UNIVERSIDADE UNIGRANRIO
ESCOLA DE CIÊNCIA E TECNOLOGIA
Sistemas de Informação

Jarf Nunes dos Santos
Kim Silva do Prado

Um Protótipo de Sistema de Apoio à Decisão baseado em Serviços de Grades

Duque de Caxias
2009

UNIVERSIDADE UNIGRANRIO
ESCOLA DE CIÊNCIA E TECNOLOGIA
Sistemas de Informação

Jarf Nunes dos Santos
Kim Silva do Prado

Um Protótipo de Sistema de Apoio à Decisão baseado em Serviços de Grades

Projeto Final de Curso apresentado como requisito final para conclusão do curso de Sistemas de Informação da Universidade do Grande Rio “Prof. José de Souza Herdy” (UNIGRANRIO).

Orientador: Profº. Alexandre de Assis Bento Lima

Duque de Caxias
2009

Um Protótipo de Sistema de Apoio à Decisão baseado em Serviços de Grades

Jarf Nunes dos Santos
Kim Silva do Prado

Projeto Final de Curso apresentado como requisito final para conclusão do curso de Sistemas de Informação da Universidade do Grande Rio “Prof. José de Souza Herdy” (UNIGRANRIO).

Aprovado em dezembro de 2009.

BANCA EXAMINADORA

_____

Profº. Dr. Alexandre de Assis Bento Lima - Orientador
UNIGRANRIO

_____

Profº. Especialista – Gilliard da Silva Alves
UNIGRANRIO

_____

Profº. Mestre – Humberto José Vieira Junior
UNIGRANRIO

Duque de Caxias
2009

Jarn Nunes dos Santos, Kim Silva do Prado

Um Protótipo de Sistema de Apoio à Decisão baseado em Serviços de Grades –
Duque de Caxias, 2009
xii, 58 p. 29,7cm. (Escola de Engenharia e Computação, 2009)

Projeto de Final de Curso - Universidade do Grande Rio, Escola de Engenharia e Computação.

1. Business Intelligence
2. Grades Computacionais
3. OLAP

Dedicamos nosso trabalho as pessoas mais importantes de nossas vidas, pais, familiares e amigos que foram essenciais por depositarem total confiança em nossa caminhada e nos incentivaram chegar até aqui.

AGRADECIMENTOS

Agradecemos primeiramente a Deus por nos dar mais essa conquista em nossas vidas, aonosso amigo orientador Profº. Dr. Alexandre de Assis Bento Lima que sempre esteve pronto a nos ajudar e passar o conhecimento necessário para alcançarmos nosso objetivo e a todos que estiveram próximo dando força e acreditando em nosso potencial.

“Feliz o homem que achou sabedoria e o homem que obtém discernimento, porque tê-la por ganho é melhor do que ter por ganho a prata, e te-la como produto [é melhor] do que o próprio ouro.”

Provérbios 3:13, 14. Bíblia Sagrada

RESUMO

Muitas organizações se encontram espelhadas fisicamente possuindo filiais. O desafio dos tomadores de decisões estratégicas é acessar e extrair informações desses dados dispersos. Com intuito de facilitar o acesso a dados dispersos geograficamente, faremos uso da tecnologia de computação em grades, que oferece um ambiente seguro dentre outros atrativos. Grades computacionais (FOSTER, 2005) possuem serviços que atuam como serviços web. Para dar suporte a leigos em informática há ferramentas que facilitam a elaboração de relatórios. Propomos um protótipo que implementa serviços que compõem uma nova arquitetura de acesso a fontes de dados de apoio à decisão. Este protótipo possui um cliente com interface amigável e acessível por meio de navegadores de internet. A ferramenta é habilitada para atuar em grades e, portanto um serviço de grade. O objetivo deste trabalho é apresentar a arquitetura e modelagem deste protótipo.

