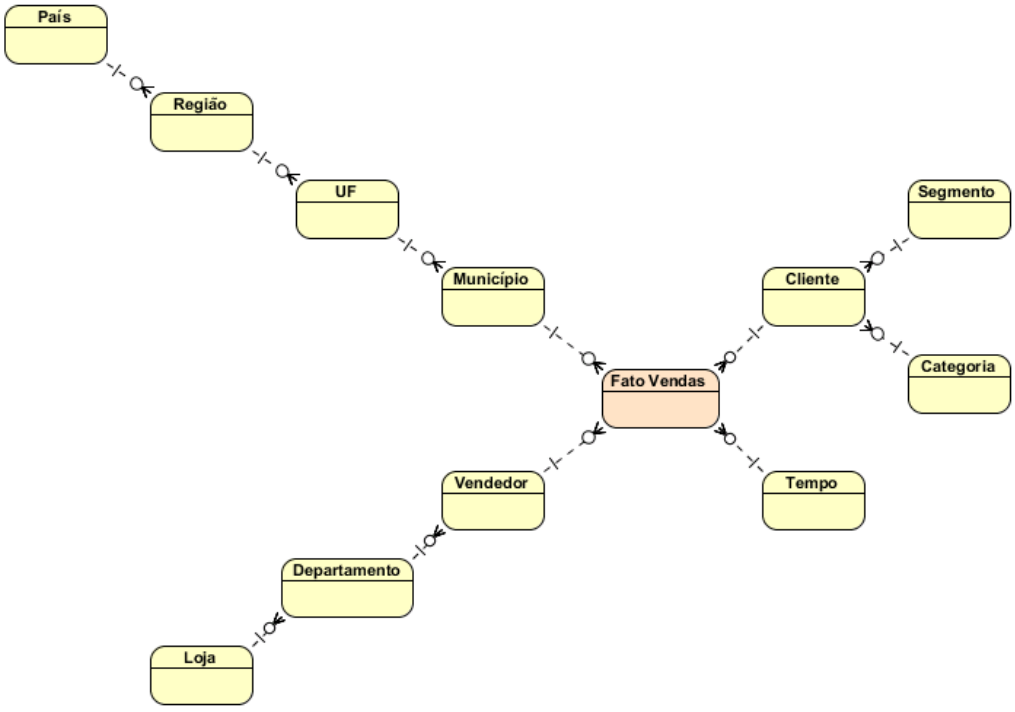


Figura 9: Exemplo de Modelo em Floco de Neve

Um dos problemas relacionados com este tipo de modelo é a quantidade de junções necessárias para responder certas consultas, comprometendo o desempenho. Este é o principal motivo que leva às críticas de Ralph Kimball.

Uma arquitetura de dados para Data Warehouse costuma ter no mínimo duas camadas: a camada do Data Warehouse onde os dados estão no nível mais atômico possível. Acima desta, temos a camada dos Data Marts onde, efetivamente, a grande maioria das consultas são realizadas. Data Marts são subconjuntos de dados de um Data Warehouse. Funcionam como se fossem "visões" do modelo principal orientadas para atenderem as consultas de uma área de negócios específica.

A transição do mundo relacional normalizado para o mundo dimensional nem sempre é fácil pois um modelo dimensional do tipo estrela rompe totalmente com os paradigmas e boas práticas da modelagem tradicional. O modelo em floco de neve, neste caso, serve como uma boa opção para a transição de uma realidade "operacional" para uma realidade "dimensional" e não raramente costuma ser a opção escolhida no momento de se modelar a camada do Data Warehouse, aplicando-se a técnica do modelo estrela a partir da camada dos Data Marts, onde trabalhamos com um nível maior de especialização e agregação.

Um outro ponto que conta à favor dos modelos em floco de neve está relacionado com o problema da reutilização das dimensões, isto é, das dimensões em conformidade. É importante salientar que o uso das dimensões em conformidade transformou o mundo dos modelos dimensionais de ilhas isoladas para um arquipélago no qual as dimensões em conformidade funcionam como pontes que unem as diversas ilhas, onde as ilhas são os modelos dimensionais individuais formados por uma única F-TAB orbitada pelas suas dimensões.

Estes arquipélagos são chamados de constelações. Constelações, são, segundo (Ghozzi, et al., 2003), extensões de modelos dimensionais. Elas agrupam diversas F-TABs de acordo com dimensões compartilhadas entre eles, as dimensões em conformidade.

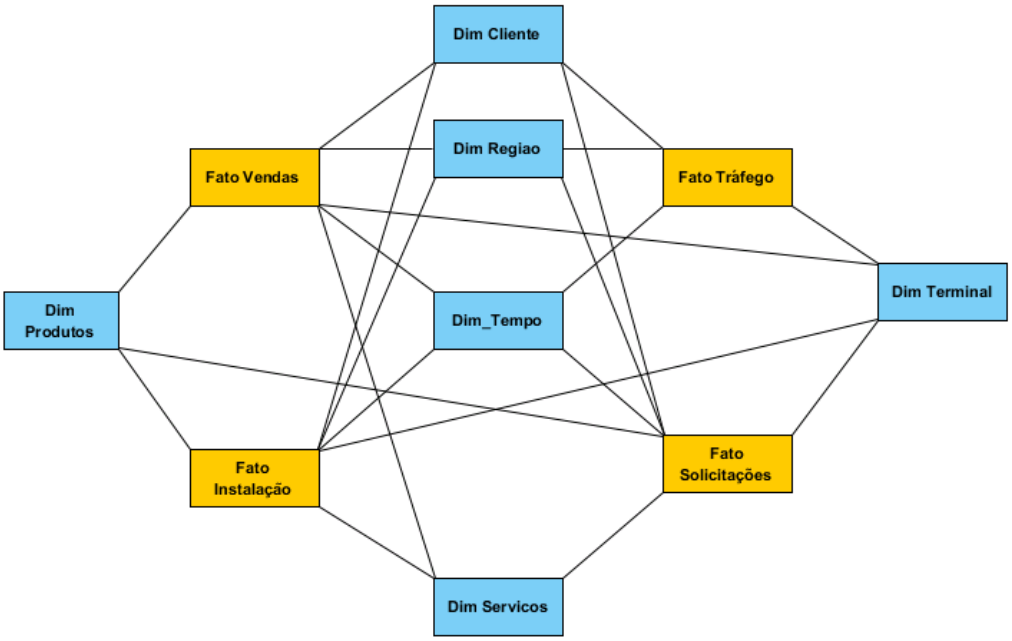


Figura 10: Exemplo de Constelação

Como, em nosso projeto, objetivamos a implantação real da ferramenta, certamente encontraremos casos em que houve o emprego do modelo em floco de neve. Esse é o principal motivo que nos leva a considerar a necessidade de tratar no escopo do presente trabalho não apenas os modelos dimensionais em formato estrela como também aqueles que se encontram estruturados em flocos de neve, mesmo que essa técnica de modelagem confronte algumas das boas práticas que consideramos como orientadoras da formulação das nossas heurísticas. Mais adiante, no capítulo 5, ficará claro que, apesar da aparente contradição entre as boas práticas adotadas neste trabalho e os modelos em floco de neve, estes modelos também serão facilmente percebidos e identificados pelas heurísticas definidas a partir das boas práticas de modelagem dimensional.

2.1.3
O Grão das F-TABs

O grão de uma F-TAB segundo Ralph Kimball é "a definição de negócio do evento que está sendo medido e que cria um registro na F-TAB. O grão é determinado exclusivamente pela realidade física da fonte dos dados"[26].

Essa visão apresentada por Kimball está fortemente influenciada pela realidade por ele encontrada durante os projetos de Data Warehouse com os quais se envolveu. Com essa simples declaração, o que Kimball procura dizer-nos é que dependemos do nível de detalhe registrado nos sistemas legados para definirmos o grão que iremos adotar para representar determinado evento em uma certa F-TAB. É uma visão realista uma vez que quem alimenta de informações o Data Warehouse, na grande maioria dos casos, são os sistemas legados.

Conceitualmente, no entanto, o grão de uma F-TAB é uma consequência direta do grão das dimensões que são usadas para caracterizar e analisar as medidas registradas naquela F-TAB. O grão estabelecido para a F-TAB depende daquilo que queremos medir e de quão precisa pode ser essa medição em função das informações que dispomos, ou melhor, do significado daquilo que queremos medir. A semântica associada às observações armazenadas em uma F-TAB é, portanto, consequência do seu grão e este é definido em função do grão de suas dimensões.

Mas como identificamos o grão de uma D-TAB? Como citamos anteriormente, dimensões podem possuir, entre as suas características (atributos), hierarquias. Em uma D-TAB desnormalizada, um determinado conjunto de atributos define uma particular hierarquia declarada naquela D-TAB.

Um bom exemplo de D-TAB desnormalizada é a D-TAB Geografia (ou Localidade, ou Região): costuma ser uma hierarquia onde, provavelmente, teríamos os seguintes níveis definidos: País, Unidade Federativa (ou Estado ou Província), Cidade e Bairro. Alternativamente, se incluíssemos "Continente" como um nível hierárquico, país passaria de raiz da hierarquia para seu

[26]http://www.kimballgroup.com/2007/07/30/keep-to-the-grain-in-dimensional-modeling/

primeiro nível. Se, por outro lado, incluíssemos “Endereço”, estaríamos definindo um novo grão mais detalhado para esta D-TAB. Da forma como a definimos, o grão desta D-TAB seria “Bairro”. Assim, a toda F-TAB que vincularmos esta D-TAB, no tocante à geografia ou localidade, o grão desta fato será “Bairro”, também. Da mesma forma, se o grão da D-TAB Tempo for dia, isto indica que todas as observações registradas nas fatos vinculadas a esta D-TAB terão dia como seu grão.

Na prática, nem sempre é possível manter todas as F-TABs de um modelo dimensional (principalmente na camada do Data Warehouse) com o mesmo grão “tempo”. É possível que algumas F-TABs, por exemplo, tenham grão mensal, enquanto outras tenham um grão diário, simplesmente porque nem todo evento ocorre num ciclo diário. Temos eventos que ocorrem semanal, mensal, ou anualmente, entre outras. A solução para isso é introduzir no domínio de cada um dos atributos da D-TAB tempo o valor “não se aplica”. A introdução deste artifício é utilizada para permitir que possamos trabalhar com grãos em diferentes níveis quando uma D-TAB é construída baseada em uma hierarquia. No caso da D-TAB tempo, a aplicação deste artifício poderia nos gerar instâncias do seguinte tipo:

ID	ANO	MÊS	QUINZ.	SEMANA	DIA
1	2013	N/A	N/A	N/A	N/A
2	2013	01	N/A	N/A	N/A
...	...	...	...	...	...
14	2013	01	1	N/A	N/A
15	2013	01	2	N/A	N/A
...	...	...	...	...	...
38	2013	01	1	1	N/A
39	2013	01	1	2	N/A
40	2013	01	1	3	N/A
41	2013	01	1	4	N/A
42	2013	01	1	5	N/A
...	...	...	...	...	...
88	2013	01	1	1	1
89	2013	01	1	1	2
...	...	...	...	...	...

Tabela 3: Exemplo de Instâncias da D-TAB Tempo com múltiplos grãos

Para ilustrarmos melhor esta discussão sobre o grão, vamos propor um possível modelo dimensional, representado pela figura 11. A medida ali registrada não está subdividida apenas pelas três perspectivas dadas pelas suas dimensões: tempo, categoria de pessoa e localidade.

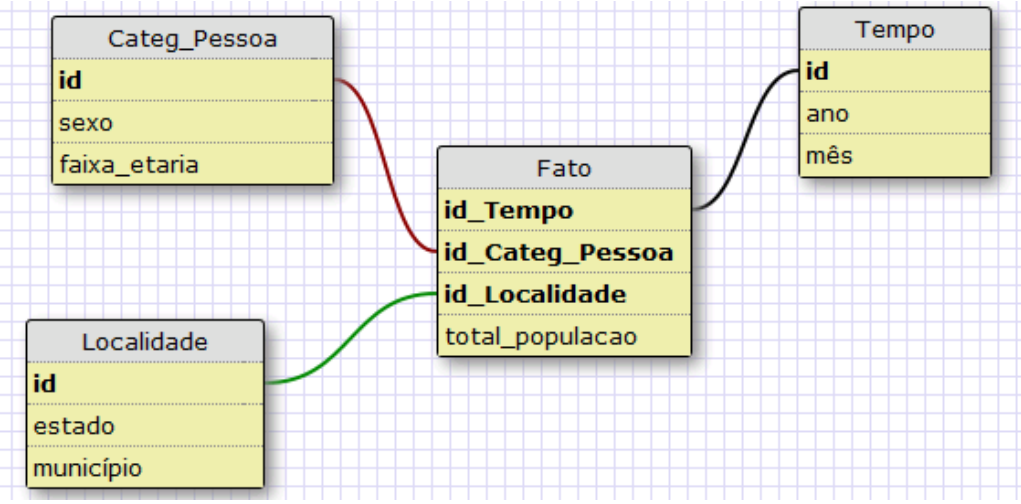


Figura 11: Exemplo de Modelo Dimensional para discussão do Grão

Na verdade o grão da F-TAB está "fatiado", segmentado, pelas diferentes perspectivas inseridas em cada D-TAB e que se traduz na combinação entre as diferentes instâncias de cada um dos atributos de suas dimensões combinados entre si. Mas vamos entender isso um pouco melhor.

Análise do grão da D-TAB Categoria Pessoa:

A D-TAB categoria pessoa possui dois atributos: sexo e faixa-etária. A cardinalidade do domínio do atributo sexo é claramente 2: masculino/feminino, por exemplo. Assumindo as faixas de idade adotadas pelo IBGE, isso nos dá 17 instâncias. A combinação entre elas nos dá 34 possibilidades. E este é o grão definido por esta D-TAB. Isso significa que cada medição realizada e registrada na F-TAB estará segmentada por estas duas pespectivas: sexo combinado com a faixa-etária.

Análise do grão da D-TAB Tempo:

A D-TAB categoria pessoa possui dois atributos: ano e mês. A cardinalidade do domínio do atributo mês é 12, sem surpresas. Já o atributo ano pode variar mas digamos que nossa série histórica começa no ano 2001 e segue até o ano de 2012. Portanto temos 12

instâncias que, combinadas com as instâncias do atributo mês, nos dá 144 possibilidades. E este é o grão desta D-TAB. Cada medição registrada na F-TAB também estará subordinada a estas duas perspectivas de tempo combinadas: ano e mês.

Análise do grão da D-TAB Localidade:

A D-TAB localidade pessoa possui dois atributos: estado e município. Nesse caso particular no entanto, não temos uma combinação de instâncias de cada atributo visto que cada município pertence a um único estado. Desta forma, o grão desta D-TAB é definido pelo atributo município exclusivamente. Segundo o IBGE, o total de municípios no Brasil em 2012 era 5570 municípios. Portanto teremos 5570 instâncias diferentes nesta D-TAB independente do número de estados existentes. Se amanhã um novo estado fosse criado e a quantidade de municípios permanecesse a mesma, isso não nos afetaria. Haveria apenas uma realocação de municípios de um (ou mais) estado(s) para o novo estado que foi criado. Portanto cada medição registrada na F-TAB diz respeito a um destes 5570 municípios.

Com base nas análises acima, podemos afirmar que, como estamos trabalhando com domínios enumerados e finitos, nossa F-TAB terá um total de 34 x 144 x 5570 observações, isto é, 27.270.720. Cada observação ali registrada informa a população total de determinado município em determinado mês, de determinado ano, agrupada por sexo e faixa etária. Para sabermos a população total de um município em determinado mês e ano, por exemplo, precisamos suprimir a D-TAB Categoria_Pessoas. Quando somamos o total de pessoas por ano, mês e município independente do sexo e faixa-etária, conseguimos saber a população total de certo município registrada para aquele mês e ano particular.

Uma observação importante a ser feita é que a observação total_população é uma medida semi-aditiva. Isso porque para sabermos a população total de um município em determinado ano, não podemos somar o registro de população para aquele município mês a mês, mas sim, considerar a

população total registrada para aquele município no mês 12 daquele ano. Assim a observação é aditiva sob a perspectiva das dimensões Categoria Pessoa e Localidade (a população total de um estado é dada pela soma das populações de cada um dos seus municípios) mas não sob a perspectiva da D-TAB tempo.

Se, neste mesmo modelo, adicionássemos uma nova observação, percentual de analfabetos, esta seria uma medida não aditiva. Isso porque um percentual não pode ser utilizado para operações comuns de soma porque neste caso, a soma das partes não nos dá o todo. Finalmente se introduzíssimos como observação a quantidade de óbitos registrados, passaríamos a ter uma medida aditiva. O total de óbitos no ano é a soma do total de óbitos registrados mensalmente, da mesma forma que o total de pessoas falecidas corresponde ao somatório de falecidos de cada sexo, por faixa etária e, o total de óbitos num estado é dado pela soma dos óbitos em cada município.

Portanto o grão de um cubo depende de como as instâncias dos atributos de suas dimensões podem ser combinadas entre si e com as demais dimensões. É fundamental entender que o grão de uma tabela é a menor unidade de informação daquela tabela. Não podemos subdividir um grão da mesma forma que não podemos subdividir um átomo (na verdade, ao contrário do grão, até podemos dividir um átomo, mas isso é altamente desaconselhável, uma vez que seria a última coisa que você faria na vida).

Portanto, como vimos, o grão de uma F-TAB é definida em função do grão de suas dimensões. Assim, se uma D-TAB é reaproveitada em mais de uma F-TAB (D-TAB em conformidade), todas as fatos que compartilham aquela D-TAB também compartilham o mesmo grão em relação a ela. Portanto o reaproveitamento de uma D-TAB só faz sentido quando o fato ali registrado tiver sua medição compatível com o grão definido naquela D-TAB.

2.2
Transição de Modelos Dimensionais para Cubos de Dados

2.2.1
Especificação de Cubos de Dados

Os modelos dimensionais são implementados fisicamente em bancos de dados relacionais ou multidimensionais. Quando implementados em bancos de dados multidimensionais, cada atributo de cada D-TAB torna-se um eixo dimensional. Os pontos nas interseções dos diversos eixos armazenam os valores das observações medidas.

A figura 12 esboça um modelo dimensional visando ilustrar o processo de transformação de um modelo dimensional em uma estrutura verdadeiramente multidimensional. No exemplo apresentado, temos apenas três dimensões, cada uma delas, por sua vez, possui apenas três atributos, além da sua chave primária. Esta estrutura de três dimensões, cada qual contendo três atributos, não é coincidência. Estamos forçando um cenário onde cada uma das dimensões irá definir um subespaço tridimensional a partir dos seus três atributos, uma vez que, conforme vimos no parágrafo anterior, cada atributo de cada D-TAB torna-se um eixo da estrutura multidimensional.

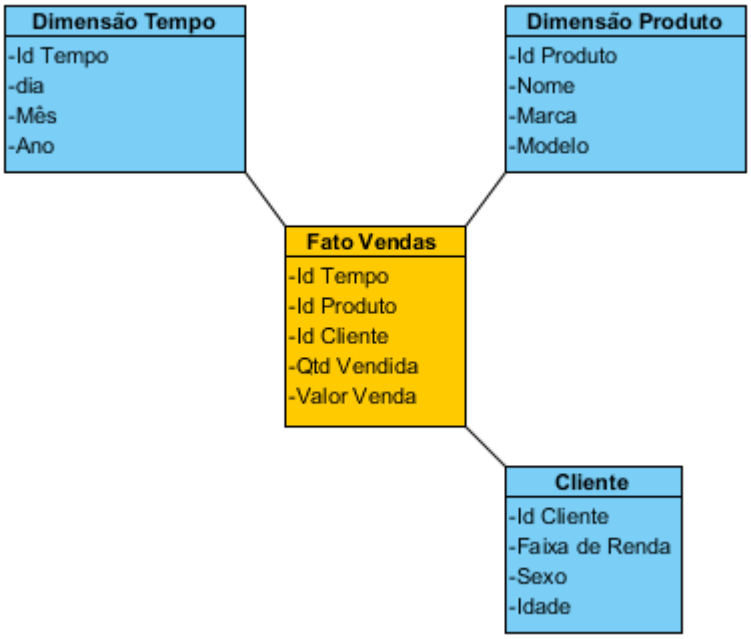


Figura 12: Exemplo de Modelo Multidimensional

Cada uma das três dimensões apresentadas no modelo da figura 12 define, através de seus atributos, um subespaço multidimensional no qual cada um dos atributos da D-TAB à exceção de sua chave primária, representa um eixo ou uma subD-TAB. As instâncias do domínio de valores do atributo representam cada um dos pontos do eixo correspondente ao atributo, como podemos observar na figura 13:

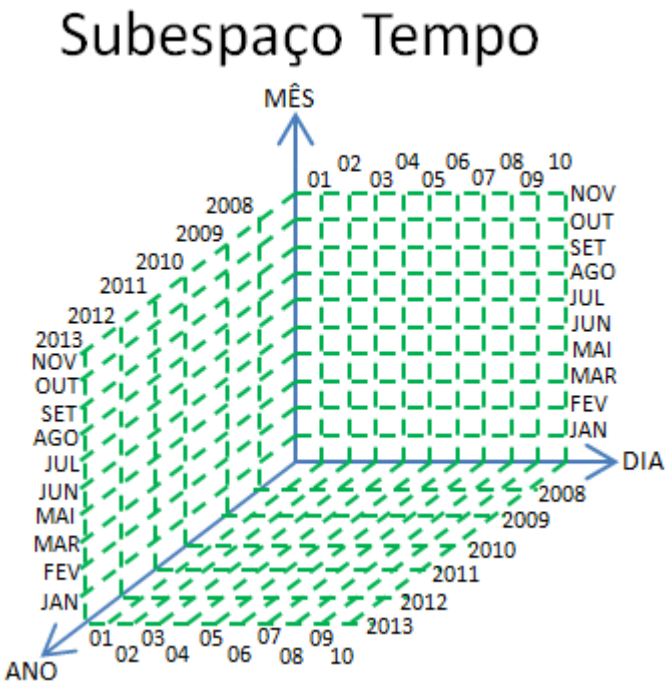


Figura 13: Subespaço definido pelos atributos da D-TAB Tempo.

Nem todas as instâncias de um atributo se combinam com as instâncias dos outros atributos. Apesar disso, todas as combinações obtidas a partir das instâncias persistidas na D-TAB são usadas para identificar os pontos que definem cada um dos subespaços. Consideremos as instâncias apresentadas nas tabelas 4, 5 e 6:

PK	DIA	MÊS	ANO
1	01	Janeiro	2013
2	02	Junho	2010
3	03	Março	2012
4	05	Maio	2011
5	08	Junho	2013

Tabela 4: Instâncias da D-TAB Tempo

PK	MARCA	MODELO	NOME
1	Apple	5	iPhone
2	Samsung	Galaxy S III	Smartphone
3	HP	14B0-60	Laptop
4	Dell	3227U5	Laptop
5	Samsung	Galaxy S IV	Smartphone

Tabela 5: Instâncias da D-TAB Produto

PK	SEXO	IDADE	FAIXA DE RENDA
1	Masculino	Até 18 anos	Até 5 S.M.
2	Feminino	Entre 25 e 35	Entre 10 e 15 S.M.
3	Feminino	Entre 25 e 35	Entre 15 e 20 S.M.
4	Feminino	Até 18 anos	Até 5 S.M.
5	Masculino	Entre 18 e 22	Até 5 S.M.

Tabela 6: Instâncias da D-TAB Cliente

Apesar de utilizarmos todas as combinações possíveis entre as diferentes instâncias de cada D-TAB para definir os subespaços dimensionais, iremos focar o restante desta discussão apenas nos pontos considerados relevantes. Os pontos ditos relevantes são aqueles que representam relacionamentos reais entre as instâncias dos atributos. Desta forma, tomando por base a tabela 6, o ponto formado pelas instâncias “Masculino”, “Entre 18 e 22” e “Até 5 S.M.” é considerado relevante. Por outro lado, o ponto formado pelas instâncias “Masculino”, “Entre 25 e 35” e “Até 5 S.M.” não é relevante, visto que não há uma combinação correspondente entre as linhas apresentadas na tabela 6.

Quando aplicamos os valores apresentados nas tabelas 4, 5 e 6 nos diversos subespaços apresentados anteriormente, encontramos os pontos “relevantes” em cada um deles. Cada subespaço é formado por três subplanos como ilustrado abaixo em relação ao subespaço definido para a D-TAB Tempo. A figura 14 nos mostra como os pontos relacionados com os valores de instâncias apresentados anteriormente na Tabela 4 definem pontos nos subplanos, para em seguida, identificar os pontos que definem a formação do subespaço que podemos projetar a partir da combinação dos pontos de cada um dos subplanos.

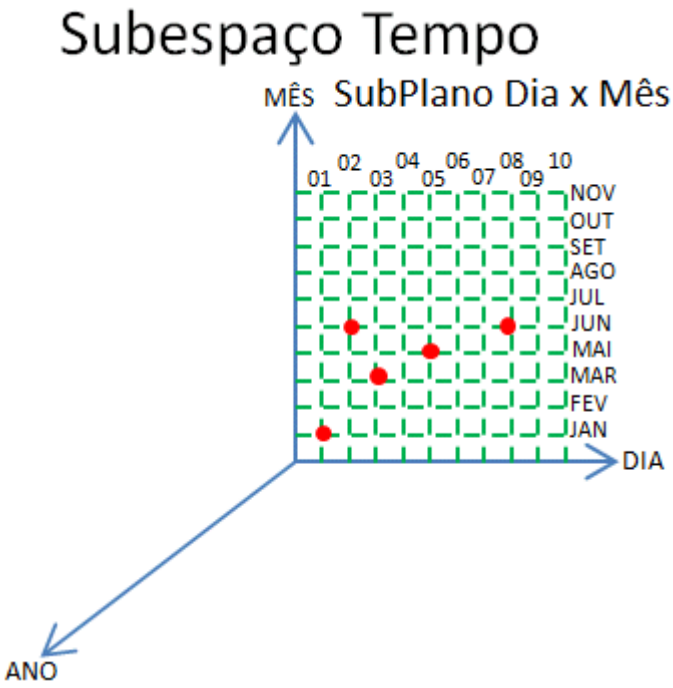


Figura 14: Exemplo de pontos definidos no Subplano Dia x Mês

Quando projetamos os pontos definidos em cada um dos subplanos e os conectamos entre si, definimos novos pontos que, desta vez, produzem um subespaço. Cada ponto deste subespaço será usado para gerarmos a projeção do cubo de dados multidimensional resultante da conexão entre os pontos projetados a partir de cada um dos subespaços dimensionais definidos a partir das dimensões. Mas é fundamental entender que os pontos que serão criados em cada subespaço correspondem apenas àqueles cuja combinações de valores entre os seus atributos equivalha à uma instância persistida na D-TAB. Desta forma, considerando a D-TAB cliente, se tomarmos as instâncias ali apresentadas como exemplo, os dois valores do domínio do atributo "sexo" são utilizados (feminino e masculino) na criação das cinco instâncias da D-TAB. No entanto, dos cinco possíveis valores definidos como domínio para o atributo "faixa de renda", apenas três são efetivamente utilizados (Até 5 S.M., Entre 10 e 15 S.M. e Entre 15 e 20 S.M.). De forma semelhante, somente três dos cinco valores definidos como domínio do atributo "Idade" são efetivamente utilizados (Até 18 anos, Entre 18 e 22 anos e Entre 25 e 35 anos). Todos os demais pontos que poderiam ser formados combinando-se cada um dos valores de domínio definidos para cada atributo com os demais valores de domínio definidos para os outros atributos não possuem qualquer relevância nem

estarão representados no subespaço. Cada um destes pontos, por sua vez, é identificado, na sua D-TAB de origem, como sendo a chave primária da instância que ele representa.

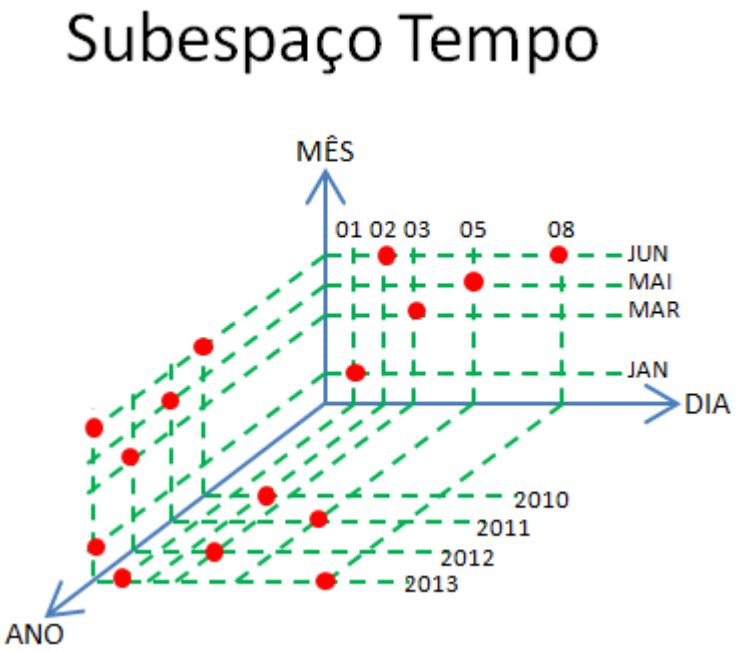


Figura 15:Pontos a partir dos quais o Subespaço Tempo será construído

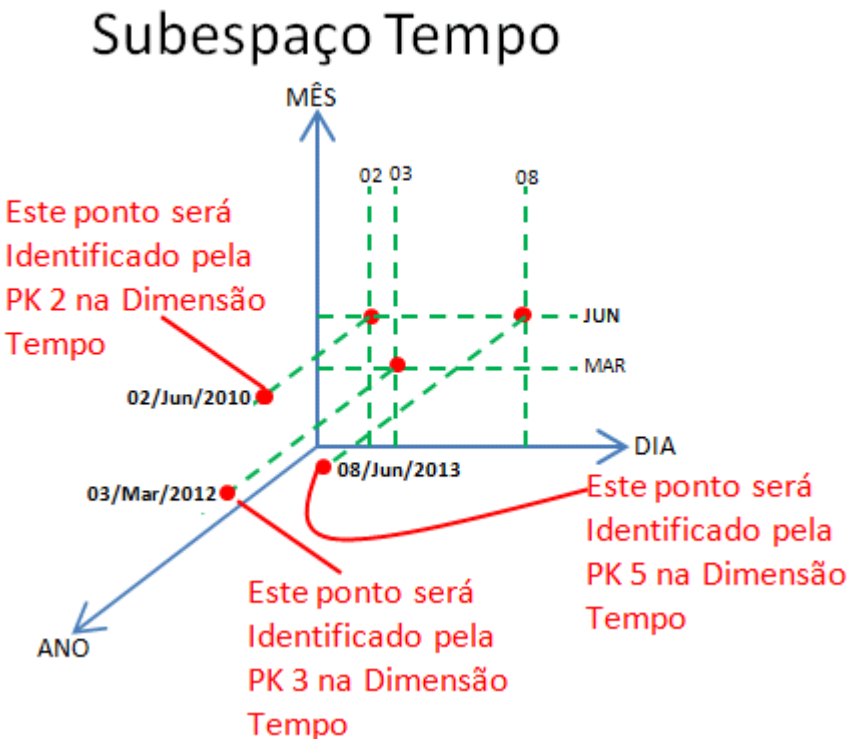


Figura 16: Projeção de alguns pontos para formar o Subespaço da D-TAB Tempo

A sequência de figuras anterior (15, e 16) ilustra este processo de construção dos subespaços a partir dos pontos projetados existentes em cada um dos subplanos constituintes dos subespaços criados a partir de cada uma das dimensões declaradas no modelo dimensional apresentado na figura 12.

Esses três subespaços são então reunidos, funcionando como se cada um deles fosse um eixo do cubo dimensional. A partir dos pontos criados em cada um dos subespaços, utilizados para defini-los, projetamos então os pontos nos cubos de acordo com as observações registradas na F-TAB. É fundamental entender que cada um desses pontos representa uma instância persistida na tabela D-TAB correspondente ao subespaço considerado e que será identificado pela chave primária correspondente à instância que ele representa, como pode ser observado na figura 16.

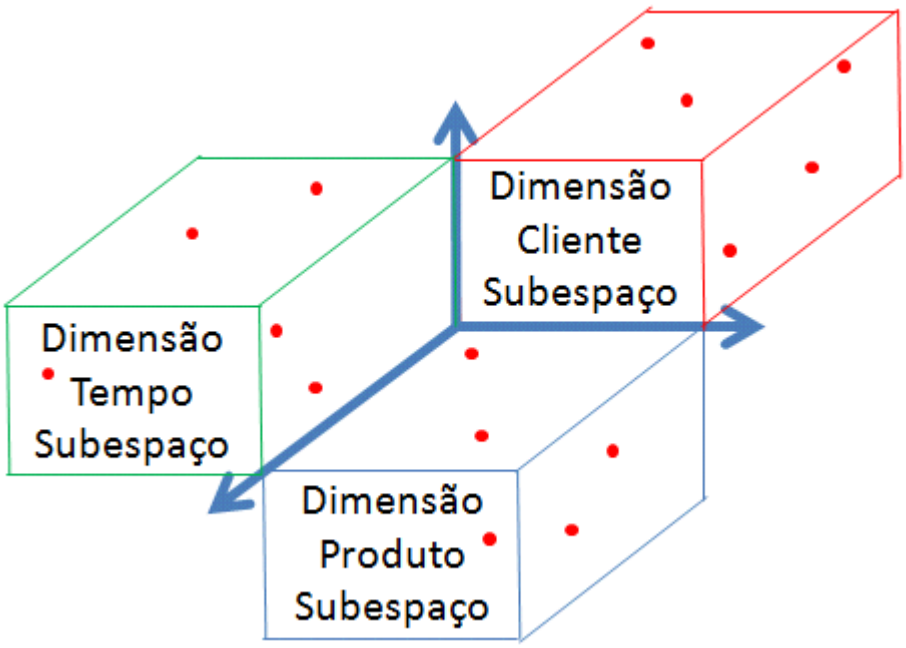


Figura 17: Subespaços formadores do Cubo Dimensional

A figura 17 nos mostra como os subespaços criados a partir dos atributos das dimensões integrantes do modelo dimensional representado pela figura 12 são utilizados para projetarmos os pontos que definem o cubo dimensional. Para definir o cubo dimensional, no entanto, precisamos da informação principal: quais instâncias de cada subespaço se relacionam entre sim, a fim de caracterizar o registro de uma observação persistida na F-TAB. Consideremos, portanto, que a tabela 7 representa o conjunto de observações registradas na F-TAB do nosso modelo.

PK DIM TEMPO	PK DIM CLIENTE	PK DIM PRODUTO	QUANTIDADE VENDIDA	VALOR VENDA
1	3	2	1	2500,00
1	4	5	2	7000,00
1	3	1	1	2500,00
1	5	1	1	1200,00
5	1	2	1	2800,00
5	1	5	1	2800,00
5	5	1	1	1800,00

Tabela 7: Representação de Instâncias da F-TAB de Vendas

Assim, podemos projetar, a partir dos pontos criados nos subespaços dimensionais, aqueles que se relacionam com os pontos das outras dimensões, produzindo, desta forma, o conjunto de pontos que representa o conjunto de observações armazenadas na F-TAB e que definem a estrutura do cubo dimensional. As instâncias de cada D-TAB podem ser representadas como pontos existentes em cada um dos eixos ortogonais que as representam e assim passamos a ter uma estrutura assim definida.

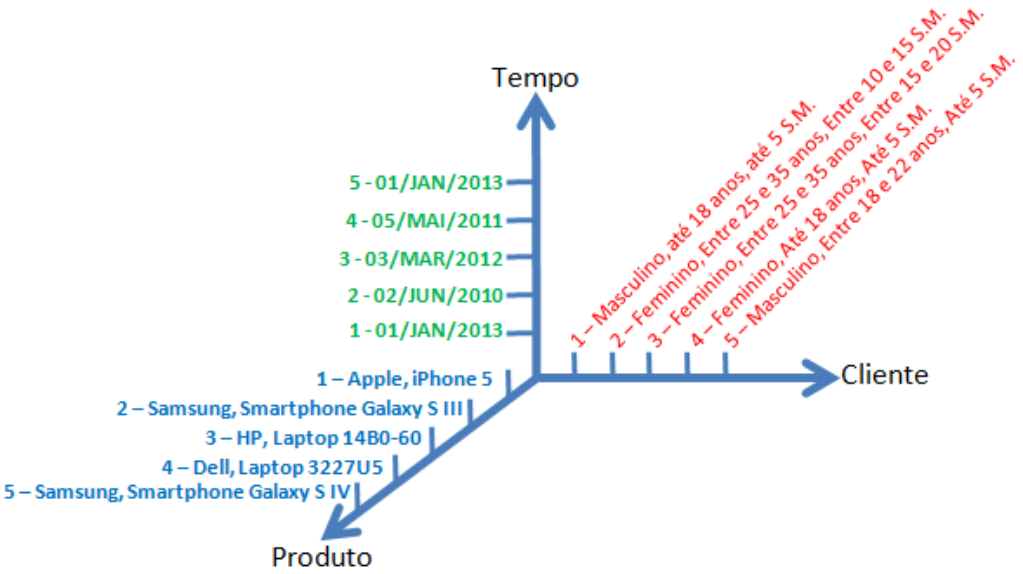


Figura 18: Estrutura do Cubo Dimensional

A produção do cubo de dados, agora, se dará pela materialização dos pontos que representam as observações registradas na F-TAB, cruzando-se as correspondentes instâncias de dimensões cujo vínculo está estabelecido no registro da observação. Assim, considerando-se as instâncias apresentadas na

tabela 7, os seguintes pontos seriam gerados, definindo o cubo dimensional onde estariam representadas as observações registradas na F-TAB. Cada um dos pontos abaixo representados será uma célula onde as medições realizadas para cada medida definida no cubo serão gravadas. No caso apresentado como exemplo, teríamos uma célula com duas entradas, uma delas representando a observação medida sobre a quantidade vendida enquanto a outra representará o valor total da venda.

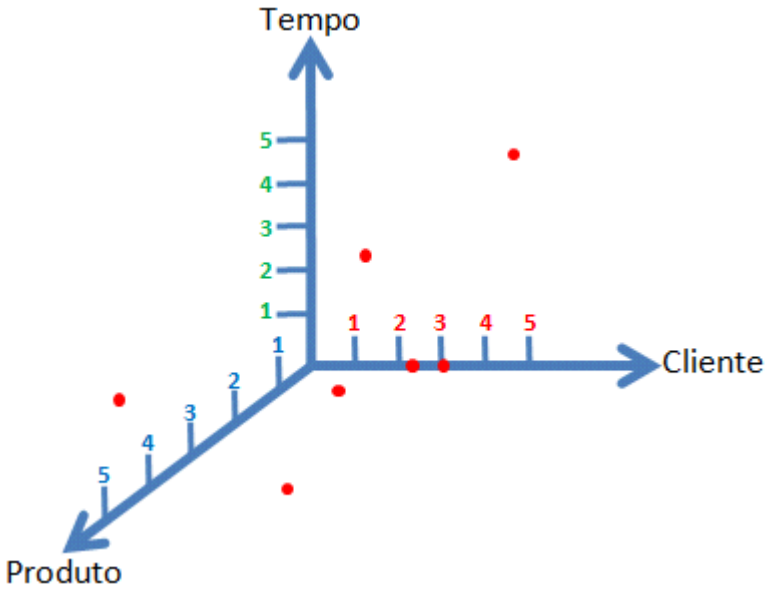


Figura 19: Pontos que representam as observações da Fato

Como mencionado anteriormente, o exemplo aqui apresentado teve como objetivo esclarecer de forma didática o processo de construção de estruturas dimensionais. O cubo dimensional real, no entanto, é formado considerando cada um dos atributos das dimensões como eixos dimensionais distintos e estabelecendo os cruzamentos entre os pontos que representam as instâncias do domínio de valores de cada atributo. Tal figura, além de ser difícil de imaginar é ainda mais difícil de representar e, portanto, utilizamos alguns artifícios para facilitar o entendimento do processo de construção da estrutura multidimensional. Mais adiante, porém, será necessário trabalharmos com o conceito real, isto é, lembrarmos que os atributos das dimensões do modelo é que são usados como eixos dimensionais para construir o cubo de dados dimensional.

2.3 Resumo

Neste capítulo discutimos parte dos fundamentos teóricos que serviram de base para o desenvolvimento desta dissertação: os conceitos fundamentais de modelagem dimensional e de cubos, a transição do modelo dimensional para os cubos dimensionais e como estes são construídos. No próximo capítulo detalharemos as linguagens e ontologias usadas para expressar estes modelos e cubos em RDF.

3.
Triplificação de Modelos Dimensionais

3.1
Introdução

As triplas utilizadas para representar os modelos dimensionais, utilizam RDF, RDFS, SKOS e Data Cube Vocabulary, a fim de identificar as classes (Fatos e Dimensões) e suas propriedades (atributos das dimensões e atributos, medidas e unidades de medida das Fatos), bem como a identificação de hierarquias entre as propriedades.

As duas próximas seções resumem o SKOS e o *Data Cube Vocabulary* pressupondo que o leitor já conhece RDF e RDFS. Se este não for o caso, o apêndice H apresenta um resumo sobre vocabulários e ontologias, descrevendo, entre outros, o RDF e o RDFS.

3.2
SKOS – Simple Knowledge Organization System Primer

SKOS, que significa Sistema Simples de Organização do Conhecimento, é um padrão W3C, baseado em outros padrões da Web Semântica (RDF e OWL), que fornece uma maneira de representar vocabulários controlados, taxonomias e tesauros. Especificamente, a própria SKOS é uma ontologia OWL e pode ser escrita em qualquer sintaxe RDF.

Um vocabulário controlado é uma lista de termos cujo emprego, representação e significado foram acordados por uma comunidade ou organização. Por exemplo: segunda, terça, quarta, quinta, sexta, sábado e domingo são os dias da semana.

Como vimos anteriormente neste trabalho, quando tratamos das ontologias, a taxonomia é um vocabulário controlado organizado em uma hierarquia. Por exemplo, podemos ter os termos Computador, Tablet e

Notebook, e Tablet e Notebook podem ser subclasses de Computador porque ambos são tipos de computadores.

Finalmente, um tesauros é uma taxonomia com mais informação sobre cada um de seus conceitos incluindo termos preferidos ou alternativos (Computer em inglês, Ordenador em espanhol, etc). Além disso um tesauros pode conter relações com conceitos relacionados, como por exemplo "computador"e "software".

Essencialmente, SKOS nos permite:

- Definir conceitos (ou recursos) através de URIs;
- Estabelecer rótulos (*labels*) com *labels* léxicos, *labels* simbólicos, *preferred, alternative, hidden, etc,* em uma ou mais linguagens naturais;
- Descrevê-los e Documentá-los utilizando definições, exemplos e diversos tipos de notas (*scope note, change note, editorial note, etc)*;
- Estabelecer relações semânticas com outros conceitos em hierarquias informais e redes de associação (*broader, narrower, related*);
- Agrupá-los em esquemas conceituais suportados por "*node labels*";

O mais importante elemento do SKOS é o conceito. Conceitos são ideentificados por URIs. Um exemplo simples de como criar um conceito usando SKOS, seria:

ex:Computador a skos:Concept.

O significado de um conceito é dado, primariamente pelos rótulos que lhe são atribuídos em linguagem natural. O uso desses rótulos é fundamental para expressar a essência dos significados vinculados ao conceito. Temos à nossa disposição dois tipos de labels: preferenciais (skos:prefLabel) ou alternativos (skos:altLabels). O label preferencial deve ser único por linguagem, enquanto que podem ser atribuídos diversos labels alternativos em uma mesma linguagem.

ex:Computador a skos:Concept;
skos:prefLabel "Computer"@en;
skos:prefLabel "Computador"@pt;

skos:prefLabel “Computador”@es;
skos:altLabel “Ordenador”@es;

O significado de um conceito pode ser aprimorado, completado, ser melhor entendido ou percebido, através da associação com outros conceitos que são obtidos através do estabelecimento de relacionamentos semânticos. Os relacionamentos semânticos podem ser mapeados em SKOS de forma hierárquica ou associativa.

As definições de hierarquias em SKOS são obtidas através do uso dos termos: skos:broader e skos:narrower. O primeiro indicando que determinado conceito é mais geral do que outro. Enquanto que o segundo indica que determinado conceito é mais específico (ou especializado) do que outro. Um “computador”, portanto seria “broader” em relação ao “notebook”. Por outro lado, o “notebook” seria “narrower” em relação ao conceito “computador”.

ex:Computador a skos:Concept;
skos:prefLabel “Computer”@en;
skos:prefLabel “Computador”@pt;
skos: prefLabel “Computador”@es;
skos: altLabel “Ordenador”@es;
skos:broader ex:Notebook;

ex:Notebook a skos:Concept;
skos:prefLabel “Laptop”@en;
skos:altLabel “Notebook”@en;
skos:prefLabel “Notebook”@pt;
skos:prefLabel “Portatil”@es;
skos:narrower ex:Computador;

As relações não hierárquicas, por sua vez, estabelecem simples relações entre conceitos, como, por exemplo “computador” e “software”.

ex:Computador a skos:Concept;
skos:prefLabel “Computer”@en;
skos:prefLabel “Computador”@pt;
skos: prefLabel “Computador”@es;
skos: altLabel “Ordenador”@es;
skos: broader ex:Notebook;
skos: related ex:Software;

Também podemos mapear relações usando skos:closeMatch and skos:exactMatch, que nos permitem representar o mapeamento entre conceitos diferentes. O skos:closeMatch indica que os conceitos são suficientemente similares e que podem ser usados de forma intercambiável, porém, não são transitivos.

ex:Netbook a skos:Concept;

skos:prefLabel “Netbook”@en;
skos:narrower “Computador”;
skos: closeMatch ex:Notebook;

Quanto ao skos:exactMatch, denota um alto grau de similaridade, indicando que os conceitos envolvidos tem o mesmo significado. Esta relação é transitiva. Suponha que o conceito “Notebook” exista em outro tesauros SKOS identificado por ex2:Portatil. Para ligar este conceito ao que criamos mais acima simplesmente incluiríamos a seguinte sentença abaixo destacada.

ex:Notebook a skos:Concept;
skos:preflabel “Laptop”@en;
skos:altLabel “Notebook”@en;
skos:prefLabel “Notebook”@pt;
skos:prefLabel “Portatil”@es;
skos: narrower ex:Computador;
skos:closeMatch ex:Netbook;
skos:exactMatch ex2:Portatil;

Antes de finalizarmos nossa apresentação do SKOS, um cuidado fundamental deve ser tomado com as propriedades skos:broader e skos:narrower pois elas, podem nos enganar. A semântica associada aos vocábulos “broader” e “narrower” são intuitivamente transitivos. Assim se tivermos declarações: A broader B e B broader C, intuitivamente implica que A broader C. No entanto, os relacionamentos skos:broader e skos:narrower **NÃO SÃO** definidos como propriedades transitivas. O objetivo é evitar efeitos indesejados em hierarquias “sujas”, para KOS que não são tesauros, por exemplo, ou para evitar o efeito “mundo aberto”.