Palavras-chave: Business Intelligence, Grades Computacionais, DRS.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

AJAX	Asynchronous JavaScript+XML
BI	Business Intelligence
DHTML	Dynamic HTML
DRS	Data Report Service
DW	Data Warehouse
GUI	Guide User Interface
HTTP	Extensible Markup Language
HTTPS	Extensible Markup Language
JSF	Java Server Faces
JSP	Java Server Pages
OGSA	Open Grid Services Architecture
OLAP	Online Analitical Processing
OLTP	Online Tansacional Processing
RPC	Remote Procedure Call
SAD	Sistema de Apoio à Decisão
SOAP	Simple Object Access Protocol
SQL	Select Query Language
UDDI	Universal Description Discovery and Integration
W3C	World Wide Web Consortium
WS	Web Services
WSDL	Web Services Description Language
WSRF	Web Services Resource Framework
XML	Extensible Markup Language

1 - Introdução

Ferramentas de apoio à decisão como as OLAP se caracterizam por permitir que usuários leigos possam fazer análises e gerar relatórios sobre fontes de dados disponíveis. Em Business Intelligence é comum a utilização do mapeamento dimensional que identifica bancos de dados como Cubos de Dados. Sobre a forma como os dados são acessados, importante é o fato de não se conhecer previamente as condições ou quais objetos serão relevantes à consulta. Estes tipos de consultas são chamadas de *ad-hoc*. Nosso protótipo implementa uma arquitetura que possui componentes em forma de serviços de grade, que cooperam para oferecerem funcionalidades. Os outros serviços são responsáveis por disponibilizar e manter um registro de fontes de dados passíveis de serem acessadas. O Serviço DRS não gera GUI, mas é um serviço capaz de processar as condições relevantes para uma consulta e criar a instrução SQL correspondente que será submetida, por meio de outros serviços, ao cubo correspondente. Foi criada uma aplicação GUI que permite interagir com os serviços para criar consultas *ad-hoc*. Esta aplicação é um conjunto de páginas internet geradas com JSP e que utiliza DHTML para ser mais eficiente.

Nos itens seguintes do nosso trabalho apresentamos o desenvolvimento que trata de temas relacionados a Business Intelligence e grades computacionais, e análise do protótipo de SAD baseado em grades computacionais.

1.1 - Objetivos

Nosso objetivo é apresentar um protótipo de Sistema de Apoio à Decisão baseado em Grades Computacionais. Mostraremos sua arquitetura e modelagem, além de seu cliente que possui uma GUI Web intuitiva e acessível por navegadores internet.

1.2 - Justificativa

Usamos grades computacionais no desenvolvimento da aplicação, pois estas fornecem acesso seguro a dados e outros recursos geograficamente dispersos. O ambiente de grades não possuía uma proposta de arquitetura e ferramenta de acesso a dados, voltada para SADs. Com nosso projeto pretendemos mudar este cenário, ao implementar serviços de uma nova arquitetura de acesso dados de apoio à decisão voltada para grades computacionais.

1.3 - Metodologia utilizada

Este trabalho foi baseado em atividades tratadas num projeto anterior de iniciação científica. Além disso, fizemos pesquisas em artigos de congressos e livros para apresentar os diversos conceitos e definições relacionadas com o nosso tema. Utilizamos um extenso tutorial(Globus Programers Tutorial) fornecido pelo grupo que desenvolve o software Globus Tookit, que implementa as funcionalidades de grades computacionais utilizada no projeto. Além destes, alguns sites web reconhecidos foram usados na obtenção de conhecimento técnico para desenvolvimento(w3schools).

2 - Data Warehouse e Modelagem Dimensional

2.1 - Data Warehouse

2.1.1 - Conceito

Grandes empresas possuem banco de dados com enorme quantidade de dados ali armazenados e que são de grande importância para as mesmas. Mas tais bancos de dados convencionais encontram dificuldades para a realização de análise dos dados neles existentes pelo fato de existirem vários sistemas não interligados entre si. Desta forma, as empresas ficam impossibilitadas de identificar as tendências e analisar comportamentos e padrões. Sendo assim, não se é possível tomar decisões sobre o histórico de dados.

Mediante essa necessidade, surgiu o Data Warehouse (Armazém de dados), que tem como função principal o armazenamento de grande quantidade de informações em um banco de dados referente a uma ou mais atividades de uma empresa de forma consolidada, voltada à tomada de decisão.

Segundo INMON (1997), “Data Warehouse (DW) é uma coleção de dados. Orientados a assuntos, integrados, variáveis com o tempo e não voláteis para suporte ao processo gerencial de decisão”.

Os dados do data warehouse são provenientes de sistemas transacionais, legados ou até mesmo de fora dos domínios da organização. Duas ou mais fontes de dados podem estar integradas no data warehouse. São preservados dados históricos, diferente do que ocorre em sistemas transacionais, e com diferentes níveis de granularidade possibilitando análises de comportamentos impossíveis de serem detectados em sistemas OLTP. Em um data warehouse os dados são organizados por áreas de interesse ou assunto, conforme a modelagem dimensional.

2.1.2 - Características

Orientado a assunto

Significa que os dados estão todos organizados de acordo com os interesses da empresa.

Ex.: Em uma empresa de telecomunicações, o assunto de principal interesse é cliente, podendo o mesmo ser residencial, empresarial, público e etc.

Integrado

Pelo fato de ter consistência do domínio das variáveis, de nomes, entre outros, na medida em que dados originários de diferentes fontes são transformados em um estado uniforme. Exemplo muito utilizado para ilustrar esse fato de integração é o elemento de dado sexo, que pode ser encontrado em algumas aplicações codificadas como H/M, em outras como M/F e também como 0/1. Desta forma, no momento em que os dados são levados ao Data Warehouse, eles são convertidos para um estado uniforme. Dessa mesma forma, pode ocorrer com outros tipos de elementos de dados com medidas diferentes. Ex.: milímetro, centímetro, metro... (Onde serão modificados para uma única medida).

Variante no tempo

Os dados são armazenados no Data Warehouse para fornecer informações de uma perspectiva histórica. Desta forma, a cada mudança feita em um dado, uma nova entrada é criada e não atualizada como é normal nos sistemas tradicionais.

Não-Volátil

Significa que os dados não sofrem atualizações, os mesmos são carregados uma única vez e a partir desse momento só podem ser consultados.

2.2 - Modelagem Dimensional

2.2.1 - Conceito

Modelagem dimensional é uma técnica de modelagem desenvolvida especialmente para a implementação de um modelo de dados que permita a visualização de dados de forma intuitiva e com alto índice de aproveitamento na extração de dados (MORALES, 2009).

Este modelo nos fornece uma representação do banco de dados consistente com o modo como o usuário realiza a visualização e faz a navegação pelo DW, combinando tabelas com dados históricos em séries temporais, onde o contexto é descrito através de tabelas de dimensões.

O modelo dimensional é baseado em três componentes: FATOS, DIMENSÕES e MEDIDAS.

- FATO é a representação de um item, transação ou evento de negócio, sendo utilizado para análise. Também tudo aquilo que reflete a evolução do dia-a-dia.
- DIMENSÃO é a possível forma de visualizar os dados. Exemplo: Por país, por produto, por cliente, por mês, por ano, etc.
- MEDIDA é o atributo que fornece quantidade a um fato, representando a performance de um indicador em relação às dimensões que participam do fato.

Cada modelo dimensional é composto por uma tabela, chamada tabela fato, e um conjunto de outras, chamadas tabelas dimensão. Cada tabela dimensão possui uma chave primária, que atua como chave estrangeira, para o relacionamento com a tabela de fato.

Desta forma, possuímos o modelo estrela que é a estrutura básica de um modelo de dados multidimensional. De forma que a tabela fato possui correspondentes às dimensões, sempre expressando uma relação muitos-para-muitos. As tabelas de fato mais úteis possuem uma ou mais medidas numéricas, ou "fatos", que ocorrem para a combinação de dados que definem cada registro.

Na figura 2.1 temos um modelo dimensional estrela Vendas. A tabela fato Vendas é o elemento central, tendo as tabelas Dimensão Tempo, Produto e Loja, associadas a ela. Este esquema é superior ao modelo flocos de neve, pois facilita o entendimento do modelo

e diminui a quantidade de junções nas consultas. Mesmo sendo normalizado até a Terceira Forma Normal, um o modelo flocos de neve não apresenta vantagens muito grandes comparado ao esquema estrela, no que diz respeito ao uso de espaço do banco de dados.