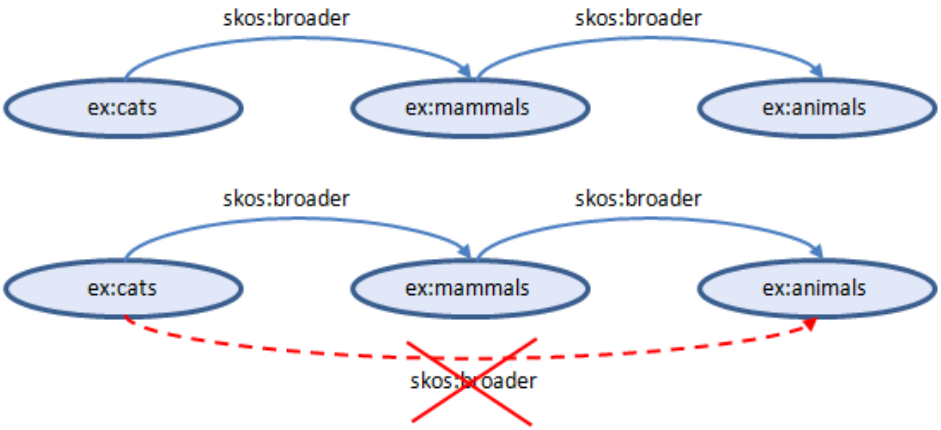


Figura 20: Ilustração da não transitividade de skos:broader

Assim a transitividade precisa ser explicitada. Fazemos isso através das propriedades: skos:broaderTransitive e skos:narrowerTransitive. Aplicando-se estas propriedades no exemplo ilustrado pela figura anterior, temos:

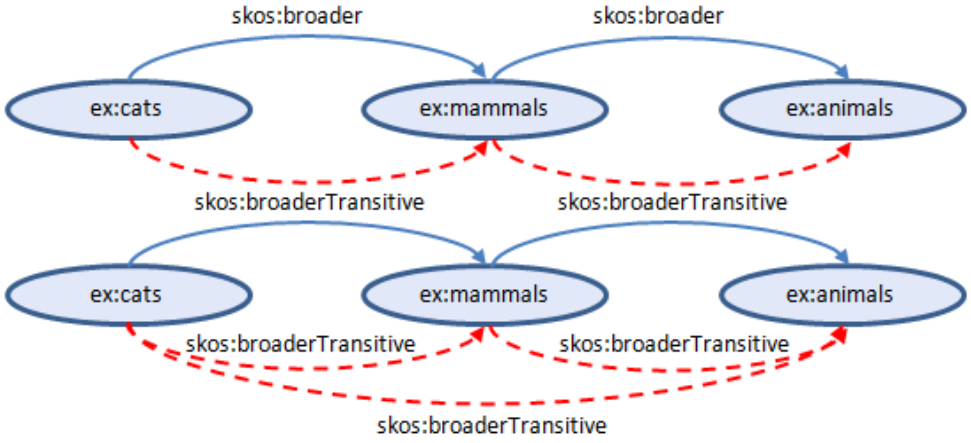


Figura 21: Ilustração do Uso das propriedades transitivas em SKOS

As propriedades skos:broaderTransitive e skos:narrowerTransitive são, respectivamente, super propriedades de skos:broader e skos:narrower. Sua "lógica de aplicação" se dá segundo esses "passos":

ex:animals skos:prefLabel "animals"@en.
ex:mammals skos:prefLabel "mammals"@en;
skos:broader ex:animals.
ex:cats skos:prefLabel "cats"@en;
skos:broader ex:mammals.

Super Propriedade

ex:cats skos:broaderTransitive ex:mammals.
ex:mammals skos:broaderTransitive ex:animals.

Transitividade

ex:cats skos:broaderTransitive ex:animals.

3.3
Data Cube Vocabulary

Conforme vimos no capítulo 1 do presente trabalho, há um grande interesse na publicação de dados estatísticos na Web, principalmente se pudermos relacioná-los a outros conceitos e dados.

O *Data Cube Vocabulary* foi criado com o objetivo de permitir a publicação de dados estatísticos no padrão RDF segundo os princípios de *linked data*.

O modelo por trás do *Data Cube Vocabulary* é compatível com o modelo SDMX[27] (*Statistical Data and Metadata Exchange*), padrão ISO (ISO, 2005) para intercâmbio e compartilhamento de dados e metadados estatísticos (Cyganiak, et al., 2013).

O *Data Cube Vocabulary* tem por base os seguintes vocabulários RDF:

- *Simple Knowledge Organization System* (SKOS)[28] – para definição dos esquemas;
- *Statistical Core Vocabulary* (SCOVO)[29] – para as estruturas dos dados estatísticos;
- VoID[30] - para o acesso aos dados;
- *Friend-of-a-Friend* (FOAF)[31] – para agentes;
- *Dublin Core Metadata Initiative* (DCMI)[32] - para os metadados;
- *Core Organization Ontology* (ORG)[33] – para as organizações;

O *Data Cube Vocabulary* foi projetado para ser um elemento central a partir do qual outros vocabulários poderiam ser estendidos de forma a complementar aspectos de publicação de dados estatísticos que ainda não tivessem sido cobertos por ele.

O *Data Cube Vocabulary* interpreta um cubo de dados de forma bastante simples e objetiva: trata-se de um conjunto de **observações**, indexado por **dimensões** (*code lists*), que descrevem **medidas** (*measures*) que devem ser **interpretadas** de acordo com os seus **atributos** (*attributes*), onde:

- *Code Lists* – representam os possíveis valores que podem ser assumidos por cada D-TAB. Em outras palavras, tratam-se das instâncias de uma D-TAB. Podem ser listas de países ou de produtos.

[27] http://sdmx.org/

[28] http://www.w3.org/2004/02/skos/

[29] http://sw.joanneum.at/scovo/schema.html

[30] http://www.w3.org/TR/void/

[31] http://xmlns.com/foaf/spec/

[32] http://dublincore.org/documents/2012/06/14/dcmi-terms/?v=terms

[33] http://www.epimorphics.com/public/vocabulary/org.html

O projeto, manutenção e padronização destas *Code Lists* constituem-se em importantes atividades na organização da produção de estatísticas. O motivo disso é que, quanto maior for a possibilidade de se compartilhar uma *Code List* entre diferentes cubos de dados maiores são as possibilidades de podermos estabelecer comparações diretas entre as estatísticas publicadas por estes diferentes cubos.

- *Measures* – medidas representam (como vimos anteriormente) os fenômenos que estão sendo observados. Que tipo de quantidade está sendo medida ou contada em certa observação? Cada cubo de dado pode conter uma ou mais medidas. Segundo (Cyganiak, et al., 2013) quando um cubo de dados contém múltiplas medidas, convém adicionar-lhe uma D-TAB "medida". O emprego de tal técnica no entanto é questionável uma vez que irá aumentar consideravelmente a quantidade de linhas no cubo de dados afetando, desta forma, o desempenho das consultas.
- *Attributes* – ainda segundo (Cyganiak, et al., 2013), observações devem possuir atributos associados que permitam interpretar corretamente o seu conteúdo. De fato, como saber exatamente o que representa um determinado valor registrado em certa medida? Pode expressar um valor em moeda ou um volume, um peso e assim por diante. E ainda que saibamos que o conteúdo da medida representa um valor em moeda, qual a precisão a ser adotada? O valor está expresso em unidades, em dezenas de unidade, centenas de unidades, milhares de unidades? Tal informação é de extrema relevância para expressar a semântica da medida. No entanto cabe esclarecer que estas informações, em modelos dimensionais, costumam existir como metadados sobre as medidas e não como colunas fisicamente criadas em F-TABs. E este ponto será de extrema relevância mais adiante, no capítulo 5 deste trabalho.

A figura 28 ilustra estes elementos principais que formam um cubo de dados estatísticos e que são o foco primordial de publicação do *Data Cube Vocabulary*

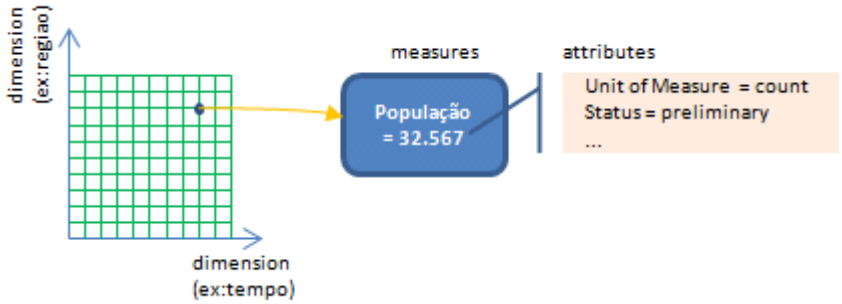


Figura 22: Elementos centrais de um cubo de dados

A figura 23 apresenta os componentes do *Data Cube Vocabulary*.

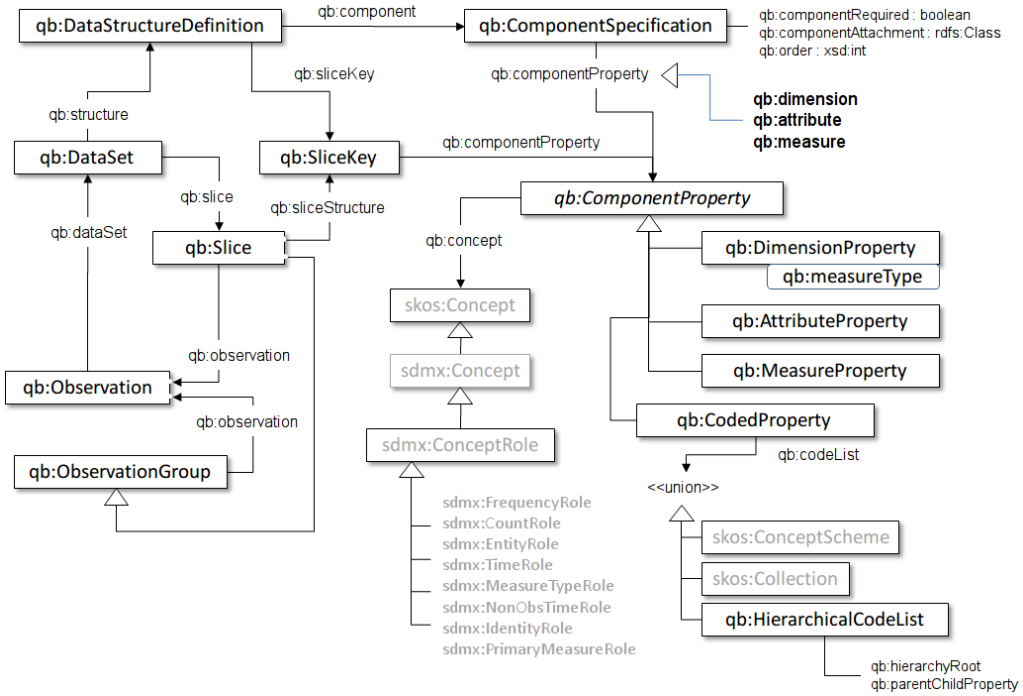


Figura 23: Estrutura do Data Cube Vocabulary

A seguir será feita uma descrição destas classes que compõem o *Data Cube Vocabulary*, para isso, no entanto, precisamos entender melhor o que está representado no diagrama acima.

Cada "caixinha" do diagrama, representa uma entidade RDF, que pode ser uma classe, um predicado ou um indivíduo. O nome de cada uma dessas entidades corresponde a uma URI. Estas URIs estão representadas por uma notação compactada na qual os *namespaces* foram substituídos por um prefixo que os representa, conforme podemos observar na tabela 8.

PREFIXO	NAMESPACE
Qb	http://purl.org/linked-data/cube#
Skos	http://www.w3.org/2004/02/skos/core#
Rdfs	http://www.w3.org/2000/01/rdf-schema#
Xsd	http://www.w3.org/2001/XMLSchema#
Sdmx	http://purl.org/linked-data/sdmx#
Ex	http://www.example.com/concepts#
Dc	http://purl.org/dc/elements/1.1/

Tabela 8: Prefixos e Namespaces

DATASETS

Class: qb:DataSet *Sub class of:* qb:Attachable *Equivalent to: s*covo:Dataset

Representam uma coleção de observações, possivelmente organizadas em diversas visões, de acordo com alguma estrutura dimensional.

OBSERVAÇÕES

Class: qb:observation *Sub class of:* qb:Attachable *Equivalent to:* scovo:Item

Representa uma observação individual registrada no cubo, podendo ter um ou mais valores de medidas associados.

Property: qb:dataSet *(Domain:* qb:Observation -> *Range:* qb:DataSet *)*

Indica o *dataset* do qual a observação faz parte.

Property: qb:observation *(Domain:* qb:Slice -> *Range:* qb:Observation *)*

Indica uma observação contida em uma visão particular de um *dataset*.

VISÕES (*SLICES)*

Class: qb:Slice *Sub class of:* qb:Attachable, qb:ObservationGroup

Representa um subconjunto de dados definido através da seleção de um subconjunto de valores nas dimensões, propriedades do componente na visão.

Class: qb:ObservationGroup

Um grupo de observações possivelmente arbitrárias.

Property: qb:slice (*Domain:* qb:DataSet -> *Range:* qb:Slice; *sub property of:* qb:observationGroup)

Representa um subconjunto de dados definido através da seleção de um subconjunto de valores nas dimensões

Property: qb:observationGroup (*Domain:* -> *Range:* qb:ObservationGroup)

Indica um grupo de observações. O domínio desta propriedade é deixada em aberto de modo que um grupo pode estar ligado a diferentes fontes e não precisa ficar restrito a um único *dataset.*

DIMENSÕES, ATRIBUTOS E MEDIDAS

Class: qb:Attachable

Superclasse abstrata para tudo que possa ter atributos e dimensões.

Class: qb:ComponentProperty Sub class of: rdf:Property

Superclasse abstrata de todas as propriedades representando dimensões, atributos e medidas.

Class: qb:DimensionProperty Sub class of: qb:ComponentProperty, qb:CodedProperty

A classe de propriedades de componentes que representam as dimensões do cubo.

Class: qb:AttributeProperty Sub class of: qb:ComponentProperty

A classe de propriedades de componentes que representam os atributos das observações no cubo, equivalente a unidade de medida da medida.

Class: qb:MeasureProperty Sub class of: qb:ComponentProperty

A classe de propriedades do componente que representa o valor medido do fenômeno que está sendo observado.

Class: qb:CodedProperty Sub class of: qb:ComponentProperty

Superclasse de todas as propriedades dos componentes codificados.

PROPRIEDADES DE COMPONENTES DE USO GERAL REUTILIZÁVEIS

Property: qb:measureType *(Domain -> Range:* qb: MeasureProperty *)*

D-TAB genérica de medida, o valor desta D-TAB indica qual medida (a partir do conjunto de medidas definidas no *Data Structure Definition*) está sendo fornecida pela observação.

DEFINIÇÃO DE ESTRUTURA DE DADOS

Class: qb:DataStructureDefinition *sub class of:* qb:ComponentSet

Define a estrutura de um *dataset* ou de uma visão.

Property: qb:structure (*Domain* qb:dataset -> *Range:* qb: DataStructureDefinition)

Indica a estrutura que está em conformidade com o *dataset*.

Property: qb:component (*Domain* qb: DataStructureDefinition -> *Range:* qb: ComponentSpecification)

Indica uma especificação de componente incluída na estrutura do *dataset*.

ESPECIFICAÇÕES DE COMPONENTES – para qualificação de componentes usados em um DSD.

Class: qb: ComponentSpecification *Sub class of:* qb:ComponentSet

Usada para definir propriedades de um componente (atributo, D-TAB, etc) que são específicas para a sua utilização em determinado DSD.

Class: qb:ComponentSet

Classe abstrata de coisas que fazem referência a uma ou mais ComponentProperties.

Property: qb: ComponentProperty (*Domain* qb:ComponentSet -> *Range:* qb: ComponentProperty)

Indica uma ComponentProperty (isto é, um atributo/D-TAB) prevista em um *dataset,* ou uma D-TAB selecionada em um SliceKey.

Property: qb:order (*Domain:* qb:ComponentSpecification -> *Range:* xsd:int)

Indica a ordem de prioridade para apresentação dos componentes – componentes que tenham essa propriedade definida (numerados) são apresentados antes daqueles que não possuem esta propriedade. Dentre aqueles que possuem a propriedade *order* definida, a ordem de apresentação é crescente, isto é, números de ordem inferior são apresentados antes daqueles que possuem ordem superior.

Property: qb:componentRequired (*Domain:* qb:ComponentSpecification -> *Range:* xsd:boolean)

Indica se a propriedade do componente é obrigatória (*True*) ou opcional (*False*) no contexto de um DSD. Aplicável apenas aos componentes que representam atributos. O valor padrão é "*False*".

Property: qb:componentAttachment (*Domain:* qb:ComponentSpecification -> *Range:* rdfs:Class)

Indica em que nível a propriedade do component deve ser anexada.

Property: qb:dimension (*Domain:->Range:* qb:DimensionProperty ; *sub property of:* qb:componentProperty)

É um predicado alternativo ao qb:componentProperty que torna explícito que o componente em questão é uma D-TAB.

Property: qb:measure (*Domain:->Range:* qb:MeasureProperty ; *sub property of:* qb:componentProperty)

É um predicado alternativo ao qb:componentProperty que torna explícito que o componente em questão é uma medida.

Property: qb:attribute (*Domain:->Range:* qb:AttributeProperty ; *sub property of:* qb:componentProperty)

É um predicado alternativo ao qb:componentProperty que torna explícito que o componente em questão é um atributo.

Property: qb:measureDimension (Domain: -> Range: qb:DimensionProperty ; *sub property of:* qb:componentProperty)

É um predicado alternativo ao qb:componentProperty que torna explícito que o componente em questão é uma D-TAB medida.

DEFINIÇÕES DAS VISÕES (*SLICE DEFINITIONS*)

Class: qb:SliceKey *Sub class of:* qb:ComponentSet

Identifica o subconjunto de propriedades dos componentes selecionados para definir uma determinada visão.

Property: qb:sliceStructure (*Domain:* qb:slice -> *Range:* qb:SliceKey)

Identifica a sliceKey de uma determinada visão.

Property: qb:sliceKey (*Domain:* qb:DataSet -> *Range:* qb:SliceKey)

Identifica a sliceKey através da qual podemos construir visões neste *dataset*.

CONCEITOS

*Property:*qb:concept (*Domain:* qb:componentProperty->*Range:* skos:Concept)

Identifica o conceito associado ao que está sendo medido ou ao que está sendo indicado por uma componentProperty.

Property: qb:codeList (*Domain:* qb:codedProperty -> *Range:* owl:UnionOf (skos:ConceptScheme skos:Collection qb:HierarchicalCodeList)

Identifica a "*code list*" associada a codedProperty. Em outras palavras, identifica as instâncias da D-TAB vinculadas à codedProperty.

HIERARQUIAS NÃO-SKOS

Class: qb:HierarquicalCodeList

Representa uma hierarquia genérica dos conceitos que podem ser utilizados para codificação. A hierarquia é definida por uma ou mais raízes, juntamente com uma propriedade que relaciona os conceitos na hierarquia ao seu conceito filho. Os mesmos conceitos podem ser membros de várias hierarquias, desde que diferentes valores qb:parentChildProperty sejam usados para cada hierarquia.

Property: qb:hierarchyRoot (*Domain:* qb:hierarchicalCodeList)

Especifica a raiz de uma hierarquia. Uma hierarquia pode ter diversas raízes, mas a existência de pelo menos uma é obrigatória.

Property: qb:parentChildProperty (*Domain:* hierarchicalCodeList -> *Range:* rdf:Property)

Especifica uma propriedade que relaciona um conceito pai na hierarquia a um conceito filho. Note-se que um filho pode ter mais de um pai.

3.4 Uso do RDF, RDFS e SKOS para representar os modelos dimensionais

O *Data Cube Vocabulay* foi elaborado tendo como objetivo a **publicação** de dados estatísticos armazenados em cubos de dados multidimensionais na *Web.* Conforme vimos na subseção 2.1.7 do presente trabalho, um cubo multidimensional é construído tomando-se os atributos das dimensões declaradas no modelo dimensional original como eixos da estrutura dimensional a partir da qual o cubo de dados será construído. Em um espaço geométrico, cada eixo representa uma D-TAB distinta. Desta maneira, cada atributo das dimensões declaradas no modelo dimensional transforma-se em uma D-TAB no cubo dimensional.

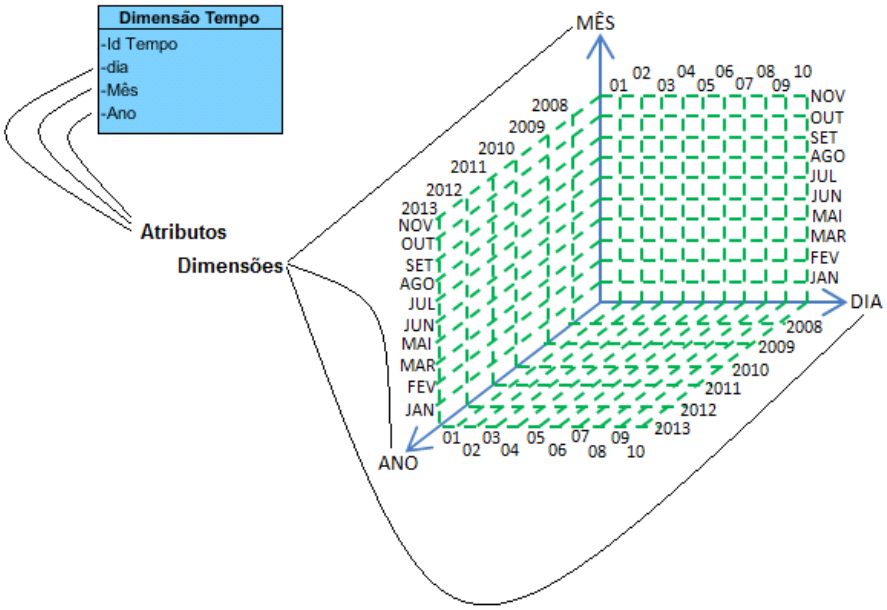


Figura 24: Atributos das Dimensões do Modelo transformam-se em Dimensões no Cubo

Como o próprio nome diz, o *Data Cube Vocabulay* foca no cubo dimensional, particularmente, em sua estrutura. Objetiva-se publicar as informações contidas neste cubo dimensional, em especial as observações ali existentes. Desta forma, quando definimos uma D-TAB através da classe *qb:DimensionProperty*, o que realmente estamos identificando é um atributo de uma D-TAB do Modelo Dimensional materializado pelo cubo. Este atributo comporta-se como um eixo dimensional utilizado para construir a estrutura dimensional do cubo. Isso possui uma implicação semântica muito grave pois a semântica expressa pelo modelo se perde ao desmembrarmos as dimensões em seus atributos e utilizá-los como eixos dimensionais independentes. Por exemplo, no caso da D-TAB Cliente integrante do modelo apresentado na figura 12, quando desmembramos seus atributos (sexo, idade e faixa de renda), transformando-os em eixos dimensionais individuais, perdemos a noção de que esse conjunto de informações representa, na verdade categorias de Clientes que são identificadas pela combinação entre as instâncias dos três atributos originalmente pertencentes a ela.

O *Data Cube Vocabulay* não oferece suporte à representação das dimensões integrantes do modelo dimensional que foi materializado no cubo. As dimensões ali declaradas são meros atributos das dimensões verdadeiramente projetadas originalmente. Portanto, há uma camada semântica

que simplesmente se extingue ao representarmos a estrutura do cubo através do *Data Cube Vocabulay*. Isso não significa que o *Data Cube Vocabulay* tenha sido mal projetado ou pouco cuidadoso. Na verdade há bons motivos para o *Data Cube Vocabulay* não se preocupar com isso: simplesmente porque existem recursos suficientes em outras ontologias para representar esta camada semântica que supostamente se perde.

É importante notar que o *Data Cube Vocabulay* não se propõe a ser uma ontologia para representar modelos dimensionais, mas sim, uma ontologia voltada para suportar a publicação de dados armazenados em cubos dimensionais. Portanto se desejamos representar as estruturas dos modelos dimensionais devemos buscar em outras ontologias elementos que suportem esta representação.

As linguagens RDF e RDFS nos oferecem suporte para representação de diagramas UML (Falkovych, et al., 2003). Um diagrama de classes UML descreve classes de objetos, seus relacionamentos e atributos. Embora um modelo dimensional não corresponda a um diagrama de classes UML, o que ele expressa é algo muito similar: entidades (dimensões), seus relacionamentos (fatos) e atributos (componentes das dimensões).

Como tudo em RDF é considerado um recurso, RDF Schema estabelece que esses recursos podem ser organizados em classes. Um recurso pode ser instância de uma ou mais classes. A propriedade rdf:type é utilizada para indicar as classes das quais um recurso é instância.

As classes podem estar organizadas em uma hierarquia de subclasses. Isso significa que qualquer recurso de um tipo que é subclasse de outro, é também considerado como sendo do tipo da superclasse. Tal relacionamento entre classes é denotado através da propriedade rdfs:subClassOf.

O sistema de tipos de RDF Schema é semelhante ao sistema de tipos do modelo de classes de UML, com algumas pequenas diferenças. Uma delas é que, ao invés de definir classes em termos das propriedades que suas instâncias devem ter, um RDF Schema define propriedades em termos das classes de recursos aos quais elas se aplicam. Este é o papel de rdfs:domain e rdfs:range.

Tomemos como exemplo a D-TAB tempo apresentada na figura 30 desta seção, Tempo seria uma classe, que possui os atributos: dia do tipo integer, mês do tipo integer e ano do tipo integer. Em RDF, teríamos três propriedades: dia,

mês e ano possuindo um domínio Tempo e tendo como range, integer. O quadro 1 dará uma ideia melhor de como essas declarações são feitas utilizando-se RDF, RDFS e SKOS.

```
ex:Dim_Tempo a rdfs:Class, skos:Concept;
  skos:prefLabel "Time Dimension"@en;
  skos:prefLabel "D-TAB Tempo"@pt.

ex:dia  a rdfs:property;
  rdfs:domain ex:Dim_Tempo;
  rdfs:range xsd:integer;
  skos:prefLabel "day"@en;
  skos:prefLabel "dia"@pt;
  skos:broader ex:mes;
  skos:broaderTransitive ex:mes;
  skos:broaderTransitive ex:ano.

ex:mes  a rdfs:property;
  rdfs:domain ex:Dim_Tempo;
  rdfs:range xsd:integer;
  skos:prefLabel "month"@en;
  skos:prefLabel "mes"@pt;
  skos:narrower ex:dia;
  skos:broader ex:ano;
  skos:narrowerTransitive ex:dia;
  skos:broaderTransitive ex:ano.

ex:ano  a rdfs:property;
  rdfs:domain ex:Dim_Tempo;
  rdfs:range xsd:integer;
  skos:prefLabel "year"@en;
  skos:prefLabel "ano"@pt;
  skos:narrower ex:mes;
  skos:narrowerTransitive ex:mes;
  skos:narrowerTransitive ex:dia.
```

Quadro 1: Declaração da Dimensão Tempo em Triplas RDF.

Como podemos observar no quadro 1, o primeiro bloco de sentenças RDF trata da declaração da D-TAB tempo. Em seguida, são apresentados blocos de sentenças que declaram os atributos desta D-TAB, nomeadamente: dia, mês e ano, estabelecendo seu vínculo como uma propriedade que possui como domínio a D-TAB tempo declarada anteriormente e estabelecendo através dos termos skos:broader e skos:narrower, a hierarquia entre eles. Obviamente existem diversas outras triplas que podemos formar e enriquecer ainda mais a semântica desta declaração, além de podermos estabelecer

conexões com termos de outros tesauros RDF. No entanto, nossa proposta nessa seção era apenas alertar para a necessidade de se declarar através de sentenças RDF a estrutura de metadados do modelo dimensional para que esta camada semântica não se perdesse. Mais adiante veremos que esta camada semântica será fundamental para que o catálogo possa vir a ser usado como base para que se possa criar SQLs dinâmicamente para acessar bancos de dados relacionais e recuperar os dado ali armazenados.

3.5
Triplificação de Cubos de Dados

Pode-se seguir diferentes estratégias para representar um cubo de dados em RDF. Em um dos extremos, temos a estratégia de triplificação de todo o cubo incluindo seus dados (conteúdo das dimensões e da F-TAB) e metadados. No outro, temos a estratégia tripla-simples que representa o cubo como um única tripla. Nesta dissertação, apenas a estratégia da triplificação completa será abordada.

Seja $R \subseteq D_1 \times \ldots \times D_m \times A_1 \times \ldots \times A_n \times O$ uma relação que representa um cubo, como vimos na seção 2.1.5. Segundo Noy e Rector (Noy, et al., 2006), podemos representar R por uma reificação, criando uma nova classe r e tratar as dimensões, atributos e observação medida como propriedades.

Assim, uma tupla $(x_1, \ldots, x_m, y_1, \ldots, y_n, z)$ em R é representada por $m+n+1$ triplas $(u, d_1, x_1), \ldots, (u, d_m, x_m), \ldots, (u, a_1, y_1), \ldots, (u, a_n, y_n), (u, o, z)$.

A identificação destas triplas, segundo os princípios de *Linked Data*, deve ser realizado através do emprego de URIs. No entanto, no caso das observações registradas em um cubo, atribuir a cada uma delas um URI normalmente será desnecessário uma vez que dificilmente há coincidência no registro de duas observações distintas. Na verdade seguindo os conceitos apresentados na seção 2.1, podemos garantir a independência de cada observação registrada em um cubo das demais observações existentes naquele cubo.

Para acessar o conteúdo das observações registradas nos cubos de dados adotaremos a linguagem de mapeamento R2RML apresentada no apêndice H desta dissertação.

3.5.1
Definição das Dimensões e Fatos

Antes de nos aprofundarmos na questão da triplificação de um modelo dimensional, é necessário entendermos que existe uma diferença entre o modelo dimensional e o cubo de dados.

Um cubo é a projeção de uma estrutura dimensional construída a partir dos atributos das dimensões e das medidas declaradas nas F-TABs apresentadas no modelo dimensional.

A construção da estrutura do cubo no *Data Cube Vocabulary* "desfaz" a estrutura das dimensões declaradas no modelo dimensional, transformando cada atributo de cada D-TAB em uma *DimensionProperty* independente, sem que haja qualquer informação que permita associar esta propriedade à estrutura da D-TAB da qual ela originalmente fazia parte.

Assim, a estrutura de um cubo declarada em *Data Cube Vocabulary* ocasiona uma perda semântica uma vez que deixamos de saber quais eram as dimensões originais declaradas no Modelo Dimensional, substituindo cada uma delas pelos seus atributos declarados como *DimensionProperty*.

Outra questão importante aqui envolvida tem a ver com a capacidade de se manipular estruturas com dezenas ou centenas de dimensões. Embora não haja limite para a quantidade de dimensões que uma determinada F-TAB possa utilizar, existe algum consenso entre os desenvolvedores de sistemas de BI, que o número deve variar de algumas poucas unidades (tipicamente 3, 4 ou 5) até umas poucas dezenas (entre 15 e 30) de dimensões.

Na prática observamos que no decorrer de cerca de 15 anos atuando no desenvolvimento de projetos de Data Warehouse, dificilmente chegamos a mais do que 15 dimensões em uma mesma F-TAB. Porém é importante salientar que muitas dimensões costumam ter dezenas de atributos. Assim, vamos supor um modelo dimensional que possua cerca de 12 dimensões, com uma média de 8 atributos por D-TAB. Neste caso, teríamos 96 atributos sendo transformados em eixos dimensionais para produzir o cubo de dados. Se utilizarmos apenas estas triplas construídas a partir do *Data Cube Vocabulary*, que empregamos na declaração da estrutura do cubo para oferecer uma visão

da estrutura dimensional ao usuário final, o que estaremos oferecendo a ele, será uma estrutura composta por 96 dimensões.

A visualização de uma estrutura composta por 96 dimensões, no lugar de uma estrutura composta por 12 dimensões, cada qual contendo 8 atributos, é muito mais difícil, não apenas, de compreender, como de se manipular.

Por fim, ao pulverizar os atributos sem manter um vínculo entre estes e a D-TAB à qual eles originalmente pertenciam, descaracterizamos as classes de informação de negócios representadas por cada D-TAB. O efeito provocado por esta descaracterização é o que chamamos de "*gap*" semântico uma vez que ela provoca uma perda de significado e de compreensão do modelo dimensional original. Isto ocorre porque o *Data Cube Vocabulary* foca na representação da estrutura do cubo dimensional e não na representação do modelo dimensional a partir do qual o cubo é produzido. Vale ressaltar que não houve desleixo por parte dos autores do *Data Cube Vocabulary* em relação ao modelo dimensional e a possibilidade de representa-lo através de triplas. Na verdade, como veremos adiante, é plenamente possível declarar o modelo dimensional utilizando outras ontologias.

Assim, a solução que propomos para superar este "*gap*" semântico é a representação dos modelos dimensionais, para oferecer uma visão organizada e estruturada deles, através da declaração de triplas que os representem. Acreditamos que tal representação se faz não apenas desejável mas necessária.

Vejamos como podemos declarar uma D-TAB através de triplas RDF, RDFS e SKOS. Tendo por base a D-TAB Dim_Causa_Morte declarada no modelo apresentado na figura 52, temos:

```
# ==================
# Declaração da Dim_Causa_Morte e seus atributos
# ==================
1   ex:dim3CausaMorte a rdfs:Class, skos:Concept;
2      rdfs:label "Dim_Causa_Morte"@pt;
3   rdfs:comment "Dimensão que contém as categorias e subcategorias de
4    óbitos."@pt.


5  ex:idDim3CausaMorte7 a rdf:Property;
```

```
6   rdfs:domain ex:dim3CausaMorte;
7   rdfs:range xsd:integer;
8   rdfs:label "id_dim_causa_morte"@pt;
9   rdfs:comment "Chave primária da D-TAB Causa Morte."@pt.

10  ex:dimAxisSubCategoria8 a rdf:Property, qb:DimensionProperty;
11    rdfs:domain ex:dim3CausaMorte;
12    rdfs:range ex-class:dimAxisSubCategoria8;
13    rdfs:label "sub_categoria"@pt;
14    rdfs:comment "Sub categoria de óbito. Ex: categoria morte natural,
15    subcategoria infarto, câncer, etc."@pt;
16    skos:broader ex:dimAxisCategoria9;
17    skos:broaderTransitive ex:dimAxisCategoria9.

18  ex:dimAxisCategoria9 a rdf:Property, qb:DimensionProperty;
19    rdfs:domain ex:dim3CausaMorte;
20    rdfs:range ex-class:dimAxisCategoria9;
21    rdfs:label "categoria"@pt;
22    rdfs:comment "Categoria de óbito. Ex: categoria morte natural,
23    assassinato, morte acidental, suicídio, etc."@pt;
24    skos:narrower ex:dimAxisSubCategoria8;
25    skos:narrowerTransitive ex:dimAxisSubCategoria8.
```

Quadro 2:Triplificação da Dimensão Causa Morte

A declaração desta D-TAB é dada pelas sentenças 1 à 4 apresentadas no quadro 2. Através destas sentenças declaramos a D-TAB Causa Morte como uma classe (rdfs:Class) e um conceito SKOS (skos:Concept) e lhe atribuímos um label e um comentário.

O próximo passo é declarar sua estrutura, isto é, os atributos que fazem parte dela. Em RDF fazemos isso criando propriedades cujo domínio é a classe que acabamos de descrever.

Há uma pequena diferença entre a declaração da chave primária e a declaração dos demais atributos da D-TAB. No caso da chave primária, a declaração é formada pelas sentenças 5 à 9.

Como podemos observar, declaramos a propriedade tendo por domínio a classe (D-TAB) Causa Morte, atribuindo-lhe um label, um comentário e um *range* que define uma restrição sobre a natureza dos valores que suas instâncias podem assumir.

Para os demais atributos, no entanto, temos algumas ligeiras diferenças na declaração das triplas, como podemos observar nas sentenças compreendidas entre as linhas 10 e 25 do quadro 2.

Neste caso, além de identificar a propriedade como uma rdf:Property, também a declaramos como uma qb:DimensionProperty para podermos utilizá-la na construção dos cubos. Outro aspecto relevante é que neste caso utilizamos uma classe (e não um *datatype*) para estabelecer o *range* da propriedade. Como veremos mais adiante este será um artifício fundamental para podermos estabelecer o vínculo entre a propriedade e a sentença SQL que utilizaremos para recuperar as instâncias de cada propriedade declarada. Como no caso anterior, atribuímos um label e um comentário à propriedade e, finalmente, como esta propriedade faz parte de uma hierarquia utilizamos skos:broader para declarar a hierarquia entre as propriedades que dela participam.

A declaração de uma F-TAB, por sua vez, pode ser obtida através do seguinte grupo de declarações:

```
# Declaração da Tabela Fato_Mortalidade e seus atributos

1  ex:fato1Mortalidade a rdfs:Class, skos:Concept;
2    rdfs:label "Fato_Mortalidade"@pt;
3   rdfs:comment "Fato que registra os óbitos por tipo de morte por município,
4   tempo e características do falecido."@pt.

5  ex:idfato1DimCaractPopulacao24 a rdf:property;
6    rdfs:domain ex:fato1Mortalidade;
7    rdfs:range xsd:integer;
8    rdfs:label "id_dim_caract_populacao"@pt;
9     rdfs:comment "Chave estrangeira da Dim_Caract_Populacao."@pt;
10   skos:exactMatch ex:idDim2CaractPopulacao3.
```

```
11  ex:idfato1DimCausaMorte25 a rdf:property;
12   rdfs:domain ex:fato1Mortalidade;
13   rdfs:range xsd:integer;
14   rdfs:label "id_dim_causa_morte"@pt;
15   rdfs:comment "Chave estrangeira da Dim_Causa_Morte."@pt;
16   skos:exactMatch ex:idDim3CausaMorte7.

17  ex:idfato1DimLocalidade26 a rdf:property;
18   rdfs:domain ex:fato1Mortalidade;
19   rdfs:range xsd:integer;
20   rdfs:label "id_dim_localidade"@pt;
21   rdfs:comment "Chave estrangeira da Dim_Localidade."@pt;
22   skos:exactMatch ex:idDim4Localidade10.

23  ex:idfato1DimTempo27 a rdf:property;
24   rdfs:domain ex:fato1Mortalidade;
25   rdfs:range xsd:integer;
26   rdfs:label "id_dim_tempo"@pt;
27   rdfs:comment "Chave estrangeira da Dim_Tempo."@pt;
28   skos:exactMatch ex:idDim5Tempo14.

29  ex:qtdObitos28 a rdf:property, qb:MeasureProperty;
30   rdfs:domain ex:fato1Mortalidade;
31   rdfs:range xsd:Integer;
32   rdfs:label "qtd_obitos"@pt;
33   rdfs:comment "Quantidade de óbitos (pessoas falecidas)."@pt;
34   sdmx-attribute:unitMeasure unit:Count.
```

Quadro 3: Triplificação da Fato Mortalidade

Tal e qual a definição da D-TAB, a fato também é decalrada como sendo uma classe RDF, como podemos observar pelas sentenças compreendidas entre as linhas 1 e 4 do quadro 3.

Conforme sabemos, uma F-TAB não possui chave artificial atuando como chave primária. Sua chave primária é formada pelo conjunto de chaves

estrangeiras nela declaradas. Além das chaves estrangeiras, as únicas colunas que ela pode possuir representam, necessariamente, as medidas que nela serão registradas. Assim a declaração padrão que adotamos para as chaves estrangeiras que compõem a chave primária da F-TAB está representada entre as linhas 11 e 16 do quadro 3.

Esta declaração segue o mesmo padrão adotado para a declaração da chave primária das dimensões com uma diferença: ela termina com uma cláulula skos:exactMatch que nos permite estabelecer o vínculo entre a chave estrangeira da F-TAB em questão e a D-TAB à qual ela pertence.

Reproduzimos no do apêndice E desta dissertação o conjunto de triplas que será gerado a partir de cada uma das dimensões e F-TABs identificados no modelo dimensional apresentado na figura 52, aplicando os conceitos anteriormente apresentados.

3.5.2
Definição dos Cubos

Conforme mencionado anteriormente, a partir dos modelos dimensionais, cubos de dados são projetados e materializam estruturas de consulta de alto desempenho que são utilizadas pelos usuários de sistemas de *BI*. Apesar dos modelos dimensionais serem necessários para capturar a semântica que deu origem ao cubo de dado, as operações de consulta são realizadas sobre o cubo de dados. Desta forma é necessário declararmos as estruturas dos cubos de dados gerados a partir das F-TABs identificadas no modelo. As triplas do cubo de dados serão construídas a partir do *Data Cube Vocabulary*. Como vimos anteriormente, começamos a introduzir declarações do *Data Cube Vocabulary* entre aquelas que formulamos para os modelos dimensionais. Já declaramos todas *dimensions properties*, *measures properties* e *units measures*.

Um típico cubo de dados declarado em *Data Cube Vocabulary* pode ser visto abaixo.

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
@prefix qb: <http://purl.org/linked-data/cube#>.
@prefix ex-class: <http://purl.org/GovDataCube/classes/>.
@prefix ex-property: <http://purl.org/GovDataCube/properties/> .
@prefix ex-resource: <http://purl.org/GovDataCube/resources/> .
```

```
@prefix sdmx-dimension: <http://purl.org/linked-data/sdmx/2009/dimension#>.
@prefix sdmx-attribute: <http://purl.org/linked-data/sdmx/2009/attribute#>.
@prefix sdmx-measure: <http://purl.org/linked-data/sdmx/2009/measure#>.
@prefix sdmx-concept: <http://purl.org/linked-data/sdmx/2009/concept#>.
```

EXEMPLO DE COMO SE DECLARA UM CUBO DE DADOS

1) Declaração do dataset

```
ex-resource:dataset-residents a qb:DataSet;
 rdfs:label "Number of residents";
 rdfs:comment "The number of residents in Brazil of each sex, race and range of age
by area and time. Dimensions: DimRace, DimSex, DimAge, DimArea and
DimYear.";
```

2) Indica que a estrutura do dataset será identificada por ex´resource:dsd-residents

```
qb:structure ex-resource:dsd-residents.
```

3) Declaração da estrutura do cubo, indicando dimensões, métrica e unidade de medida

```
ex-resource:dsd-residents a qb:DataStructureDefinition;
 # The dimensions
 qb:component [qb:dimension ex-resource:dimRace];
 qb:component [qb:dimension ex-resource:dimSex];
 qb:component [qb:dimension ex-resource:dimAge];
 qb:component [qb:dimension ex-resource:dimYear];
 qb:component [qb:dimension ex-resource:dimArea];

 # The measure

 qb:component [qb:measure ex-property:numberResidents];

 # The attributes
 qb:component [qb:attribute sdmx-attribute:unitMeasure; qb:componentAttachment
 qb:DataSet;].
```

4) Os componentes do cubo precisam ser declarados como propriedades. Abaixo temos a definição dessas propriedades com a definição dos valores que cada propriedade pode assumir

```
ex-property:numberResidents a rdf:Property, qb:MeasureProperty;
 rdfs:label "number of residents";
 rdfs:subPropertyOf sdmx-measure:obsValue;
 rdfs:range xsd:integer .

ex-resource:dimRace a qb:DimensionProperty, rdf:Property;
 rdfs:label "dimension Race";
 rdfs:range ex-class:Race.

ex-resource:dimSex a qb:DimensionProperty, rdf:Property;
 rdfs:label "dimension Sex";
 rdfs:range ex-class:Sex.

ex-resource:dimAge a qb:DimensionProperty, rdf:Property;
 rdfs:label "dimension Age";
 rdfs:range ex-class:Age.

ex-resource:dimYear a qb:DimensionProperty, rdf:Property;
 rdfs:label "dimension Year";
```

```
 rdfs:subProperyOf sdmx-dimension:refTime;
 qb:concept sdmx-concept:refTime;
 rdfs:range ex-class:Year.

ex-resource:dimArea a qb:DimensionProperty, rdf:Property;
 rdfs:label "dimension Area";
 rdfs:range ex-class:Area.


  5) Definição das classes cujas instâncias representam os valores que podem ser
     assumidos pelas propriedades
ex-class:Race rdf:type rdfs:Class;
 rdfs:comment "Represents a race";
 rdfs:label "Race".

ex-class:Sex rdf:type rdfs:Class;
 rdfs:comment "Represents a sex gender";
 rdfs:label "Sex".

ex-class:Age rdf:type rdfs:Class;
 rdfs:comment "Represents a age";
 rdfs:label "Age".

ex-class:Year rdf:type rdfs:Class;
 rdfs:comment "Represents a year";
 rdfs:label "Year".

ex-class:Area rdf:type rdfs:Class;
 rdfs:comment "Represents a area";
 rdfs:label "Area“.
```

Quadro 4: Exemplo da declaração de um Cubo de Dados em *Data Cube Vocabulary*

Analisando as triplas que declaramos no quadro 4, fica claro que os conjuntos de triplas que nos falta declarar são aqueles que definem o dataset e a sua estrutura. A partir de cada F-TAB projetamos um cubo de dados. Assim, como em nosso modelo temos quatro F-TABs declaradas, também teremos quatro cubos de dados que precisam ser declarados. Como as propriedades já foram devidamente declaradas, só precisamos focar na declaração dos datasets e da estrutura de cada um deles.

No apêndice E apresentamos as declarações dos datasets e suas estruturas para o modelo apresentado na figura 52.

3.5.3
Mapeamento dos Cubos de Dados Interligados para Tabelas

O último conjunto de triplas que precisamos fornecer ao *Catalog* para que os demais módulos do nosso *framework* possam desempenhar as suas funções são aquelas que serão escritas em R2RML. Através delas viabilizamos o acesso às instâncias tanto das dimensões como das observações registradas nas F-TABs.

Como estamos tratando de triplas confeccionadas para recuperar instâncias persistidas em algum banco de dados, necessitamos especificar as informações de conexão ao banco de dados.

```
#
#====== INICIO DECLARACAO DAS CONSULTAS R2RML
#


#===== ESTA PARTE SERÁ INCLUÍDA DENTRO DO WRAPPER ====
ex-resource:databaseBase_DW a d2rq:Database;
 d2rq:username "user";
 d2rq:password "pass";
 d2rq:jdbcDSN "jdbc:oracle:thin:@192.168.0.48:1521:XE";
 d2rq:jdbcDriver "oracle.jdbc.OracleDriver"".
#== FIM DA PARTE QUE SERÁ INCLUÍDA DENTRO DO WRAPPER ==
```

Quadro 5: Triplas R2RML para conexão ao banco de dados

Conforme mencionamos anteriormente, a fim de garantir a segurança dos dados das instituições credenciadas, criamos um serviço dentro do módulo Wrapper que irá ser acionado pelo *Mediator* e receberá as triplas R2RML confeccionadas pela DCD Tool às quais adicionará as triplas acima apresentadas para, então, utilizar o DB2Triples para recuperar as instâncias triplificadas tanto das dimensões quanto das observações registradas nas F-TABs.

Como cada atributo das dimensões originais do modelo dimensional torna-se uma propriedade (qb:DimensionProperty) do cubo de dados, precisamos declarar, para cada uma delas, um conjunto de triplas em R2RML que irá recuperar as suas instâncias.

```
#
# Declaração para recuperar as instâncias do Eixo Dimensional Faixa_Etaria
```

```
#


ex-resource:TriplesMapdimAxisFaixaEtaria4 a rr:TriplesMap;
 d2rq:dataStorage ex-resource:databaseBase_DW;
 rr:logicalTable [ rr:sqlQuery """
  SELECT DISTINCT
md5('www.fgv.br'||'BASE_DW'||'DIM_CARACT_POPULACAO'||'FAIXA_ETARIA'||TO_CH
AR(ID)||FAIXA_ETARIA) as faixa_etaria_md5,
    faixa_etaria, id
  FROM Dim_Caract_Populacao """; ];
 rr:subjectMap [
  rr:template "http://data.example.com/caract_populacao/faixa_etaria/{faixa_etaria_md5}";
  rr:class ex-class:dimAxisFaixaEtaria4; ];
 rr:predicateObjectMap [
  rr:predicate rdfs:label;
  rr:objectMap [ rr:column "faixa_etaria"; rr:language "pt" ];].
```

Quadro 6: Triplas R2RML para recuperar instâncias de Faixa Etária

Através da classe "ex-class:dimAxisFaixaEtaria4" declarada anteriormente quando geramos o conjunto de triplas de definição dos cubos, estabelecemos a conexão entre a propriedade do cubo de dados identificada por ex:dimAxisFaixaEtaria4 e que tem como rdfs:range a classe "ex-class:dimAxisFaixaEtaria4". Este procedimento será repetido para cada uma das "qb:DimensionProperty" definidas anteriormente, a fim de garantir a recuperação das instâncias de cada uma delas.

Para criar as URIs das instâncias, tanto das dimensões como das observações, utilizamos a função MD5 *Hash Generator*. Trata-se de um algoritmo de *hash* que cria uma chave de 128 bits a partir de uma string fornecida como entrada. Este algoritmo foi desenvolvido pela RSA Data Security Inc.

A string utilizada para gerar a chave de 128 bits deve evitar possíveis colisões. Assim, como independente do tamanho da entrada, uma chave de 128 bits é gerada, escolhemos a concatenação de um conjunto de informações que garantirá a unicidade da string de entrada: o domínio Web da instituição credenciada, o nome do banco de dados, o nome da tabela, o nome da coluna onde está a informação que será recuperada, o id da instância recuperada e o conteúdo da coluna. Este grupo de informações concatenada garante a

unicidade da string de entrada pois não podem haver duas tabelas com mesmo nome no mesmo banco de dados, nem duas colunas com o mesmo nome na mesma tabela e tampouco duas instâncias com o mesmo id, quando este é gerado por uma sequencia numérica. Desta forma garantimos que a chave gerada será realmente única e que não teremos problemas de colisão na geração das chaves de 128 bits que funcionarão como URIs das instâncias.

O próximo passo é declarar as triplas que nos permitirão recuperar as instâncias das observações registradas em algum dos cubos de dados que declaramos anteriormente através das triplas redigidas em *Data Cube Vocabulary*. Para realizar esta declaração, precisamos, antes de tudo, declarar todas as *triplesmap* correspondentes a cada atributo de cada D-TAB que faz parte do modelo dimensional que dá origem ao cubo. A declaração em R2RML da consulta à F-TAB faz referência às *triplesmaps* de cada propriedade (qb:DimensionProperty) declarada para o cubo de dados estabelecendo o vínculo entre as observações medidas ali registradas e as respectivas instâncias das propriedades que definem aquele registro.

```
  #========== CUBO POPULACAO_ABSOLUTA ============
01 ex:TriplesMapFatoPopulacaoAbsoluta a rr:TriplesMap;
02  d2rq:dataStorage ex:databaseResidents;
03  rr:logicalTable [ rr:sqlQuery "“”
04       SELECT id_dim_caract_populacao, id_tempo, id_localidade, qtd_pessoas,
qtd_matriculas_escolar
05      FROM fato_populacao_absoluta """; ];
06  rr:subjectMap [
07  rr:template
08
"http://purl.org/GovDataCube/resources/cubopopulacaoabsoluta/observations/{id_
dim_caract_populacao}_{id_tempo}_{id_localidade}_{qtd_pessoas}_{qtd_matric
ulas_escolar}";
09  rr:class qb:observation; ];
10  rr:predicateObjectMap [
11  rr:predicate qb:dataSet;
12  rr:objectMap [rr:constant ex:cubo-populacao-absoluta]; ];
13  rr:predicateObjectMap [
```

```
14  rr:predicate ex:dimAxisFaixaEtaria4;
15  rr:objectMap [
16  rr:parentTriplesMap ex:TriplesMapdimAxisFaixaEtaria4;
17  rr:joinCondition [
18  rr:child "id_dim_caract_populacao";
19  rr:parent "id"; ];];];
20  rr:predicateObjectMap [
21  rr:predicate ex:dimAxisFlagUrbana5;
21  rr:objectMap [
22  rr:parentTriplesMap ex:TriplesMapdimAxisFlagUrbana5;
23  rr:joinCondition [
24  rr:child "id_dim_caract_populacao";
25  rr:parent "id"; ];];];
26  rr:predicateObjectMap [
27  rr:predicate ex:dimAxisSexo6;
28  rr:objectMap [
29  rr:parentTriplesMap ex:TriplesMapdimAxisSexo6;
30  rr:joinCondition [
31  rr:child "id_dim_caract_populacao";
32  rr:parent "id"; ];];];
33  rr:predicateObjectMap [
34  rr:predicate ex:dimAxisMunicipio11;
35  rr:objectMap [
36  rr:parentTriplesMap ex:TriplesMapdimAxisMunicipio11;
37  rr:joinCondition [
38  rr:child "id_dim_localidade";
39  rr:parent "id"; ];];];
40  rr:predicateObjectMap [
41  rr:predicate ex:dimAxisUf12;
42  rr:objectMap [
43  rr:parentTriplesMap ex:TriplesMapdimAxisUf12;
44  rr:joinCondition [
45  rr:child "id_dim_localidade";
46  rr:parent "id"; ];];];
47  rr:predicateObjectMap [
```

```
48  rr:predicate ex:dimAxisRegiao13;
49  rr:objectMap [
50  rr:parentTriplesMap ex:TriplesMapdimAxisRegiao13;
51  rr:joinCondition [
52  rr:child "id_dim_localidade";
53  rr:parent "id"; ];];];
54  rr:predicateObjectMap [
55  rr:predicate ex:dimAxisBimestre15;
56  rr:objectMap [
57  rr:parentTriplesMap ex:TriplesMapdimAxisBimestre15;
58  rr:joinCondition [
59  rr:child "id_dim_tempo";
60  rr:parent "id"; ];];];
61  rr:predicateObjectMap [
62  rr:predicate ex:dimAxisMesNome16;
63  rr:objectMap [
64  rr:parentTriplesMap ex:TriplesMapdimAxisMesNome16;
65  rr:joinCondition [
66  rr:child "id_dim_tempo";
67  rr:parent "id"; ];];];
68  rr:predicateObjectMap [
69  rr:predicate ex:dimAxisSemestre17;
70  rr:objectMap [
71  rr:parentTriplesMap ex:TriplesMapdimAxisSemestre17;
72  rr:joinCondition [
73  rr:child "id_dim_tempo";
74  rr:parent "id"; ];];];
75  rr:predicateObjectMap [
76  rr:predicate ex:dimAxisTrimestre18;
77  rr:objectMap [
78  rr:parentTriplesMap ex:TriplesMapdimAxisTrimestre18;
79  rr:joinCondition [
80  rr:child "id_dim_tempo";
81  rr:parent "id"; ];];];
82  rr:predicateObjectMap [
```

```
83  rr:predicate ex:dimAxisMesNum19;
84  rr:objectMap [
85  rr:parentTriplesMap ex:TriplesMapdimAxisMesNum19;
86  rr:joinCondition [
87  rr:child "id_dim_tempo";
88  rr:parent "id"; ];];];
89  rr:predicateObjectMap [
90  rr:predicate ex:dimAxisAno20;
91  rr:objectMap [
92  rr:parentTriplesMap ex:TriplesMapdimAxisAno20;
93  rr:joinCondition [
94  rr:child "id_dim_tempo";
95  rr:parent "id"; ];];];
96  rr:predicateObjectMap [
97  rr:predicate ex:qtdPessoas32;
98  rr:objectMap [rr:column "qtd_pessoas"]; ];
99  rr:predicateObjectMap [
100 rr:predicate ex:qtdMatriculasEscolar33;
101 rr:objectMap [rr:column "qtd_matriculas_escolar"]; ].
```

Quadro 7: Triplas R2RML para recuperar as triplas do Cubo População Absoluta

Segundo o modelo da figura 52 temos três dimensões: Dim_Localidade, Dim_Caract_População e Dim_Tempo definindo os fatos registrados na Fato_População_Absoluta. Para cada um dos atributos destas dimensões, precisamos definir consultas R2RML para recuperar suas instâncias. É precisamente o que fazemos no apêndice G desta dissertação, onde todas as triplas R2RML, para cada atributo de D-TAB estão declaradas e “ligadas” às propriedades (qb:DimensionProperty) definidas como dimensões dos cubos declarados no apêndice F. Também temos neste apêndice as declarações das sentenças em R2RML que definem as consultas que recuperam as instâncias das observações dos cubos.

A conexão entre as observações e as instâncias das dimensões é obtida pelo seguinte conjunto de triplas apresentados entre as linhas 10 e 19 do quadro 7.

A declaração rr:parentTriplesMap formaliza a conexão entre as instâncias recuperadas através da declaração rr:TriplesMap identificada por ex:TriplesMapdimAxisAno20 e as instâncias das observações recuperadas pela sentença SQL usando como critério de junção, as colunas “id” e “id_dim_tempo”. Além disso, a declaração rr:predicate estabelece a conexão com a qd:DimensionProperty ex:dimAxisAno20 declarada anteriormente no apêndice F (triplas declaradas em *Data Cube Vocabulary*).

Assim, como podemos constatar, temos uma perfeita conexão entre o conjunto de triplas em RDF, RDFS e SKOS que definem os modelos dimensionais, as triplas delaradas em *Data Cube Vocabulary* que definem os cubos de dados projetados a partir dos modelos dimensionais e, finalmente, as triplas declaradas em R2RML que recuperam as instâncias das observações e das propriedades dimensões dos cubos.

As triplas geradas são então carrgadas para o *Catalog* do *framework Olap2DataCube Catalog on Demand* permitindo que os demais componentes possam desempenhar suas tarefas e oferecer às aplicações usuárias um acesso direto ao conteúdo de múltiplas bases governamentais através de um único canal.

3.6 Resumo

Neste capítulo apresentamos os vocabulários SKOS e *Data Cube Vocabulary* que utilizamos na triplificação de modelos dimensionais e cubos de dados. Mostramos, não apenas a necessidade, mas também, como complementar as declarações em *Data Cube Vocabulary* com triplas em RDF, RDFS e SKOS, a fim de estabelecermos um mapeamento entre a estrutura do cubo de dados e o modelo dimensional que lhe deu origem. Por fim, apresentamos as declarações em R2RML que permitem a recuperação das instâncias dos cubos.

4.
O Framework OLAP2DataCube Catalog On Demand

4.1
O Framework OLAP2DataCube Catalog on Demand

A fim de viabilizar o compartilhamento dos dados estatísticos residentes em múltiplas plataformas e bancos distribuídos através das diversas instituições governamentais, criamos um framework intitulado *OLAP2DataCube Catalog On Demand*. Este framework possui diversos módulos, dentre os quais aquele que é objeto da presente dissertação: o *Data Cube Discovery Tool*. Através deste framework, aplicações "clientes" poderão visualizar qualquer um dos cubos de dados nele catalogados.

Este framework propõe-se a oferecer um ambiente de implementação para consultas federadas (Maia, et al., 2012). Esta abordagem propõe-se a disponibilizar, para usuários ou aplicações, o acesso a múltiplas fontes de dados autônomas através de um vocabulário padrão especificado em uma ontologia de domínio. As consultas realizadas são enviadas para um mediador que utiliza informações carregadas em um catálogo que permite localizar quais as bases que possuem dados pertinentes à consulta solicitada, além das necessárias para conectar e acessar a fonte de origem. Com base nessas informações, o mediador decompõe a consulta original em subconsultas, direcionando cada uma delas às suas fontes de origem. Após a recuperação dos dados nas múltiplas fontes de origem, o mediador consolida a resposta à consulta solicitada usando o mesmo vocabulário e entrega a informação consolidada para consumo da aplicação usuária.

A arquitetura proposta no framework *OLAP2DataCube Catalog On Demand* também sofreu forte influência do padrão *Crawling Data* (Heath, et al., 2011), que, por sua vez, é inspirada na clássica arquitetura de motores de busca da Web como Google e Yahoo.

A seguir, iremos detalhar os seis módulos que compõem o *framework OLAP2DataCube Catalog On Demand*: *Client Application*, *Mediator*, *Catalog*, *Enriching*, *DCD Tool* e *Wrapper*.

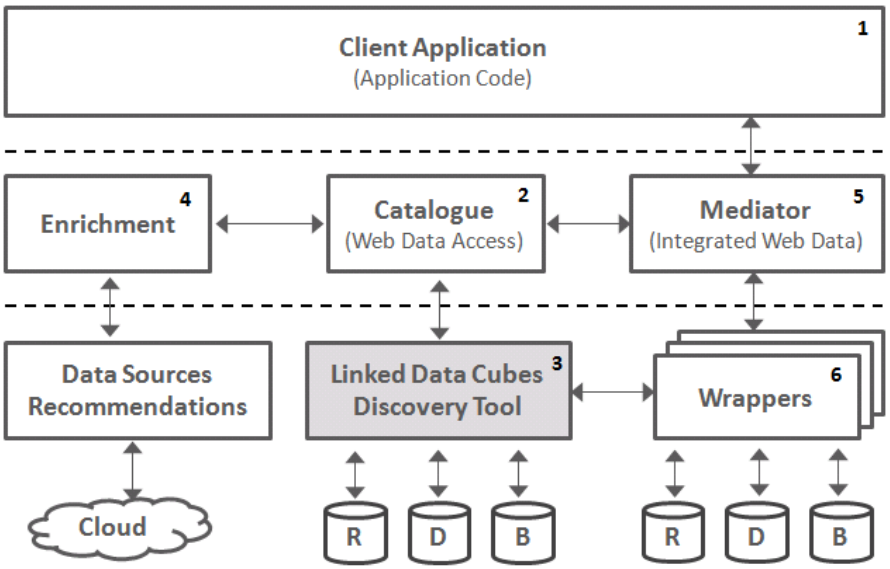


Figura 25: Arquitetura do OLAP2DataCube Catalog On Demand

4.1.1 Client Application

É a camada onde estão localizadas as aplicações clientes que se comunicam com o mediador. O protocolo de comunicação entre essas aplicações e o mediador está baseado nas ontologias utilizadas pelo *framework*, tais como o *Data Cube Vocabulary, o Skos e o R2RML*. Uma *Client Application* se comunica com o *Mediator* para identificar quais cubos estão disponíveis através do framework em um dado instante, bem como os tipos de informações armazenadas em cada um deles. Também são informadas as estruturas de cada cubo (dimensões e medidas) e seus metadados.

A apresentação destas informações disponibilizadas pelo *Mediator* para a *Client Application* é de responsabilidade exclusiva desta última. Além de permitir que um usuário recupere informações sobre quais cubos estão disponíveis e sua estrutura, a *client application* também deve ser capaz de solicitar ao *Mediator* a recuperação dos dados de um cubo específico. Por fim, a *client application* deve oferecer ao usuário funcionalidades voltadas para a

exploração dos cubos e de formatação dos dados recuperados através de relatórios estáticos ou dinâmicos.

4.1.2 Catalog

Estrutura central onde estão armazenados os cubos disponíveis para manipulações do usuário, provê ao *Mediator* as informações que este repassa a qualquer *Client Application* bem como os dados necessários para recuperar as observações solicitadas por ela a partir de determinado cubo de dados.

O mapeamento entre os cubos e os seus respectivos RDB's é estabelecido através da definição dos metadados que permitem representar os modelos dimensionais correspondentes à visão de cada cubo de dado, bem como da estrutura dos RDB's onde podem ser encontradas as observações registradas em cada um deles.

O *Catalog* estabelece a relação entre cada entidade do modelo dimensional e a(s) respectiva(s) tabela(s) onde as instâncias daquela entidade encontram-se persistidas no banco de dados de origem. Por último o *Catalog* contém informações sobre o banco de dados onde os modelos dimensionais foram persistidos que permitirão ao *Mediator* solicitar ao *Wrapper* a execução de consultas que retornarão as instâncias triplificadas dos dados armazenados nos bancos de dados de origem.

Além do *Mediator* que, unicamente, consome os dados aqui armazenados, o *Catalog* também interage com outros dois módulos do *framework*: o *Enriching* e o *DCD Tool*.

Esta camada constitui-se exclusivamente de um banco de dados de triplas disponibilizado para o acesso do *Mediator*.

4.1.3 Mediator

O *Mediator* é o módulo que realiza toda intermediação entre as aplicações clientes e o *framework*. Ele é o elemento central da arquitetura que utiliza o *Catalog* para responder às solicitações realizadas pelas aplicações clientes. Quando a solicitação realizada pela aplicação cliente necessitar

acessar os dados armazenados nas bases de origem, é o *Mediator* que recupera do *Catalog* as triplas necessárias para realizar a solicitação ao *Wrapper*, recebe a resposta deste e organiza as triplas que serão retornadas à aplicação cliente como resposta à solicitação realizada inicialmente.

Da mesma forma também é através do *Mediator* que uma *Client Application* consegue obter informações sobre cada cubo disponível e o seu conteúdo.

4.1.4 Wrapper

O *Wrapper* é o módulo responsável por realizar a comunicação entre o *framework* e os bancos de dados originais onde estão armazenadas as informações estatísticas. A fim de garantir a segurança no acesso ao dado no banco de dados de origem, o *Wrapper* será implementado como um *Web Service*, e será dividido em dois módulos: um módulo cliente e outro servidor.

O módulo cliente será responsável por direcionar as solicitações de recuperação dos dados ao servidor correto. O módulo servidor estará configurado em um servidor disponibilizado pela instituição proprietária do banco de dados onde as informações estatísticas residem e conterá uma pequena base de dados construída em MySQL contendo as informações de conexão com os bancos daquela instituição, garantindo assim que tais dados sigilosos não fiquem expostos fora do ambiente seguro da própria instituição.

O *Wrapper* também irá interagir fortemente com o *DCD Tool*. Será através do *Wrapper* que este módulo irá acessar os bancos de dados originais para pesquisar em seus catálogos os possíveis modelos dimensionais ali existentes, bem como obter as instâncias das tabelas dimensões destes modelos para gerar as triplas que precisam estar disponibilizadas no *catalog* para que o módulo *enriching* possa realizar as suas funções.

4.1.5 Enriching

O *Enriching* é o módulo responsável por criar triplas de *sameAs* que vinculam recursos declarados através das descrições dos *linked data cubes*

armazenadas no catálogo (metadados e o conteúdo das instâncias das dimensões) com outros localizados em fontes de dados externas ao *framework* e espalhadas pela *Web*, como por exemplo a *DBpedia* (Sören, et al., 2007).

4.1.6
Data Cube Discovery Tool

Este módulo é o objeto de estudo do presente trabalho e será detalhado no capítulo 5. Resumidamente podemos dizer que trata-se do módulo que povoa o catálogo com as triplas necessárias para o *Enriching* e o *Mediator* realizarem as suas atividades. Para que isso seja possível, entretanto, o *DCD Tool* necessitará realizar um conjunto bastante extenso de funções especializadas. Além de popular o catálogo este módulo interage e depende fortemente do *Wrapper* para poder realizar suas funções.

4.2
Funcionamento do Framework OLAP2DataCube Catalog On Demand

O funcionamento do *framework* proposto está estruturado fundamentalmente sobre dois grandes processos: alimentação do *Catalog e* consumo do *Catalog* e das informações contidas nos RDB´s onde as informações estatísticas estão persistidas.

Nesta seção faremos uma breve descrição destes dois grandes processos, começando pela alimentação do *Catalog*.

A alimentação do *Catalog* pode ser subdividida em cinco etapas: *Parameterizing, Catalog Exploration, Metadata Enriching, Triples Generating e sameAs Enriching*, que serão apresentadas a seguir.

4.2.1
Alimentação do *Catalog*

Esse macro-processo é suportado pelos módulos *Wrapper, DCD Tool e Enrichment.* Ele abrange toda a geração das informações do catálogo, desde a formação das triplas contendo os metadados sobre modelos dimensionais e

RDB´s como o conteúdo correspondente à persistência das instâncias das dimensões.

4.2.1.1 Etapa 1: Parameterizing

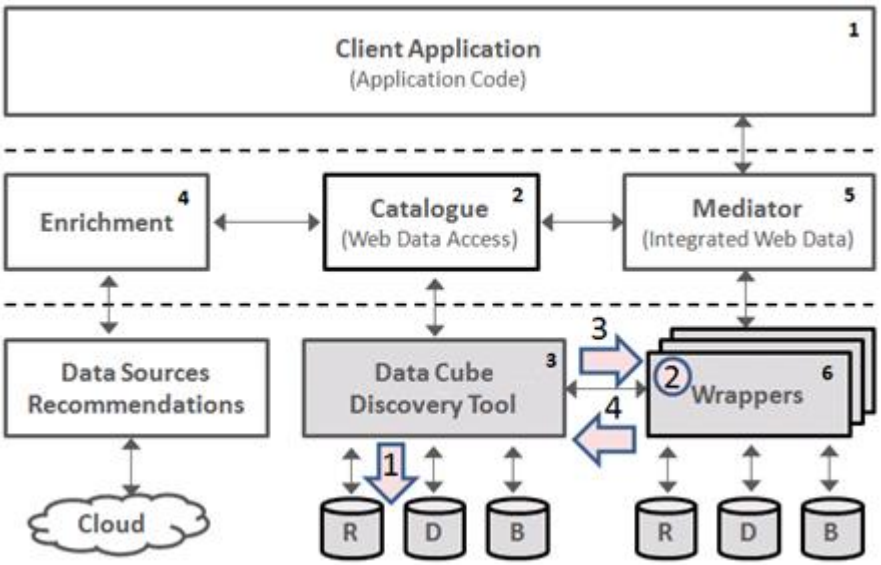


Figura 26: Etapa Parameterizing do Processo Alimentação do Catalog

Essa etapa, ilustrada pela Figura 26, se inicia com o administrador do *framework* informando a instituição (cadastrando-a caso ela seja nova) bem como os dados do(s) RDB(s) ao(s) qual(is) o *framework* terá acesso e criando os usuários da instituição (passo 1). Em seguida, o módulo servidor do *Wrapper* deverá ser instalado no servidor da instituição através do qual o *Web service será* disponibilizado bem como uma pequena base MySQL necessária para o funcionamento deste módulo (passo 2). Após a parametrização e a instalação do módulo servidor do *Wrapper* deverá ser realizado um teste de conexão ilustrado pelos passos 3 e 4, através dos quais uma solicitação é direcionada ao *wrapper* e respondida por ele.

4.2.1.2 Etapa 2: Catalog Exploration

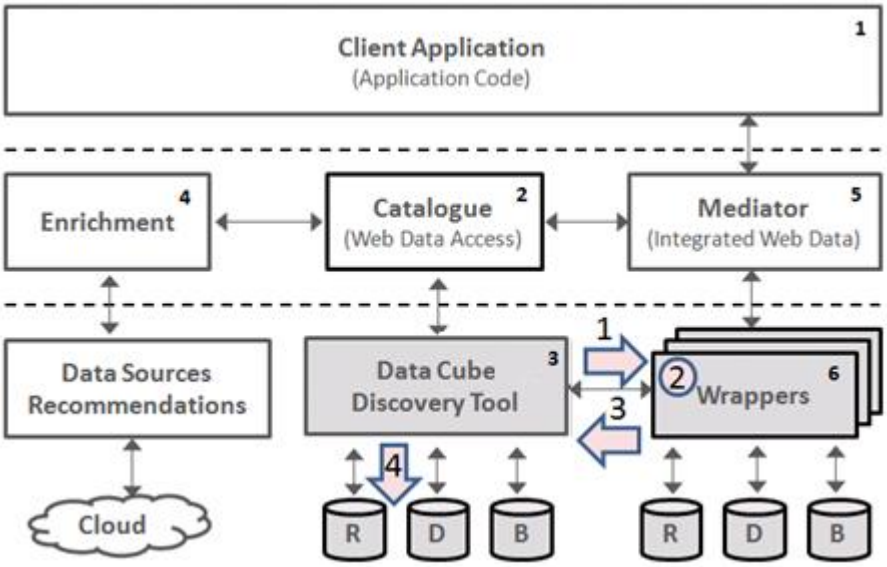


Figura 27: Etapa Catalog Exploration do Processo Alimentação do Catalog

Nessa etapa um usuário envia uma solicitação de exploração do catálogo para o *Wrapper* (passo1). O *Wrapper* executa o procedimento de exploração do catálogo do banco de dados (passo2) e devolve as informações dos modelos dimensionais identificados durante este processo (passo 3).

4.2.1.3 Etapa 3: Metadata Enriching

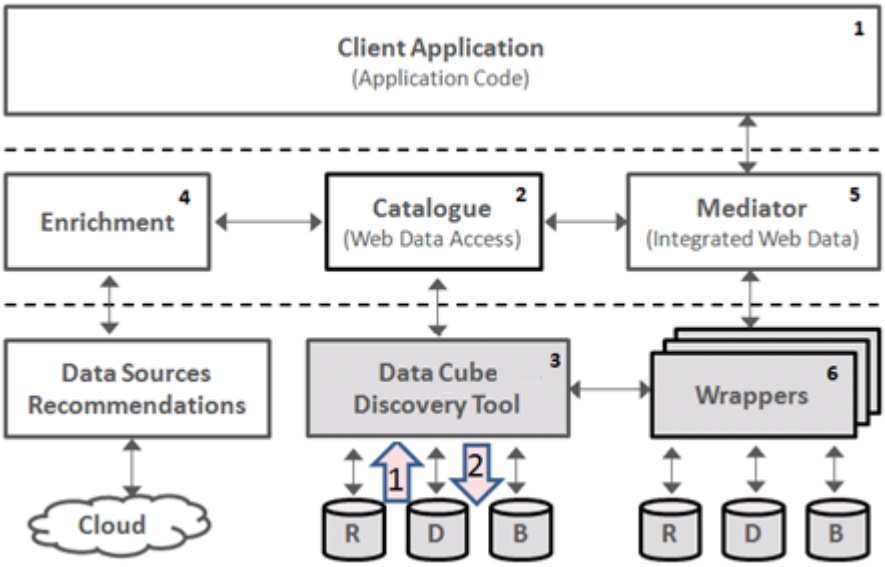


Figura 28: Etapa Metadata Enriching do Processo Alimentação do Catalog

O enriquecimento dos metadados dos modelos dimensionais se dá através da atualização e complementação de informações consideradas obrigatórias para uma boa formação e estruturação do *Catalog*. Nesta etapa, o usuário apenas realiza a recuperação das informações de um dado modelo dimensional (passo 1) e atualiza as informações obrigatórias e, caso deseje, as opcionais também (passo 2).

4.2.1.4 Etapa 4: Triples Generate

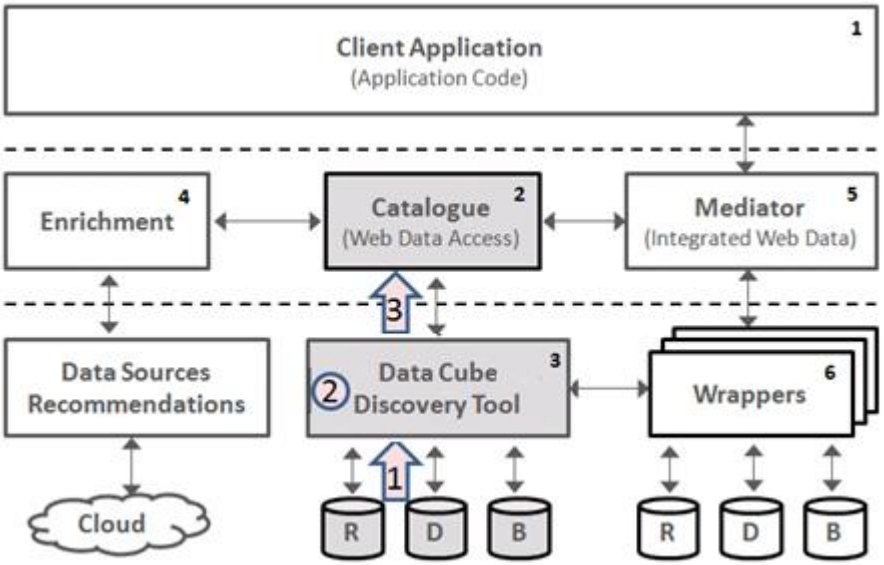


Figura 29: Etapa Triples Generate do Processo Alimentação do Catalog

A geração das triplas para o *Catalog* é a etapa que conclui a colaboração do módulo *DCD Tool* na solução do *framework* proposto. Nesta etapa, inicialmente, são obtidas as informações saneadas de algum modelo dimensional escolhido por um usuário autorizado da instituição credenciada (passo 1). Em seguida, o módulo *DCD Tool* realiza a triplificação de metadados e dados utilizando os vocabulários padrões definidos (R2RML, *Data Cube Vocabulary*, *Skos*, etc) (passo 2) e, finalmente, armazena no *Catalog* as triplas geradas (passo 3).

4.2.1.5 Etapa 5: sameAs Enriching

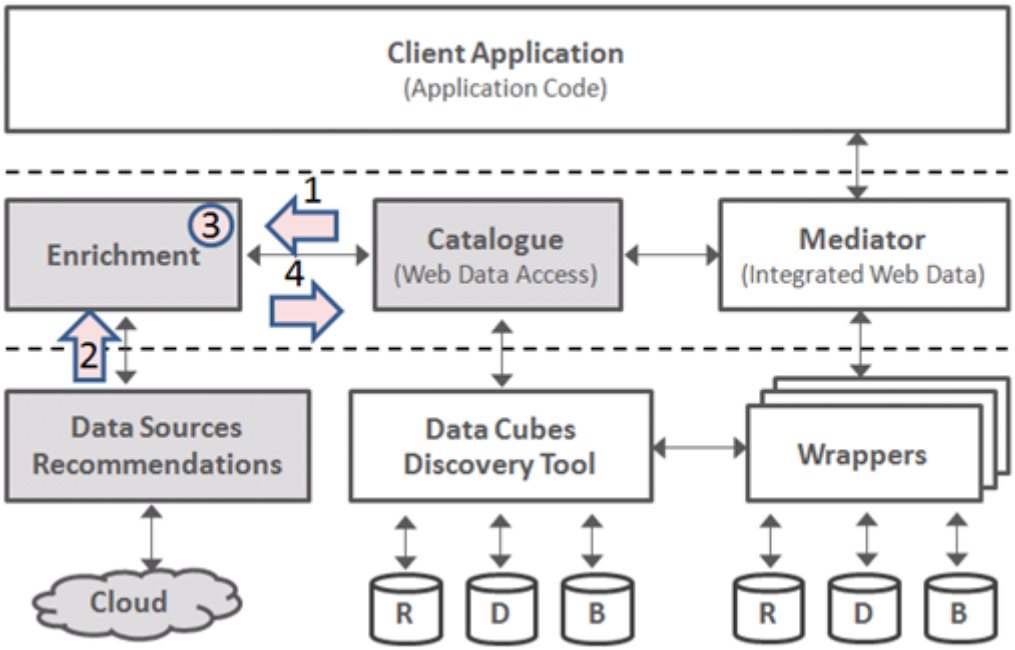


Figura 30 Etapa *sameAs Enriching* do Processo Alimentação do Catalog

O módulo responsável pelo enriquecimento das informações contidas no Catálogo, estabelece a ligação dos conceitos ali registrados com seus equivalentes localizados em outras fontes de dados espalhadas pela *Web*. Ele é o grande responsável pela integração do conteúdo do catálogo de nosso *framework* na nuvem LOD.

Inicialmente, ele identifica e recupera a partir do catálogo as classes, instâncias e propriedades que utilizará para buscar fontes externas (passo 1). Em seguida, obtém-se a partir do módulo de recomendação (externo ao *framework*) uma lista de recomendações de fontes externas (passo 2). Realizam-se procedimentos de configuração e execução de processos que realizam a busca dos dados disponíveis nas fontes recomendadas, e geram medidas de similaridade baseadas nas configurações realizadas (passo3). Em seguida as triplas de sameAs geradas são armazenadas no catálogo (passo 4) e juntamente com os dados relacionados ao processo de comparação.

4.2.2
Consumo do *Catalog*

Esse macro-processo é suportado pelos módulos *Client Application*, *Mediator*, *Catalog* e *Wrapper*. Ele tem por objetivo responder a todas as solicitações realizadas por aplicações cliente e usa os dados disponíveis no catálogo para informar quais cubos ele possui, a estrutura destes cubos, além de permitir que consultas sejam realizadas sobre os dados armazenados nos cubos. Na primeira versão do *Mediator* serão disponibilizadas apenas queries que retornam o conteúdo completo de um cubo em triplas RDF.

Este processo de consumo do *catalog*, pode ser dividido em três etapas: *Search and Choose*, *Production and Request*, *Transform and Responds* (Salas, et al., 2012) apresentados a seguir.

4.2.2.1
Etapa 1: Search and Choose

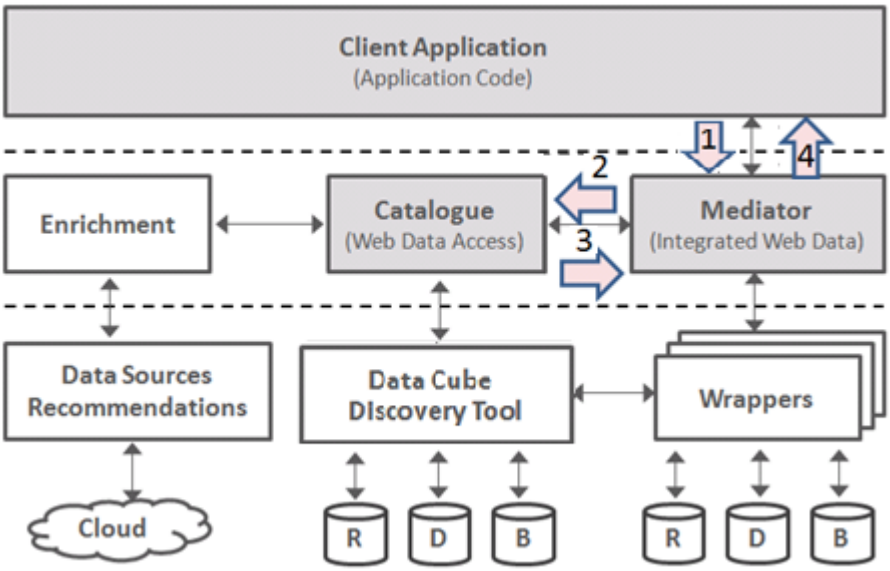


Figura 31: Etapa Search and Choose do *Framework OLAP2DataCube Catalog On Demand*

Nesta etapa a aplicação cliente solicita informações sobre cubos que tratem determinado assunto, informando a palavra-chave desejada (passo 1). O *Mediator*, então, efetua uma consulta em SPARQL sobre o *Catalog* (passo 2), para oter os cubos compatíveis com a palavra-chave e os devolve ao *Mediator*.

Este repassa as triplas que definem os cubos selecionados (passo 3) para o usuário (passo 4).

4.2.2.2 Etapa 2: Production and Request

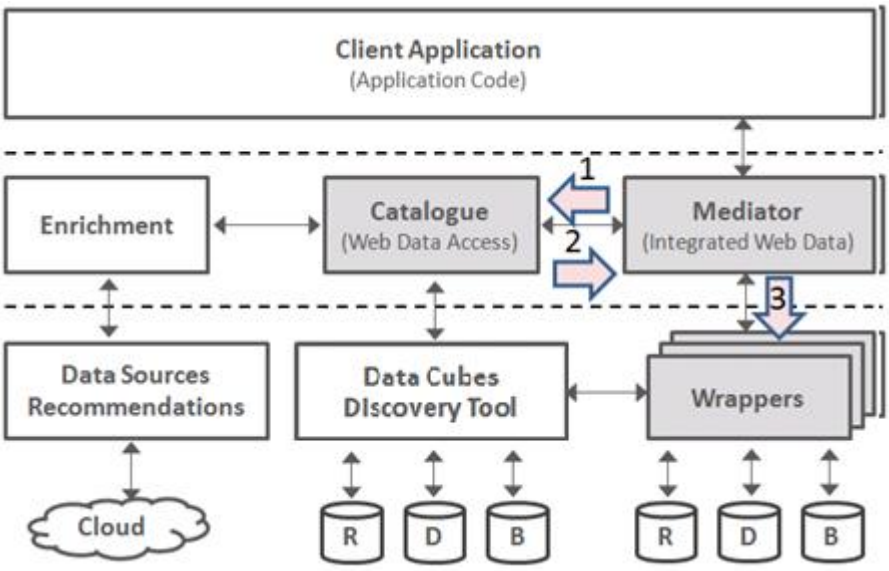


Figura 32: Etapa Production and Request do *Framework OLAP2DataCube Catalog On Demand*

A *Client Application* informa o cubo escolhido pelo usuário. O *Mediator* busca (passo 1) no *Catalog* as triplas R2RML relativas ao cubo que se deseja recuperar (passo 2) e comunica-se com o *Wrapper* do servidor institucional, que utiliza o *DB2Triples* para recuperar as observações do cubo desejado já em formato de triplas RDF (passo 3).

4.2.2.3 Etapa 3: Transform and Respond

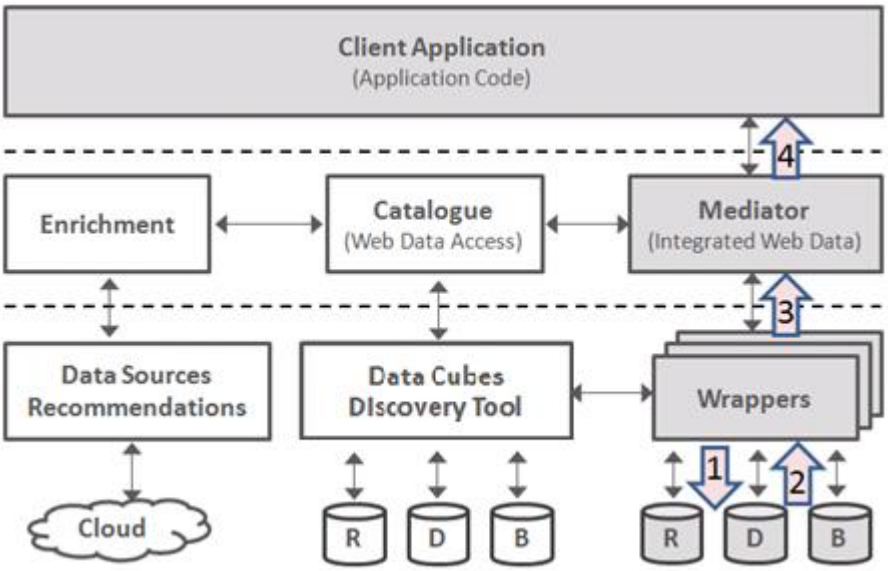


Figura 33: Etapa Transform and Respond do *Framework OLAP2DataCube Catalog On Demand*

A última etapa do processo de consumo dos cubos se inicia com a resposta do RDB a solicitação efetuada pelo *Wrapper*, via *DB2Triple* (passo 1). Os dados recuperados são triplificados pela ferramenta *DB2Triple* e gera as visões RDF dos dados relacionais solicitados pelo usuário (passo 2). Essas visões são repassadas ao *Mediator* (passo 3), no formato de arquivo RDF, que por sua vez o repassa para o *Client Application* (passo 4).

A partir do arquivo RDF recebido, a aplicação cliente exibe os dados retornados para o usuário, permitindo que este o manipule de acordo com as possibilidades oferecidas pela aplicação cliente.

4.3 Resumo

Neste capítulo apresentamos o *framework OLAP2DataCube Catalog On Demand*, descrevendo suas camadas, detalhando seus componentes, a comunicação entre estes e o seu processo produtivo e colaborativo.

5.
Data Cube Discovery Tool

Conforme mencionado, o módulo DC Discovery Tool, componente do *framework OLAP2DataCube Catalog On Demand*, é, na realidade um conjunto de ferramentas elaboradas com o objetivo de apoiar os processos necessários para obtenção, saneamento e organização das informações que deverão ser utilizadas para popular o catálogo do *framework*.

Este capítulo está assim organizado: apresentação do processo, arquitetura interna da DCD Tool, Exploração de Catálogos Relacionais e o Processo de Descoberta de Modelos Dimensionais, o processo de Enriquecimento de Metadados, e, por fim, apresentação do processo de geração das triplas a partir das informações armazenadas no catálogo interno do DCD Tool.

5.1
Descrição dos Processos

O processo colaborativo consagrado na construção da Web de dados através do crescimento progressivo da nuvem LOD, foi fonte de inspiração para a elaboração do processo que aqui apresentamos.

Viabilizar a integração de diferentes bases de dados governamentais, oriundas de diferentes instituições públicas ou privadas a serviço do Estado, é um obstáculo real mas que precisa ser transposto. O maior obstáculo à tal proposta está certamente concentrado em três pontos fundamentais:

- Quanto custa? – custo em termos financeiros e esforço de pessoal.
- O que eu ganho? – que vantagens isso irá me proporcionar?
- E a segurança dos meus dados? – qual o risco para a minha segurança?

Para cada uma dessas questões, felizmente, temos boas respostas para apresentar. O custo para uma determinada instituição utilizar nosso *framework*

para publicação de seus dados e integração destes à nuvem LOD resume-se ao esforço necessário para a implantação das bases que deseja publicar, o que será detalhado adiante. O custo de desenvolvimento ou customização é exatamente zero. Todos os componentes estão plenamente desenvolvidos e disponíveis para sua implantação. Os componentes são leves e pesam muito pouco no servidor da instituição credenciada. Eles poderão ser baixados diretamente do site e instalados de acordo com a conveniência da instituição.

No tocante à segunda questão, consideramos que qualquer instituição que se dispuser a publicar seus dados através do nosso *framework*, ganha, simplesmente a integração de seus dados diretamente à nuvem LOD sem precisar desenvolver uma única linha de código. Basta fazer uma simples instalação de componentes seguindo as instruções disponíveis no site e realizar os procedimentos de instalação da base de dados verdade nosso "público alvo" são justamente instituições que já fazem isso ou instituições que desejem vir a fazê-lo. Estamos falando de instituições que seguem os princípios de transparência de *e-government* e cujos dados que serão publicados, na verdade, são considerados de "domínio público". Em outras palavras, são instituições que já publicam dados na Web e que estão acostumadas com o processo de publicação de dados armazenados em seus bancos de dados. Bom, se tais organizações já possuem este processo, o que ganham em usar nosso *framework*?

Para começar muitas dessas instituições possuem aplicações *Web* construídas para viabilizar esta disponibilização de dados. Tais aplicações consumem recursos de desenvolvimento, manutenção, melhorias e por aí em diante. Mais do que isso, os dados disponibilizados através destas aplicações não estão integrados à nuvem LOD e não se conectam a outros dados similares. A semântica deles não se encontra disponível para integração. Ao utilizar o nosso *framework*, a instituição credenciada terá a garantia da integração dos seus dados na *Web* 3.0 com sua semântica devidamente disponibilizada. Além disso, custos de manutenção ou desenvolvimento de novas melhorias no *framework* simplesmente não os afetará. O único custo para essas instituições é o custo da manutenção e atualização dos metadados das bases que ela vier a disponibilizar o acesso através do *framework*.

Finalmente temos a questão da segurança. Nenhuma organização se sentirá muito à vontade de disponibilizar uma porta de entrada a algum servidor para que uma aplicação Web acesse diretamente seus bancos de dados. No entanto como a maioria delas já está acostumada a publicar os seus dados, já possui experiência com soluções que, apesar de permitirem o acesso aos seus bancos de dados, ainda assim garantem a segurança deste acesso. Na arquitetura da solução proposta, toda comunicação com o(s) banco(s) de dados das instituições é realizada através de um *Web Service* especialmente desenvolvido com o objetivo de trazer uma camada de segurança que garante o acesso sem expor a instituição. Esta questão da segurança será mais bem desenvolvida quando tratarmos especificamente do *Web Service*.

Superados os pontos que inicialmente poderiam provocar receio na adesão ao *framework*, passamos então a detalhar os processos que deverão ser conhecidos por aqueles que se dispuserem a utilizar a nossa solução.

Na figura 40 ilustramos um diagrama UML apresentando as principais funcionalidades disponíveis no Web site da ferramenta DCD Tool. Estas funcionalidades suportam os processos que serão identificados e descritos a seguir.

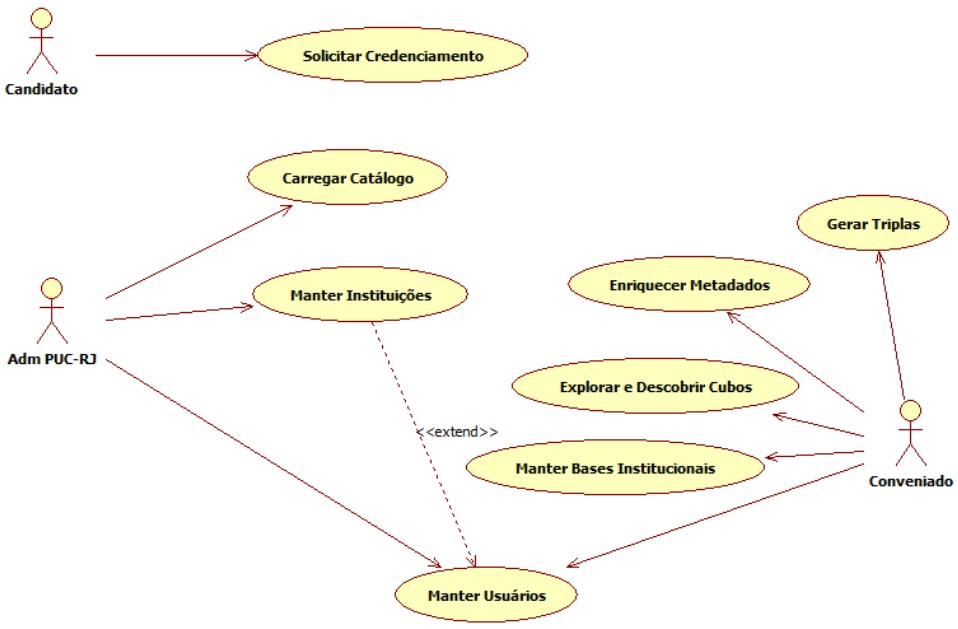


Figura 34: Diagrama UML das principais funcionalidades da DCD Tool

Uma visão geral do processo pode ser obtida a partir da seguinte descrição:

1. Um representante da instituição interessada em usar o *framework* preenche uma solicitação de credenciamento – caso de uso "Solicitar Credenciamento".
2. Após o cadastro ser confirmado, a Instituição e seus usuários são cadastrados – casos de uso "Manter Instituições" e "Manter Usuários".
3. Os usuários cadastrados para a instituição podem, então, realizar o download e instalação dos componentes.
4. Uma vez com a instalação concluída, deve-se proceder ao cadastro dos bancos de dados onde se encontram os modelos cujos dados se deseja publicar – caso de uso "Manter Bases Institucionais".
5. Com as bases de dados cadastradas, a próxima etapa é cadastrar o agendamento de identificação e extração dos modelos dimensionais e verificar o resultado deste processamento – caso de uso "Explorar e Descobrir Cubos".
6. Na etapa anterior foram identificados e gravados no catálogo interno da DCD Tool, os modelos dimensionais encontrados no banco de dados da instituição. Agora o usuário da instituição deverá preencher os metadados complementares para que estas estruturas e seus dados possam ser publicadas – caso de uso "Enriquecer Metadados".
7. Após concluir o preenchimento das informações de metadados, o usuário pode gerar as triplas que serão carregadas no catálogo do *framework* – caso de uso "Gerar Triplas".
8. Finalmente, o administrador do *framework* irá validar as triplas geradas e carregá-las no catálogo do *framework* – caso de uso "Carregar Catálogo".

Nas próximas seções deste capítulo apresentaremos a arquitetura interna da ferramenta e descreveremos cada um de seus módulos. Durante as descrições dos módulos componentes da ferramenta, abordaremos os aspectos processuais suportados por ele, bem como as soluções técnicas empregadas no seu desenvolvimento.

5.2
Credenciamento e acesso à ferramenta

Antes de descrevermos a arquitetura do Linked Data Cube Discovery Tool, detalhando os seus processos mais importantes, apresentaremos o processo através do qual as instituições poderão nos contatar e credenciarem-se a utilizar o nosso *framework*.

O primeiro passo é acessar o site que iremos disponibilizar, cuja tela principal na sua versão atual está apresentada na figura 41. Esta tela apresenta uma mensagem de boas vindas, um menu de opções e uma área de login para os usuários que já estão cadastrados. O menu, inicialmente possui habilitadas apenas as opções: Home, About e Credenciamento→Solicitação. Todas as demais opções somente se tornam acessíveis caso um login bem sucedido tenha ocorrido.

Figura 35: Home do site do DCD Tool

Através do menu principal o candidato a se conveniar conosco pode acessar o nosso formulário de contato através da opção “Solicitação”, acessível pela opção “Credenciamento” do menu principal.

Figura 36:Opção de Solicitação de Credenciamento.

Selecionada esta opção o formulário abaixo é apresentado para preenchimento pelo candidato. Apenas algumas informações básicas são exigidas para que possamos contatar o interessado. Entre elas, é necessário o fornecimento do nome da Instituição, nome da pessoa de contato na Instituição com a qual faremos contato para definir o credenciamento (ou não) da instituição, Web site da instituição e email da pessoa de contato. Além desses dados, um telefone de contato com ramal e um celular podem (ou não) serem fornecidos para facilitar o processo de credenciamento.

Figura 37: Formulário de Solicitação de Credenciamento

O credenciamento da instituição candidata, é realizado através da opção de cadastramento de instituições pela seguinte opção do menu principal:

Figura 38: Acesso ao Cadastro de Instituições

Através da opção acima, o sistema dá acesso ao formulário de cadastramento das instituições. O cadastro é simples possuindo apenas as informações essenciais para identificação das instituições. Ainda assim, é fundamental, pois todos os demais objetos que serão cadastrados ficarão vinculados a uma instituição.

O cadastramento é realizado através do formulário abaixo apresentado e que será preenchido pelo administrador do *framework*.

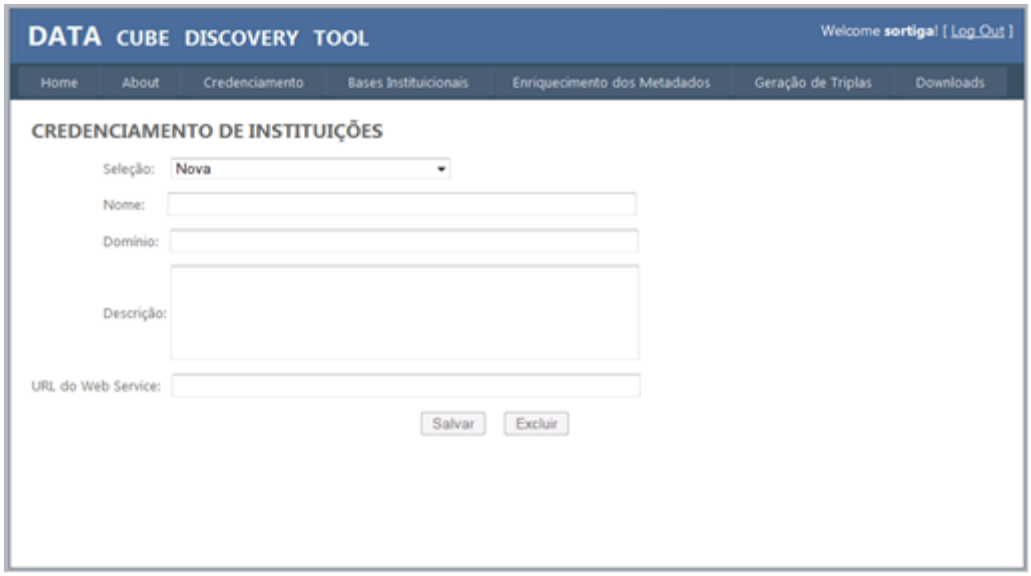


Figura 39: Formulário de Credenciamento de Instituição

Em seguida, deve-se cadastrar um ou mais usuários para aquela instituição através do seguinte formulário. Os usuários da instituição podem ser “master” ou “comum”. O usuário “master” pode criar e atualizar dados de outros usuários da sua instituição enquanto que o “comum” não pode realizar estas ações. As demais funcionalidades são comuns à ambos os tipos.

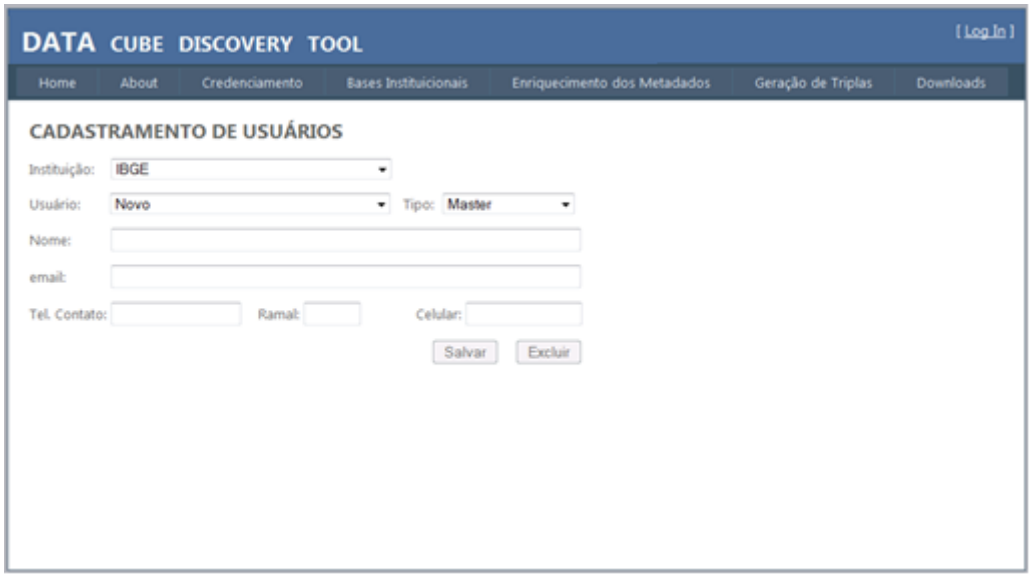


Figura 40: Formulário de Cadastramento de Usuário

Com a instituição e seus usuários cadastrados, começamos a povoar o modelo de dados do catálogo interno da DCD Tool.

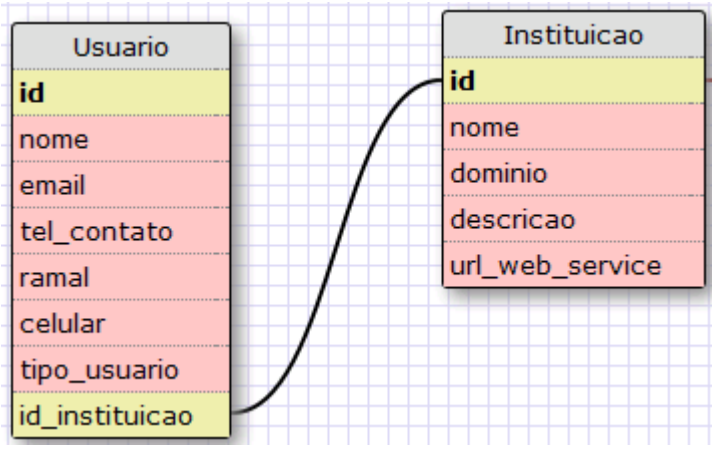


Figura 41: Fragmento do Modelo de Dados - Instituição e Usuário

A partir do momento que essas entidades estão preenchidas, o uso das funcionalidades da DCD Tool passa para a Instituição e, os próximos passos serão realizados quase que exclusivamente por seus usuários.

O próximo passo é cadastrar as bases institucionais onde iremos buscar os modelos dimensionais que contém os dados estatísticos que deverão ser publicados. Para isso, no entanto, um usuário já cadastrado na etapa anterior, deve entrar na área de download e baixar os componentes que precisam ser instalados em algum servidor da instituição credenciada: um pequeno banco de dados MySQL, reunindo as tabelas necessárias para o *wrapper* desempenhar suas funções de forma adequada, um módulo Web para cadastramento de bancos de dados institucionais e o *wrapper*, implementado através de um *Web service* para permitir a comunicação entre o *framework* e a instituição.

Após a instalação desses componentes, um usuário da instituição credenciada deve informar ao adminsitrador do *framework* a url através da qual o wrapper pode ser acessado e realizar o cadastramento do(s) banco(s) de dados onde estão armazenados os modelos dimensionais, indicando o tipo de banco, informação esta, fundamental para que os processos subsequentes possam ser executados. Este cadastramento será feito a partir do módulo de cadastramento de bancos de dados baixado do site onde a DCD Tool encontra-se disponibilizada. Trata-se de uma interface Web simples que apenas permite ao usuário cadastrar bancos de dados no banco MySQL que também foi baixado do site. Não há procedimento de controle de acesso nesse caso por se tratar de uma aplicação que deverá estar restrita ao responsável pelo cadastramento dessas bases de dados na instituição credenciada.

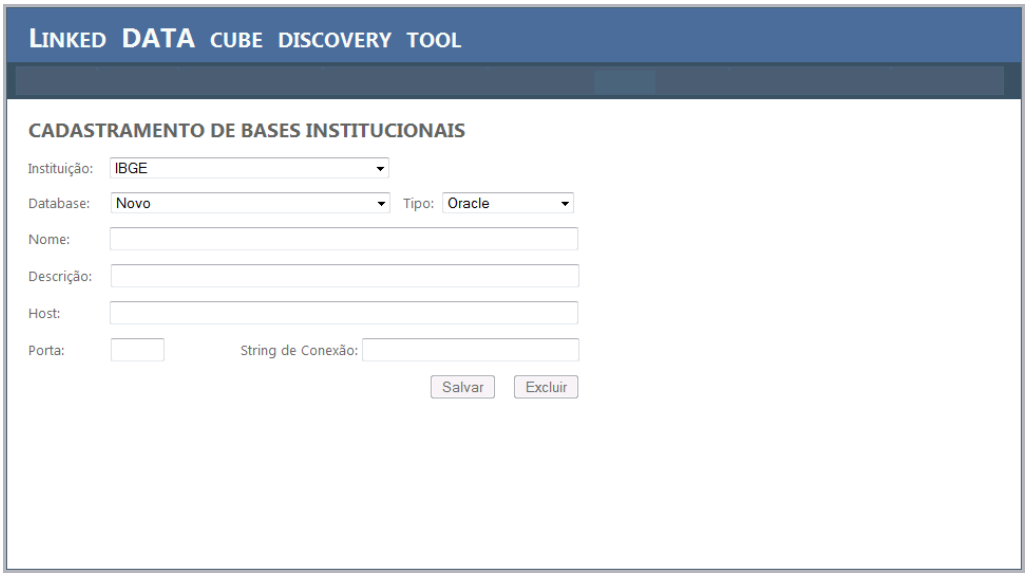


Figura 42: Cadastramento dos Bancos de Dados Institucionais

Após o cadastramento destes bancos de dados, o usuário deverá entrar no site da DCD Tool e efetuar uma operação de sincronismo (só será possível se a url de publicação do *Web service* já tiver sido cadastrada pelo administrador do *framework* e disponibilizada pela instituição credenciada). Esta operação simplesmente atualiza o catálogo interno da DCD Tool com o nome e id dos bancos de dados cadastrados para a instituição a partir de uma solicitação que é encaminhada ao *Web Service – Provider* instalado no servidor da instituição credenciada. Um exemplo do arquivo XML produzido pela ferramenta é apresentado no quadro 8:

```
<?xml version="1.0" encoding="UTF-8"?>
<bancosinstitucionais>
 <instituicao>
  <nome>FGV</nome>
  <databases>
  <database id="1">
   <nome>BASE_DW</nome>
   <descricao>Banco do Data Warehouse da FGV</descricao>
   <tipo>Oracle</tipo>
  </database>
  <database id="2">
   <nome>BASE_BI</nome>
   <descricao>Bancos das aplicações BI</descricao>
   <tipo>Oracle</tipo>
  </database>
  </databases>
 </instituicao>
</bancosinstitucionais>
```

Quadro 8: Exemplo de XML gerado na Sincronização

Após a operação de sincronismo, o catálogo interno da DCD Tool estará devidamente atualizado com os bancos de dados da instituição credenciada e a partir desse momento, começa efetivamente o ciclo de implantação das bases estatísticas residentes nos bancos de dados cadastrados.

Não é incomum que grandes instituições possuam diferentes bancos de dados utilizados em suas instalações. A ferramenta DCD Tool oferece suporte aos seguintes bancos de dados: Oracle, MySQL, SQLServer e PostgreSQL. O "segredo" para suportar estas diferentes tecnologias é bastante simples na verdade. A estrutura apresentada na figura 49 revela a solução que criamos para suporte a diferentes tecnologias de bancos de dados.

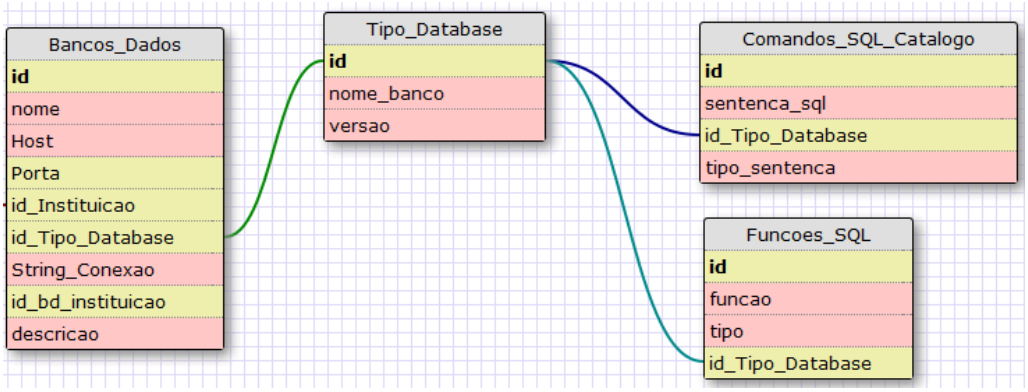


Figura 43: Fragmento de Modelo - Solução de Suporte a diferentes Databases

Como pode ser observado pela figura 43, cada tipo de banco de dados suportado pela ferramenta é cadastrado na tabela "Tipo_Database". Basta sabermos o nome do banco (Oracle, MySql, SqlServer, etc) e a versão (11g, 9i, etc), para o caso de haver alterações significativas que suscitem a alteração das queries de uma versão para outra. Na tabela "Comandos_SQL_Catalogo" armazenamos as diferentes consultas por tipo (coluna "tipo_sentenca").

Também pode vir a ser necessário o conhecimento específico de funções do banco de dados para a construção das queries que utilizaremos mais adiante para recuperar as instâncias das F-TABs e dimensões através das triplas R2RML. Por isso, criamos uma tabela ("Funções_SQL") onde armazenamos as funções que possam vir a ser úteis, agrupadas por tipo de função, por exemplo, funções para obter substring, funções para transformar valor em string, e assim por diante, para cada um dos tipos de bancos de dados cadastrados.

As tabelas "Tipo_Database", "Comandos_SQL_Catalogo" e "Funções_SQL" foram inicializadas com os dados referentes aos bancos de dados inicialmente suportados pela ferramenta, já mencionados acima, e, para se implantar novos bancos, basta atualizar estas três tabelas e o sistema estará apto a trabalhar com o novo banco de dados. Por questões de segurança, a distribuição do suporte a novos bancos de dados será realizada versionando-se o banco MySQL disponível na área de download. Assim, caso uma instituição credenciada decida adotar uma nova tecnologia de banco de dados suportada pela nossa ferramenta, bastará baixar a versão do banco que contém os comandos SQL desta tecnologia e instalá-lo.

5.3
Arquitetura do Linked Data Cube Discovery Tool

No capítulo 4, apresentamos a arquitetura do *framework* do qual a nossa ferramenta é apenas um componente. Nesta seção iremos descrever a arquitetura interna da ferramenta, que orientou o desenvolvimento de seus módulos, os quais serão descritos detalhadamente nas próximas seções.

Também apresentaremos a arquitetura de dados utilizada para suportar o funcionamento de cada um dos módulos. O detalhamento das tabelas componentes da arquitetura de dados será realizado nas seções onde os módulos serão detalhados. O conjunto de entidades utilizadas para persistir os dados necessários para o funcionamento daquele módulo serão detalhadas junto com a descrição de cada módulo.

A figura 44 seguir representa a arquitetura interna da DCD Tool. Ela possui três camadas sendo a primeira delas, responsável por realizar a pesquisa e reconhecimento de modelos dimensionais em bases de dados relacionais, implementada parcialmente no domínio da instituição credenciada. Esta camada foi implementada através de componentes localizados no site da DCD Tool e de um *Web Service (wrapper)*, responsável por realizar operações no servidor da instituição credenciada para atender as solicitações realizadas através do site da DCD Tool.

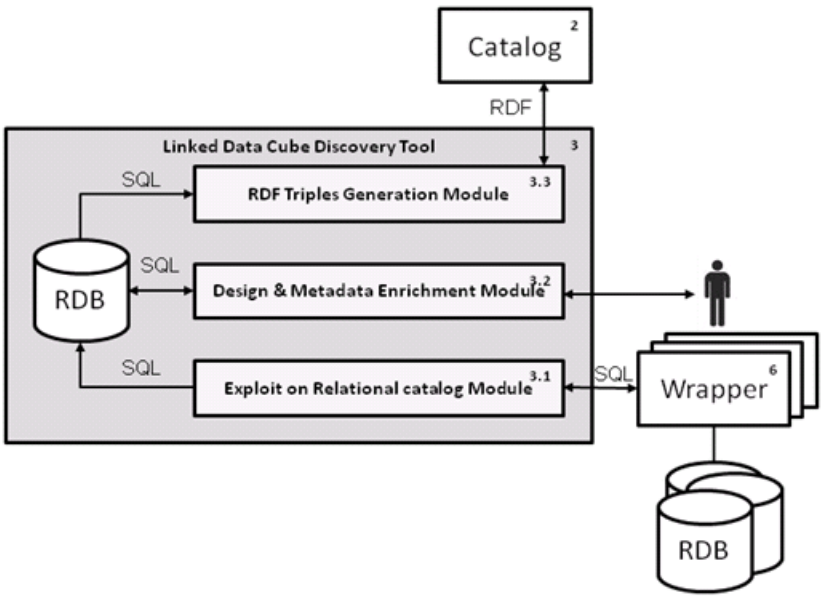


Figura 44: Arquitetura Interna da DCD Tool.

A utilização de *Web Services* em aplicações de *e-Commerce* e de *m-Commerce* está mais do que consagrada. Este tem sido o padrão de comunicação entre aplicações no ambiente *Web* desde o final da década de 1990.

Um *Web service* cria uma camada protetora e permite que apenas solicitações pré-programadas possam ser atendidas, limitando o acesso de qualquer aplicação ou usuário àquilo que é disponibilizado pelos serviços implementados por ele. Não existe acesso direto ao banco de dados nem é necessário para qualquer aplicação externa possuir informações que coloquem em risco a segurança do ambiente de tecnologia das instituições credenciadas tais como usuário e senha de banco. Também não é necessário abrir uma porta para o "mundo externo" a fim de permitir acesso direto a um ou mais bancos de dados. Tudo isso fica encapsulado pelo *Web service* e as informações necessárias para a realização das consultas a estas bases permanece protegida no ambiente seguro da própria instituição.

5.3.1
Camada Exploit on Relational Catalog

Um dos aspectos vitais para tornar viável o uso do *framework* é o balanceamento das cargas de processamento. Assim, se planejamos oferecer os serviços de nosso *framework* a diversas instituições, se concentrássemos a carga de processamento em nosso servidor, em pouco tempo estaríamos incapacitados a prestar um serviço de qualidade à cada uma das instiuições credenciadas. Motivados pela questão de desempenho decidimos dividir a solução da camada de exploração e descoberta de cubos em quatro componentes básicos (além, obviamente da aplicação principal que é executada em nosso site):

1. O *Web Service* instalado no site da instituição credenciada;
2. Dois serviços Windows que ficam em permanente execução em nosso servidor
3. Um *Web Service* instalado no servidor da DCD TOOL.

O serviço "*exploit_discovery_request*" do *Web Service* instalado no servidor do cliente será responsável por explorar todo o catálogo do banco de

dados informado como parâmetro de entrada deste serviço e encontrar os modelos dimensionais. Tal processamento pode demorar algum tempo levando-se em conta o tamanho do catálogo a ser processado, além das operações de atualização dos controles internos e de geração do arquivo XML que será devolvido com a descrição dos modelos encontrados. Por este motivo decidimos implementar um processo assíncrono baseado em comunicação entre os dois *Web Services* mencionados anteriormente.

Este processo será detalhado a partir da figura 45. Apresentamos, através dela, as etapas constituintes do ciclo completo de exploração do catálogo de um banco de dados no site da instituição credenciada, identificação de modelos dimensionais, tradução destes modelos em estrutura XML, devolução deste XML para o site da DCD TOOL e atualização do catálogo interno da DCD Tool com estes dados.

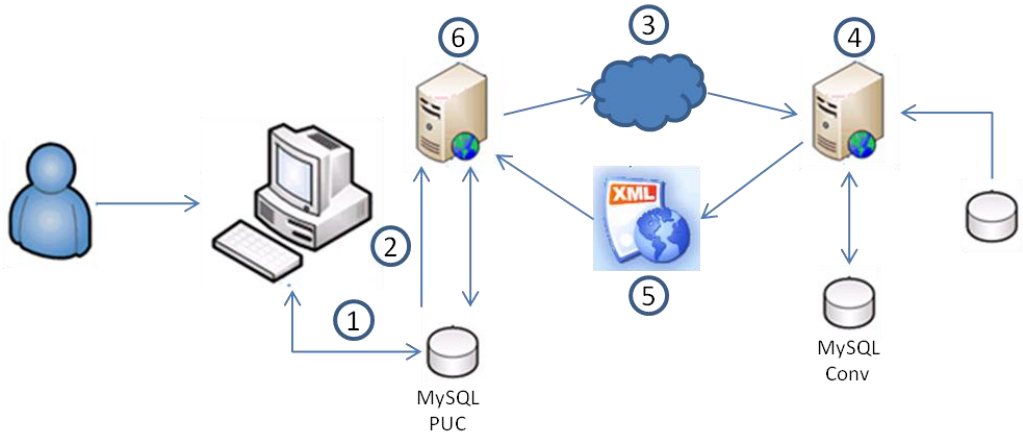


Figura 45: Esquema da camada “Exploit on Relational Catalog”

1. O usuário deve logar no site da DCD Tool e acessar a opção “Agendamento” conforme a figura 46:

Figura 46: Acesso ao agendamento do Exploit on Relational Catalog

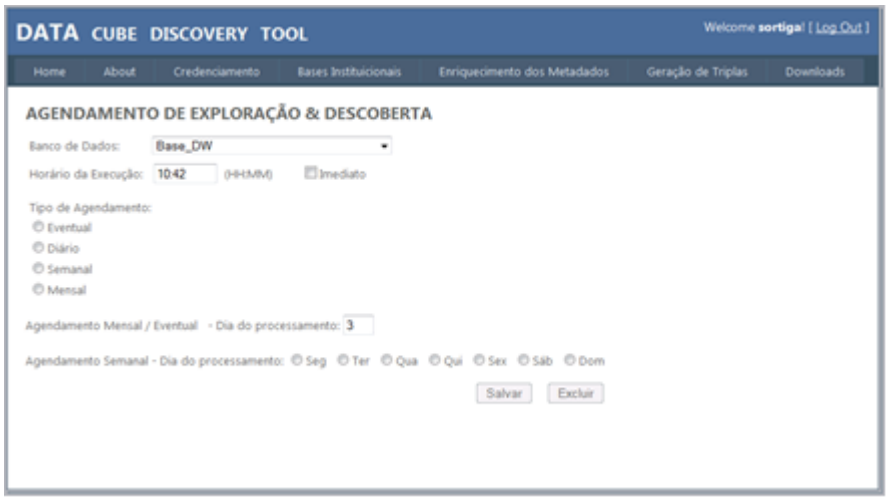


Figura 47: Formulário de Cadastro de Agendamento

O agendamento permite ao usuário realizar um único procedimento de exploração e descoberta de modelos dimensionais ou deixar programada uma regra periódica de exploração do catálogo do banco relacional em questão.

Qualquer que seja a opção, ele é obrigado a indicar um banco de dados da sua instituição para a realização do agendamento. Para efetuar um agendamento que deverá ser executado apenas naquele instante, basta marcar o checkbox “Imediato” e clicar no botão “Salvar”. Se, no entanto, o usuário quiser executar o procedimento apenas uma vez, ele deverá preencher o horário em que deseja que o processamento seja realizado, marcar a opção “Eventual” no “Tipo de Agendamento” e preencher o dia em que o processamento deve

ocorrer em “Agendamento Mensal / Eventual – Dia do processamento”.

O agendamento eventual só está disponível para a faixa de dias do mês corrente a partir da data atual até o final do mês. Não se podem programar agendamentos eventuais para meses subsequentes. As outras modalidades de agendamento enquadram-se na “regra periódica de exploração do catálogo”.

Podem-se realizar processamentos de verificação: diários, indicando apenas o horário em que estes devem acontecer; semanais, indicando-se o dia da semana e o horário; ou mensais, indicando-se qual o dia do mês e o horário. Quando o usuário clica em “salvar”, é gerado um registro de agendamento na tabela “Agendamento”.

Para um determinado banco de dados de uma determinada instituição só pode haver um registro de agendamento **por tipo** de agendamento. Se já houver um certo tipo de agendamento registrado para determinado banco de dados, o sistema apresenta uma tela contendo os dados que estão sendo modificados e pergunta se o usuário confirma a alteração.

Quando um registro é gravado ou atualizado na tabela “Agendamento”, um trigger é disparado para atualizar ou inserir registros na tabela “Fila_Processamento”. Essa tabela contém as solicitações de processamento decorrentes dos agendamentos cadastrados pelos usuários.

Todas as informações necessárias para se realizar a chamada do *Web Service* da instituição estão contidas na solicitação. Toda solicitação fica vinculada ao agendamento que lhe originou e quando o agendamento define uma “regra periódica de exploração do catálogo”, criamos um vínculo entre cada nova solicitação e a anterior. Assim podemos rastrear uma sequência de agendamentos vinculados a uma mesma regra.

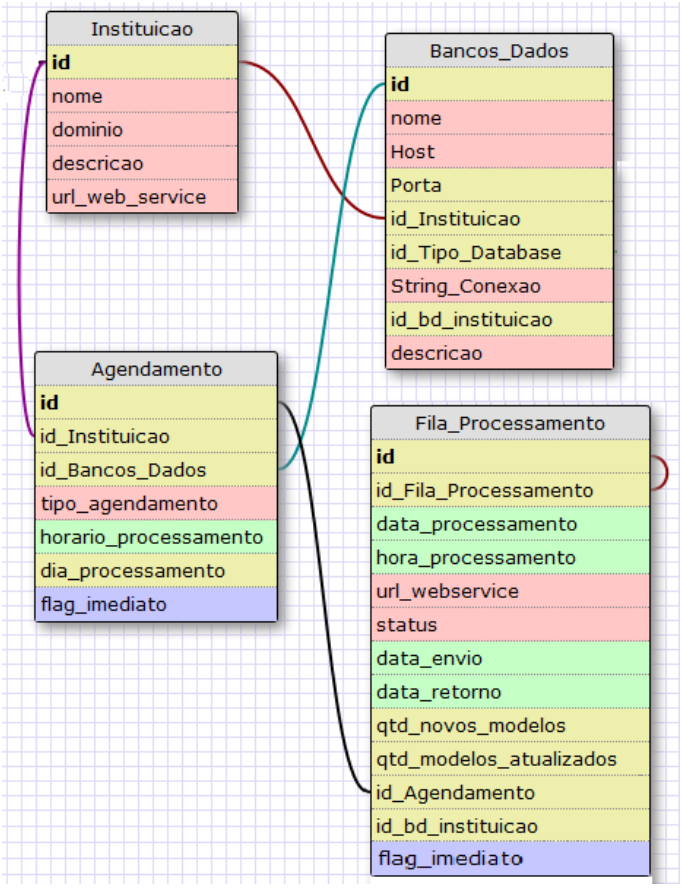


Figura 48: Fragmento do Modelo de dados sobre Agendamento

2. O segundo passo do esquema da camada "Exploit on Relational Catalog" é realizado por um serviço Windows instalado no servidor da DCD TOOL. Trata-se, na verdade de um pequeno programa que realiza pooling em um loop infinito verificando se há alguma solicitação que precise ser encaminhada ao *Web Service* de alguma instituição. O pseudo-código deste programa é apresentado no quadro 9.

```
Fim ← False
Enquanto não Fim Faça
  Se existe solicitação pendente com flag_imediato Então
   Chama Serviço exploit_discovery_request usando
    a url_Webservice, Informando o id_bd_instituição,
    o id da fila de processamento, dt_ultima_solicitacao
    e tipo_agendamento
   Atualiza status da solicitação para "Submetida"
```

```
 Senão
  Se existe solicitação pendente Então
   Se data_processamento = "hoje"
    e hora_processamento = "agora" Então
    Chama Serviço exploit_discovery_request usando a
     url_Webservice Informando o id_bd_instituição
     o id da fila de processamento, dt_ultima_solicitacao
     e tipo_agendamento
    Atualiza status da solicitação para "Submetida"
   Fim Se
  Fim Se
Fim Enquanto
```

Quadro 9: Pseudo-código Schedule de Exploração de Catálogo Relacional

3. Representa a chamada do serviço "*exploit_discovery_request*" do *Web Service* instalado na instituição credenciada.
4. Nesta etapa é executado o serviço "*exploit_discovery_request*" que é o responsável por identificar os modelos dimensionais, recuperando seus componentes e gravando um arquivo XML contendo a descrição destes componentes. O detalhamento deste serviço será apresentado na próxima seção deste capítulo.
5. Ao final da geração do arquivo XML contendo o resultado do processamento do serviço "*exploit_discovery_request*", este efetua uma solicitação do serviço "*save_models_request*" implementado no *Web Service* que está instalado no site da DCD TOOL. Este serviço apenas recebe o arquivo XML e o armazena em um diretório padrão.
6. Outro serviço Windows instalado no servidor da DCD TOOL é responsável por fazer pooling no diretório padrão e realizar o processamento dos arquivos XML que vão sendo devolvidos para o servidor da DCD TOOL. Após identificar a existência de um arquivo, o programa começa o seu processamento identificando qual a solicitação que originou o arquivo XML para poder atualizar seu status e informações de retorno do processamento tais como:

data_retorno, qtd_novos_modelos e qtd_modelos_atualizados. Esta atualização se dá ao final do processamento em que os novos dados são inseridos nas tabelas do catálogo interno da DCD Tool, cuja estrutura do modelo de dados é apresentada no apêndice C desta dissertação.

Para saber o resultado de um processamento, o usuário deve acessar a opção "Consultar processamentos agendados" conforme ilustrado pela figura 49.

Figura 49:Acesso à consulta dos processamentos agendados

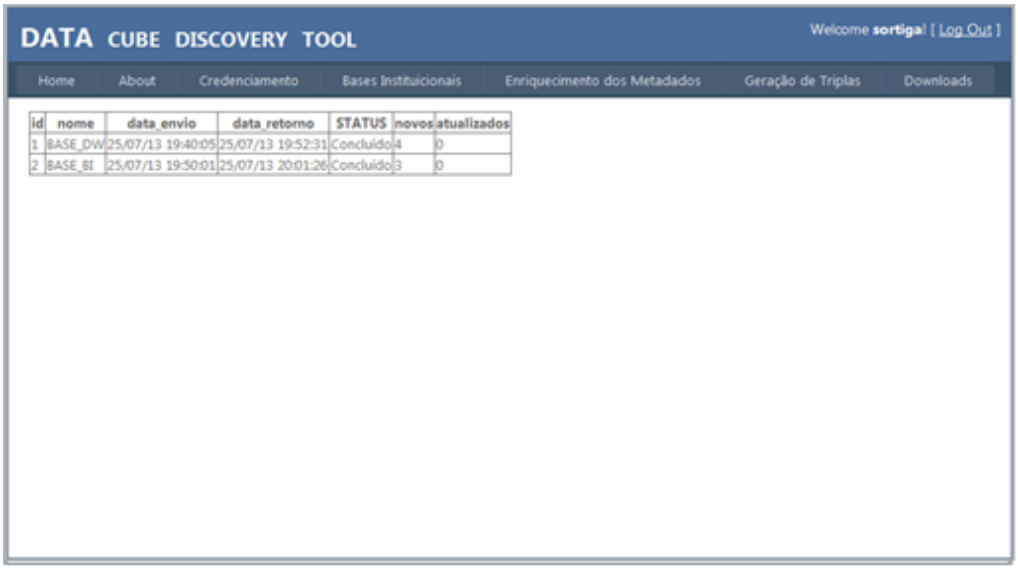


Figura 50: Consulta aos Agendamentos Programados

São informados o nome do banco de dados para o qual o agendamento foi realizado, o id do agendamento, datas de envio e retorno, status, quantidade de novos modelos localizados e quantidade de modelos atualizados.

5.3.2
Camada Design and Metadata Enrichment

Esta camada é responsável por oferecer todas as funcionalidades para o enriquecimento semântico dos metadados obtidos a partir da exploração do catálogo relacional. Importante notar que os metadados recuperados dos catálogos relacionais são meras declarações de estruturas. Há pouca ou nenhuma semântica que se possa obter a partir da declaração das estruturas das tabelas físicas onde os modelos dimensionais encontram-se construídos. Assim, é necessário oferecer um ferramental capaz de permitir aos usuários das instituições credenciadas atribuírem semântica aos metadados estruturais.

As funcionalidades aqui apresentadas somente podem ser executadas depois de pelo menos uma execução bem sucedida do serviço "*exploit_discovery_request"* para algum banco de dados da instituição credenciada. Após uma carga bem sucedida de modelos dimensionais, o usuário deve complementar os metadados semânticos, sem os quais, não é possível efetuar a geração das triplas que serão utilizadas para alimentar o catálogo do *framework*.

No apêndice D deste trabalho, apresentamos cada uma das funcionalidades oferecidas pela ferramenta para o enriquecimento de metadados.

5.3.3
Camada RDF Triples Generation

Esta camada é responsável pela geração dos arquivos contendo as triplas que serão carregadas no catálogo do *framework*. A geração das triplas somente é possível se as informações de metadados obrigatórias estiverem preenchidas. São gerados diversos conjuntos de triplas atendendo a diversas necessidades dos demais módulos do *framework*.

O detalhamento do processo de triplificação será apresentado na seção 5.5, neste mesmo capítulo. Aqui apresentaremos apenas a funcionalidade através da qual o processo de triplificação é realizado e seu funcionamento.

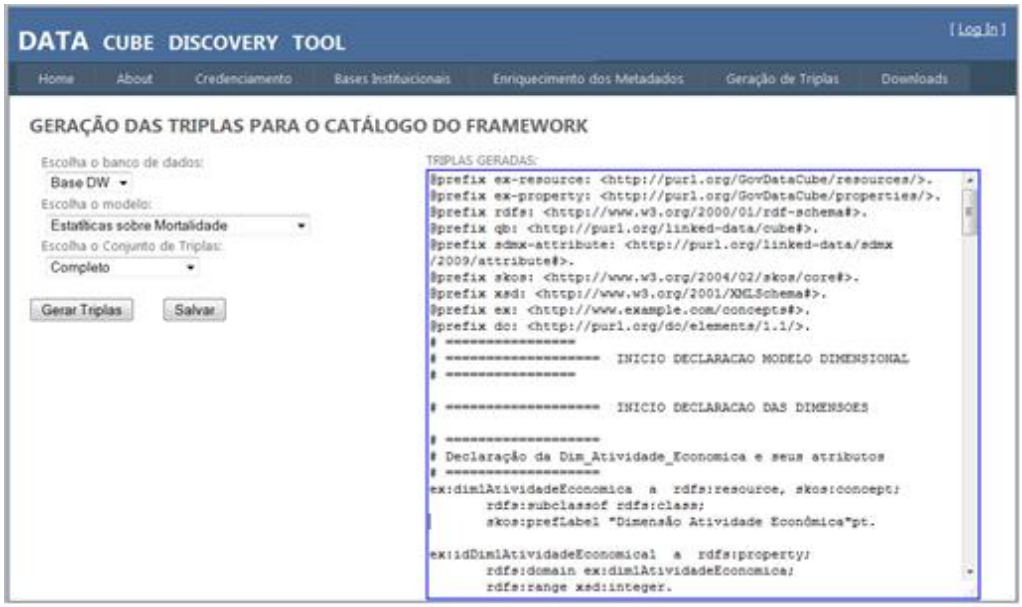


Figura 51: Tela da Geração de Triplas

O usuário deve escolher o banco de dados e o modelo que será triplificado. Além disso deve escolher o conjunto de triplas que serão gerados: todas, apenas as triplas do modelo dimensional, apenas as triplas dos cubos ou apenas as triplas R2RML. Clicando no botão “Gerar Triplas”, o sistema processa o programa de geração das triplas, apresentando o resultado na caixa de texto localizada no lado direito da tela. O usuário pode examinar as triplas geradas e gravar o resultado em um arquivo. Este arquivo será utilizado para ser efetuada a carga do catálogo do *framework*.

5.4 Resumo

Neste capítulo apresentamos os processos desenvolvidos para viabilizar a implantação e uso da ferramenta por instituições credenciadas, uma descrição da arquitetura interna da ferramenta e dos seus principais componentes e de como esses interagem entre si e são utilizados pelo usuário para produzir as triplas.

6.
Heurísticas para Identificação de Modelos Dimensionais

6.1
Formulação das Heurísticas

Recorde da seção 2.1.1 que os modelos dimensionais devem exibir certas propriedades, resumidas abaixo:

A. Os atributos próprios de uma F-TAB, isto é, aqueles que não são chave-estrangeira de uma das suas dimensões, serão, sempre, de natureza numérica, representando exclusivamente observações medidas.
B. As chaves artificiais a serem utilizadas nas dimensões deverão ser sequências simples numéricas.
C. Todas as colunas de uma F-TAB são numéricas.
D. Toda coluna de uma F-TAB que não for chave-estrangeira representará uma observação medida.
E. Se duas observações medidas coexistem em um mesmo ambiente dimensional e possuem o mesmo nome, ainda que pertençam a diferentes F-TABs, elas tem o mesmo significado.
F. Em um dado ambiente dimensional, cada D-TAB é única e, portanto, equivalências entre dimensões só serão possíveis quando estivermos comparando dimensões de diferentes ambientes dimensionais.
G. A chave-primária de uma F-TAB é definida pela concatenação das chaves-estrangeiras de suas dimensões.

Além destas propriedades de modelos dimensionais, observamos que:

H. Apenas modelos dimensionais serão tratados no presente trabalho, devendo ser desprezado todo e qualquer modelo E-R clássico, normalizado ou não.
I. Modelos dimensionais estrela e em floco de neve serão, ambos, tratados.

Com base nestas assertivas, formulamos as seguintes heurísticas que nortearam o desenvolvimento da solução de identificação de cubos de dados:

1. Uma F-TAB é uma tabela composta exclusivamente de colunas com tipos de dados numéricos.
2. A chave primária de uma F-TAB é uma chave composta, isto é, deve ser resultado de uma combinação das suas colunas.
3. Cada uma das colunas componentes da chave primária de uma F-TAB deve ser uma chave estrangeira de alguma outra tabela.
4. As colunas que não participam da chave primária de uma F-TAB representam as observações medidas naquela F-TAB.
5. As tabelas cujas chaves primárias são chaves estrangeiras em uma F-TAB são consideradas dimensões daquela F-TAB.
6. Se uma tabela D-TAB possui uma chave estrangeira, então o modelo dimensional está normalizado (formato floco de neve); a tabela cuja chave primária está declarada como chave estrangeira na D-TAB possui uma relação de hierarquia com a D-TAB. Tabelas nessas condições serão consideradas *outrigger table*, que são extensões da D-TAB primária que se encontra conectada à F-TAB.
7. Como uma F-TAB é a materialização de um relacionamento n-ário, a sua cardinalidade, isto é, a quantidade de tabelas dimensões com as quais se relaciona deve ser maior que dois. De fato, modelos dimensionais típicos envolvendo a D-TAB tempo, como aqueles que estão no escopo do nosso estudo, e possuem no mínimo três dimensões. Portanto, se uma tabela candidata a F-TAB materializar o relacionamento entre apenas duas candidatas a dimensões e não possuir ao menos um atributo próprio (medida potencial) além das chaves das dimensões potenciais, então ela será considerada um relacionamento N:M.
8. A assertiva F acima leva em consideração o uso de dimensões compartilhadas entre F-TABs em um mesmo ambiente dimensional. Em particular, a D-TAB tempo deve apresentar uma alta cardinalidade (quantidade de F-TABs com as quais se relaciona), uma vez que as dimensões em conformidade são únicas em um ambiente dimensional. Mais do que isso, como lidamos com modelos dimensionais onde a D-

TAB tempo é obrigatória, podemos afirmar que a cardinalidade da D-TAB tempo não apenas é a maior possível como, também, que é o número exato das F-TABs que existem naquele ambiente dimensional.

As heurísticas acima podem ser implementadas em uma única sentença SQL conforme pode ser observado no Apêndice A.

A fim de validarmos esta sentença SQL, criamos um modelo dimensional representado pela figura 52 em um banco de dados relacional contendo 66.695 objetos. As únicas estruturas dimensionais neste banco são aquelas apresentadas na figura e, consequentemente, apenas as F-TABs ali representadas deveriam retornar após a submissão do SQL.

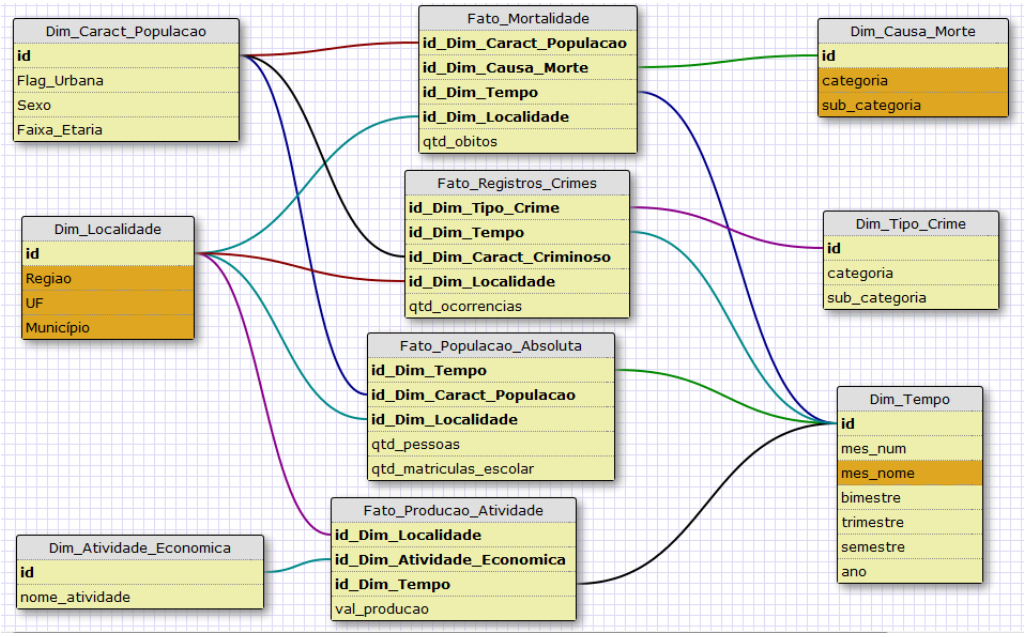


Figura 52: Modelo proposto para validação do serviço Exploit_Discovery

No entanto, a referida sentença SQL recupera não apenas F-TABs mas também gera alguns “falsos positivos”: tabelas que implementam relacionamentos triplos ou relacionamentos N:M em tradicionais modelos de dados normalizados.

Assim, precisamos conseguir separar esses “falsos positivos” das verdadeiras F-TABs. Para isso precisaremos de um algoritmo e de uma estrutura de dados que irá nos auxiliar a identificar as verdadeiras F-TABs. Para ilustrarmos adequadamente esta situação, iremos reproduzir abaixo as F-TABs potenciais recuperadas a partir do processamento de teste que realizamos

em uma base onde criamos os modelos dimensionais apresentados no início desta seção, misturado com modelos normalizados tradicionais.

	TABLE_NAME
1	CAUSUARIO_ESCRITORIO
2	DOMINIO_JUSTIF_EXCECAO_PRECO
3	ERRO_INCONS_SUPERMERCADO
4	ESTRUTURA_REFEICAO
5	FATO_MORTALIDADE
6	FATO_POPULACAO_ABSOLUTA
7	FATO_PRODUCAO_ATIVIDADE
8	FATO_REGISTROS_CRIMES
9	IBR_CLIINF_CONT_MEIO_CONT
10	IBR_CONT_AREA_INTERESSE
11	IBR_CONT_FORMACAO
12	INSUMO_FORM_PESQ_DEMANDA
13	REUTILIZA

Figura 53: Potenciais F-TABs identificadas pela query SQL

Como podemos observar na figura 53, apenas as linhas 5, 6, 7 e 8 correspondem a verdadeiras F-TABs. As demais linhas correspondem às tabelas que implementam relacionamentos N:M ou triplos. Conforme mencionamos anteriormente no final da seção 2.1.1, relacionamentos do tipo 1:N ou 1:1 não são objeto de interesse desse estudo, uma vez que uma F-TAB corresponde a uma implementação de um relacionamento n-ário.

6.2
Uma solução para os “falsos positivos”

Para podermos distinguir os “falsos positivos”, anteriormente mencionados, das verdadeiras F-TABs, utilizaremos uma matriz “Fato X Dimensão” cujo elemento possui a seguinte estrutura:

LABEL	VALOR
ATIVO	TOTAL ATRIB

Figura 54: Layout do elemento da Matriz “Fato X Dimensão”

Todo elemento situado na linha 0 da matriz corresponde a uma tabela D-TAB potencial e todo elemento situado na coluna 0 da matriz corresponde a

uma F-TAB potencial que foi identificada através da execução da sentença SQL apresentada anteriormente.

Conforme mencionamos anteriormente, o resultado da sentença SQL pode apresentar falsos positivos (relacionamentos triplos e N:M). Portanto consideramos que as tabelas resultantes da execução da sentença são candidatas a F-TABs. Analogamente, as dimensões vinculadas a tais tabelas serão consideradas, em um primeiro momento, como candidatas a dimensões pois podem na verdade representar simples tabelas não dimensionais.

Quando a linha for 0 ou a coluna for 0, o campo **label** do elemento da matriz estará preenchido com o nome da tabela (D-TAB ou F-TAB, conforme o caso) e o campo **valor** estará preenchido com a cardinalidade daquela tabela. A *cardinalidade* de uma F-TAB potencial é definida como o número de dimensões à ela vinculadas. A cardinalidade de uma D-TAB potencial é definida pelo número de F-TABs que utilizam aquela D-TAB.

Quando a linha for maior que 0 e a coluna menor que 0, o campo label do elemento da matriz estará com valor string.empty e o campo valor do elemento da matriz será 1, se a D-TAB representada naquela coluna está vinculada à F-TAB representada naquela linha, e 0 caso não esteja.

O campo **total atrib** do elemento aplica-se exclusivamente às F-TABs potenciais e será inicializado com a quantidade de colunas não chave encontradas na F-TAB potencial. Já o campo **ativo** aplica-se tanto às F-TABs como às dimensões e servirá para identificar quais tabelas dentre as F-TABs potenciais serão consideradas verdadeiras F-TABs, tendo o mesmo uso para as dimensões. É inicializado como "falso" para todas as F-TABs e dimensões potenciais.

O preenchimento da matriz "Fato X Dimensão" é realizado através do seguinte algoritmo.