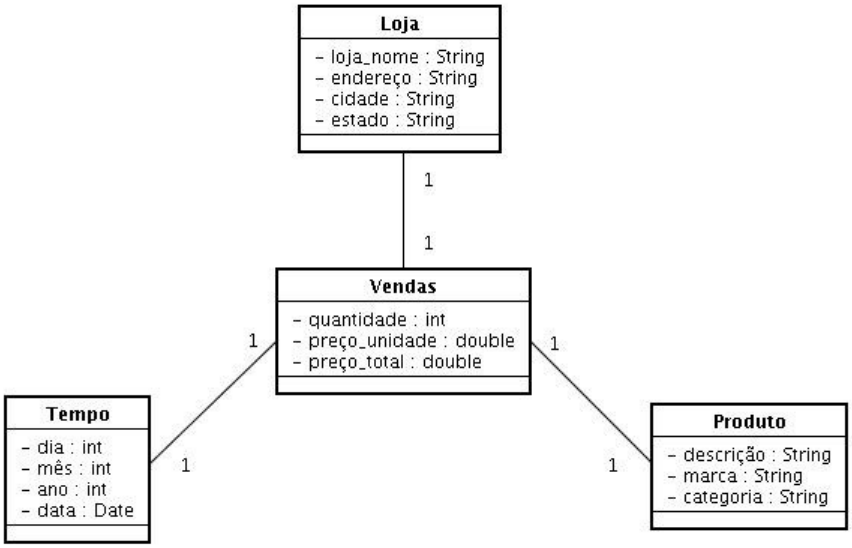


Figura 2.1 - Modelo dimensional Estrela do esquema Vendas

Os fatos mais usados nas tabelas fato são numéricos e aditivos. Aditividade é crucial porque as aplicações de data warehouse quase nunca recuperam um único registro da tabela de fato, mas sim centenas, milhares ou mesmo milhões desses registros em um momento e a única coisa útil para fazer com tantos registros é adicioná-los de forma crescente.

As tabelas de dimensão, pelo contrário, muitas vezes contêm informações textuais descritivos. Os atributos de dimensão são usados como fonte da maior parte das restrições interessante em consultas de data warehouse.

2.2.2 - Granularidade

Granularidade é o nível de detalhamento específico utilizado para armazenar os dados no DW. Quanto maior o detalhamento, mais baixo o nível de granularidade.

Exemplo granularidade baixa (ALVES, 2009):

Ano	(2009)
Semestre	(2)
Mês	(11)
Semana	(46)
Dia	(9)

Características de granularidade alta e baixa:

Alta

- Economia de espaço em disco
- Redução na capacidade de atender consultas

Baixa

- Grande quantidade de espaço em disco
- Aumento na capacidade de resposta de qualquer questão

3 - Grades Computacionais

3.1 - Definições

O nome grades computacionais (grid computing) foi adotado pela semelhança com a rede elétrica que no inglês significa grid, tendo em vista que a rede elétrica encontra-se distribuída por toda casa ou outros locais que vamos e podemos ter acesso a mesma. Para isso, basta que tenhamos uma tomada para conectar um aparelho elétrico e utilizar a energia que se encontra ali, na rede, a todo o momento.

Desta mesma forma são as Grades Computacionais, que são redes computacionais capazes de alcançar uma alta taxa de processamento dividindo as tarefas entre diversas máquinas (que formam uma Organização Virtual), podendo ser local, estadual, nacional ou mundial. Esses processamentos são executados no momento em que as máquinas não estão sendo utilizadas por seus usuários, assim é evitado o desperdício de processamento da máquina utilizada.

Podemos citar como exemplo, duas empresas sediadas em países com fuso-horário bem diferentes como Brasil e Japão que poderiam formar um grid (para isso, é necessário que estejam em uma mesma organização virtual), combinando seus servidores, de maneira que uma possa utilizar os benefícios de processamento das máquinas da outra enquanto a máquina não está sendo utilizada. Assim, quando a empresa do Brasil está em “horário de serviço”, a empresa do Japão está “fechada” e vice-versa, desta forma, a empresa que se encontra em “horário de serviço”, pode utilizar as máquinas da outra para realizar processamentos nos quais apenas suas máquinas não dariam conta de realizar.