```
-- cria a lista contendo todas as fatos potenciais – Etapa 1
Lista_Fatos_Potenciais.Carrega(sentenca_sql)

-- cria a lista contendo todas as dimensões potenciais – Etapa 2
Para cada fato_potencial na Lista_Fatos_Potenciais
 -- obtem a relação de dimensões potenciais associadas à fato potencial
   Lista_Dimensoes_Fato_Potenciais.LimparConteudo
   Lista_Dimensoes_Fato_Potenciais.Carregar(fato_potencial)
```

```
 -- cria a lista de dimensões potenciais
   Para cada dimensao_potencial na Lista_Dimensoes_Fato_Potenciais
      Se Não (Lista_Dimensoes_Potenciais.Encontrou(dimensao_potencial))
      Então
        Lista_Dimensoes_Potenciais.Insere(dimensao_potencial)
      Fim Se
   Fim Para
Fim Para

-- Aloca a matriz_fato_D-TAB – Etapa3
Dimensiona matriz_fato_dimensao as New tipo_matriz_fato_dimensao
 (Lista_Fatos_Potenciais.TotalElementos+1, Lista_Dimensoes_Potenciais.TotalElementos+1)

-- Coloca as dimensões em cada coluna da linha 0 – Etapa 4
Para indice = 0 até Lista_Dimensoes_Potenciais.TotalElementos - 1
   coluna = indice + 1
   matriz_fato_dimensao.no(0,coluna).label = Lista_Dimensoes_Potenciais(indice)
   matriz_fato_dimensao.no(0,coluna).valor = 0
   matriz_fato_dimensao.no(0,coluna).ativo = falso
Fim Para

-- Coloca as fatos em cada linha da coluna 0 – Etapa 5
Para indice = 0 até Lista_Fatos_Potenciais.TotalElementos - 1
   linha = indice + 1
   matriz_fato_dimensao.no(linha,0).label = Lista_Fatos_Potenciais(indice)
   matriz_fato_dimensao.no(linha,0).valor = 0
   matriz_fato_dimensao.no(linha,0).ativo = falso
   matriz_fato_dimensao.no(linha,0).tot_atrib =
      Lista_Fatos_Potenciais .ObtemTotAtrib(índice)
Fim Para

-- Preenche o interior da Matriz Fato x Dimensao
-- e incrementa a cardinalidade das fatos e dimensoes potenciais
– Etapa 6
Para linha = 1 até Lista_Fatos_Potenciais.TotalElementos
 -- obtem a relação de dimensões potenciais associadas à fato potencial
   Lista_Dimensoes_Fato_Potenciais.LimparConteudo
   Lista_Dimensoes_Fato_Potenciais.Carregar(matriz_fato_dimensao(linha,0).label)
   Para cada dimensao_potencial na Lista_Dimensoes_Fato_Potenciais
      Para coluna = 1 até Lista_Dimensoes_Potenciais.TotalElementos Faça
```

```
        Se matriz_fato_dimensao(0,coluna).label = dimensao_potencial então
          matriz_fato_dimensao(linha, coluna).valor = 1
          matriz_fato_dimensao(0,coluna).valor += 1
        Senão
          matriz_fato_dimensao(linha, coluna).valor = 0
        Fim Se
      Fim Para
      matriz_fato_dimensao(linha,0).valor += 1
   Fim Para
   matriz_fato_dimensao.no(linha,0).label = Lista_Fatos_Potenciais(indice)
   matriz_fato_dimensao.no(linha,0).valor = 0
Fim Para
```

Quadro 10: Algoritmo de identificação de F-TABs - parte 1

A execução do algoritmo acima gera a matriz "Fato X Dimensão" preenchida indicando quais dimensões potenciais estão vinculadas a quais F-TABs potenciais, a cardinalidade de cada D-TAB e F-TAB potencial e o flag **ativo** marcado como falso para todas as F-TABs potenciais.

As tabelas dimensões potenciais são obtidas executando-se o seguinte comando SQL para cada uma das F-TABs potenciais:

```
Select distinct table_name
From ALL_CONSTRAINTS
where constraint_name in
( Select r_constraint_name
  From ALL_CONSTRAINTS
Where constraint_type = 'R'
  and table_name = <NOME-DA-FATO-POTENCIAL>
```

Quadro 11: Sentença SQL para obtenção das dimensões de uma F-TAB

A consulta acima é utilizada pelo método "Carregar" da lista de dimensões "Lista_Dimensoes_Fato_Potenciais" apresentada no algoritmo anterior.

Na tabela 9 apresentamos as dimensões potenciais que foram carregadas durante a Etapa 2 do algoritmo apresentado no quadro 10, para cada uma das F-TABs potenciais obtidas pela execução da sentença SQL na Etapa 1 do mesmo algoritmo.

FATOS POTENCIAIS	DIMENSÕES POTENCIAIS
CAUSUARIO_ESCRITORIO	INFORMANTE_WEB
	USUARIO_FGV
DOMINIO_JUSTIF_EXCECAO_PRECO	JUSTIFICATIVA_EXCECAO_PRECO
	DOMINIO_JUSTIFICATIVA
ERRO_INCONS_SUPERMERCADO	ERRO_SUPERMERCADO
	INCONSISTENCIA_SUPERMERCADO
ESTRUTURA_REFEICAO	CLIENTE
	COMPOSICAO_REFEICAO
FATO_MORTALIDADE	DIM_CARAC_POPULACAO
	DIM_CAUSA_MORTE
	DIM_LOCALIDADE
	DIM_TEMPO
FATO_POPULACAO_ABSOLUTA	DIM_CARAC_POPULACAO
	DIM_LOCALIDADE
	DIM_TEMPO
FATO_PRODUCAO_ATIVIDADE	DIM_ATIVIDADE_ECONOMICA
	DIM_LOCALIDADE
	DIM_TEMPO
FATO_REGISTRO_CRIMES	DIM_CARAC_POPULACAO
	DIM_TIPO_CRIME
	DIM_LOCALIDADE
	DIM_TEMPO
IBR_CLIINF_CONT_MEIO_CONT	IBR_CLIINF_CONTATO
	IBR_CLIINF_MEIO
IBR_CONT_AREA_INTERESSE	IBR_AREA_INTERESSE
	IBR_CLIINF_CONTATO
IBR_CONT_FORMACAO	IBR_CLIINF_CONTATO
	IBR_FORMACAO_ACAD
INSUMO_FORM_PESQ_DEMANDA	INSUMO_FORMULARIO_PESQUISA
	IBR_DEMANDA
REUTILIZA	COMPOSICAO_ELEMENTAR
	ITEM_ELEMENTAR

Tabela 9: Relação das dimensões potenciais para as F-TABs potenciais

Dutante a Etapa 2 do algoritmo é gerada uma lista das dimensões sem repetição. Ao fim desta segunda etapa possuímos uma lista de F-TABs potenciais e uma lista de dimensões potenciais que utilizamos para definir o tamanho da matriz "Fato X Dimensão" que iremos alocar e, posteriormente, para orientar o processo de carga dos elementos desta matriz.

Lista_Fatos_Potenciais	
F1	CAUSUARIO_ESCRITORIO
F2	DOMINIO_JUSTIF_EXCECAO_PRECO
F3	ERRO_INCONS_SUPERMERCADO
F4	ESTRUTURA_REFEICAO
F5	FATO_MORTALIDADE
F6	FATO_POPULACAO_ABSOLUTA
F7	FATO_PRODUCAO_ATIVIDADE
F8	FATO_REGISTRO_CRIMES
F9	IBR_CLIINF_CONT_MEIO_CONT
F10	IBR_CONT_AREA_INTERESSE
F11	IBR_CONT_FORMACAO
F12	INSUMO_FORM_PESQ_DEMANDA
F13	REUTILIZA

Lista_Dimensoes_Potenciais	
D1	INFORMANTE_WEB
D2	USUARIO_FGV
D3	JUSTIFICATIVA_EXCECAO_PRECO
D4	DOMINIO_JUSTIFICATIVA
D5	ERRO_SUPERMERCADO
D6	INCONSISTENCIA_SUPERMERCADO
D7	CLIENTE
D8	COMPOSICAO_REFEICAO
D9	DIM_CARAC_POPULACAO
D10	DIM_CAUSA_MORTE
D11	DIM_LOCALIDADE
D12	DIM_TEMPO
D13	DIM_ATIVIDADE_ECONOMICA
D14	DIM_TIPO_CRIME
D15	IBR_CLIINF_CONTATO
D16	IBR_CLIINF_MEIO
D17	IBR_AREA_INTERESSE
D18	IBR_FORMACAO_ACAD
D19	INSUMO_FORMULARIO_PESQUISA
D20	IBR_DEMANDA
D21	COMPOSICAO_ELEMENTAR
D22	ITEM_ELEMENTAR

Tabela 10: Listas das F-TABs e D-TAB Potenciais

A Etapa 3 do algoritmo utiliza o tamanho das listas acima para alocar o tamanho exato da matriz “Fato X Dimensão”. Em seguida, a Etapa 4 preenche as colunas da linha 0 da matriz com as dimensões potenciais, conforme podemos observar na figura 55. O label do elemento corresponde à informação localizada mais acima e a cardinalidade, campo valor aparece abaixo.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
		D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15	D16	D17	D18	D19	D20	D21	D22
0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F

Figura 55: Preenchimento da Linha 0 da Matriz “Fato X Dimensão”

A Etapa 5 preenche as linhas da coluna 0 da matriz com as F-TABs potenciais, conforme ilustrado pela figura 56, destacando que neste caso os quatro campos do elemento estão preenchidos. O primeiro deles é o label, o segundo é o campo valor, representando a cardinalidade da F-TAB potencial, o

terceiro é o flag ativo e, finalmente, o quarto corresponde ao total de atributos não chave existentes naquela F-TAB potencial.

	0			
0				
1	F1	0	F	0
2	F2	0	F	0
3	F3	0	F	0
4	F4	0	F	0
5	F5	0	F	1
6	F6	0	F	2
7	F7	0	F	1
8	F8	0	F	1
9	F9	0	F	0
10	F10	0	F	0
11	F11	0	F	0
12	F12	0	F	0
13	F13	0	F	0

Figura 56: Preenchimento da Coluna 0 da Matriz "Fato X Dimensão"

A sexta e última etapa deste algoritmo é responsável por marcar nas células internas da matriz, linha 1 e coluna 1 em diante, quais F-TABs potenciais se relacionam com quais dimensões potenciais e atualizar a cardinalidade delas.

	0			1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
0				D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15	D16	D17	D18	D19	D20	D21	D22
				1	1	1	1	1	1	0	0	4	2	4	4	1	1	1	1	1	1	1	1	1	1
				F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
1	F1	2	F	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	F2	2	F	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	F3	2	F	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	F4	2	F	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0
5	F5	4	F	1	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0
6	F6	3	F	2	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0
7	F7	3	F	1	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0
8	F8	4	F	1	0	0	0	0	0	0	0	1	0	1	1	0	1	0	0	0	0	0	0	0	0
9	F9	2	F	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0
10	F10	2	F	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
11	F11	2	F	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0
12	F12	2	F	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
13	F13	2	F	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figura 57: Matriz "Fato X Dimensão" preenchida

A segunda parte do algoritmo que identifica as F-TABs faz, simplesmente, um percurso pela matriz aplicando a heurística 8, formulada anteriormente nesta seção, para identificar as F-TABs que realmente devem ser consideradas como tais.

Aplicando-se o critério formulado na heurística 8, devemos percorrer a linha 0 da matriz e identificar qual a D-TAB potencial com maior cardinalidade e marcamos seu flag **ativo** para "verdade". A partir desta D-TAB, localizaremos as F-TABs potenciais à ela vinculadas e também a marcaremos com o flag **ativo** "verdade". A partir de cada uma das F-TABs marcadas como "verdade", identificaremos suas outras dimensões e as marcaremos como "verdade". Após este procedimento, todas as demais candidatas a dimensões e a F-TABs que sobrarem com flag **ativo** "falso" serão descartadas.

Caso haja empate e mais de uma D-TAB tenha a maior cardinalidade, adotaremos o procedimento acima para todas, lembrando que, com base na assertiva F e na regra 1.3 da tabela 1 (Melhores Práticas em Modelagem Dimensional), a D-TAB tempo deve ser única em um dado modelo dimensional e, ainda que alguma outra D-TAB possa estar sendo compartilhada por todas as F-TABs do modelo, no final, ambas estarão vinculadas às mesmas F-TABs o que não invalida o procedimento aqui proposto.

Aplicando-se o procedimento acima à matriz "Fato X Dimensão" que confeccionamos anteriormente, temos:

	0				1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
0					D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15	D16	D17	D18	D19	D20	D21	D22
					1	1	1	1	1	1	0	0	4	2	4	4	1	1	1	1	1	1	1	1	1	1
					F	F	F	F	F	F	F	F	V	V	V	V	V	V	F	F	F	F	F	F	F	F
1	F1	2	F	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	F2	2	F	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	F3	2	F	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	F4	2	F	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0
5	F5	4	V	1	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0
6	F6	3	V	2	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0
7	F7	3	V	1	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0
8	F8	4	V	1	0	0	0	0	0	0	0	0	1	0	1	1	0	1	0	0	0	0	0	0	0	0
9	F9	2	F	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0
10	F10	2	F	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
11	F11	2	F	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0
12	F12	2	F	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
13	F13	2	F	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figura 58: Matriz "Fato X Dimensão" após aplicação do Algoritmo de Identificação das Fatos

Podemos constatar que as tabelas identificadas correspondem exatamente àquelas que constituem o modelo dimensional que utilizamos para teste.

No caso de um modelo estrela, a tarefa pode ser considerada concluída. No entanto, desejamos tratar também os modelos em floco de neve. Na verdade, ainda não sabemos se o modelo acima é uma estrela ou um floco de neve. Para sabermos isso, iremos utilizar outra estrutura de dados: uma árvore.

6.3
Extração de modelos dimensionais de um esquema relacional

A partir da matriz "Fato X Dimensão", construiremos uma árvore, tendo no seu primeiro nível as F-TABs. Usaremos um algoritmo de busca em profundidade para orientar tanto a construção como o percurso na árvore.

Para cada F-TAB que tiver o flag **ativo** "verdadeiro", o algoritmo cria um elemento filho a partir da raiz da árvore. Em seguida, para cada D-TAB vinculada à F-TAB, o algoritmo percorre a árvore, pesquisando se aquela D-TAB já existe em algum ramo da árvore. Caso exista, o elemento da F-TAB recém adicionado, ganhará uma ligação com o elemento da D-TAB já existente. Caso a D-TAB ainda não esteja inserida na árvore, ela é inserida como um novo ramo filho do elemento da F-TAB recém acrescentada.

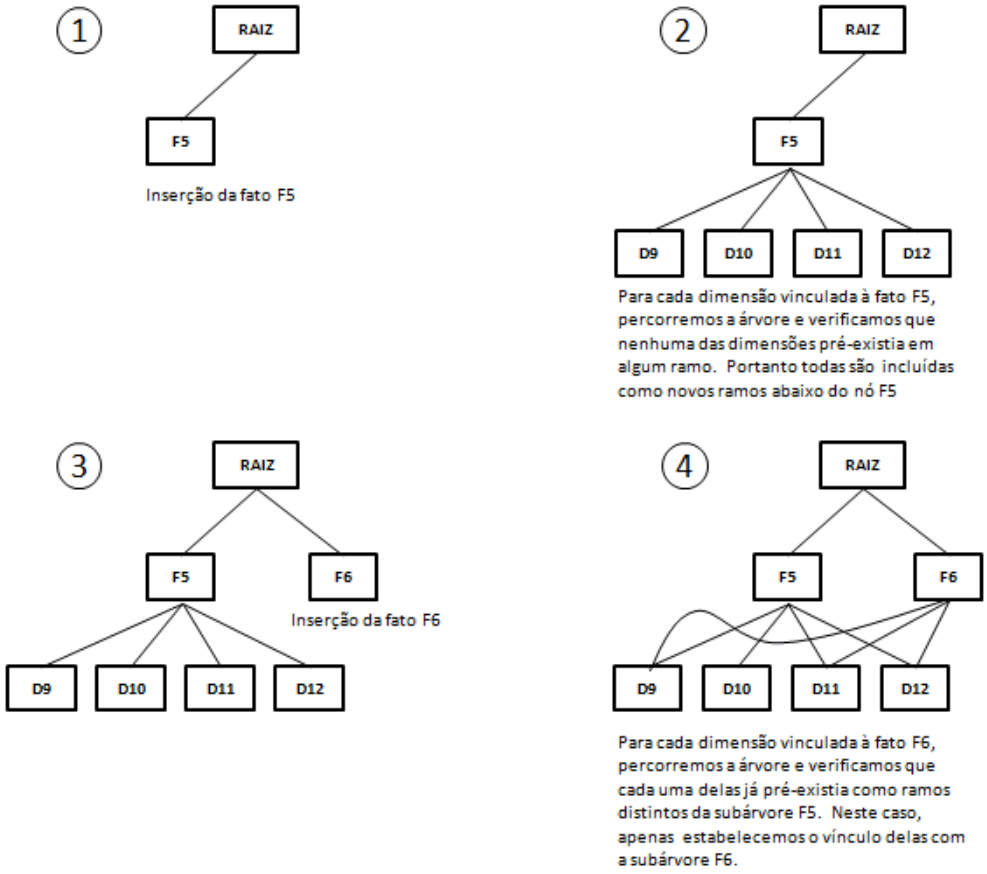


Figura 59: Construção da Árvore Dimensional - Parte 1

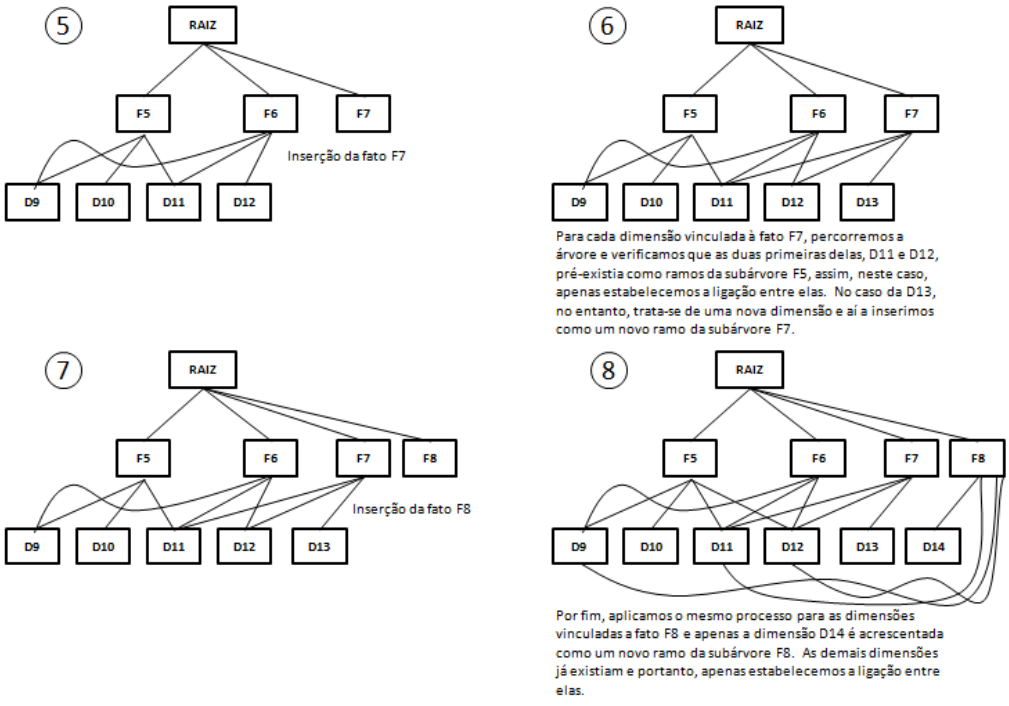


Figura 60: Construção da Árvore Dimensional - Parte 2

Ocorre que podemos estar trabalhando com modelos em floco de neve ao invés de modelos em formato estrela. Em ambos os casos, temos uma F-TAB central orbitada por tabelas dimensões diretamente vinculadas a F-TAB. No entanto, quando lidamos com modelos em floco de neve, ao menos uma das dimensões vinculadas à F-TAB deverá estar normalizada, possuindo, portanto, ao menos uma D-TAB secundária (outrigger) vinculada à D-TAB primária num relacionamento 1:N, onde a chave da D-TAB secundária aparecerá como uma chave estrangeira na D-TAB primária.

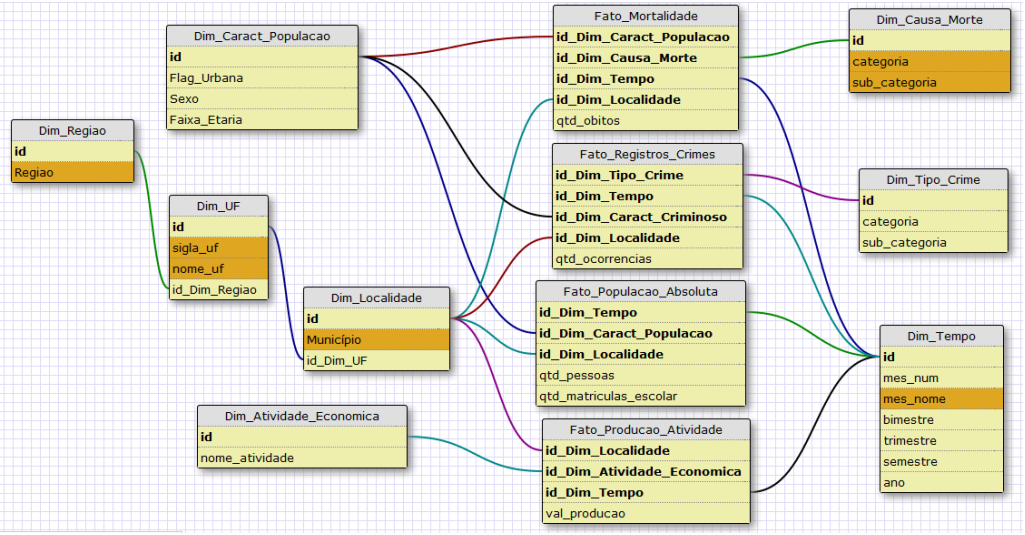


Figura 61: Modelo Dimensional em Floco de Neve

Na figura 61 normalizamos a D-TAB localidade do nosso modelo original para ilustrar como o modelo dimensional que utilizamos poderia ser transformado em um modelo em floco de neve.

Um ponto importante a destacar é que a normalização aplicada a tabela D-TAB localidade não mudaria em absolutamente nada todo o processamento que realizamos anteriormente. Se o nosso modelo fosse um modelo em floco de neve, como o acima proposto, teríamos criado exatamente a mesma matriz "Fato X D-TAB", identificado exatamente as mesmas fatos e dimensões e criado exatamente a mesma estrutura de árvore que apresentamos no quadro 8 da figura 60. O motivo para isso é bem simples na verdade: comparando os dois modelos dimensionais, podemos reparar que as dimensões que estão vinculadas diretamente às F-TABs são as mesmas. O fato de haver ou não dimensões secundárias (*outrigger*) não afeta em nada o resultado do processamento.

Portanto, podemos perceber que o nosso trabalho ainda não se encerrou e que para atingirmos o objetivo a que nos propusemos de tratarmos, igualmente, modelos em estrela e modelos em floco de neve, precisamos de mais um procedimento, o qual batizamos de procedimento de verificação de dimensões secundárias.

O procedimento de verificação de dimensões secundárias é baseado em um percurso em profundidade na árvore, visitando cada uma das dimensões, isto é, subárvores situadas a partir do nível 2, considerando a raiz como nível 0 e as subárvores imediatamente vinculadas à raiz (F-TABs) como nível 1. Para cada uma dessas dimensões, verificamos se a mesma possui alguma chave estrangeira declarada em sua estrutura. Caso possua, inserimos um elemento filho abaixo dela com a identificação da D-TAB secundária identificada. Para cada D-TAB inserida, tornamos a verificar se esta possui uma chave estrangeira em sua estrutura e repetimos o procedimento até que não haja mais dimensões inseridas com chaves estrangeiras em sua estrutura. Neste momento, passamos para a próxima D-TAB (caso ainda não tenhamos visitado todas) e repetimos o procedimento até que todas as dimensões declaradas no nível 2 da estrutura da árvore tenham sido visitadas e verificadas e, opcionalmente, tenham a sua estrutura de dimensões secundárias identificadas e anexadas à árvore.

No nosso exemplo, a D-TAB localidade passou a possui duas outras dimensões secundárias, vinculadas à si: a DIM_UF e a DIM_REGIÃO. A D-TAB localidade em nosso modelo é identificada pelo código D11. Como o último código das dimensões potenciais foi o código D24, os códigos que serão atribuídos a estas duas novas tabelas serão: D25 e D26, respectivamente.

Aplicando o procedimento acima descrito ao nosso caso teríamos:

1. Começamos o percurso posicionando-nos no ramo mais à esquerda da sub árvore definida pela F-TAB F5.

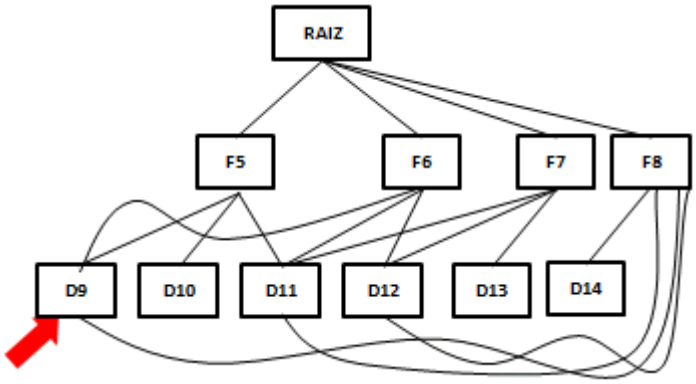


Figura 62: Percurso na árvore - passo 1.

2. Como não há chave estrangeira na D-TAB D9, iremos buscar a próxima D-TAB, no caso a D10.

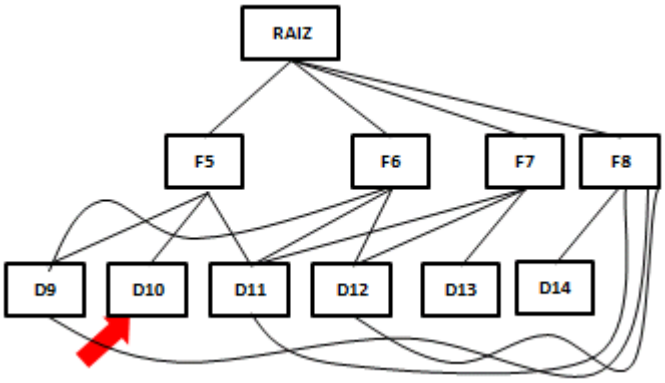


Figura 63: Percurso na árvore - passo 2

3. Como D10 também não possui chave estrangeira, passamos para D11.

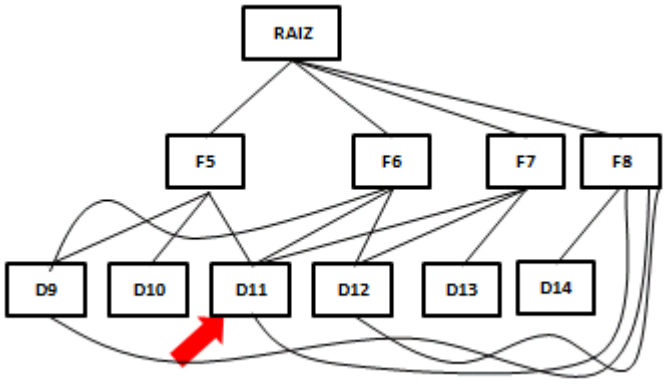


Figura 64: Percurso na árvore - passo 3

4. D11 é a D-TAB localidade e possui uma chave estrangeira. Assim, recuperamos a D-TAB UF à qual atribuímos o código D25 e a inserimos na nossa estrutura.

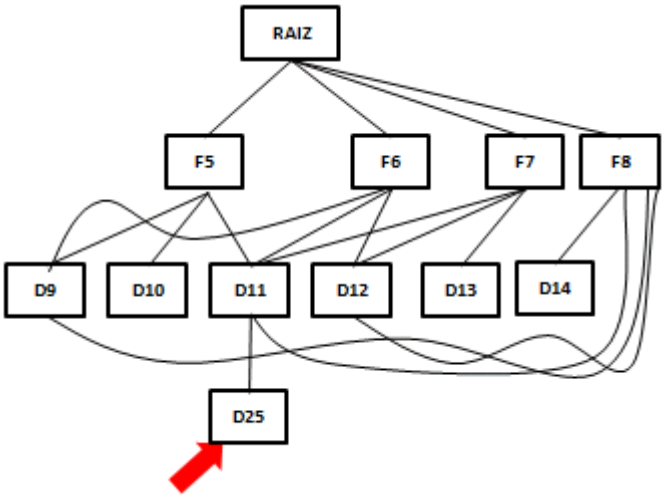


Figura 65: Percurso na árvore - Inserção de D-TAB Secundária

5. Verificamos que a D-TAB recém inserida também possui uma chave estrangeira e adicionamos à estrutura essa nova D-TAB secundária.

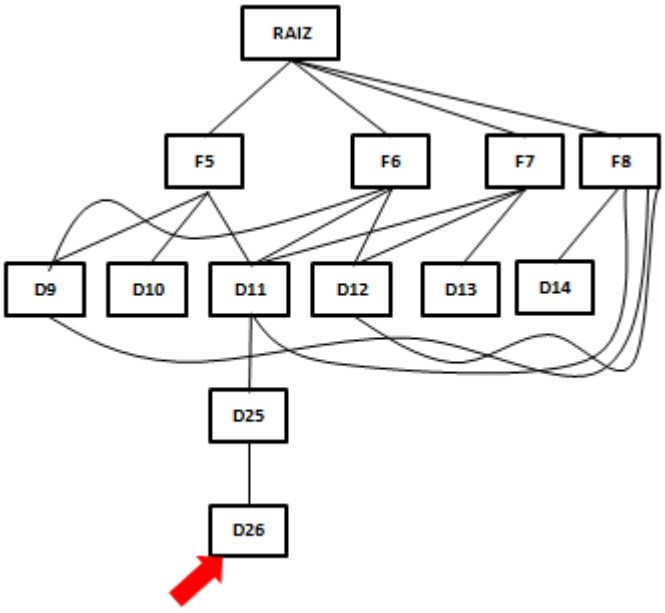


Figura 66: Percurso na árvore - Inserção de Nova D-TAB Secundária

6. Inserimos a D-TAB D26, D-TAB Região, na estrutura e verificamos que ela não possui chave estrangeira em sua estrutura. A partir daí, seguiremos sucessivamente para as dimensões D12, D13 e D14

Esta estrutura de árvore será utilizada para gerarmos o arquivo XML com a estrutura do catálogo declarada e que deverá ser enviado para o servidor da DCD TOOL a fim de permitir a atualização das tabelas do catálogo interno.

Antes, porém, de explicarmos como essa estrutura nos auxiliará a gerar o arquivo XML que utilizaremos para transmitir as informações coletadas no catálogo do banco, precisamos revelar um último e fundamental detalhe da construção desta estrutura. A cada operação de inserção de um elemento na árvore, além das informações próprias daquele elemento (nome da tabela, tipo – D-TAB ou F-TAB, etc) também criamos uma lista contendo todas as colunas componentes da estrutura daquela tabela, com todas as informações importantes sobre aquela coluna, tais como nome, tipo de dado, se é ou faz parte da chave primária, posição da coluna na chave primária, se é chave ou faz parte de chave estrangeira, tamanho máximo e etc. A figura 73 ilustra essa estrutura.

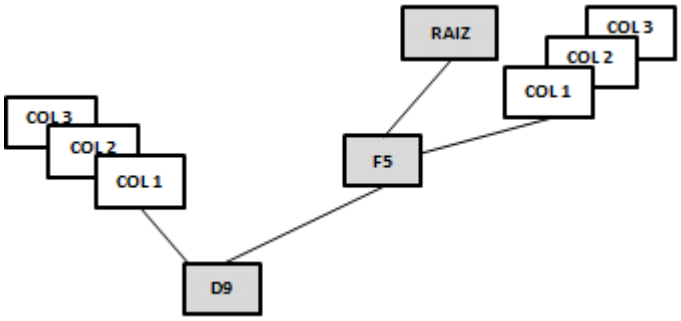


Figura 67:Visão da Lista de colunas na árvore dimensional

Finalmente, percorremos a estrutura da árvore uma última vez para gerar o arquivo XML que será devolvido para o site da DCD Tool através da solicitação do serviço "*save_models_request*" disponibilizado pelo Web service que instalamos no nosso site para viabilizar a comunicação assíncrona.

Cada subárvore definida pelos elementos localizados no nível 1 da estrutura da árvore e que correspondem às F-TABs, será considerado um modelo dimensional independente. Isto porque a partir de cada F-TAB projetamos um único cubo dimensional independente. A partir deste elemento percorremos o ramo definido a partir dele e iremos criando os elementos da estrutura XML com toda a riqueza de detalhes necessária e que já se encontra disponível na estrutura que construímos anteriormente. O arquivo XML correspondente ao modelo dimensional que apresentamos como estudo de caso nesta seção foi disponibilizado no apêndice B deste trabalho.

6.4 Resumo

Neste capítulo detalhamos o processo que elaboramos para realizar a identificação de modelos dimensionais em catálogos de bancos de dados relacionais.

7.
Conclusão

7.1
Contribuições

Nesta dissertação apresentamos o conjunto de ferramnetas DCD, concebidas para pesquisar e identificar modelos dimensionais em catálogos de bancos de dados relacionais, descrever sua estrutura, permitir o enriquecimento de seus metados, suportar a definição de equivalência de conceitos através dos vocábulos skos:exactMatch e skos:closeMatch e gerar triplas baseadas em RDF e RDFS e outras ontologias padronizadas como SKOS, R2RML e o *Data Cube Vocabulary* para alimentar o catálogo do *framework OLAP2DataCube Catalog On Demand*. A despeito das múltiplas facetas da ferramenta e da abrangência do seu escopo, seu principal objetivo é gerar as descrições dos cubos de dados interligados que permitirão aos demais módulos do *framework* desempenharem suas funções.

Durante o processo de desenvolvimento da ferramenta identificamos a oportunidade de agregar algumas funcionalidades à camada de enriquecimento de metadados, tais como as identificações de equivalência entre colunas e a declaração de hierarquias, particularmente útil em modelos desnormalizados como os modelos em formato estrela.

Consideramos como principal contribuição deste trabalho, a ferramenta de geração de triplas, em particular de triplas R2RML. Destacamos o fato dela ser capaz de produzir triplas que partem da declaração das estruturas dos modelos dimensionais, passando pela estrutura dos cubos gerados a partir de tais modelos, até chegar nas triplas que permitem a recuperação das instâncias armazenadas nas dimensões e F-TABs materializadas em um banco de dados relacional.

Muitas ferramentas de mercado não oferecem a mesma abrangência na produção de triplas, conforme ilustrado no apêndice G.

Identificamos e solucionamos o gap semântico deixado pelo *Data Cube Vocabulary* em relação à representação de modelos dimensionais, uma vez que o foco desta ontologia concentra-se na declaração de estrutura de cubos. Para isso, empregamos RDF, RDFS e SKOS, e viabilizamos a representação dos modelos dimensionais através de triplas, preenchendo este gap (apêndices E e F).

Para viabilizar o processo de triplificação, elaboramos uma estrutura genérica de catálogo para bancos relacionais (apêndice C) que pode ser utilizada não apenas com o fim aqui apresentado mas como uma estrutura auxiliar para a realização de migração de bases de dados de uma tecnologia de banco de dados para outra.

Oferecemos ainda ferramentas de apoio ao enriquecimento dos metadados dos modelos dimensionais (apêndice D)

Ao contrário de outros trabalhos realizados na mesma área, nossa ferramenta é capaz de tratar tanto modelos estrelas quanto em floco de neve. Na verdade, qualquer que seja o modelo dimensional, estamos aptos a identificá-lo e distingui-lo de modelos tradicionais que apenas suportam processos operacionais.

Também desenvolvemos um gerador de representação de modelos relacionais e dimensionais em XML (apêndice B).

Por fim, para que tudo isso fosse possível, elaboramos e implementamos um conjunto de heurísticas e algoritmos capazes de identificar e distinguir modelos dimensionais em catálogos de bancos de dados relacionais (apêndice A e Capítulo 6).

7.2
Trabalhos Futuros

O uso de RDF e RDFS ainda nos impõem restrições que poderiam ser superadas com o uso de owl. Nas declarações de *rdfs:range*, só podemos atribuir múltiplos *ranges* ou *domains* à uma propriedade se utilizarmos *owl:unionOf*.

Também precisamos aprimorar o algoritmo de identificação de F-TABs para distinguir as F-TABs primárias (aquelas que possuem grão no nível

atômico) daquelas que são, simplesmente, agregações geradas a partir das primeiras.

A introdução de elementos gráficos que representem os modelos dimensionais irá certamente tornar as ferramentas de enriquecimento de metadados mais atrativas e interessantes para se trabalhar.

No que se refere especificamente ao *framework*, vislumbramos a possibilidade de torná-lo mais flexível, capacitando-o não apenas a suportar definições de consultas realizadas dinamicamente pelas aplicações usuárias, como também oferecer a possibilidade de oferecer visões baseadas em mashup de dados, implementando, assim, o suporte à consultas federadas.

8. Referências Bibliográficas

Auer, S., Dietzold, S. and Riechert, T. 2006. Ontowiki - A Tool for Social, Semantic Collaboration. **The 5th International Semantic Web Conference**. 2006.

Berners-Lee, T. 2006. **Linked Data**. W3C. [Online] 07 27, 2006. [Cited: 07 12, 2013.] http://www.w3.org/DesignIssues/LinkedData.html.

Bizer, C., et al. 2007. Interlinking Open Data on the Web. **4th European Semantic Web Conference**, 2007. Poster Paper. 2007.

Bizer, C., Heath, T. and Bernes-Lee, T. 2009. **Linked Data - The Story so Far.** 2009.

Brickley, G. and Guha, R. V. 2004. **RDF Vocabulary Description Language 1.0:** RDF Schema. [Online] W3C, 2004. http://www.w3.org/TR/rdf-schema/.

Carrol, J. J., et al. 2004. Jena: implementing the semantic Web recommendations. **In International World Wide Web Conference**, pages 74-83. New York, NY, USA : s.n., 2004.

Cerbah, F. 2008. Learning Highly Structured Semantic Repositories from Relational Databases - The RDBtoOnto Tool. **In: 5th European Semantic Web Conference**, pages 777-781. Tenerife, Spain : s.n., 2008.

Cyganiak, R. and Jentzsch, A. 2011. **The Linking Open Data Cloud Diagram.** [Online] 2011. http://richard.cyganiak.de/2007/10/lod/.

Cyganiak, R., Reynolds, T. and Tennison, J. 2013. **The RDF Data Cube Vocabulary.** W3C working draft. [Online] 05 2013. [Cited: 05 17, 2013.] http://www.w3.org/TR/vocab-data-cube/.

Erling, O. and Mikhailov, I. 2009. Rdf support in the virtuoso dbms. **Networked Knowledge-Networked Media***, pages 7–24.* 2009.

Falkovych, K., Sabou, M. and Stuckenschmidt, H. 2003. **UML for the Semantic Web:** Transformation Based Approaches. [book auth.] B. Omelayenko and M. Klein. Knowledge Transformation for the Semantic

Web, Frontiers in Artificial Intelligence and Applications*, Vol. 95.* Amsterdam : IOS Press, 2003.

Ghozzi, F., et al. 2003. Constraints and Multidimensional Databases. *ICEIS'03:* **5th International Conference on Enterprise Information Systems***, pages 104-111.* 2003.

Hartig, O., Bizzer, C. and Freytag, J. 2009. Executing SPARQL Queries over the Web of Linked Data. **8th International Semantic Web Conference***.* Chantilly, VA, USA : s.n., 2009.

Heath, T. and Bizer, C. 2011. **Linked Data***.* s.l. : Morgan & Claypool Publishers, 2011.

—. 2011. **Linked Data: Evolving the Web into a Global Data Space**. 2011.

ISO. 2005. **Statistical data em metadata exchange (sdmx)***. Technical report.* 2005. ISO/TS 17369:2005.

Maia, Macedo S., et al. 2012. Junções Adaptativas em Consultas Federadas sobre Linked Data*.* 2012. **Simpósio Brasileiro de Banco de Dados**.

Kendall, G. C., Feigenbaum, L. and Torres, E. 2008. **SPARQL Protocol for RDF**. [Online] W3C, 2008. http://www.w3.org/TR/rdf-sparql-protocol/.

Kimball,Ralph. 2013. : http://www.kimballgroup.com/2013/02/05/design-tip-152-slowly-changing-dimension-types-0-4-5-6-7/. [Online] Fevereiro 05, 2013. [Cited: Maio 01, 2013.]

—. 1996. http://www.kimballgroup.com/1996/04/02/slowly-changing-dimensions-2/. [Online] Abril 02, 1996. [Cited: Abril 20, 2013.]

—. 1998. http://www.kimballgroup.com/1998/05/02/surrogate-keys/. [Online] Maio 02, 1998. [Cited: Maio 05, 2013.]

—. 2000. http://www.kimballgroup.com/2000/09/15/design-tip-13-when-a-fact-table-can-be-used-as-a-dimension-table/. [Online] Setembro 15, 2000. [Cited: Abril 20, 2013.]

—. 2001. http://www.kimballgroup.com/2001/10/24/what-not-to-do/. [Online] Outubro 24, 2001. [Cited: Maio 05, 2013.]

—. 2005. http://www.kimballgroup.com/2005/03/10/slowly-changing-dimensions-are-not-always-as-easy-as-1-2-3/. [Online] Março 10, 2005. [Cited: Abril 20, 2013.]

—. 2009. http://www.kimballgroup.com/2009/05/29/the-10-essential-rules-of-dimensional-modeling/. [Online] Maio 29, 2009. [Cited: Maio 02, 2013.]

—. 2011. http://www.kimballgroup.com/2011/06/01/design-tip-135-conformed-dimensions-as-the-foundation-for-agile-data-warehousing/. [Online] Junho 01, 2011. [Cited: Maio 02, 2013.]

—. 2012. http://www.kimballgroup.com/2012/02/01/design-tip-142-building-bridges/. [Online] Fevereiro 01, 2012. [Cited: Maio 04, 2013.]

—. 2002. **The Data Warehouse Toolkit - Segunda Edição***.* Rio de Janeiro : Campus, 2002. 85-352-1129-2.

Machado, Felipe Nery Rodrigues. 2006. **Tecnologia e Projeto de Data Warehouse:** Uma visão multidimensional*.* Tatuapé : Érica, 2006.

Martínez, Ana, et al. 2004. Las categorías o facetas fundamentales: una metodología para el diseño de taxonomías corporativas de sitios Web argentinos. **Ciência da Informação***.* 2004, Vol. 33, 2.

Moody, Daniel L. and Kortink, Mark A. R. 2000. **From Enterprise Models to Dimensional Models: A Methodology for Data Warehouse and Data Mart Design**. Melbourne : s.n., 2000.

Noy, N., et al. 2006. **Defining n-ary relations on the semantic Web**. [Online] 2006. [Cited: 07 10, 2013.] http://www.w3.org/TR/swbp-n-aryRelations/.

Pesce, Márcia. 2011. **RdXel um conjunto de ferramentas para manipulação de dados estatísticos.** Rio de Janeiro : s.n., 2011.

Ramalho, R. A. S., Vidotti, S. A. B. G. and Fujita, M. S. L. 2005. **Web semântica: aspectos interdisciplinares para a organização e recuperação de informação***.* Florianópolis : PGCIN/UFSC, 2005.

Ruback, Livia, et al. 2013. A Mediator for Statistical Linked Data*.* Coimbra : s.n., 2013. **28th Symposium On Applied Computing**.

Salas, Percy. 2011. **StdTrip: An a priori design approach and process for publishing Linked Data***.* Rio de Janeiro : s.n., 2011.

Salas, Percy, et al. 2012. **OLAP2DataCube: An Ontowiki Plug-In for Statistical Data Publishing.** TOPI. 2012.

Seaborne, A. and Bizer, C. 2004. D2RQ - treating non-RDF databases as virtual RDF graphs. **The 4th International Semantic Web Conference**. 2004.

Sören, Auer, et al. 2007. **DBpedia: A Nucleus for a Web of Open Data**. 2007.

Thomsen, E. 1997. OLAP **Solutions: Building Multidimensional Information Systems.** 1997.

Van Rees, Reinout. 2003. **Clarity in the usage of the terms ontology, taxonomy.** [Online] 2003. [Cited: 07 13, 2013.] http://reinout.vanrees.org/_downloads/2003_cib.pdf.

Vital, Luciane Paula and Café, Ligia Maria Arruda. 2011. Ontologias e Taxonomias: diferenças. **Perspectivas em Ciência da Informação.** 2011, Vol. 16, 2.

9. Apêndice A: SQL para identificação de Fatos Potenciais

Comando SQL que permite identificar potenciais F-TABs no catálogo de um banco de dados Oracle.

```
Select distinct table_name from
 (
 select a.TABLE_NAME
  from ALL_CONSTRAINTS a
  where a.OWNER NOT IN ('SQLTXPLAIN', 'SYS', 'SYSTEM', 'ORDSYS',
          'OLAPSYS', 'PERFSTAT')
  /* SELECIONA TABELAS QUE TENHAM CHAVE PRIMÁRIA OU ÚNICA */
  and a.Constraint_Type in ('P','U')
  /* E QUE POSSUEM CHAVE ESTRANGEIRA */
  and a.table_name in ( Select table_name from all_constraints where
    Constraint_Type = 'R')
  /* E ONDE A CHAVE PRIMÁRIA SEJA COMPOSTA */
  and 1 < (Select count(*) from
      (select column_name from all_cons_columns
      where constraint_name in
       (select constraint_name from all_constraints
       where Constraint_Type in ('P','U')
        and table_name = a.table_name)
      )
    )
 /* E ONDE AS COLUNAS DA CHAVE PRIMÁRIA SEJAM AS CHAVES ESTRANGEIRAS */
  and ( select column_name from all_cons_columns
    where constraint_name in
     (select constraint_name from all_constraints
      where Constraint_Type in ('P','U')
      and table_name = a.table_name)
    minus
    select column_name from all_cons_columns
    where constraint_name in
     (select Constraint_Name from all_constraints
      where Constraint_Type = 'R'
```

```
    and table_name = a.table_name)
  ) IS NULL
/* E ONDE AS COLUNAS DA CHAVE PRIMÁRIA SEJAM NUMÉRICAS */
and 0 = ( SELECT SUM(TOT_CHAVES_PRIMARIAS) FROM
    (
     Select (Select count(*) from all_tab_columns
        where column_name in
         (select column_name from all_cons_columns
         where constraint_name in
          (select constraint_name from all_constraints
           where Constraint_Type in ('P','U')
           and table_name = a.table_name)
         )
         and table_name = a.table_name
       ) tot_chaves_primarias
      from DUAL
     UNION ALL
     select (Select count(*) from all_tab_columns
        where column_name in
         (select column_name from all_cons_columns
         where constraint_name in
          (select constraint_name from all_constraints
           where Constraint_Type in ('P','U')
           and table_name = a.table_name)
         )
         and table_name = a.table_name
         and (data_type in ('INTEGER', 'SHORTINTEGER', 'LONGINTEGER')
           or
           (data_type in ('NUMBER','DECIMAL','SHORTDECIMAL')
           and NVL(data_scale,0) = 0))
       ) * (-1) AS tot_chaves_primarias
      From dual
    )
  )
/* E ONDE AS DEMAIS COLUNAS SEJAM NUMÉRICAS */
and
(0 = (SELECT SUM(TOT_COLUNAS) FROM
  (
  select count(*) AS tot_colunas from all_tab_cols
  where table_name = a.table_name
```

```
    UNION ALL
    select count(*) * (-1) AS TOTCOLUNA from all_tab_cols
    where table_name = a.table_name
     and data_type in ('INTEGER','SHORTINTEGER','LONGINTEGER','NUMBER', 'DECIMAL',
          'SHORTDECIMAL')
    )
   )
 )
)
```

10. Apêndice B: Exemplo de Documento XML contendo declaração da estrutura do modelo dimensional

Exemplo dodocumento XML que representa o modelo dimensional apresentado na figura 58 deste trabalho.

```
<?xml version="1.0" encoding="UTF-8"?>
<bancosinstitucionais>
 <instituicao>
  <nome>FGV</nome>
  <solicitação>1</solicitação>
  <databases>
  <database id="1">
   <modelos>
    <fato id="1">
    <nome>FATO_MORTALIDADE</nome>
    </fato>
    <colunas>
    <coluna id="1">
     <nome>ID_DIM_CARAC_POPULACAO</nome>
     <tipo_dado>xsd:integer</tipo_dado>
     <nulos>falso</nulos>
     <ehpk>verdade</ehpk>
     <pkcomposta>verdade</pkcomposta>
     <pkordem>1</pkordem>
     <ehfk>verdade</ehfk>
     <fktable>DIM_CARAC_POPULACAO</fktable>
     <fkcoluna>ID</fkcoluna>
    </coluna>
    <coluna id="2">
     <nome>ID_DIM_CAUSA_MORTE</nome>
     <tipo_dado>xsd:integer</tipo_dado>
     <nulos>falso</nulos>
     <ehpk>verdade</ehpk>
     <pkcomposta>verdade</pkcomposta>
     <pkordem>2</pkordem>
     <ehfk>verdade</ehfk>
```

```
 <fktable>DIM_CAUSA_MORTE</fktable>
 <fkcoluna>ID</fkcoluna>
</coluna>
<coluna id="3">
 <nome>ID_DIM_TEMPO</nome>
 <tipo_dado>xsd:integer</tipo_dado>
 <nulos>falso</nulos>
 <ehpk>verdade</ehpk>
 <pkcomposta>verdade</pkcomposta>
 <pkordem>3</pkordem>
 <ehfk>verdade</ehfk>
 <fktable>DIM_TEMPO</fktable>
 <fkcoluna>ID</fkcoluna>
</coluna>
<coluna id="4">
 <nome>ID_DIM_LOCALIDADE</nome>
 <tipo_dado>xsd:integer</tipo_dado>
 <nulos>falso</nulos>
 <ehpk>verdade</ehpk>
 <pkcomposta>verdade</pkcomposta>
 <pkordem>4</pkordem>
 <ehfk>verdade</ehfk>
 <fktable>DIM_LOCALIDADE</fktable>
 <fkcoluna>ID</fkcoluna>
</coluna>
<coluna id="5">
 <nome>QTD_OBITOS</nome>
 <tipo_dado>xsd:integer</tipo_dado>
 <nulos>verdade</nulos>
 <ehpk>falso</ehpk>
 <ehfk>falso</ehfk>
</coluna>
</colunas>
<dimensoes>
<dimensao id="1">
 <nome>DIM_CARAC_POPULACAO</nome>
 <colunas>
  <coluna id="1">
   <nome>ID</nome>
   <tipo_dado>xsd:integer</tipo_dado>
```

```
<nulos>falso</nulos>
<ehpk>verdade</ehpk>
<pkcomposta>falso</pkcomposta>
<pkordem>1</pkordem>
<ehfk>Falso</ehfk>
</coluna>
<coluna id="2">
<nome>FLAG_URBANA</nome>
<tipo_dado>xsd:boolean</tipo_dado>
<nulos>falso</nulos>
<ehpk>falso</ehpk>
<ehfk>Falso</ehfk>
</coluna>
<coluna id="3">
<nome>SEXO</nome>
<tipo_dado>xsd:string</tipo_dado>
<tamanho>1</tamanho>
<nulos>falso</nulos>
<ehpk>falso</ehpk>
<ehfk>Falso</ehfk>
</coluna>
<coluna id="4">
<nome>FAIXA_ETARIA</nome>
<tipo_dado>xsd:string</tipo_dado>
<tamanho>20</tamanho>
<nulos>falso</nulos>
<ehpk>falso</ehpk>
<ehfk>Falso</ehfk>
</coluna>
</colunas>
</dimensao>
<dimensao id="2">
<nome>DIM_CAUSA_MORTE</nome>
<colunas>
<coluna id="1">
<nome>ID</nome>
<tipo_dado>xsd:integer</tipo_dado>
<nulos>falso</nulos>
<ehpk>verdade</ehpk>
<pkcomposta>falso</pkcomposta>
```

```
 <pkordem>1</pkordem>
 <ehfk>Falso</ehfk>
</coluna>
<coluna id="2">
 <nome>CATEGORIA</nome>
 <tipo_dado>xsd:string</tipo_dado>
 <tamanho>20</tamanho>
 <nulos>falso</nulos>
 <ehpk>falso</ehpk>
 <ehfk>Falso</ehfk>
</coluna>
<coluna id="3">
 <nome>SUB_CATEGORIA</nome>
 <tipo_dado>xsd:string</tipo_dado>
 <tamanho>20</tamanho>
 <nulos>falso</nulos>
 <ehpk>falso</ehpk>
 <ehfk>Falso</ehfk>
</coluna>
</colunas>
</dimensao>
<dimensao id="3">
 <nome>DIM_TEMPO</nome>
 <colunas>
 <coluna id="1">
  <nome>ID</nome>
  <tipo_dado>xsd:integer</tipo_dado>
  <nulos>falso</nulos>
  <ehpk>verdade</ehpk>
  <pkcomposta>falso</pkcomposta>
  <pkordem>1</pkordem>
  <ehfk>Falso</ehfk>
 </coluna>
 <coluna id="2">
  <nome>MES_NUM</nome>
  <tipo_dado>xsd:integer</tipo_dado>
  <range>1;2;3;4;5;6;7;8;9;10;11;12;</range>
  <nulos>falso</nulos>
  <ehpk>verdade</ehpk>
  <ehfk>Falso</ehfk>
```

```
</coluna>
<coluna id="3">
 <nome>MES_NOME</nome>
 <tipo_dado>xsd:string</tipo_dado>
 <range>"JAN";"FEV";"MAR";"ABR";"MAI";"JUN";"JUL";"AGO";"SET";
   "OUT";"NOV";"DEZ";</range>
 <tamanho>3</tamanho>
 <nulos>falso</nulos>
 <ehpk>falso</ehpk>
 <ehfk>Falso</ehfk>
</coluna>
<coluna id="4">
 <nome>BIMESTRE</nome>
 <tipo_dado>xsd:integer</tipo_dado>
 <range>1;2;3;4;</range>
 <nulos>falso</nulos>
 <ehpk>verdade</ehpk>
 <ehfk>Falso</ehfk>
</coluna>
<coluna id="5">
 <nome>TRIMESTRE</nome>
 <tipo_dado>xsd:integer</tipo_dado>
 <range>1;2;3;4;</range>
 <nulos>falso</nulos>
 <ehpk>verdade</ehpk>
 <ehfk>Falso</ehfk>
</coluna>
<coluna id="6">
 <nome>SEMESTRE</nome>
 <tipo_dado>xsd:integer</tipo_dado>
 <range>1;2;</range>
 <nulos>falso</nulos>
 <ehpk>verdade</ehpk>
 <ehfk>Falso</ehfk>
</coluna>
<coluna id="7">
 <nome>ANO</nome>
 <tipo_dado>xsd:integer</tipo_dado>
 <nulos>falso</nulos>
 <ehpk>verdade</ehpk>
```

```
  <ehfk>Falso</ehfk>
 </coluna>
</colunas>
</dimensao>
<dimensao id="4">
 <nome>DIM_LOCALIDADE</nome>
 <colunas>
  <coluna id="1">
   <nome>ID</nome>
   <tipo_dado>xsd:integer</tipo_dado>
   <nulos>falso</nulos>
   <ehpk>verdade</ehpk>
   <pkcomposta>falso</pkcomposta>
   <pkordem>1</pkordem>
   <ehfk>Falso</ehfk>
  </coluna>
  <coluna id="2">
   <nome>MUNICIPIO</nome>
   <tipo_dado>xsd:string</tipo_dado>
   <tamanho>200</tamanho>
   <nulos>falso</nulos>
   <ehpk>falso</ehpk>
   <ehfk>Falso</ehfk>
  </coluna>
  <coluna id="3">
   <nome>UF</nome>
   <tipo_dado>xsd:string</tipo_dado>
   <range>"AC";"AL";"AM";"AP";"BA";"CE";"DF";"ES";"GO";"MA";"MG";
     "MS";"MT";"PA";"PB";"PE";"PI";"PR";"RJ";"RN";"RO";"RR";"RS";
     "SC";"SE";"SP";"TO";</range>
   <tamanho>2</tamanho>
   <nulos>falso</nulos>
   <ehpk>falso</ehpk>
   <ehfk>Falso</ehfk>
  </coluna>
  <coluna id="4">
   <nome>REGIAO</nome>
   <tipo_dado>xsd:string</tipo_dado>
   <range>"NO";"NE";"CO";"SE";"SU";</range>
   <tamanho>2</tamanho>
```

```
  <nulos>falso</nulos>
  <ehpk>falso</ehpk>
  <ehfk>Falso</ehfk>
 </coluna>
</colunas>
</dimensao>
</dimensoes>
<fato id="2">
<nome>FATO_POPULACAO_ABSOLUTA</nome>
</fato>
<colunas>
<coluna id="1">
 <nome>ID_DIM_TEMPO</nome>
 <tipo_dado>xsd:integer</tipo_dado>
 <nulos>falso</nulos>
 <ehpk>verdade</ehpk>
 <pkcomposta>verdade</pkcomposta>
 <pkordem>3</pkordem>
 <ehfk>verdade</ehfk>
 <fktable>DIM_TEMPO</fktable>
 <fkcoluna>ID</fkcoluna>
</coluna>
<coluna id="2">
 <nome>ID_DIM_CARAC_POPULACAO</nome>
 <tipo_dado>xsd:integer</tipo_dado>
 <nulos>falso</nulos>
 <ehpk>verdade</ehpk>
 <pkcomposta>verdade</pkcomposta>
 <pkordem>1</pkordem>
 <ehfk>verdade</ehfk>
 <fktable>DIM_CARAC_POPULACAO</fktable>
 <fkcoluna>ID</fkcoluna>
</coluna>
<coluna id="3">
 <nome>ID_DIM_LOCALIDADE</nome>
 <tipo_dado>xsd:integer</tipo_dado>
 <nulos>falso</nulos>
 <ehpk>verdade</ehpk>
 <pkcomposta>verdade</pkcomposta>
 <pkordem>4</pkordem>
```

```
 <ehfk>verdade</ehfk>
 <fktable>DIM_LOCALIDADE</fktable>
 <fkcoluna>ID</fkcoluna>
</coluna>
<coluna id="4">
 <nome>QTD_PESSOAS</nome>
 <tipo_dado>xsd:integer</tipo_dado>
 <nulos>verdade</nulos>
 <ehpk>falso</ehpk>
 <ehfk>falso</ehfk>
</coluna>
<coluna id="5">
 <nome>QTD_MATRICULAS_ESCOLAR</nome>
 <tipo_dado>xsd:integer</tipo_dado>
 <nulos>verdade</nulos>
 <ehpk>falso</ehpk>
 <ehfk>falso</ehfk>
</coluna>
</colunas>
<dimensoes>
<dimensao id="5">
 <refer>dimensao id="1"</refer>
</dimensao>
<dimensao id="6">
 <refer>dimensao id="3"</refer>
</dimensao>
<dimensao id="7">
 <refer>dimensao id="4"</refer>
</dimensao>
</dimensoes>
<fato id="3">
<nome>FATO_PRODUCAO_ATIVIDADE</nome>
</fato>
<colunas>
<coluna id="1">
 <nome>ID_DIM_LOCALIDADE</nome>
 <tipo_dado>xsd:integer</tipo_dado>
 <nulos>falso</nulos>
 <ehpk>verdade</ehpk>
 <pkcomposta>verdade</pkcomposta>
```

```
 <pkordem>4</pkordem>
 <ehfk>verdade</ehfk>
 <fktable>DIM_LOCALIDADE</fktable>
 <fkcoluna>ID</fkcoluna>
</coluna>
<coluna id="2">
 <nome>ID_DIM_ATIVIDADE_ECONOMICA</nome>
 <tipo_dado>xsd:integer</tipo_dado>
 <nulos>falso</nulos>
 <ehpk>verdade</ehpk>
 <pkcomposta>verdade</pkcomposta>
 <pkordem>1</pkordem>
 <ehfk>verdade</ehfk>
 <fktable>DIM_ATIVIDADE_ECONOMICA</fktable>
 <fkcoluna>ID</fkcoluna>
</coluna>
<coluna id="3">
 <nome>ID_DIM_TEMPO</nome>
 <tipo_dado>xsd:integer</tipo_dado>
 <nulos>falso</nulos>
 <ehpk>verdade</ehpk>
 <pkcomposta>verdade</pkcomposta>
 <pkordem>3</pkordem>
 <ehfk>verdade</ehfk>
 <fktable>DIM_TEMPO</fktable>
 <fkcoluna>ID</fkcoluna>
</coluna>
<coluna id="4">
 <nome>VAL_PRODUCAO</nome>
 <tipo_dado>xsd:decimal</tipo_dado>
 <tamanho>10</tamanho>
 <precisao>2</precisao>
 <nulos>verdade</nulos>
 <ehpk>falso</ehpk>
 <ehfk>falso</ehfk>
</coluna>
</colunas>
<dimensoes>
<dimensao id="8">
 <refer>dimensao id="3"</refer>
```

```
</dimensao>
<dimensao id="9">
 <nome>DIM_ATIVIDADE_ECONOMICA</nome>
 <colunas>
  <coluna id="1">
   <nome>ID</nome>
   <tipo_dado>xsd:integer</tipo_dado>
   <nulos>falso</nulos>
   <ehpk>verdade</ehpk>
   <pkcomposta>falso</pkcomposta>
   <pkordem>1</pkordem>
   <ehfk>Falso</ehfk>
  </coluna>
  <coluna id="2">
   <nome>NOME_ATIVIDADE</nome>
   <tipo_dado>xsd:string</tipo_dado>
   <tamanho>200</tamanho>
   <nulos>falso</nulos>
   <ehpk>falso</ehpk>
   <ehfk>Falso</ehfk>
  </coluna>
 </colunas>
</dimensao>
<dimensao id="10">
 <refer>dimensao id="4"</refer>
</dimensao>
</dimensoes>
<fato id="4">
<nome>FATO_REGIOSTROS_CRIMES</nome>
</fato>
<colunas>
<coluna id="1">
 <nome>ID_DIM_TIPO_CRIME</nome>
 <tipo_dado>xsd:integer</tipo_dado>
 <nulos>falso</nulos>
 <ehpk>verdade</ehpk>
 <pkcomposta>verdade</pkcomposta>
 <pkordem>2</pkordem>
 <ehfk>verdade</ehfk>
 <fktable>DIM_TIPO_CRIME</fktable>
```

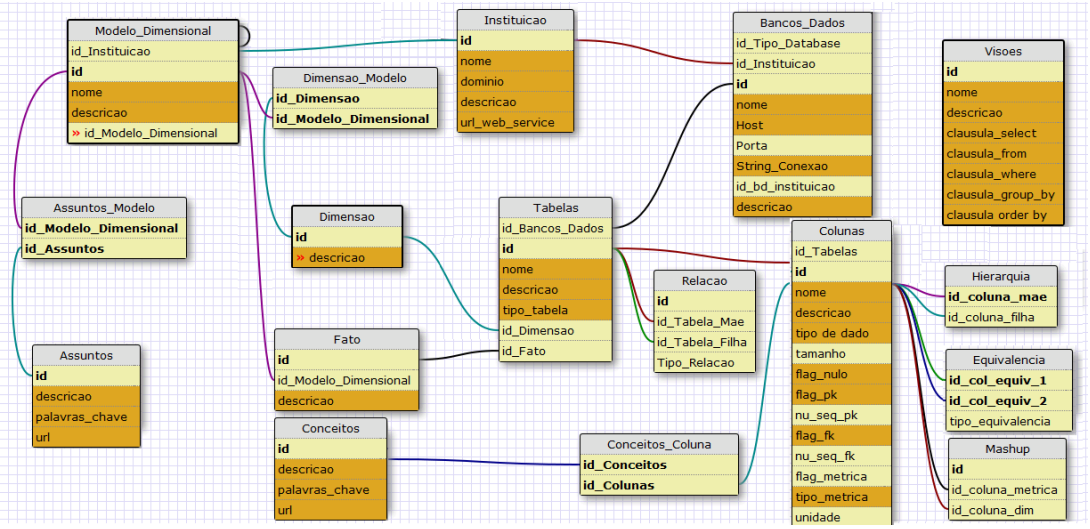
```
 <fkcoluna>ID</fkcoluna>
</coluna>
<coluna id="2">
 <nome>ID_DIM_TEMPO</nome>
 <tipo_dado>xsd:integer</tipo_dado>
 <nulos>falso</nulos>
 <ehpk>verdade</ehpk>
 <pkcomposta>verdade</pkcomposta>
 <pkordem>3</pkordem>
 <ehfk>verdade</ehfk>
 <fktable>DIM_TEMPO</fktable>
 <fkcoluna>ID</fkcoluna>
</coluna>
<coluna id="3">
 <nome>ID_DIM_CARACT_CRIMINOSO</nome>
 <tipo_dado>xsd:integer</tipo_dado>
 <nulos>falso</nulos>
 <ehpk>verdade</ehpk>
 <pkcomposta>verdade</pkcomposta>
 <pkordem>1</pkordem>
 <ehfk>verdade</ehfk>
 <fktable>DIM_CARAC_POPULACAO</fktable>
 <fkcoluna>ID</fkcoluna>
</coluna>
<coluna id="4">
 <nome>ID_DIM_LOCALIDADE</nome>
 <tipo_dado>xsd:integer</tipo_dado>
 <nulos>falso</nulos>
 <ehpk>verdade</ehpk>
 <pkcomposta>verdade</pkcomposta>
 <pkordem>4</pkordem>
 <ehfk>verdade</ehfk>
 <fktable>DIM_LOCALIDADE</fktable>
 <fkcoluna>ID</fkcoluna>
</coluna>
<coluna id="5">
 <nome>QTD_OCORRENCIAS</nome>
 <tipo_dado>xsd:integer</tipo_dado>
 <nulos>verdade</nulos>
 <ehpk>falso</ehpk>
```

```
 <ehfk>falso</ehfk>
</coluna>
</colunas>
<dimensoes>
<dimensao id="11">
 <nome>DIM_TIPO_CRIME</nome>
 <colunas>
  <coluna id="1">
   <nome>ID</nome>
   <tipo_dado>xsd:integer</tipo_dado>
   <nulos>falso</nulos>
   <ehpk>verdade</ehpk>
   <pkcomposta>falso</pkcomposta>
   <pkordem>1</pkordem>
   <ehfk>Falso</ehfk>
  </coluna>
  <coluna id="2">
   <nome>CATEGORIA</nome>
   <tipo_dado>xsd:string</tipo_dado>
   <tamanho>30</tamanho>
   <nulos>falso</nulos>
   <ehpk>falso</ehpk>
   <ehfk>Falso</ehfk>
  </coluna>
  <coluna id="3">
   <nome>SUB_CATEGORIA</nome>
   <tipo_dado>xsd:string</tipo_dado>
   <tamanho>50</tamanho>
   <nulos>falso</nulos>
   <ehpk>falso</ehpk>
   <ehfk>Falso</ehfk>
  </coluna>
 </colunas>
</dimensao>
<dimensao id="12">
 <refer>dimensao id="3"</refer>
</dimensao>
<dimensao id="13">
 <refer>dimensao id="1"</refer>
</dimensao>
```

```
    <dimensao id="14">
     <refer>dimensao id="4"</refer>
    </dimensao>
    </dimensoes>
   </modelos>
  </database>
  </databases>
 </instituicao>
</bancosinstitucionais>
```

11. Apêndice C: Catálogo interno da DCD Tool.

Com base nas informações contidas no arquivo XML apresentado no apêndice B, as seguintes tabelas, constituintes do catálogo interno da DCD Tool são carregadas.



Abaixo segue uma breve descrição sobre cada uma das entidades acima.

Assuntos Onde serão catalogados os assuntos tratados pelos modelos dimensionais. Um assunto deve ter uma descrição e ao menos uma palavra-chave associada. Todo modelo dimensional deve ter ao menos um assunto vinculado. Um assunto pode ter uma URL associada.
Assuntos_Modelo Estabelece o vínculo entre um assunto e um modelo.
Modelo_Dimensional Onde serão catalogados os modelos dimensionais e os meta modelos dimensionais. Ambos modelos são criados automaticamente pelo processo de carga e recebem um nome padrão “Modelo” + id. O meta modelo é um modelo formado exclusivamente de modelos dimensionais, ou seja, ele não possui F-TABs nem dimensões diretamente vinculados a ele, apenas modelos dimensionais. Já os modelos dimensionais, podem possuir diversas dimensões mas uma única F-TAB vinculada. Para o meta modelo, o usuário deve preencher apenas a sua descrição e atualizar o nome padrão que lhe foi dado. Para os modelos dimensionais, ele deve atualizar o

nome, preencher a descrição e vincular ao menos um assunto ao modelo antes de tentar triplificá-lo. O processo de triplificação só acontece quando um modelo possui um nome diferente do padrão e as demais informações preenchidas. Os vínculos entre meta modelos e modelos dimensionais é estabelecido automaticamente durante a carga e não pode ser mudado, bem como o vínculo entre F-TAB e dimensões e o modelo dimensional.
Dimensão As dimensões são criadas durante a carga e podem reunir mais de uma tabela no caso de esquemas em floco de neve. Toda D-TAB precisa de uma descrição que deve ser preenchida antes da triplificação. Uma D-TAB pode estar vinculada a mais de um modelo. O vínculo entre D-TAB e F-TAB pode ser obtido tanto através do modelo dimensional como através das relações registradas entre as tabelas que as implementam. Estes vínculos são estabelecidos automaticamente no processo de carga e não podem ser alterados.
Dimensão_Modelo Conjunto de dimensões de um modelo dimensional. Atualização automática realizada durante o processo de carga.
Fato São criadas durante a carga, pertencem a um único modelo dimensional e também são únicas nele. Toda F-TAB precisa de uma descrição que deve ser preenchida antes da triplificação.
Tabelas São as entidades físicas que existem em um banco de dados e materializam uma F-TAB ou uma D-TAB. Apenas a descrição deve ser fornecida pelo usuário antes da triplificação. Os demais campos, nome, tipo, a qual D-TAB ou F-TAB está vinculada e o banco de dados a que pertence, são preenchidas em tempo de carga e não podem ser alteradas.
Relação Identifica uma relação entre tabelas que pode ser um vínculo entre uma D-TAB e uma F-TAB ou uma relação de hierarquia, no caso de dimensões primárias e secundárias (outriggers). Preenchida automaticamente durante a carga.
Bancos_Dados Na verdade existem duas versões desta tabela. A versão completa existe apenas no servidor da instituição credenciada e contém todas as informações preenchidas. A versão "leve" contendo apenas o id do banco de dados, seu nome e descrição e o id

dele na versão desta tabela que foi criada no MySQL local da instituição, fica armazenada no MySQL do servidor da DCD TOOL. Carga automática realizada através da sincronização.
Conceitos Onde serão catalogados os conceitos associados às informações contidas em uma dada coluna de alguma tabela. Um conceito deve ter uma descrição e ao menos uma palavra-chave associada, além disso um conceito pode ter uma URL associada que será usada para criar um vínculo de sameAs entre a coluna e uma fonte externa. O vínculo entre colunas e conceitos é opcional. Um conceito pode estar associado a mais de uma coluna e cada coluna pode ter mais de um conceito a ela vinculado.
Conceitos_Coluna Estabelece o vínculo entre um conceito e uma coluna.
Colunas Onde estão registrados os metadados obtidos no catálogo do banco de dados da instituição. Cada coluna pertence a uma única tabela, um nome, tipo de dado, tamanho (quando aplicável), indicador se aceita nulos ou não, indicador se faz parte da pk e seu posicionamento na composição da pk, caso esta seja composta, um indicador de se é ou faz parte de uma chave estrangeira e seu posicionamento na composição da chave estrangeira caso positivo e a chave seja composta e um indicador que informa se a coluna é uma medida. Todos esses dados são obtidos a partir do catálogo. O usuário deve complementar esses dados preenchendo os campos descrição, o tipo de medida (aditiva, semi-aditiva ou não-aditiva) e a unidade de medida da medida, no caso da coluna ser uma medida.
Equivalência Existem dois tipos de relações de equivalência: aquele que indica que a chave estrangeira em determinada tabela corresponde à chave primária de outra, preenchido automaticamente na carga das tabelas, e, outro, que é formalizado pelo usuário indicando que duas colunas quaisquer, pertencentes a diferentes tabelas e, possivelmente, diferentes bancos de dados, são equivalentes entre si, isto é, possuem o mesmo significado e domínio de valores.
Hierarquia Estabelece relações hierárquicas entre colunas, independente das tabelas às quais estas colunas pertençam.
Mashup Trata-se de uma tabela de relacionamento onde registramos relacionamentos independentes entre cada medida aditiva que exista cadastrada e cada uma das

colunas de cada uma das dimensões que estão vinculadas à F-TAB à qual a medida pertence. Será através desta tabela de relacionamento, acrescida das tabelas de relacionamento Hierarquia e Equivalência que iremos viabilizar futuramente uma solução de mashup de dados.

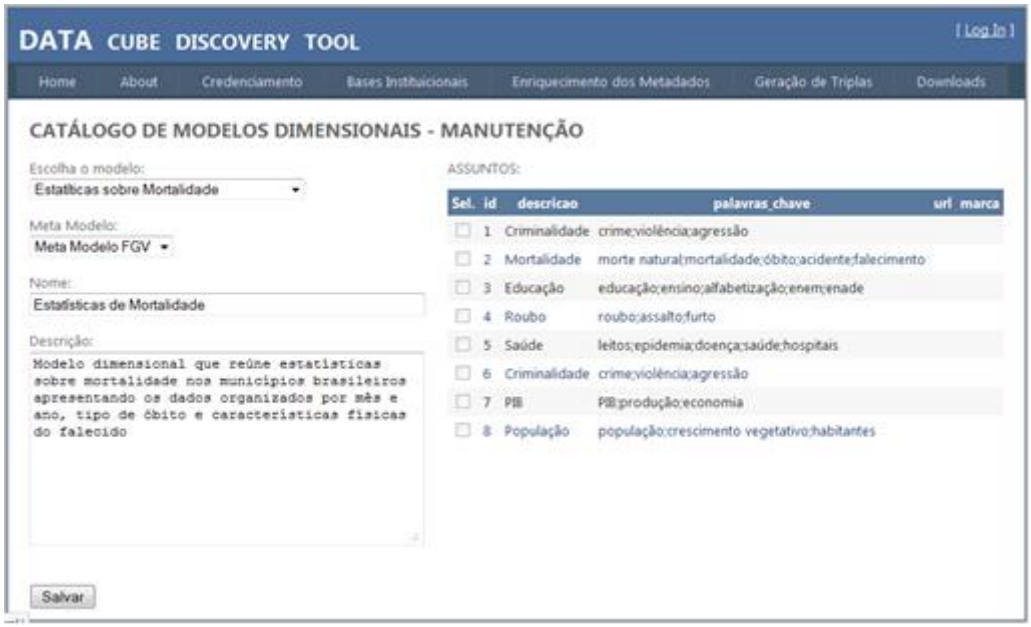
Visões

Registra as visões que serão oferecidas para as aplicações usuárias a partir do catálogo do *framework*. Para cada modelo dimensional, no momento da sua carga, construímos automaticamente uma visão contendo todos os elementos daquele modelo. Esta é a visão padrão daquele modelo dimensional, que recebe um nome "padrão" também e que deve ser alterado pelo usuário e precisa de uma breve descrição também obrigatória, sem a qual o processo de triplificação não será processado. O usuário pode cadastrar novas visões além das "visões padrões", e, futuramente, visões de mashup poderão ser criadas. As visões são construídas a partir de comandos SQL construídos pelo usuário para materializar o resultado por ele almejado quando aquela visão for selecionada pela aplicação usuária do *framework*.

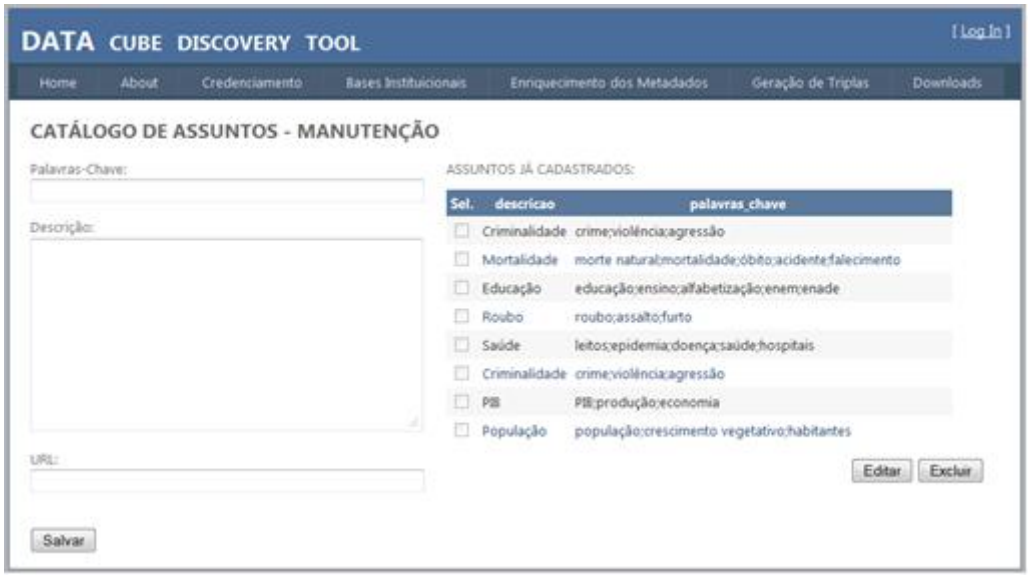
12. Apêndice D: Funcionalidades da camada *Enrichment Metadatas* da DCD Tool.

12.1.1.1 Catálogo dos Modelos Dimensionais

O catálogo dos modelos dimensionais permite editar o nome e a descrição destes. O vínculo com o meta modelo é definido no momento da carga e não pode ser alterado. Pode-se vincular um ou mais assuntos previamente cadastrados pela funcionalidade "Catálogo de Assuntos", ao modelo dimensional.

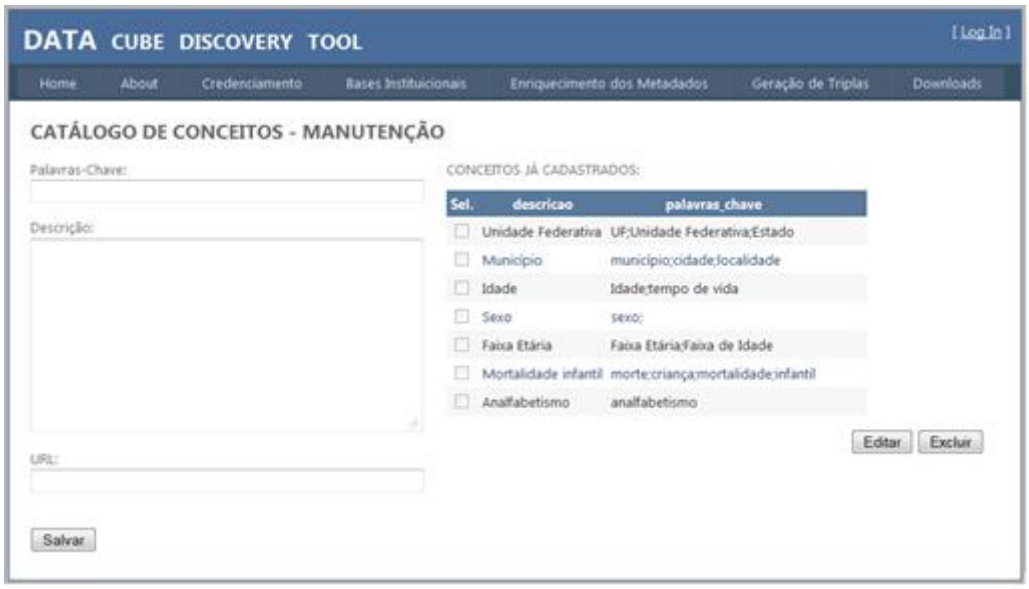


12.1.1.2 Catálogo dos Assuntos



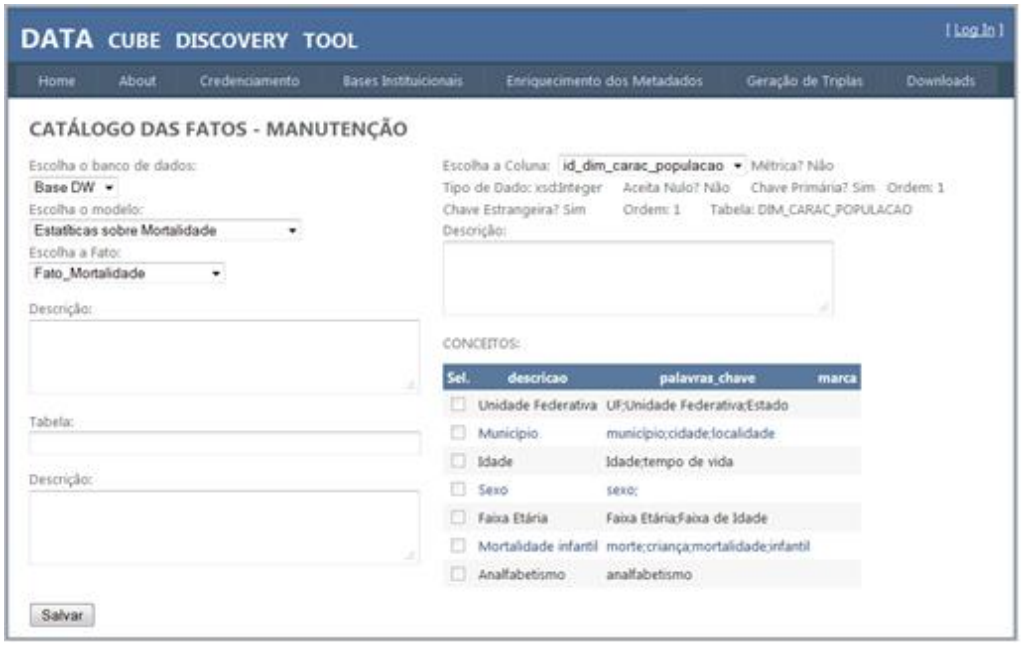
O catálogo dos assuntos permite incluir, excluir e editar assuntos para serem vinculados aos modelos dimensionais. A exclusão só é permitida quando o assunto não se encontra vinculado a nenhum modelo dimensional.

12.1.1.3 Catálogo dos Conceitos



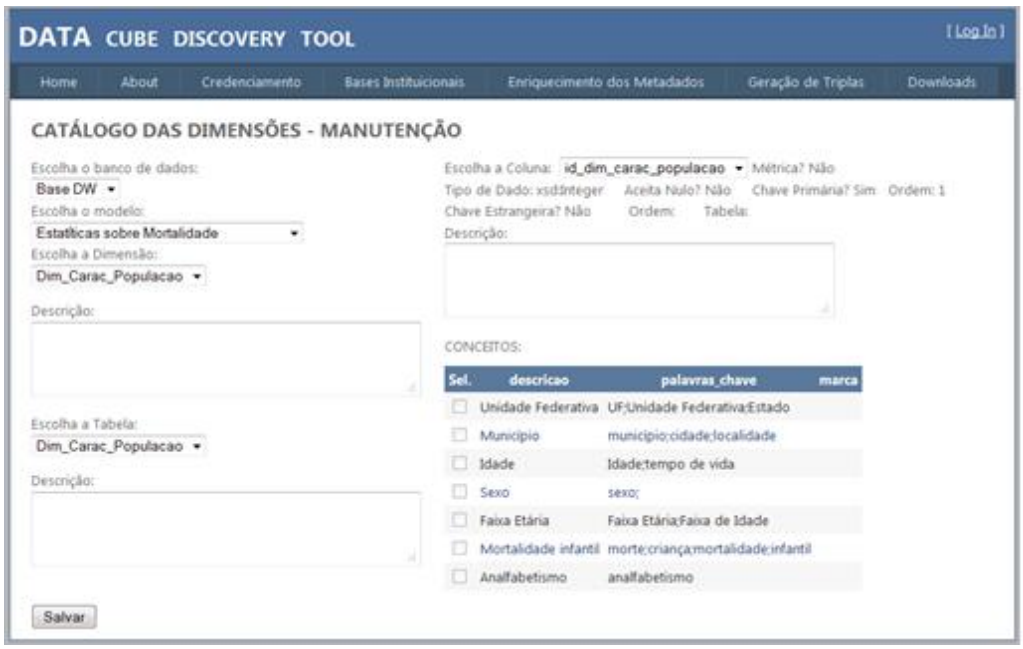
O catálogo dos conceitos permite incluir, excluir e editar conceitos para serem vinculados às colunas das tabelas dos modelos dimensionais. A exclusão só é permitida quando o conceito não se encontra vinculado a nenhuma coluna.

12.1.1.4 Catálogo das Fatos



O catálogo das fatos permite editar a descrição tanto da F-TAB como da tabela que a materializa no banco de dados da instituição credenciada. Também podemos editar a descrição de cada coluna desta tabela, definir o tipo de medida e a unidade, no caso da coluna ser uma medida. Finalmente, podemos associar conceitos a cada coluna.

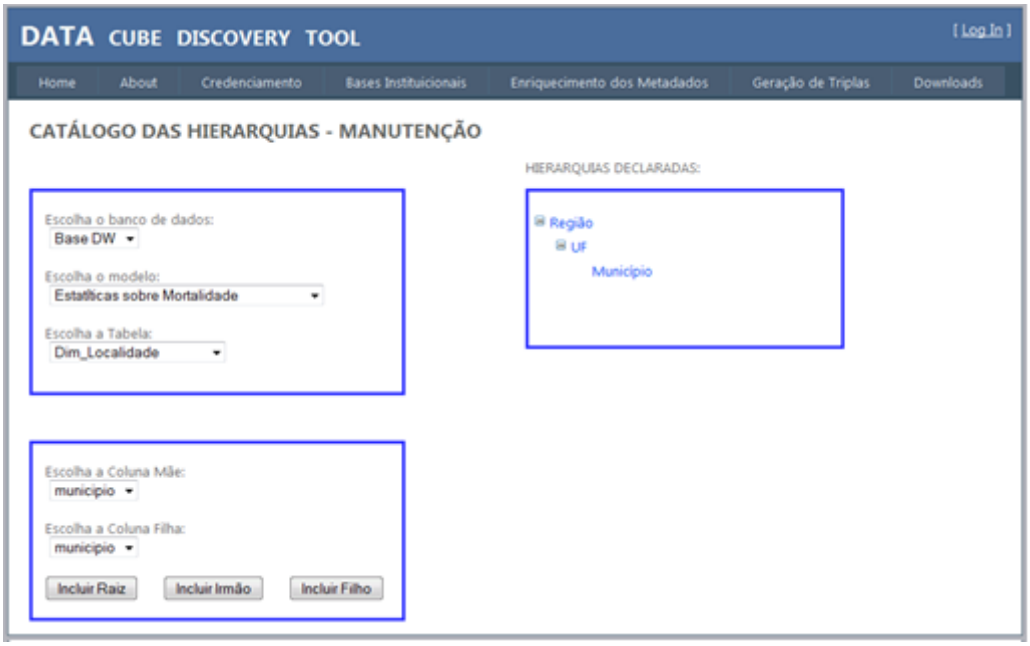
12.1.1.5 Catálogo das Dimensões



O catálogo das dimensões permite editar a descrição tanto da D-TAB como da(s) tabela(s) que a materializa no banco de dados da instituição

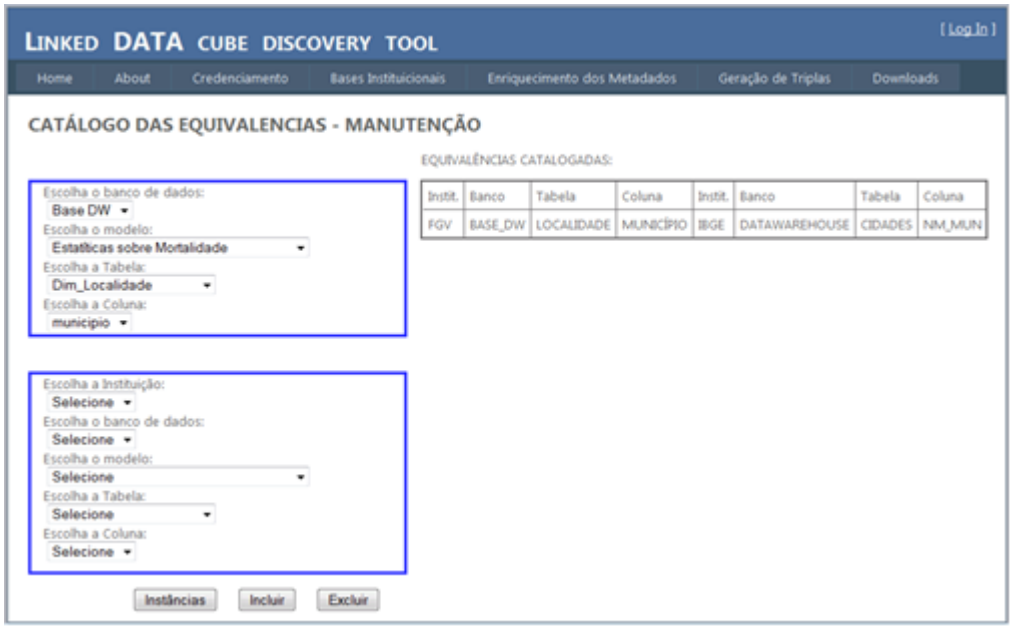
credenciada. Também podemos editar a descrição de cada coluna desta tabela, definir o tipo de medida e a unidade, no caso da coluna ser uma medida. Finalmente, podemos associar conceitos a cada coluna. A diferença para o catálogo de Fatos é que neste caso, carregamos uma dropdownlist contendo as tabelas vinculadas à D-TAB (no caso de floco de neve).

12.1.1.6 Catálogo das Hierarquias

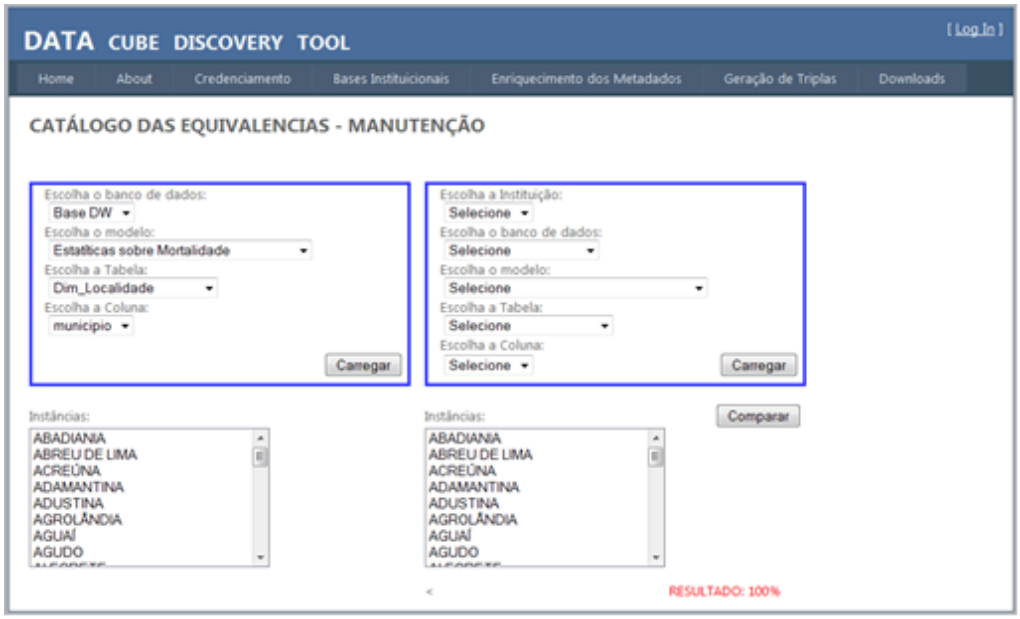


O catálogo das hieraquias permite estabelecer as relações através das quais as hierarquias entre colunas de uma mesma tabela são declaradas. Em estruturas em floco de neve as hierarquias são construídas automaticamente em tempo de carga.

12.1.1.7 Catálogo das Equivalências



O catálogo das equivalências permite registrar equivalências entre colunas de dados de tabelas diferentes em um mesmo banco de dados, diferentes bancos de dados e até mesmo de diferentes bancos de dados de diferentes instituições. Para avaliar o grau de equivalência entre as colunas, o usuário deve clicar no botão “instâncias” e será direcionado para a tela abaixo apresentada.



Nesta tela, o usuário pode submeter consultas às instâncias de determinada coluna em certa tabela em algum dos bancos de dados cadastrados no sistema e depois comandar uma comparação entre o conteúdo destas instâncias para se certificar do nível de equivalência entre elas.

As equivalências onde os domínios de valores coincidiram em 100% das suas instâncias darão origem a triplas utilizando o vocábulo *skos:exactMatch*. Aquelas que tiveram menos 100% serão declaradas com *skos:closeMatch*.

13. Apêndice E: Exemplo de Triplificação de um Modelo Dimensional.

Abaixo descrevemos o conjunto de triplas que representa cada F-TAB e D-TAB presentes no modelo dimensional apresentado na figura 58 deste trabalho.

```
@prefix ex-resource: <http://purl.org/GovDataCube/resources/>.
@prefix ex-property: <http://purl.org/GovDataCube/properties/>.
@prefix ex-class: <http://purl.org/GovDataCube/classes/>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix qb: <http://purl.org/linked-data/cube#>.
@prefix sdmx-attribute: <http://purl.org/linked-data/sdmx/2009/attribute#>.
@prefix rr: <http://www.w3.org/ns/r2rml#>.
@prefix d2rq: <http://www.wiwiss.fu-berlin.de/suhl/bizer/D2RQ/0.1#>.
@prefix skos: <http://www.w3.org/2004/02/skos/core#>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
@prefix ex: <http://www.example.com/concepts#>.
@prefix dc: <http://purl.org/dc/elements/1.1/>.
@prefix owl:  <http://www.w3.org/2002/07/owl#>.
#=======================================================
#====== INICIO DECLARACAO MODELO DIMENSIONAL =======
#=======================================================
#====== INICIO DECLARACAO DAS DIMENSOES ==============
# ==================
# Declaração da Dim_Atividade_Economica e seus atributos
# ==================
ex:dim1AtividadeEconomica a rdfs:Class, skos:Concept;
  rdfs:label "Dim_Atividade_Economica"@pt;
  rdfs:comment "Dimensão Atividade Econômica"@pt.
```

```
ex:idDim1AtividadeEconomica1 a rdf:property;
  rdfs:domain ex:dim1AtividadeEconomica;
  rdfs:range xsd:integer;
  rdfs:label "id_dim_atividade_economica"@pt;
  rdfs:comment "Chave primária da D-TAB atividade econômica."@pt.
ex:dimAxisNomeAtividade2 a rdf:property, qb:DimensionProperty;
  rdfs:domain ex:dim1AtividadeEconomica;
  rdfs:range ex-class:dimAxisNomeAtividade2;
  rdfs:label "nome_atividade"@pt;
  rdfs:comment "Define qual é a atividade econômica produtiva: serviços,
indústria, agricultura, etc."@pt.
ex-class:dimAxisNomeAtividade2 a rdfs:Class;
  rdfs:label "nome_atividade"@pt;
  rdfs:comment "Define qual é a atividade econômica produtiva: serviços,
indústria, agricultura, etc."@pt.
# ==================
# Declaração da Dim_Caract_Populacao e seus atributos
# ==================
ex:dim2CaractPopulacao a rdfs:Class, skos:Concept;
  rdfs:label "Dim_Caract_Populacao"@pt;
  rdfs:comment " Dimensão Característica da População."@pt.
ex:idDim2CaractPopulacao3 a rdf:property;
  rdfs:domain ex:dim2CaractPopulacao;
  rdfs:range xsd:integer;
  rdfs:label "id_dim_caract_populacao"@pt;
  rdfs:comment "Chave primária da D-TAB Características da População."@pt.
ex:dimAxisFaixaEtaria4 a rdf:property, qb:DimensionProperty;
  rdfs:domain ex:dim2CaractPopulacao;
  rdfs:range ex-class:dimAxisFaixaEtaria4;
  rdfs:label "faixa_etaria"@pt;
  rdfs:comment "Faixas de idade no padrão do IBGE."@pt.

ex-class:dimAxisFaixaEtaria4 a rdfs:Class;
  rdfs:label "faixa_etaria"@pt;
```

```
  rdfs:comment "Faixas de idade no padrão do IBGE."@pt.
ex:dimAxisFlagUrbana5 a rdf:property, qb:DimensionProperty;
  rdfs:domain ex:dim2CaractPopulacao;
  rdfs:range ex-class:dimAxisFlagUrbana5;
  rdfs:label "flag_urbana"@pt;
  rdfs:comment "Identifica se a pessoa mora em área urbana ou rural."@pt.
ex-class:dimAxisFlagUrbana5 a rdfs:Class;
  rdfs:label "flag_urbana"@pt;
  rdfs:comment "Identifica se a pessoa mora em área urbana ou rural."@pt.
ex:dimAxisSexo6 a rdf:property, qb:DimensionProperty;
  rdfs:domain ex:dim2CaractPopulacao;
  rdfs:range ex-class:dimAxisSexo6;
  rdfs:label "sexo"@pt;
  rdfs:comment "Identifica se a pessoa é homem ou mulher."@pt.
ex-class:dimAxisSexo6 a rdfs:Class;
  rdfs:label "sexo"@pt;
  rdfs:comment "Identifica se a pessoa é homem ou mulher."@pt.
# ==================
# Declaração da Dim_Causa_Morte e seus atributos
# ==================
ex:dim3CausaMorte a rdfs:Class, skos:Concept;
  rdfs:label "Dim_Causa_Morte"@pt;
  rdfs:comment "Dimensão que contém as categorias e subcategorias de
óbitos."@pt.

ex:idDim3CausaMorte7 a rdf:property;
  rdfs:domain ex:dim3CausaMorte;
  rdfs:range xsd:integer;
  rdfs:label "id_dim_causa_morte"@pt;
  rdfs:comment "Chave primária da D-TAB Causa Morte."@pt.
ex:dimAxisSubCategoria8 a rdf:property, qb:DimensionProperty;
  rdfs:domain ex:dim3CausaMorte;
  rdfs:range ex-class:dimAxisSubCategoria8;
  rdfs:label "sub_categoria"@pt;
```

```
  rdfs:comment "Sub categoria de óbito. Ex: categoria morte natural, subcategoria infarto, câncer, etc."@pt;
  skos:broader ex:dimAxisCategoria9;
  skos:broaderTransitive ex:dimAxisCategoria9.
ex-class:dimAxisSubCategoria8 a rdfs:Class;
  rdfs:label "sub_categoria"@pt;
  rdfs:comment "Sub categoria de óbito. Ex: categoria morte natural, subcategoria infarto, câncer, etc."@pt.
ex:dimAxisCategoria9 a rdf:property, qb:DimensionProperty;
  rdfs:domain ex:dim3CausaMorte;
  rdfs:range ex-class:dimAxisCategoria9;
  rdfs:label "categoria"@pt;
  rdfs:comment "Categoria de óbito. Ex: categoria morte natural, assassinato, morte acidental, suicídio, etc."@pt;
  skos:narrower ex:dimAxisSubCategoria8;
  skos:narrowerTransitive ex:dimAxisSubCategoria8.
ex-class:dimAxisCategoria9 a rdfs:Class;
  rdfs:label "categoria"@pt;
  rdfs:comment "Categoria de óbito. Ex: categoria morte natural, assassinato, morte acidental, suicídio, etc."@pt.
# ==================
# Declaração da Dim_Localidade e seus atributos
# ==================
ex:dim4Localidade a rdfs:Class, skos:Concept;
  rdfs:label "Dim_Localidade"@pt;
  rdfs:comment "Dimensão que contém os municípios brasileiros agrupados por uf e região."@pt.
ex:idDim4Localidade10 a rdf:property;
  rdfs:domain ex:dim4Localidade;
  rdfs:range xsd:integer;
  rdfs:label "id_dim_localidade"@pt;
  rdfs:comment "Chave primária da D-TAB Localidade."@pt.
ex:dimAxisMunicipio11 a owl:DatatypeProperty , rdf:property, qb:DimensionProperty;
```

```
  rdfs:domain ex:dim4Localidade;
  rdfs:range ex-class:dimAxisMunicipio11;
  rdfs:label "municipio"@pt;
  rdfs:comment "Municípios brasileiros."@pt;
  skos:broader ex:dimAxisUf12;
  skos:broaderTransitive ex:dimAxisUf12.
ex-class:dimAxisMunicipio11 a rdfs:Class;
  rdfs:label "municipio"@pt;
  rdfs:comment "Municípios brasileiros."@pt;
ex:dimAxisUf12 a rdf:property, qb:DimensionProperty;
  rdfs:domain ex:dim4Localidade;
  rdfs:range ex-class:dimAxisUf12;
  rdfs:label "uf"@pt;
  rdfs:comment "Unidades Federativas brasileiras."@pt;
  skos:narrower ex:dimAxisMunicipio11;
  skos:narrowerTransitive ex:dimAxisMunicipio11;
  skos:broader ex:dimAxisRegiao13;
  skos:broaderTransitive ex:dimAxisRegiao13.
ex-class:dimAxisUf12 a rdfs:Class;
  rdfs:label "uf"@pt;
  rdfs:comment "Unidades Federativas brasileiras."@pt.
ex:dimAxisRegiao13 a rdf:property, qb:DimensionProperty;
  rdfs:domain ex:dim4Localidade;
  rdfs:range ex-class:dimAxisRegiao13;
  rdfs:label "regiao"@pt;
  rdfs:comment "Regiões geo-políticas brasileiras."@pt;
  skos:narrower ex:dimAxisUf12;
  skos:narrowerTransitive ex:dimAxisUf12;
ex-class:dimAxisRegiao13 a rdfs:Class;
  rdfs:label "regiao"@pt;
  rdfs:comment "Regiões geo-políticas brasileiras."@pt.
# ==================
# Declaração da Dim_Tempo e seus atributos
# ==================
```

```
ex:dim5Tempo a rdfs:Class, skos:Concept;
  rdfs:label "Dim_Tempo"@pt;
  rdfs:comment "Dimensão Tempo."@pt.
ex:idDim5Tempo14 a rdf:property;
  rdfs:domain ex:dim5Tempo;
  rdfs:range xsd:integer;
  rdfs:label "id_dim_tempo"@pt;
  rdfs:comment "Chave primária da D-TAB Tempo."@pt.
ex:dimAxisBimestre15 a rdf:property, qb:DimensionProperty;
  rdfs:domain ex:dim5Tempo;
  rdfs:range ex-class:dimAxisBimestre15;
  rdfs:label "bimestre"@pt;
  rdfs:comment "Identifica o bimestre em que a medição foi registrada."@pt.
ex-class:dimAxisBimestre15 a rdfs:Class;
  rdfs:label "bimestre"@pt;
  rdfs:comment "Identifica o bimestre em que a medição foi registrada."@pt.
ex:dimAxisMesNome16 a rdf:property, qb:DimensionProperty;
  rdfs:domain ex:dim5Tempo;
  rdfs:range ex-class:dimAxisMesNome16;
  rdfs:label "mes_nome"@pt;
  rdfs:comment "Nome por extenso do mês em que a medição foi
registrada."@pt.
ex-class:dimAxisMesNome16 a rdfs:Class;
  rdfs:label "mes_nome"@pt;
  rdfs:comment "Nome por extenso do mês em que a medição foi
registrada."@pt.
ex:dimAxisSemestre17 a rdf:property, qb:DimensionProperty;
  rdfs:domain ex:dim5Tempo;
  rdfs:range ex-class:dimAxisSemestre17;
  rdfs:label "semestre"@pt;
  rdfs:comment "Identifica o semestre em que a medição foi registrada."@pt.
ex-class:dimAxisSemestre17 a rdfs:Class;
  rdfs:label "semestre"@pt;
  rdfs:comment "Identifica o semestre em que a medição foi registrada."@pt.
```

```
ex:dimAxisTrimestre18 a rdf:property, qb:DimensionProperty;
  rdfs:domain ex:dim5Tempo;
  rdfs:range ex-class:dimAxisTrimestre18;
  rdfs:label "trimestre"@pt;
  rdfs:comment "Identifica o trimestre em que a medição foi registrada."@pt.
ex-class:dimAxisTrimestre18 a rdfs:Class;
  rdfs:label "trimestre"@pt;
  rdfs:comment "Identifica o trimestre em que a medição foi registrada."@pt.
ex:dimAxisMesNum19 a rdf:property, qb:DimensionProperty;
  rdfs:domain ex:dim5Tempo;
  rdfs:range ex-class:dimAxisMesNum19;
  rdfs:label "mes_num"@pt;
  rdfs:comment "Identifica o mês em que a medição foi registrada (mês numérico)."@pt;
  skos:broader ex:dimAxisAno20;
  skos:broaderTransitive ex:dimAxisAno20.
ex-class:dimAxisMesNum19 a rdfs:Class;
  rdfs:label "mes_num"@pt;
  rdfs:comment "Identifica o mês em que a medição foi registrada (mês numérico)."@pt.
ex:dimAxisAno20 a rdf:property, qb:DimensionProperty;
  rdfs:domain ex:dim5Tempo;
  rdfs:range ex-class:dimAxisAno20;
  rdfs:label "ano"@pt;
  rdfs:comment "Identifica o ano em que a medição foi registrada."@pt;
  skos:narrower ex:dimAxisMesNum19;
  skos:narrowerTransitive ex:dimAxisMesNum19.
ex-class:dimAxisAno20 a rdfs:Class;
  rdfs:label "ano"@pt;
  rdfs:comment "Identifica o ano em que a medição foi registrada."@pt.
# ==================
# Declaração da Dim_Tipo_Crime e seus atributos
# ==================
ex:dim6TipoCrime a rdfs:Class, skos:Concept;
```

```
  rdfs:label "Dim_Tipo_Crime"@pt;
  rdfs:comment "Dimensão que contém as categorias e subcategorias de Crimes."@pt.
ex:idDim6TipoCrime21 a rdf:property;
  rdfs:domain ex:dim6TipoCrime;
  rdfs:range xsd:integer.
  rdfs:label "id_dim_tipo_crime"@pt;
  rdfs:comment "Chave primária da D-TAB Tipo de Crime."@pt.
ex:dimAxisSubCategoria22 a rdf:property, qb:DimensionProperty;
  rdfs:domain ex:dim3CausaMorte;
  rdfs:range ex-class:dimAxisSubCategoria22;
  rdfs:label "sub_categoria"@pt;
  rdfs:comment "Sub categoria de crime. Ex: categoria Assalto, subcategoria roubo, furto, latrocínio, etc."@pt;
  skos:broader ex:dimAxisCategoria23;
  skos:broaderTransitive ex:dimAxisCategoria23.
ex-class:dimAxisSubCategoria22 a rdfs:Class;
  rdfs:label "sub_categoria"@pt;
  rdfs:comment "Sub categoria de crime. Ex: categoria Assalto, subcategoria roubo, furto, latrocínio, etc."@pt.
ex:dimAxisCategoria23 a rdf:property, qb:DimensionProperty;
  rdfs:domain ex:dim6TipoCrime;
  rdfs:range xsd:string { maxLength="100" };
  rdfs:label "categoria"@pt;
  rdfs:comment "Categoria de crime. Ex: Assassinato, Assalto, Violência Sexual, Violência Doméstica, etc."@pt;
  skos:narrower ex:dimAxisSubCategoria22;
  skos:narrowerTransitive ex:dimAxisSubCategoria22.
ex-class:dimAxisCategoria23 a rdfs:Class;
  rdfs:label "sub_categoria"@pt;
  rdfs:comment "Sub categoria de crime. Ex: categoria Assalto, subcategoria roubo, furto, latrocínio, etc."@pt.
# ========== FIM DECLARACAO DAS DIMENSOES ===========
# ========== INICIO DECLARACAO DAS FATOS ==============
```

```
# ==================
# Declaração da Tabela Fato_Mortalidade e seus atributos
# ==================
ex:fato1Mortalidade a rdfs:Class, skos:Concept;
  rdfs:label "Fato_Mortalidade"@pt;
  rdfs:comment "Fato que registra os óbitos por tipo de morte por município, tempo e características do falecido."@pt.
ex:idfato1DimCaractPopulacao24 a rdf:property;
  rdfs:domain ex:fato1Mortalidade;
  rdfs:range xsd:integer;
  rdfs:label "id_dim_caract_populacao"@pt;
  rdfs:comment "Chave estrangeira da Dim_Caract_Populacao."@pt;
  skos:exactMatch ex:idDim2CaractPopulacao3.
ex:idfato1DimCausaMorte25 a rdf:property;
  rdfs:domain ex:fato1Mortalidade;
  rdfs:range xsd:integer;
  rdfs:label "id_dim_causa_morte"@pt;
  rdfs:comment "Chave estrangeira da Dim_Causa_Morte."@pt;
  skos:exactMatch ex:idDim3CausaMorte7.
ex:idfato1DimLocalidade26 a rdf:property;
  rdfs:domain ex:fato1Mortalidade;
  rdfs:range xsd:integer;
  rdfs:label "id_dim_localidade"@pt;
  rdfs:comment "Chave estrangeira da Dim_Localidade."@pt;
  skos:exactMatch ex:idDim4Localidade10.
ex:idfato1DimTempo27 a rdf:property;
  rdfs:domain ex:fato1Mortalidade;
  rdfs:range xsd:integer;
  rdfs:label "id_dim_tempo"@pt;
  rdfs:comment "Chave estrangeira da Dim_Tempo."@pt;
  skos:exactMatch ex:idDim5Tempo14.
ex:qtdObitos28 a rdf:property, qb:MeasureProperty;
  rdfs:domain ex:fato1Mortalidade;
  rdfs:range xsd:Integer;
```

```
  rdfs:label "qtd_obitos"@pt;
  rdfs:comment "Quantidade de óbitos (pessoas falecidas)."@pt;
  sdmx-attribute:unitMeasure unit:Count.
# ==================
# Declaração da Tabela Fato_Populacao_Absoluta e seus atributos
# ==================
ex:fato2PopulacaoAbsoluta a rdfs:Class, skos:Concept;
  rdfs:label "Fato_Mortalidade"@pt;
  rdfs:comment "Fato que registra a população absoluta por município, tempo e
características da população."@pt.
ex:idfato2DimCaractPopulacao29 a rdf:property;
  rdfs:domain ex:fato2PopulacaoAbsoluta;
  rdfs:range xsd:integer;
  rdfs:label "id_dim_caract_populacao"@pt;
  rdfs:comment "Chave estrangeira da Dim_Caract_Populacao."@pt;
  skos:exactMatch ex:idDim2CaractPopulacao3.
ex:idfato2DimLocalidade30 a rdf:property;
  rdfs:domain ex:fato2PopulacaoAbsoluta;
  rdfs:range xsd:integer;
  rdfs:label "id_dim_localidade"@pt;
  rdfs:comment "Chave estrangeira da Dim_Localidade."@pt;
  skos:exactMatch ex:idDim4Localidade10.
ex:idfato2DimTempo31 a rdf:property;
  rdfs:domain ex:fato2PopulacaoAbsoluta;
  rdfs:range xsd:integer;
  rdfs:label "id_dim_tempo"@pt;
  rdfs:comment "Chave estrangeira da Dim_Tempo."@pt;
  skos:exactMatch ex:idDim5Tempo14.
ex:qtdPessoas32 a rdf:property, qb:MeasureProperty;
  rdfs:domain ex:fato2PopulacaoAbsoluta;
  rdfs:range xsd:Integer;
  rdfs:label "qtd_obitos"@pt;
  rdfs:comment "Quantidade de pessoas que integram a população de
determinado lugar."@pt;
```

```
  sdmx-attribute:unitMeasure unit:Count.
ex:qtdMatriculasEscolar33 a rdf:property, qb:MeasureProperty;
  rdfs:domain ex:fato2PopulacaoAbsoluta;
  rdfs:range xsd:Integer;
  rdfs:label "qtd_obitos"@pt;
  rdfs:comment "Quantidade de matrículas na rede escolar em determinado
lugar."@pt;
  sdmx-attribute:unitMeasure unit:Count.
# ==================
# Declaração da Tabela Fato_Producao_Atividade e seus atributos
# ==================
ex:fato3ProducaoAtividade a rdfs:Class, skos:Concept;
  rdfs:label "Fato_Producao_Atividade"@pt;
  rdfs:comment "Fato que registra a produção econômica, por setor de atividade
em cada município brasieliro."@pt.
ex:idfato3DimAtividadeEconomica34 a rdf:property;
  rdfs:domain ex:fato3ProducaoAtividade;
  rdfs:range xsd:integer;
  rdfs:label "id_dim_atividade_economica"@pt;
  rdfs:comment "Chave estrangeira da Dim_Atividade_Economica."@pt.
  skos:exactMatch ex:idDim1AtividadeEconomica1.
ex:idfato3DimLocalidade35 a rdf:property;
  rdfs:domain ex:fato3ProducaoAtividade;
  rdfs:range xsd:integer;
  rdfs:label "id_dim_localidade"@pt;
  rdfs:comment "Chave estrangeira da Dim_Localidade."@pt;
  skos:exactMatch ex:idDim4Localidade10.
ex:idfato3DimTempo36 a rdf:property;
  rdfs:domain ex:fato3ProducaoAtividade;
  rdfs:range xsd:integer;
  rdfs:label "id_dim_tempo"@pt;
  rdfs:comment "Chave estrangeira da Dim_Tempo."@pt;
  skos:exactMatch ex:idDim5Tempo14.
ex:valProducao37 a rdf:property, qb:MeasureProperty;
```

```
  rdfs:domain ex:fato3ProducaoAtividade;
  rdfs:range xsd:decimal { maxLength="18" fractionDigits="2" };
  rdfs:label "val_producao"@pt;
  rdfs:comment "Valor da produção econômica oriúnda das atividades humanas."@pt;
  sdmx-attribute:unitMeasure unit:http://dbpedia.org/ontology/currency.
# ====================
# Declaração da Tabela Fato_Registros_Crimes e seus atributos
# ====================
ex:fato4RegistrosCrimes a rdfs:Class, skos:Concept;
  rdfs:label "Fato_Registros_Crimes"@pt;
  rdfs:comment "Fato que registra a quantidade de crimes ocorridos nos municípios brasieliros por características do criminoso e tipo de crime, mês a mês."@pt.
ex:idfato4DimCaractCriminoso38 a rdf:property;
  rdfs:domain ex:fato4RegistrosCrimes;
  rdfs:range xsd:integer;
  rdfs:label "id_dim_caract_criminoso"@pt;
  rdfs:comment "Chave estrangeira da Dim_Caract_Populacao."@pt;
  skos:exactMatch ex:idDim2CaractPopulacao3.
ex:idfato4DimLocalidade39 a rdf:property;
  rdfs:domain ex:fato4RegistrosCrimes;
  rdfs:range xsd:integer;
  rdfs:label "id_dim_localidade"@pt;
  rdfs:comment "Chave estrangeira da Dim_Localidade."@pt;
  skos:exactMatch ex:idDim4Localidade10.
ex:idfato4DimTempo40 a rdf:property;
  rdfs:domain ex:fato4RegistrosCrimes;
  rdfs:range xsd:integer;
  rdfs:label "id_dim_tempo"@pt;
  rdfs:comment "Chave estrangeira da Dim_Tempo."@pt;
  skos:exactMatch ex:idDim5Tempo14.
ex:idfato4DimTipoCrime41 a rdf:property;
  rdfs:domain ex:fato4RegistrosCrimes;
```

```
  rdfs:range xsd:integer;
  rdfs:label "id_tipo_crime"@pt;
  rdfs:comment "Chave estrangeira da Dim_Tipo_Crime."@pt;
  skos:exactMatch ex:idDim6TipoCrime21.
ex:qtdOcorrencias42 a rdf:property, qb:MeasureProperty;
  rdfs:domain ex:fato4RegistrosCrimes;
  rdfs:range xsd:integer;
  rdfs:label "qtd_ocorrencias"@pt;
  rdfs:comment "Quantidade de eventos criminosos registrados."@pt;
  sdmx-attribute:unitMeasure unit:Count.
# ============ FIM DA DECLARACAO DAS FATOS ===========
#=====================================================
# ====== FIM DECLARACAO MODELO DIMENSIONAL ==========
#=====================================================
```

14. Apêndice F: Exemplo de Triplas que representam os Cubos Dimensionais.

Abaixo descrevemos o conjunto de triplas que representam cada cubo dimensional que se projeta a partir dos modelos dimensionais (figura 76) cujas triplas foram apresentadas no apêndice E deste trabalho.

```
#=======================================================
# ============ INICIO DECLARACAO DOS CUBOS ============
#=======================================================

# ====================
# Declaração do Cubo Mortalidade
# ====================
ex:cubo-mortalidade a qb:DataSet;
  rdfs:label "Indicador de Mortalidade no Brasil"@pt;
  rdfs:comment "Quantidade de óbitos registrados nos municípios brasileiros.";
  qb:structure ex:dsd-cubo-mortalidade.

ex:dsd-cubo-mortalidade a qb:DataStructureDefinition;
 # Dimensões do Cubo
 qb:component
  [qb:dimension dimAxisFaixaEtaria4; qb:order 1],
  [qb:dimension dimAxisFlagUrbana5; qb:order 2],
  [qb:dimension dimAxisSexo6; qb:order 3],
  [qb:dimension dimAxisSubCategoria8; qb:order 4],
  [qb:dimension dimAxisCategoria9; qb:order 5],
  [qb:dimension dimAxisMunicipio11; qb:order 6],
  [qb:dimension dimAxisUf12; qb:order 7],
  [qb:dimension dimAxisRegiao13; qb:order 8],
```

```
  [qb:dimension dimAxisBimestre15; qb:order 9],
  [qb:dimension dimAxisMesNome16; qb:order 10],
  [qb:dimension dimAxisSemestre17; qb:order 11],
  [qb:dimension dimAxisTrimestre18; qb:order 12],
  [qb:dimension dimAxisMesNum19; qb:order 13],
  [qb:dimension dimAxisAno20; qb:order 14],
 # medidas do Cubo
  [qb:measure ex:qtdObitos28];
 # Atributo da medida
  [qb:attribute sdmx-attribute:unitMeasure; qb:componentAttachment
qb:DataSet;].

# ==================
# Declaração do Cubo Populacao_Absoluta
# ==================
ex:cubo-populacao-absoluta a qb:DataSet;
  rdfs:label "Indicador de População no Brasil"@pt;
  rdfs:comment "Quantidade de residentes e matrícula escolar registrados nos
municípios brasileiros.";
  qb:structure ex:dsd-populacao-absoluta.

ex:dsd-cubo-populacao-absoluta a qb:DataStructureDefinition;
 # Dimensões do Cubo
 qb:component
  [qb:dimension dimAxisFaixaEtaria4; qb:order 1],
  [qb:dimension dimAxisFlagUrbana5; qb:order 2],
  [qb:dimension dimAxisSexo6; qb:order 3],
  [qb:dimension dimAxisMunicipio11; qb:order 6],
  [qb:dimension dimAxisUf12; qb:order 7],
  [qb:dimension dimAxisRegiao13; qb:order 8],
  [qb:dimension dimAxisBimestre15; qb:order 9],
  [qb:dimension dimAxisMesNome16; qb:order 10],
  [qb:dimension dimAxisSemestre17; qb:order 11],
  [qb:dimension dimAxisTrimestre18; qb:order 12],
```

```
 [qb:dimension dimAxisMesNum19; qb:order 13],
 [qb:dimension dimAxisAno20; qb:order 14],
# medidas do Cubo
 [qb:measure ex:qtdPessoas32],
 [qb:measure ex:qtdMatriculasEscolar33],
# Atributo da medida
 [qb:attribute sdmx-attribute:unitMeasure; qb:componentAttachment
qb:DataSet;].

# ====================
# Declaração do Cubo Producao_Atividade
# ====================
ex:cubo-producao-atividade a qb:DataSet;
 rdfs:label "Indicador da Produção Econômica no Brasil"@pt;
 rdfs:comment "Riqueza produzida por setor de atividade nos municípios
brasileiros.";
 qb:structure ex:dsd-producao-atividade.

ex:dsd-cubo-producao-atividade a qb:DataStructureDefinition;
# Dimensões do Cubo
qb:component
 [qb:dimension dimAxisFaixaEtaria4; qb:order 1],
 [qb:dimension dimAxisFlagUrbana5; qb:order 2],
 [qb:dimension dimAxisSexo6; qb:order 3],
 [qb:dimension dimAxisMunicipio11; qb:order 6],
 [qb:dimension dimAxisUf12; qb:order 7],
 [qb:dimension dimAxisRegiao13; qb:order 8],
 [qb:dimension dimAxisBimestre15; qb:order 9],
 [qb:dimension dimAxisMesNome16; qb:order 10],
 [qb:dimension dimAxisSemestre17; qb:order 11],
 [qb:dimension dimAxisTrimestre18; qb:order 12],
 [qb:dimension dimAxisMesNum19; qb:order 13],
 [qb:dimension dimAxisAno20; qb:order 14],
# medidas do Cubo
```

```
 [qb:measure ex:ex:valProducao37],
 # Atributo da medida
 [qb:attribute sdmx-attribute:unitMeasure; qb:componentAttachment
qb:DataSet;].

# ====================
# Declaração do Cubo Registros Crimes
# ====================
ex:cubo-registros-crimes a qb:DataSet;
 rdfs:label "Indicador de Criminalidade no Brasil"@pt;
 rdfs:comment "Quantidade de crimes registrados nos municípios brasileiros.";
 qb:structure ex:dsd-cubo-registros-crimes.

ex:dsd-cubo-registros-crimes a qb:DataStructureDefinition;
 # Dimensões do Cubo
 qb:component
 [qb:dimension dimAxisFaixaEtaria4; qb:order 1],
 [qb:dimension dimAxisFlagUrbana5; qb:order 2],
 [qb:dimension dimAxisSexo6; qb:order 3],
 [qb:dimension dimAxisSubCategoria22; qb:order 4],
 [qb:dimension dimAxisCategoria23; qb:order 5],
 [qb:dimension dimAxisMunicipio11; qb:order 6],
 [qb:dimension dimAxisUf12; qb:order 7],
 [qb:dimension dimAxisRegiao13; qb:order 8],
 [qb:dimension dimAxisBimestre15; qb:order 9],
 [qb:dimension dimAxisMesNome16; qb:order 10],
 [qb:dimension dimAxisSemestre17; qb:order 11],
 [qb:dimension dimAxisTrimestre18; qb:order 12],
 [qb:dimension dimAxisMesNum19; qb:order 13],
 [qb:dimension dimAxisAno20; qb:order 14],
 # medidas do Cubo
 [qb:measure ex:qtdOcorrencias42];
 # Atributo da medida
```

[qb:attribute sdmx-attribute:unitMeasure; qb:componentAttachment qb:DataSet;].

#===
============== FIM DECLARACAO DOS CUBOS =============
#===

15. Apêndice G: Triplas para recuperação das observações registradas nos Cubos Dimensionais.

Abaixo descrevemos o conjunto de triplas que permitem a recuperação das instâncias das dimensões e as observações das F-TABs representadas no modelo dimensional apresentado na figura 76 deste trabalho.

```
#=======================================================
# ====== INICIO DECLARACAO DAS CONSULTAS R2RML =======
#=======================================================

#==== ESTA PARTE SERÁ INCLUÍDA DENTRO DO WRAPPER ====
ex:databaseBase_DW a d2rq:Database;
 d2rq:username "root";
 d2rq:password "root";
 d2rq:jdbcDSN "jdbc:oracle:thin:@192.168.0.48:1521:XE";
 d2rq:jdbcDriver "oracle.jdbc.OracleDriver"".
#== FIM DA PARTE QUE SERÁ INCLUÍDA DENTRO DO WRAPPER ===

#=======================================================
# = RECUPERACAO DAS INSTÂNCIAS DAS DIMENSÕES EM R2RML
#=======================================================
# ==================
# Declaração para recuperar as instâncias do Eixo Dimensional Nome_Atividade da Dim_Atividade_Economica
# ==================

ex:TriplesMapdimAxisNomeAtividade2 a rr:TriplesMap;
 d2rq:dataStorage ex:databaseBase_DW;
 rr:logicalTable [ rr:sqlQuery """
  SELECT DISTINCT
```

```
md5('www.fgv.br'||'BASE_DW'||'DIM_ATIVIDADE_ECONOMICA'||'NOME
_ATIVIDADE'||TO_CHAR(ID)||NOME_ATIVIDADE)                    as
nome_atividade_md5,
    nome_atividade, id
  FROM Dim_Atividade_Economica """; ];
 rr:subjectMap [
  rr:template
"http://data.example.com/atividade_economica/nome_atividade/{nome_atividade_md5}";
  rr:class ex-class:dimAxisNomeAtividade2; ];
 rr:predicateObjectMap [
  rr:predicate rdfs:label;
  rr:objectMap [ rr:column "nome_atividade"; rr:language "pt" ];].

# ====================
# Declaração para recuperar as instâncias do Eixo Dimensional Faixa_Etaria da
Dim_Caract_Populacao
# ====================

ex:TriplesMapdimAxisFaixaEtaria4 a rr:TriplesMap;
 d2rq:dataStorage ex:databaseBase_DW;
 rr:logicalTable [ rr:sqlQuery """
  SELECT DISTINCT
md5('www.fgv.br'||'BASE_DW'||'DIM_CARACT_POPULACAO'||'FAIXA_ETARIA'||TO_CHAR(ID)||FAIXA_ETARIA) as faixa_etaria_md5,
    faixa_etaria, id
  FROM Dim_Caract_Populacao """; ];
 rr:subjectMap [
  rr:template
"http://data.example.com/caract_populacao/faixa_etaria/{faixa_etaria_md5}";
  rr:class ex-class:dimAxisFaixaEtaria4; ];
 rr:predicateObjectMap [
  rr:predicate rdfs:label;
  rr:objectMap [ rr:column "faixa_etaria"; rr:language "pt" ];].
```

```
# ==================
# Declaração para recuperar as instâncias do Eixo Dimensional Flag_Urbana da Dim_Caract_Populacao
# ==================

ex:TriplesMapdimAxisFlagUrbana5 a rr:TriplesMap;
 d2rq:dataStorage ex:databaseBase_DW;
 rr:logicalTable [ rr:sqlQuery """
  SELECT DISTINCT
md5('www.fgv.br'||'BASE_DW'||'DIM_CARACT_POPULACAO'||'FLAG_URBANA'||TO_CHAR(ID)||FLAG_URBANA) as flag_urbana_md5,
    flag_urbana, id
  FROM Dim_Caract_Populacao """; ];
 rr:subjectMap [
  rr:template
"http://data.example.com/caract_populacao/flag_urbana/{flag_urbana_md5}";
  rr:class ex-class:dimAxisFlagUrbana5; ];
 rr:predicateObjectMap [
  rr:predicate rdfs:label;
  rr:objectMap [ rr:column "flag_urbana"; rr:language "pt" ];].

# ==================
# Declaração para recuperar as instâncias do Eixo Dimensional Sexo da Dim_Caract_Populacao
# ==================

ex:TriplesMapdimAxisSexo6 a rr:TriplesMap;
 d2rq:dataStorage ex:databaseBase_DW;
 rr:logicalTable [ rr:sqlQuery """
  SELECT DISTINCT
md5('www.fgv.br'||'BASE_DW'||'DIM_CARACT_POPULACAO'||'SEXO'||TO_CHAR(ID)||SEXO) as flag_urbana_md5,
    sexo, id
```

```
 FROM Dim_Caract_Populacao """; ];
 rr:subjectMap [
  rr:template "http://data.example.com/caract_populacao/sexo/{sexo_md5}";
  rr:class ex-class:dimAxisSexo6; ];
 rr:predicateObjectMap [
  rr:predicate rdfs:label;
  rr:objectMap [ rr:column "sexo"; rr:language "pt" ];].

# ====================
# Declaração para recuperar as instâncias do Eixo Dimensional Sub_Categoria
da Dim_Causa_Morte
# ====================

ex:TriplesMapdimAxisSubCategoria8 a rr:TriplesMap;
 d2rq:dataStorage ex:databaseBase_DW;
 rr:logicalTable [ rr:sqlQuery """
  SELECT DISTINCT
md5('www.fgv.br'||'BASE_DW'||'DIM_CAUSA_MORTE'||'SUB_CATEGORI
A'||TO_CHAR(ID)||SUB_CATEGORIA) as sub_categoria_md5,
    sub_categoria, id
  FROM Dim_Causa_Morte """; ];
 rr:subjectMap [
  rr:template
"http://data.example.com/causa_morte/sub_categoria/{sub_categoria_md5}";
  rr:class ex-class:dimAxisSubCategoria8; ];
 rr:predicateObjectMap [
  rr:predicate rdfs:label;
  rr:objectMap [ rr:column "sub_categoria"; rr:language "pt" ];].

# ====================
# Declaração para recuperar as instâncias do Eixo Dimensional Categoria da
Dim_Causa_Morte
# ====================
```

```
ex:TriplesMapdimAxisCategoria9 a rr:TriplesMap;
 d2rq:dataStorage ex:databaseBase_DW;
 rr:logicalTable [ rr:sqlQuery """
  SELECT DISTINCT
md5('www.fgv.br'||'BASE_DW'||'DIM_CAUSA_MORTE'||'CATEGORIA'||TO
_CHAR(ID)||CATEGORIA) as categoria_md5,
    categoria, id
  FROM Dim_Causa_Morte """; ];
 rr:subjectMap [
  rr:template
"http://data.example.com/causa_morte/categoria/{categoria_md5}";
  rr:class ex-class:dimAxisCategoria9; ];
 rr:predicateObjectMap [
  rr:predicate rdfs:label;
  rr:objectMap [ rr:column "categoria"; rr:language "pt" ];].

# ====================
# Declaração para recuperar as instâncias do Eixo Dimensional Municipio da
Dim_Localidade
# ====================

ex:TriplesMapdimAxisMunicipio11 a rr:TriplesMap;
 d2rq:dataStorage ex:databaseBase_DW;
 rr:logicalTable [ rr:sqlQuery """
  SELECT DISTINCT
md5('www.fgv.br'||'BASE_DW'||'DIM_LOCALIDADE'||'MUNICIPIO'||TO_C
HAR(ID)||MUNICIPIO) as municipio_md5,
    municipio, id
  FROM Dim_Localidade """; ];
 rr:subjectMap [
  rr:template
"http://data.example.com/localidade/municipio/{municipio_md5}";
  rr:class ex-class:dimAxisMunicipio11; ];
 rr:predicateObjectMap [
```

```
  rr:predicate rdfs:label;
  rr:objectMap [ rr:column "municipio"; rr:language "pt" ];].

# ====================
# Declaração para recuperar as instâncias do Eixo Dimensional Uf da
Dim_Localidade
# ====================

ex:TriplesMapdimAxisUf12 a rr:TriplesMap;
 d2rq:dataStorage ex:databaseBase_DW;
 rr:logicalTable [ rr:sqlQuery """
  SELECT DISTINCT
md5('www.fgv.br'||'BASE_DW'||'DIM_LOCALIDADE'||'UF'||TO_CHAR(ID)||
UF) as uf_md5,
    uf, id
  FROM Dim_Localidade """; ];
 rr:subjectMap [
  rr:template "http://data.example.com/localidade/uf/{uf_md5}";
  rr:class ex-class:dimAxisUf12; ];
 rr:predicateObjectMap [
  rr:predicate rdfs:label;
  rr:objectMap [ rr:column "uf"; rr:language "pt" ];].

# ====================
# Declaração para recuperar as instâncias do Eixo Dimensional Regiao da
Dim_Localidade
# ====================

ex:TriplesMapdimAxisRegiao13 a rr:TriplesMap;
 d2rq:dataStorage ex:databaseBase_DW;
 rr:logicalTable [ rr:sqlQuery """
  SELECT DISTINCT
md5('www.fgv.br'||'BASE_DW'||'DIM_LOCALIDADE'||'REGIAO'||TO_CHA
R(ID)||REGIAO) as regiao_md5,
```

```
   regiao, id
  FROM Dim_Localidade """; ];
 rr:subjectMap [
  rr:template "http://data.example.com/localidade/regiao/{regiao_md5}";
  rr:class ex-class:dimAxisRegiao13; ];
 rr:predicateObjectMap [
  rr:predicate rdfs:label;
  rr:objectMap [ rr:column "regiao"; rr:language "pt" ];].

# ==================
# Declaração para recuperar as instâncias do Eixo Dimensional Bimestre da
Dim_Tempo
# ==================

ex:TriplesMapdimAxisBimestre15 a rr:TriplesMap;
 d2rq:dataStorage ex:databaseBase_DW;
 rr:logicalTable [ rr:sqlQuery """
  SELECT DISTINCT
md5('www.fgv.br'||'BASE_DW'||'DIM_TEMPO'||'BIMESTRE'||TO_CHAR(ID)
||BIMESTRE) as bimestre_md5,
    bimestre, id
  FROM Dim_Tempo """; ];
 rr:subjectMap [
  rr:template "http://data.example.com/tempo/bimestre/{bimestre_md5}";
  rr:class ex-class:dimAxisBimestre15; ];
 rr:predicateObjectMap [
  rr:predicate rdfs:label;
  rr:objectMap [ rr:column "bimestre"; rr:language "pt" ];].

# ==================
# Declaração para recuperar as instâncias do Eixo Dimensional Mes_Nome da
Dim_Tempo
# ==================
```

```
ex:TriplesMapdimAxisMesNome16 a rr:TriplesMap;
 d2rq:dataStorage ex:databaseBase_DW;
 rr:logicalTable [ rr:sqlQuery """
  SELECT DISTINCT
md5('www.fgv.br'||'BASE_DW'||'DIM_TEMPO'||'MES_NOME'||TO_CHAR(I
D)||MES_NOME) as mes_nome_md5,
    mes_nome, id
  FROM Dim_Tempo """; ];
 rr:subjectMap [
  rr:template "http://data.example.com/tempo/mes_nome/{mes_nome_md5}";
  rr:class ex-class:dimAxisMesNome16; ];
 rr:predicateObjectMap [
  rr:predicate rdfs:label;
  rr:objectMap [ rr:column "mes_nome"; rr:language "pt" ];].

# ====================
# Declaração para recuperar as instâncias do Eixo Dimensional Bimestre da
Dim_Tempo
# ====================

ex:TriplesMapdimAxisSemestre17 a rr:TriplesMap;
 d2rq:dataStorage ex:databaseBase_DW;
 rr:logicalTable [ rr:sqlQuery """
  SELECT DISTINCT
md5('www.fgv.br'||'BASE_DW'||'DIM_TEMPO'||'SEMESTRE'||TO_CHAR(ID
)||SEMESTRE) as semestre_md5,
    semestre, id
  FROM Dim_Tempo """; ];
 rr:subjectMap [
  rr:template "http://data.example.com/tempo/semestre/{semestre_md5}";
  rr:class ex-class:dimAxisSemestre17; ];
 rr:predicateObjectMap [
  rr:predicate rdfs:label;
  rr:objectMap [ rr:column "semestre"; rr:language "pt" ];].
```

```
# ==================
# Declaração para recuperar as instâncias do Eixo Dimensional Trimestre da Dim_Tempo
# ==================

ex:TriplesMapdimAxisTrimestre18 a rr:TriplesMap;
 d2rq:dataStorage ex:databaseBase_DW;
 rr:logicalTable [ rr:sqlQuery """
  SELECT DISTINCT
md5('www.fgv.br'||'BASE_DW'||'DIM_TEMPO'||'TRIMESTRE'||TO_CHAR(ID)||TRIMESTRE) as trimestre_md5,
    trimestre, id
  FROM Dim_Tempo """; ];
 rr:subjectMap [
  rr:template "http://data.example.com/tempo/trimestre/{trimestre_md5}";
  rr:class ex-class:dimAxisTrimestre18; ];
 rr:predicateObjectMap [
  rr:predicate rdfs:label;
  rr:objectMap [ rr:column "trimestre"; rr:language "pt" ];].

# ==================
# Declaração para recuperar as instâncias do Eixo Dimensional Mes_Num da Dim_Tempo
# ==================

ex:TriplesMapdimAxisMesNum19 a rr:TriplesMap;
 d2rq:dataStorage ex:databaseBase_DW;
 rr:logicalTable [ rr:sqlQuery """
  SELECT DISTINCT
md5('www.fgv.br'||'BASE_DW'||'DIM_TEMPO'||'MES_NUM'||TO_CHAR(ID)||MES_NUM) as mes_num_md5,
    mes_num, id
  FROM Dim_Tempo """; ];
```

```
 rr:subjectMap [
  rr:template "http://data.example.com/tempo/mes_num/{mes_num_md5}";
  rr:class ex-class:dimAxisMesNum19; ];
 rr:predicateObjectMap [
  rr:predicate rdfs:label;
  rr:objectMap [ rr:column "mes_num"; rr:language "pt" ];].

# ====================
# Declaração para recuperar as instâncias do Eixo Dimensional Ano da
Dim_Tempo
# ====================

ex:TriplesMapdimAxisAno20 a rr:TriplesMap;
 d2rq:dataStorage ex:databaseBase_DW;
 rr:logicalTable [ rr:sqlQuery """
  SELECT DISTINCT
md5('www.fgv.br'||'BASE_DW'||'DIM_TEMPO'||'ANO'||TO_CHAR(ID)||ANO
) as ano_md5,
    ano, id
  FROM Dim_Tempo """; ];
 rr:subjectMap [
  rr:template "http://data.example.com/tempo/ano/{ano_md5}";
  rr:class ex-class:dimAxisAno20; ];
 rr:predicateObjectMap [
  rr:predicate rdfs:label;
  rr:objectMap [ rr:column "ano"; rr:language "pt" ];].

# ====================
# Declaração para recuperar as instâncias do Eixo Dimensional Sub_Categoria
da Dim_Tipo_Crime
# ====================

ex:TriplesMapdimAxisSubCategoria22 a rr:TriplesMap;
 d2rq:dataStorage ex:databaseBase_DW;
```

```
 rr:logicalTable [ rr:sqlQuery """
  SELECT DISTINCT
md5('www.fgv.br'||'BASE_DW'||'DIM_TIPO_CRIME'||'SUB_CATEGORIA'||T
O_CHAR(ID)||SUB_CATEGORIA) as sub_categoria_md5,
    sub_categoria, id
  FROM Dim_Tipo_Crime """; ];
 rr:subjectMap [
  rr:template
"http://data.example.com/tipo_crime/sub_categoria/{sub_categoria_md5}";
  rr:class ex-class:dimAxisSubCategoria22; ];
 rr:predicateObjectMap [
  rr:predicate rdfs:label;
  rr:objectMap [ rr:column "sub_categoria"; rr:language "pt" ];].

# ====================
# Declaração para recuperar as instâncias do Eixo Dimensional Categoria da
Dim_Tipo_Crime
# ====================

ex:TriplesMapdimAxisCategoria23 a rr:TriplesMap;
 d2rq:dataStorage ex:databaseBase_DW;
 rr:logicalTable [ rr:sqlQuery """
  SELECT DISTINCT
md5('www.fgv.br'||'BASE_DW'||'DIM_TIPO_CRIME'||'CATEGORIA'||TO_C
HAR(ID)||CATEGORIA) as categoria_md5,
    categoria, id
  FROM Dim_Tipo_Crime """; ];
 rr:subjectMap [
  rr:template
"http://data.example.com/Tipo_Crime/categoria/{categoria_md5}";
  rr:class ex-class:dimAxisCategoria23; ];
 rr:predicateObjectMap [
  rr:predicate rdfs:label;
  rr:objectMap [ rr:column "categoria"; rr:language "pt" ];].
```

```
#=======================================================
#== RECUPERACAO DAS INSTÂNCIAS DOS CUBOS EM R2RML ==
#=======================================================

# ================== CUBO MORTALIDADE ==============

ex:TriplesMapFatoMortalidade a rr:TriplesMap;
 d2rq:dataStorage ex:databaseResidents;
 rr:logicalTable [ rr:sqlQuery "“”
     SELECT id_dim_caract_populacao, id_dim_causa_morte, id_tempo,
id_localidade, qtd_obitos
     FROM fato_mortalidade """; ];
 rr:subjectMap [
  rr:template

"http://purl.org/GovDataCube/resources/cubomortalidade/observations/{id_di
m_caract_populacao}_{id_dim_causa_morte}_{id_tempo}_{id_localidade}_{
qtd_obitos}";
  rr:class qb:observation; ];
 rr:predicateObjectMap [
  rr:predicate qb:dataSet;
  rr:objectMap [rr:constant ex:cubo-mortalidade]; ];
 rr:predicateObjectMap [
  rr:predicate ex:dimAxisFaixaEtaria4;
  rr:objectMap [
   rr:parentTriplesMap ex:TriplesMapdimAxisFaixaEtaria4;
   rr:joinCondition [
   rr:child "id_dim_caract_populacao";
   rr:parent "id"; ];];];
 rr:predicateObjectMap [
  rr:predicate ex:dimAxisFlagUrbana5;
  rr:objectMap [
   rr:parentTriplesMap ex:TriplesMapdimAxisFlagUrbana5;
```

```
 rr:joinCondition [
 rr:child "id_dim_caract_populacao";
 rr:parent "id"; ];];];
rr:predicateObjectMap [
 rr:predicate ex:dimAxisSexo6;
 rr:objectMap [
 rr:parentTriplesMap ex:TriplesMapdimAxisSexo6;
 rr:joinCondition [
 rr:child "id_dim_caract_populacao";
 rr:parent "id"; ];];];
rr:predicateObjectMap [
 rr:predicate ex:dimAxisSubCategoria8;
 rr:objectMap [
 rr:parentTriplesMap ex:TriplesMapdimAxisSubCategoria8;
 rr:joinCondition [
 rr:child "id_dim_causa_morte";
 rr:parent "id"; ];];];
rr:predicateObjectMap [
 rr:predicate ex:dimAxisCategoria9;
 rr:objectMap [
 rr:parentTriplesMap ex:TriplesMapdimAxisCategoria9;
 rr:joinCondition [
 rr:child "id_dim_causa_morte";
 rr:parent "id"; ];];];
rr:predicateObjectMap [
 rr:predicate ex:dimAxisMunicipio11;
 rr:objectMap [
 rr:parentTriplesMap ex:TriplesMapdimAxisMunicipio11;
 rr:joinCondition [
 rr:child "id_dim_localidade";
 rr:parent "id"; ];];];
rr:predicateObjectMap [
 rr:predicate ex:dimAxisUf12;
 rr:objectMap [
```

```
  rr:parentTriplesMap ex:TriplesMapdimAxisUf12;
  rr:joinCondition [
  rr:child "id_dim_localidade";
  rr:parent "id"; ];];];
 rr:predicateObjectMap [
  rr:predicate ex:dimAxisRegiao13;
  rr:objectMap [
  rr:parentTriplesMap ex:TriplesMapdimAxisRegiao13;
  rr:joinCondition [
  rr:child "id_dim_localidade";
  rr:parent "id"; ];];];
 rr:predicateObjectMap [
  rr:predicate ex:dimAxisBimestre15;
  rr:objectMap [
  rr:parentTriplesMap ex:TriplesMapdimAxisBimestre15;
  rr:joinCondition [
  rr:child "id_dim_tempo";
  rr:parent "id"; ];];];
 rr:predicateObjectMap [
  rr:predicate ex:dimAxisMesNome16;
  rr:objectMap [
  rr:parentTriplesMap ex:TriplesMapdimAxisMesNome16;
  rr:joinCondition [
  rr:child "id_dim_tempo";
  rr:parent "id"; ];];];
 rr:predicateObjectMap [
  rr:predicate ex:dimAxisSemestre17;
  rr:objectMap [
  rr:parentTriplesMap ex:TriplesMapdimAxisSemestre17;
  rr:joinCondition [
  rr:child "id_dim_tempo";
  rr:parent "id"; ];];];
 rr:predicateObjectMap [
  rr:predicate ex:dimAxisTrimestre18;
```

```
 rr:objectMap [
  rr:parentTriplesMap ex:TriplesMapdimAxisTrimestre18;
  rr:joinCondition [
  rr:child "id_dim_tempo";
  rr:parent "id"; ];];];
 rr:predicateObjectMap [
 rr:predicate ex:dimAxisMesNum19;
 rr:objectMap [
  rr:parentTriplesMap ex:TriplesMapdimAxisMesNum19;
  rr:joinCondition [
  rr:child "id_dim_tempo";
  rr:parent "id"; ];];];
 rr:predicateObjectMap [
 rr:predicate ex:dimAxisAno20;
 rr:objectMap [
  rr:parentTriplesMap ex:TriplesMapdimAxisAno20;
  rr:joinCondition [
  rr:child "id_dim_tempo";
  rr:parent "id"; ];];];
 rr:predicateObjectMap [
 rr:predicate ex:qtdObitos28;
 rr:objectMap [rr:column "qtd_obitos"]; ].

#=============== CUBO POPULACAO_ABSOLUTA =============

ex:TriplesMapFatoPopulacaoAbsoluta a rr:TriplesMap;
 d2rq:dataStorage ex:databaseResidents;
 rr:logicalTable [ rr:sqlQuery "“”
     SELECT id_dim_caract_populacao, id_tempo, id_localidade, qtd_pessoas,
qtd_matriculas_escolar
     FROM fato_populacao_absoluta """; ];
 rr:subjectMap [
  rr:template
```

```
"http://purl.org/GovDataCube/resources/cubopopulacaoabsoluta/observations/{
id_dim_caract_populacao}_{id_tempo}_{id_localidade}_{qtd_pessoas}_{qtd
_matriculas_escolar}";
 rr:class qb:observation; ];
rr:predicateObjectMap [
 rr:predicate qb:dataSet;
 rr:objectMap [rr:constant ex:cubo-populacao-absoluta]; ];
rr:predicateObjectMap [
 rr:predicate ex:dimAxisFaixaEtaria4;
 rr:objectMap [
  rr:parentTriplesMap ex:TriplesMapdimAxisFaixaEtaria4;
  rr:joinCondition [
  rr:child "id_dim_caract_populacao";
  rr:parent "id"; ];];];
rr:predicateObjectMap [
 rr:predicate ex:dimAxisFlagUrbana5;
 rr:objectMap [
  rr:parentTriplesMap ex:TriplesMapdimAxisFlagUrbana5;
  rr:joinCondition [
  rr:child "id_dim_caract_populacao";
  rr:parent "id"; ];];];
rr:predicateObjectMap [
 rr:predicate ex:dimAxisSexo6;
 rr:objectMap [
  rr:parentTriplesMap ex:TriplesMapdimAxisSexo6;
  rr:joinCondition [
  rr:child "id_dim_caract_populacao";
  rr:parent "id"; ];];];
rr:predicateObjectMap [
 rr:predicate ex:dimAxisMunicipio11;
 rr:objectMap [
  rr:parentTriplesMap ex:TriplesMapdimAxisMunicipio11;
  rr:joinCondition [
```

```
  rr:child "id_dim_localidade";
  rr:parent "id"; ];];];
 rr:predicateObjectMap [
  rr:predicate ex:dimAxisUf12;
  rr:objectMap [
   rr:parentTriplesMap ex:TriplesMapdimAxisUf12;
   rr:joinCondition [
   rr:child "id_dim_localidade";
   rr:parent "id"; ];];];
 rr:predicateObjectMap [
  rr:predicate ex:dimAxisRegiao13;
  rr:objectMap [
   rr:parentTriplesMap ex:TriplesMapdimAxisRegiao13;
   rr:joinCondition [
   rr:child "id_dim_localidade";
   rr:parent "id"; ];];];
 rr:predicateObjectMap [
  rr:predicate ex:dimAxisBimestre15;
  rr:objectMap [
   rr:parentTriplesMap ex:TriplesMapdimAxisBimestre15;
   rr:joinCondition [
   rr:child "id_dim_tempo";
   rr:parent "id"; ];];];
 rr:predicateObjectMap [
  rr:predicate ex:dimAxisMesNome16;
  rr:objectMap [
   rr:parentTriplesMap ex:TriplesMapdimAxisMesNome16;
   rr:joinCondition [
   rr:child "id_dim_tempo";
   rr:parent "id"; ];];];
 rr:predicateObjectMap [
  rr:predicate ex:dimAxisSemestre17;
  rr:objectMap [
   rr:parentTriplesMap ex:TriplesMapdimAxisSemestre17;
```

```
  rr:joinCondition [
  rr:child "id_dim_tempo";
  rr:parent "id"; ];];];
 rr:predicateObjectMap [
  rr:predicate ex:dimAxisTrimestre18;
  rr:objectMap [
   rr:parentTriplesMap ex:TriplesMapdimAxisTrimestre18;
   rr:joinCondition [
   rr:child "id_dim_tempo";
   rr:parent "id"; ];];];
 rr:predicateObjectMap [
  rr:predicate ex:dimAxisMesNum19;
  rr:objectMap [
   rr:parentTriplesMap ex:TriplesMapdimAxisMesNum19;
   rr:joinCondition [
   rr:child "id_dim_tempo";
   rr:parent "id"; ];];];
 rr:predicateObjectMap [
  rr:predicate ex:dimAxisAno20;
  rr:objectMap [
   rr:parentTriplesMap ex:TriplesMapdimAxisAno20;
   rr:joinCondition [
   rr:child "id_dim_tempo";
   rr:parent "id"; ];];];
 rr:predicateObjectMap [
  rr:predicate ex:qtdPessoas32;
  rr:objectMap [rr:column "qtd_pessoas"]; ];
 rr:predicateObjectMap [
  rr:predicate ex:qtdMatriculasEscolar33;
  rr:objectMap [rr:column "qtd_matriculas_escolar"]; ].

#============== CUBO PRODUCAO_ATIVIDADE =============

ex:TriplesMapFatoProducaoAtividade a rr:TriplesMap;
```

```
 d2rq:dataStorage ex:databaseResidents;
 rr:logicalTable [ rr:sqlQuery "“”
     SELECT id_dim_atividade_economica, id_dim_tempo, id_dim_localidade,
val_producao
     FROM fato_producao_atividade """; ];
 rr:subjectMap [
  rr:template

"http://purl.org/GovDataCube/resources/cuboproducaoatividade/observations/{
id_dim_atividade_economica}_{id_tempo}_{id_localidade}_{val_producao}"
;
  rr:class qb:observation; ];
 rr:predicateObjectMap [
  rr:predicate qb:dataSet;
  rr:objectMap [rr:constant ex:cubo-producao-atividade]; ];
 rr:predicateObjectMap [
  rr:predicate ex:dimAxisNomeAtividade2;
  rr:objectMap [
   rr:parentTriplesMap ex:TriplesMapdimAxisNomeAtividade2;
   rr:joinCondition [
   rr:child "id_dim_atividade_economica";
   rr:parent "id"; ];];];
 rr:predicateObjectMap [
  rr:predicate ex:dimAxisMunicipio11;
  rr:objectMap [
   rr:parentTriplesMap ex:TriplesMapdimAxisMunicipio11;
   rr:joinCondition [
   rr:child "id_dim_localidade";
   rr:parent "id"; ];];];
 rr:predicateObjectMap [
  rr:predicate ex:dimAxisUf12;
  rr:objectMap [
   rr:parentTriplesMap ex:TriplesMapdimAxisUf12;
   rr:joinCondition [
```

```
  rr:child "id_dim_localidade";
  rr:parent "id"; ];];];
 rr:predicateObjectMap [
  rr:predicate ex:dimAxisRegiao13;
  rr:objectMap [
   rr:parentTriplesMap ex:TriplesMapdimAxisRegiao13;
   rr:joinCondition [
   rr:child "id_dim_localidade";
   rr:parent "id"; ];];];
 rr:predicateObjectMap [
  rr:predicate ex:dimAxisBimestre15;
  rr:objectMap [
   rr:parentTriplesMap ex:TriplesMapdimAxisBimestre15;
   rr:joinCondition [
   rr:child "id_dim_tempo";
   rr:parent "id"; ];];];
 rr:predicateObjectMap [
  rr:predicate ex:dimAxisMesNome16;
  rr:objectMap [
   rr:parentTriplesMap ex:TriplesMapdimAxisMesNome16;
   rr:joinCondition [
   rr:child "id_dim_tempo";
   rr:parent "id"; ];];];
 rr:predicateObjectMap [
  rr:predicate ex:dimAxisSemestre17;
  rr:objectMap [
   rr:parentTriplesMap ex:TriplesMapdimAxisSemestre17;
   rr:joinCondition [
   rr:child "id_dim_tempo";
   rr:parent "id"; ];];];
 rr:predicateObjectMap [
  rr:predicate ex:dimAxisTrimestre18;
  rr:objectMap [
   rr:parentTriplesMap ex:TriplesMapdimAxisTrimestre18;
```

```
  rr:joinCondition [
  rr:child "id_dim_tempo";
  rr:parent "id"; ];];];
 rr:predicateObjectMap [
  rr:predicate ex:dimAxisMesNum19;
  rr:objectMap [
  rr:parentTriplesMap ex:TriplesMapdimAxisMesNum19;
  rr:joinCondition [
  rr:child "id_dim_tempo";
  rr:parent "id"; ];];];
 rr:predicateObjectMap [
  rr:predicate ex:dimAxisAno20;
  rr:objectMap [
  rr:parentTriplesMap ex:TriplesMapdimAxisAno20;
  rr:joinCondition [
  rr:child "id_dim_tempo";
  rr:parent "id"; ];];];
 rr:predicateObjectMap [
  rr:predicate ex:valProducao37;
  rr:objectMap [rr:column "val_producao"]; ].

#================== CUBO REGISTROS_CRIME ==============

ex:TriplesMapFatoMortalidade a rr:TriplesMap;
 d2rq:dataStorage ex:databaseResidents;
 rr:logicalTable [ rr:sqlQuery "“”
     SELECT     id_dim_caract_populacao,     id_dim_tipo_crime,     id_tempo,
id_localidade, qtd_ocorrencias
     FROM fato_registros_crime """; ];
 rr:subjectMap [
  rr:template

"http://purl.org/GovDataCube/resources/cuboregistroscrime/observations/{id_d
```

```
im_caract_populacao}_{id_dim_tipo_crime}_{id_tempo}_{id_localidade}_{q
td_ocorrencias}";
  rr:class qb:observation; ];
 rr:predicateObjectMap [
  rr:predicate qb:dataSet;
  rr:objectMap [rr:constant ex:cubo-registros-crime]; ];
 rr:predicateObjectMap [
  rr:predicate ex:dimAxisFaixaEtaria4;
  rr:objectMap [
   rr:parentTriplesMap ex:TriplesMapdimAxisFaixaEtaria4;
   rr:joinCondition [
   rr:child "id_dim_caract_populacao";
   rr:parent "id"; ];];];
 rr:predicateObjectMap [
  rr:predicate ex:dimAxisFlagUrbana5;
  rr:objectMap [
   rr:parentTriplesMap ex:TriplesMapdimAxisFlagUrbana5;
   rr:joinCondition [
   rr:child "id_dim_caract_populacao";
   rr:parent "id"; ];];];
 rr:predicateObjectMap [
  rr:predicate ex:dimAxisSexo6;
  rr:objectMap [
   rr:parentTriplesMap ex:TriplesMapdimAxisSexo6;
   rr:joinCondition [
   rr:child "id_dim_caract_populacao";
   rr:parent "id"; ];];];
 rr:predicateObjectMap [
  rr:predicate ex:dimAxisSubCategoria22;
  rr:objectMap [
   rr:parentTriplesMap ex:TriplesMapdimAxisSubCategoria22;
   rr:joinCondition [
   rr:child "id_dim_tipo_crime";
   rr:parent "id"; ];];];
```

```
rr:predicateObjectMap [
 rr:predicate ex:dimAxisCategoria23;
 rr:objectMap [
  rr:parentTriplesMap ex:TriplesMapdimAxisCategoria23;
  rr:joinCondition [
  rr:child "id_dim_tipo_crime";
  rr:parent "id"; ];];];
rr:predicateObjectMap [
 rr:predicate ex:dimAxisMunicipio11;
 rr:objectMap [
  rr:parentTriplesMap ex:TriplesMapdimAxisMunicipio11;
  rr:joinCondition [
  rr:child "id_dim_localidade";
  rr:parent "id"; ];];];
rr:predicateObjectMap [
 rr:predicate ex:dimAxisUf12;
 rr:objectMap [
  rr:parentTriplesMap ex:TriplesMapdimAxisUf12;
  rr:joinCondition [
  rr:child "id_dim_localidade";
  rr:parent "id"; ];];];
rr:predicateObjectMap [
 rr:predicate ex:dimAxisRegiao13;
 rr:objectMap [
  rr:parentTriplesMap ex:TriplesMapdimAxisRegiao13;
  rr:joinCondition [
  rr:child "id_dim_localidade";
  rr:parent "id"; ];];];
rr:predicateObjectMap [
 rr:predicate ex:dimAxisBimestre15;
 rr:objectMap [
  rr:parentTriplesMap ex:TriplesMapdimAxisBimestre15;
  rr:joinCondition [
  rr:child "id_dim_tempo";
```

```
 rr:parent "id"; ];];];
rr:predicateObjectMap [
 rr:predicate ex:dimAxisMesNome16;
 rr:objectMap [
  rr:parentTriplesMap ex:TriplesMapdimAxisMesNome16;
  rr:joinCondition [
  rr:child "id_dim_tempo";
  rr:parent "id"; ];];];
rr:predicateObjectMap [
 rr:predicate ex:dimAxisSemestre17;
 rr:objectMap [
  rr:parentTriplesMap ex:TriplesMapdimAxisSemestre17;
  rr:joinCondition [
  rr:child "id_dim_tempo";
  rr:parent "id"; ];];];
rr:predicateObjectMap [
 rr:predicate ex:dimAxisTrimestre18;
 rr:objectMap [
  rr:parentTriplesMap ex:TriplesMapdimAxisTrimestre18;
  rr:joinCondition [
  rr:child "id_dim_tempo";
  rr:parent "id"; ];];];
rr:predicateObjectMap [
 rr:predicate ex:dimAxisMesNum19;
 rr:objectMap [
  rr:parentTriplesMap ex:TriplesMapdimAxisMesNum19;
  rr:joinCondition [
  rr:child "id_dim_tempo";
  rr:parent "id"; ];];];
rr:predicateObjectMap [
 rr:predicate ex:dimAxisAno20;
 rr:objectMap [
  rr:parentTriplesMap ex:TriplesMapdimAxisAno20;
  rr:joinCondition [
```

```
  rr:child "id_dim_tempo";
  rr:parent "id"; ];];];
 rr:predicateObjectMap [
  rr:predicate ex:qtdOcorrencias42;
  rr:objectMap [rr:column "qtd_ocorrencias"]; ].

#=========================================================
#======== FIM DECLARACAO DAS CONSULTAS R2RML ========
#=========================================================
```

16. Apêndice H: Linguagens e Ontologias.

RDF

O RDF (Brickley, et al., 2004) é um padrão do W3C que tem por objetivo viabilizar a descrição de recursos na Web. Trata-se de um modelo de dados utilizado para representar um recurso e seus atributos, bem como a relação entre atributos. Um recurso em RDF pode ser qualquer coisa. Em outras palavras, um recurso em RDF pode ser entendido como um sinônimo de "entidade", um termo genérico para qualquer coisa em um determinado domínio[34]. RDF baseia-se na ideia de que todas as coisas podem e são descritas através de propriedades que possuem valores. Estas descrições são realizadas através de declarações. Uma típica declaração RDF possui o seguinte formato: (Sujeito, Predicado, Objeto) ou (Recurso, Propriedade, Valor [de propriedade]), onde:

- Sujeito ou Recurso: identifica o objeto da declaração;
- Propriedade: identifica uma propriedade ou característica do objeto;
- Valor: identifica o valor assumido para o objeto naquela propriedade.

Este formato de representação é conhecido como "tripla". Em uma "tripla" RDF o predicado é usado para ligar o sujeito ao objeto ou, em outras palavras, através de uma propriedade conectamos um recurso ao valor a ele atribuído para aquela propriedade. Graficamente representamos as triplas como:

Figura 68: Representação Gráfica de uma Tripla RDF

Uma exigência que temos em RDF é que cada recurso tenha uma identificação única. A Web Semântica provê uma forma geral de identificação

[34] http://www.w3.org/TR/2004/REC-rdf-schema-20040210/

chamada URI[35] (Uniform Resource Identifier) que pode ser aplicada a qualquer objeto esteja ele, ou não, na Web, desde que seja necessário referenciá-lo em alguma declaração RDF. De forma mais precisa, RDF usa "URI references" (URIref), a qual caracteriza-se pela adição opcional de um identificador de fragmento no final da URI.

A especificação RDF restringe os valores permitidos para cada uma das partes da tripla:

• Sujeito : PODE ser uma URIref ou um nó vazio,

• Predicado : DEVE ser uma URIref,

• Objeto : PODE ser uma URIref, um literal ou um nó vazio.

No nosso dia a dia, usamos a linguagem natural para fazer declarações que expressam fatos a respeito de algo. Podemos querer expressar, por exemplo, o fato de que uma cidade é capital de um país, através da seguinte declaração: "Brasília é a capital do Brasil". Esta declaração pode ser subdividida nas seguintes partes:

- "Brasil" representa o recurso que está sendo descrito pela declaração.
- "capital" representa a propriedade específica do recurso que a declaração descreve.
- "Brasília" representa o valor da propriedade mencionada na declaração.

Esta declaração pode ser representada sob o formato de uma tripla RDF onde o sujeito (Brasil) está relacionado ao objeto (Brasília) através do predicado (temCapital), cujo relacionamento encontra-se ilustrado na figura abaixo:

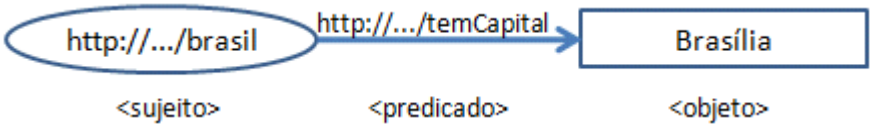


Figura 69: Representação Gráfica de uma Tripla RDF

A figura anterior representa a tripla através de um grafo dirigido rotulado onde os nós representam o sujeito e o objeto e o arco representa o predicado.

A partir dessa representação básica baseada em tripla, podemos formular declarações mais complexas sobre recursos. Abaixo ilustramos um grafo RDF contendo várias declarações.

[35] http://www.w3.org/2011/rdf-wg/wiki/IRIs/RDFConceptsProposal

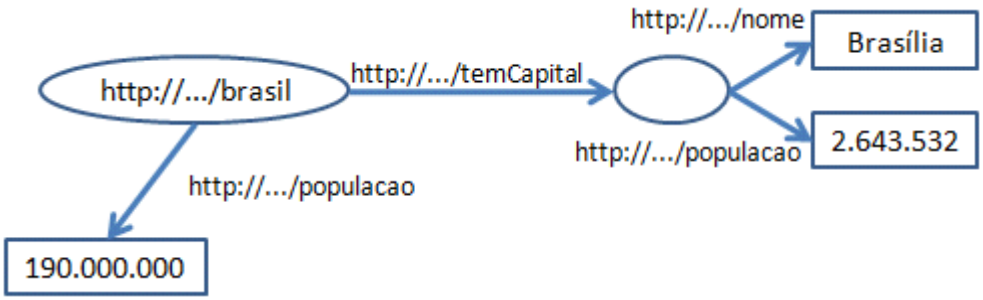


Figura 70: Exemplo de Grafo RDF

No grafo acima, as elipses são usadas para identificar sujeitos ou objetos representados por uma URI. Por exemplo, a URI http://../Brasil representa o sujeito “Brasil”, que está sendo descrito através da declaração das propriedades http://.../temCapital e http://.../populacao com seus respectivos valores associados. Estes valores podem ser representados de 2 maneiras distintas: através de um retângulo, usado para indicar valores literais ou de uma elipse, usada para indicar valores que referem-se a URIs. A identificação dos arcos (predicados) é sempre feita através do uso de URI.

Além das declarações sobre recursos, com RDF também podemos fazer declarações que referenciam outras declarações (declarações sobre declarações). Denominamos esta característica do RDF de reificação.

Tomemos por exemplo, a seguinte declaração em linguagem natural: Brasília é a capital do Brasil desde 1962.

Aplicando a esta declaração o conceito de reificação, obtemos a seguinte declaração aninhada (<(<Brasil> <temCapital> <Brasilia>)> <ehCapitalDesde> <1962>) onde a declaração (<Brasil> <temCapital> <Brasilia>) desempenha o papel de sujeito na declaração como um todo..

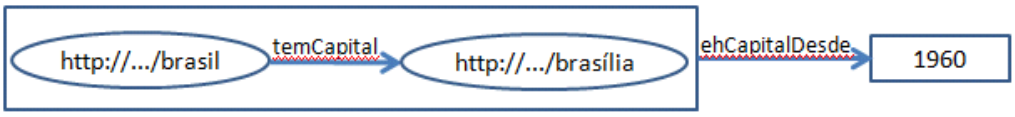


Figura 71: Exemplo do conceito de reificação em RDF

O uso do conceito de reificação é útil pois permite o uso de declarações que tenham sido feitas em determinado domínio, referenciando-as como recursos na formulação de outras declarações expressas em outros domínios.

A afirmação expressa em um grafo RDF corresponde à afirmação de todas as suas respectivas triplas. No entanto, para que as declarações RDF

possam ser processadas por máquinas, precisamos, além do uso de URIs para identificar unicamente cada recurso, de uma representação que seja mais adequada para este propósito, além de algum mecanismo para a definição de tipos de dados, uma vez que a especificação RDF não provê este tipo de mecanismo.

Apesar de não possuir mecanismos para representação de tipos de dados, RDF permite que sejam usados tipos de dados definidos externamente, como por exemplo, esquemas XML[36].

Na elaboração de documentos RDF é permitido o uso de 2 tipos de dados. O primeiro tipo refere-se ao uso das URIs e o segundo, ao uso de literais, sendo permitido string e tipos de dados definidos com XML Schema.

Em RDF, podemos fazer uso de diferentes formatos para serializar as declarações contidas nos grafos. Alguns dos principais formatos disponíveis são: RDF/XML, Turtle, N3 e RXR , sendo que RDF/XML é a recomendação oficial da W3C. Abaixo apresentamos a serialização em RDF/XML da tripla da figura 69.

```
<rdf:Description rdf:about="http://.../ppgsc/paises#Brasil">
        <ppgsc:temCapital>Brasilia</ppgsc:temCapital>
</rdf:Description>
```

Em resumo, vimos que o RDF complementa o XML oferecendo significado para o aninhamento de tags e um modelo de dados para descrever recursos. Por outro lado, o RDF possui uma expressividade muito limitada estando restrito a expressar declarações binárias. Além disso, sua especificação não impõe restrições rígidas a respeito das declarações que podem ser feitas, o que permite a construção de declarações sintaticamente válidas mas que contêm falhas do ponto de vista semântico: (<Brasília> <temCapital> <Brasil>). Isso ocorre porque na especificação RDF tudo é considerado recurso e estes, por sua vez, não possuem tipagem.

RDF Schema

[36] http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/

O RDF Schema (Brickley, et al., 2004) surgiu com o objetivo de permitir ao RDF superar em parte as suas limitações. Com o RDFS, podemos estabelecer primitivas que permitem a definição de vocabulários específicos de domínio, possibilitando a tipagem de recursos e das propriedades especificadas, viabilizando, assim a validação das declarações contidas no grafo RDF e a prevenção contra declarações semanticamente incorretas.

Podemos considerar o RDFS como uma linguagem de ontologia simples para a definição de vocabulários específicos de domínio usado com RDF. Através dela podemos elaborar vocabulários para comunidades específicas.

No RDFS descrevemos os recursos na forma de hierarquias de classes e propriedades. Através das classes, que seguem o mesmo conceito das classes da orientação a objetos, o RDFS nos permite especificar quais as propriedades que descrevem um determinado recurso.

Ela nos permite definir classes, propriedades, hierarquias de classes e de propriedades, bem como restrições de propriedades (domínio e imagem).

O mecanismo de tipagem adotado pelo RDFS é bem similar ao do XMLS, fornecendo, porém, informações sobre a interpretação das declarações contidas no modelo de dados RDF, sem impor uma ordem ou combinação de tags como faz o XMLS.

Em RDFS usamos as classes para definir um sistema de tipos para os recursos RDF. Nele, os recursos *rdfs:Resource*, *rdf:Property* e *rdfs:Class* são classes. Além das classes, o RDFS define também os seguintes predicados reservados: *rdf:type* e *rdfs:subClassOf*. Usamos estes predicados para definir, respectivamente, a classe à qual um recurso pertence e o relacionamento de subclasse entre dois tipos.

Como vimos, tudo aquilo que pode ser descrito por expressões RDF chamamos de recurso. Todo recurso é considerado uma instância da classe *rdfs:Resource*. O subconjunto de recursos que são propriedades são representados pela classe *rdf:Property*. Quando um esquema define uma nova classe, o recurso que representa aquela classe deve ter uma propriedade *rdf:type* cujo valor seja o recurso *rdfs:Class*. Os recursos RDF podem ser definidos como instâncias de uma ou mais classes usando a propriedade *rdfs:type*.

As hierarquias de classes podem ser definidas através dos predicados *rdfs:Class* e *rdfs:subClassOf*. Da mesma forma, as hierarquias de propriedade são definidas através dos predicados *rdf:Property* e *rdfs:subPropertyOf*.

Além de hierarquias, em RDFS podemos definir restrições nos tipos de valores usados com as propriedades. Os predicados *rdfs:range* e *rdfs:domain* são usados para este fim. Através do rdfs:range definimos que o valor de determinada propriedade deve ser um recurso de determinada classe. Em outras palavras, definimos os valores que uma determinada propriedade pode assumir. Podemos ilustrar o uso desse predicado, definindo para a propriedade "temCapital" um domínio de valores restrito à classe "Cidade" O predicado *rdfs:domain*, por sua vez, identifica que certa propriedade pode ser usada com sujeitos de determinada classe. Assim podemos estabelecer que a propriedade "temCapital" pode ser usada apenas por sujeitos da classe País, através do uso deste predicado.

A semântica do RDFS é caracterizada através de triplas axiomáticas e regras de consequência lógica que inferem novas triplas, conforme apresentado a seguir:

1. Se um grafo RDF possui as triplas: (S, *rdfs:subClassOf*, C) e (S', *rdf:type*, S), então podemos inferir que (S', *rdf:type*, C) – garantia da pertinência de recursos em superclasses;
2. Se um grafo RDF possui as triplas (S, *rdfs:subClassOf*, C) e (C, *rdfs:subClassOf*, C') então podemos inferir que (S, *rdfs:subClassOf*, C') – transitividade do relacionamento de subclasses;
3. Se um grafo RDF possui as triplas (P, *rdfs:domain*, D) e (R, P, S) então podemos inferir que (R, *rdf:type*, D) – inferência do tipo de recurso através do domínio definido para uma propriedade;
4. Se um grafo RDF possui as triplas (P, *rdfs:range*, D) e (R, P, S) então podemos inferir que (S, *rdf:type*, D) – inferência do tipo de informação em relação à imagem da propriedade.

Em resumo, o RDFS é uma linguagem para descrever vocabulários para domínios específicos, isto é, trata-se de uma linguagem primitiva para descrição de ontologias. Ela provê um mecanismo para tipagem de recursos em grafos RDF e nos permite aplicar novas regras semânticas para a inferência de

novas triplas. Para os fins do presente trabalho, as linguagens RDF e RDFS são suficientes para traduzir em formato de triplas a semântica dos modelos dimensionais. Mais adiante veremos como fazer isso.

Linked Data

O termo *Linked Data* (Bizer, et al., 2009) pode ser definido como um conjunto de regras, princípios e melhores práticas para possibilitar a publicação e interligação de dados estruturados na Web.

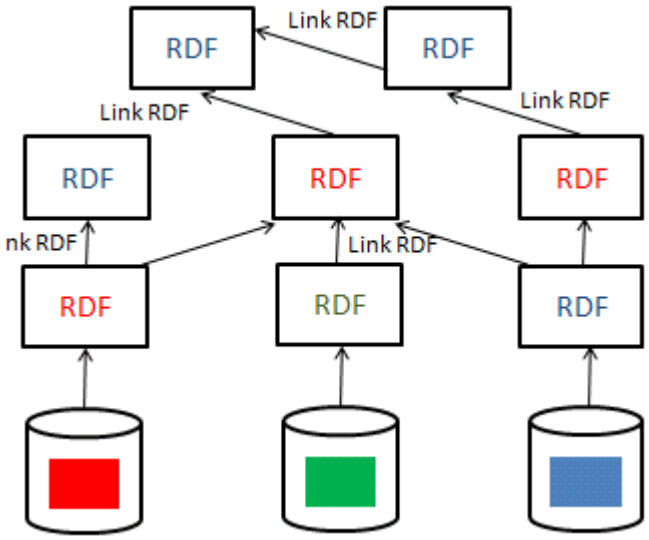


Figura 72: Ilustração de Linked Data na Web

Mas como estabelecemos essas ligações entre os dados? Existem quatro princípios básicos do Linked Data que enunciamos a seguir:

1. Usar URIs para nomear recursos;
2. Usar URIs HTTP para localizar os nomes dos recursos;
3. Quando alguém procurar por uma URI, providenciar, através dela, informações úteis através do emprego de predicados RDF;
4. Incluir sentenças RDF que estabeleçam ligações com outras URIs de forma que se possam descobrir mais recursos;

Esses princípios são tidos como um guia de como publicar, e conectar dados por meio da Web. Eles levam em consideração que o principal objetivo do *Linked Data* é permitir que as pessoas compartilhem dados estruturados na Web da mesma forma que compartilham documentos hoje em dia (Heath, et al., 2011).

Mas vejamos como essas ligações entre dados vão sendo estabelecidas através do uso dos predicados RDF e as declarações de triplas construídas com eles estabelecendo conexões entre esses dados através da sequência abaixo apresentada:

1. Temos uma pessoa chamada Richard Cyganiak[37] de cujo arquivo foaf[38] extraímos as seguintes triplas:

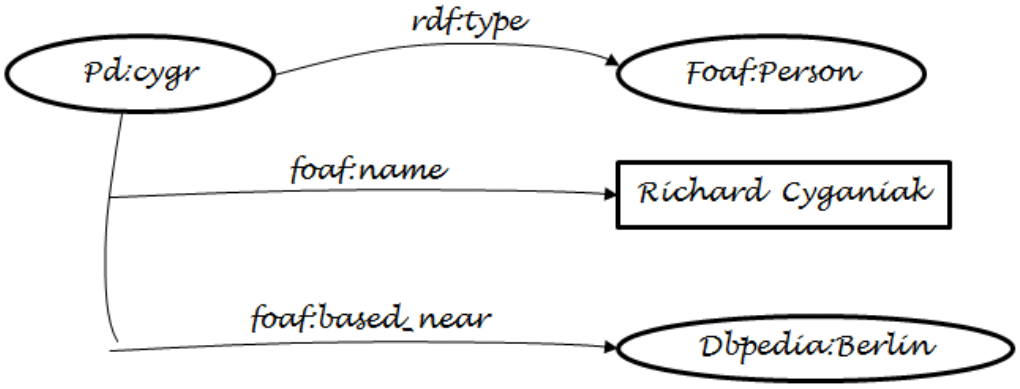


Figura 73: FOAF de Richard Cyganiak

2. Porém, na DBpedia, estão disponíveis diversas informações adicionais sobre a cidade de Berlin, onde se encontra Richard, dentre elas, a população de Berlin e o fato dela ser uma cidade alemã.

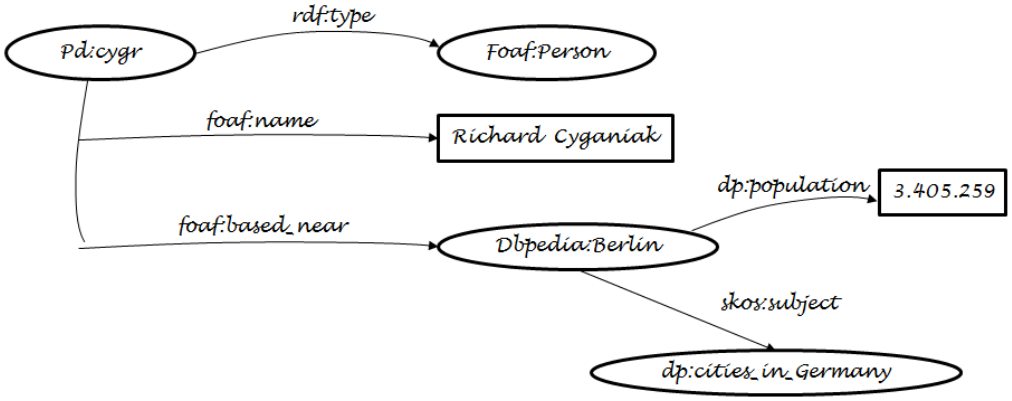


Figura 74: Ligando dados do FOAF com a DBpedia

3. Porém a Alemanha possui várias outras cidades como Hamburgo e Munique e assim por diante...

[37] http://richard.cyganiak.de/

[38] http://richard.cyganiak.de/foaf.rdf

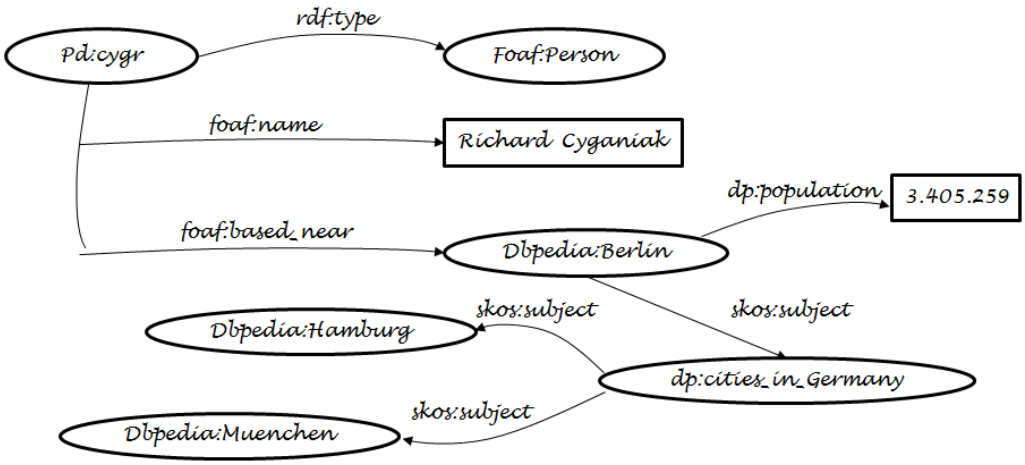


Figura 75: Incluindo novas ligações com a DBpedia

Através dos dados e metadados estabelecemos ligações ou deduzimos novas ligações entre eles utilizando as declarações criadas no formato de triplas. Essa rede de conexões que estabelecemos "passo a passo" acima, pode se estender quase que infinitamente desde que haja conteúdo suficiente para isso. E, na verdade, até já há, afinal a nuvem LOD não para de crescer.

Vocabulários Padrões

Usamos vocabulários para descrevermos os objetos do mundo através de classes criadas para representá-los, definindo suas propriedades e o relacionamento entre eles. Fazemos isso usando os termos oferecidos por eles para realizar essas descrições.

Existem dois sistemas de organização e representação da informação: as taxonomias e as ontologias (Vital, et al., 2011). A diferença entre eles é muito tênue sendo que muitas vezes eles se confundem.

Para (Martínez, et al., 2004) "a taxonomia, em um sentido amplo, é a criação da estrutura (ordem) e dos rótulos (nomes) que ajudam a localizar a informação relevante. Em um sentido mais específico, é o ordenamento e rotulação de metadados, que permite organizar sistematicamente a informação primária".

No campo da Web semântica, onde buscamos formas de aperfeiçoar o processo de recuperação da informação, construindo categorias em linguagens, que tenham sentido para o computador (Ramalho, et al., 2005), as ontologias, sob o aspecto estrutural, são definidas, de acordo com o W3C, como um

conjunto de definições legíveis por máquina que criam uma taxonomia de classes e subclasses e os relacionamentos entre elas.

O fato de as ontologias construírem subclasses baseadas nas relações taxonômicas pode ser uma das causas da falta de clareza na distinção entre estes termos. Porém, (Van Rees, 2003) esclarece que, "[...] uma vez que uma série de propriedades são adicionadas em uma estrutura hierárquica, o termo ontologia é mais adequado que taxonomia".

Os vocabulários são adotados por certas comunidades de acordo com o seu domínio e o suporte que seus termos oferecem para descrever e publicar os dados de interesse ou necessários para aquela comunidade. Assim existe uma considerável oferta de vocabulários especializados que são utilizados para construir a Web de dados. A reutilização, sempre que possível, de termos de vocabulários conhecidos no padrão RDF é considerada uma boa prática segundo (Heath, et al., 2011).

Maximizar o uso de vocabulários padrões é, portanto, recomendável e desejável. Isto porque essa prática aumenta a possibilidade de que os dados descritos através destes vocabulários sejam consumidos por aplicações que reconheçam e sejam capazes de processar seus termos.

Listamos, abaixo os principais vocabulários que utilizaremos para descrever os cubos de dados, ou modelos dimensionais. O reuso de tais vocabulários, sempre que possível, é recomendado para garantir a interoperabilidade (Bizer, et al., 2007).

- *R2RML*[39] – é uma linguagem de mapeamento de bancos de dados relacionais para RDF;
- *Data Cube Vocabulary*[40] - é uma linguagem para descrever dados estatísticos multidimensionais e seus metadados;
- *SKOS*[41] - é uma linguagem voltada para representação de esquemas conceituais, simples, extensível e compreensível por máquinas.

[39] http://www.w3.org/TR/r2rml/

[40] http://www.w3.org/TR/vocab-data-cube/

[41] http://www.w3.org/TR/skos-primer/

Corroborando a recomendação de reuso, a criação de novos vocabulários deve ser evitada ao máximo e só empregada quando os termos desejados não possam ser representados por nenhum dos vocabulários existentes (Bizer, et al., 2007).

Nas seções seguintes descreveremos estes dois vocabulários que serão fundamentais para o presente trabalho.

R2RML

É uma linguagem utilizada para realizar mapeamentos de bancos de dados relacionais para RDF. Tais mapeamentos possibilitam visualizar os dados relacionais no modelo de dados RDF, expressos na estrutura e vocabulário escolhidos pelo autor do mapeamento.

Cada mapeamento R2RML é adaptado para um esquema de banco de dados específico e para um vocabulário alvo. A entrada para um mapeamento R2RML é um banco de dados relacional que está de acordo com esse esquema. A saída é um *dataset* RDF, conforme definido no SPARQL (Kendall, et al., 2008), que usa predicados e tipos do vocabulário alvo. O mapeamento é conceitual; processadores R2RML são livres para materializar os dados de saída, ou para disponibilizá-los de forma virtual através de uma interface que consulta o banco de dados de origem, ou para oferecer qualquer outra forma de acesso ao *dataset* RDF de saída.

Um R2RML *mapping* define um mapeamento a partir de um banco de dados relacional para um *dataset* RDF. Esse mapeamento é representado como um grafo RDF. Em outras palavras, RDF é usado não apenas como o modelo de dados alvo do mapeamento, mas também como um formalismo para representar o mapeamento R2RML. Um documento de mapeamento R2RML deve ser escrito na sintaxe *Turtle* (tartaruga) RDF.

Um mapeamento R2RML consiste em uma ou mais estruturas chamadas *TriplesMaps*. Cada *TriplesMap* contém uma referência a uma tabela lógica no banco de dados de entrada. A tabela lógica pode ser um dos itens seguintes:

1. A tabela base que existe no esquema SQL de entrada.
2. Uma visão que existe no esquema SQL de entrada.
3. Uma consulta SQL válida no esquema de entrada.

Além disso, um *TriplesMap* contém as regras para o mapeamento de uma linha da tabela lógica para um conjunto de triplas RDF. Ele consiste de uma estrutura *SubjectMap* e uma ou mais estruturas *PredicateObjectMap*. Todas as triplas RDF geradas a partir de uma mesma linha compartilham o mesmo Sujeito. A estrutura *SubjectMap* em um *TriplesMap* contém as regras para gerar o Sujeito para uma linha.

Cada estrutura *PredicateObjectMap* em um *TriplesMap* contém as regras para a geração de um par (objeto, predicado) a partir dos valores na linha da tabela. Ela consiste de uma estrutura *PredicateMap* e de uma estrutura *ObjectMap*. A estrutura *PredicateMap* contém as regras para gerar o predicado e a estrutura *ObjectMap* contém as regras para gerar o objeto.

Um *TriplesMap* é usado para gerar triplas RDF a partir de uma linha na tabela lógica, combinando o único sujeito da linha, gerados usando o *SubjectMap*, com o par (objeto, predicado) gerado a partir da linha usando o *PredicateObjectMap*.

A saída de um mapeamento R2RML é um dataset RDF. As triplas RDF no dataset são o resultado da aplicação das regras de mapeamento de cada *TriplesMap* para as linhas de sua tabela lógica.