Organização Virtual

O real problema específico e que subjaz ao conceito de grade é coordenar o compartilhamento de recursos e resolução de problemas em dinâmica, multi-institucional das organizações virtuais (FOSTER, 2003). A partilha não é primariamente de intercâmbio de arquivos, mas sim o acesso direto a computadores para processamentos, software, dados e outros recursos, como é exigido por uma gama de resolução colaborativa de problemas e de recursos estratégicos de intermediações emergentes na indústria, ciência e engenharia.

Este compartilhamento é necessára e altamente controlado, com fornecedores de recursos e consumidores definidos claramente e cuidadosamente. São definidos o que é compartilhado, quem tem permissão para partes, e as condições em que ocorre a partilha. Esse conjunto de indivíduos e/ou instituições definidos pelas regras da partilha de tal forma que é chamado de Organização Virtual.

3.2 - Serviços Web X Serviços de Grade

Ao lado da WEB 2.0, serviços Web são uma das tecnologias mais comentadas pelos desenvolvedores nos últimos tempos. Apesar de o nome apontar para algo genérico, representam uma nova maneira de criar aplicações distribuídas. Um Web service é uma aplicação de software que pode ser acessada por clientes remotos usando diferentes linguagens baseadas em XML (POTTS; KOPACK, 2003). Normalmente para se identificar um Web Service é usada uma URL, como fazemos para localizar um site Web. A diferença entre um site Web e um Web service está no tipo de informação que podem fornecer. Um web site pode oferecer conteúdos que são próprios para serem exibidos em um navegador de internet, de forma que podemos dizer que seus clientes são pessoas. Os Web Services são aplicações de software projetadas para oferecerem informações para outras aplicações. Seus clientes são aplicativos.

Os clientes de Web services enviam requisições em documentos XML num formato especial de acordo com as regras da especificação SOAP. Uma mensagem SOAP pode conter uma chamada a um método e quaisquer parâmetros necessários.

A grande vantagem dos serviços Web é permitir a interoperabilidade de qualquer plataforma de hardware e software capaz de enviar e receber arquivos contendo texto em ASCII.Ou seja, qualquer plataforma de hardware ou software pode ser um servidor ou cliente em um contexto de serviços Web. Independentemente da linguagem, sistema operacional ou distância entre cliente e servidor, eles podem se comunicar.

Esta flexibilidade despertou o interesse de grandes fornecedores de ferramentas de desenvolvimento, como Microsoft, IBM e Sun Microsystens. Eles se aliaram a consórcios de padronização, como o W3C, para publicar as especificações que dão suporte aos serviços Web.

Dentre os padrões que dão suporte aos serviços Web os mais importantes são:

SOAP

O SOAP é uma gramática da Tecnologia XML, ou seja, um conjunto de regras definidas que compõem um documento no formato XML. Seu objetivo é descrever um formato de mensagens que não esteja vinculado a qualquer arquitetura de hardware ou software, mas que possibilite máquinas de qualquer plataforma enviar mensagens para qualquer outra plataforma. O corpo de uma mensagem SOAP pode conter uma Remote Procedure Call (RPC). Uma RPC é uma chamada de método que deve ser executada no computador do Web Service. Uma RPC é similar a uma chamada de método comum numa linguagem comum, mas com o diferencial de ocorrer por meio de uma rede.

XML

A XML é uma linguagem de marcação que permite a auto-descrição de documentos que a empregam. SOAP, WSDL e UDDI são especificações que empregam XML na elaboração de seus documentos.

HTTP

O HTTP é usado como meio de transportar as mensagens SOAP e os arquivos WSDL que são trocados entre servidor e clientes. Entretanto este protocolo maduro não é a única opção de transporte. O SOAP permite que outros protocolos também sejam usados, como FTP,SMTP e JMS. Essas implementações, embora possíveis, são menos comuns do que com o HTTP.

WSDL

Descreve a interface do serviço e como ele pode ser acessado. Esta interface é composta dos métodos expostos pelo serviço Web para que eventuais clientes possam executar RPCs. Esta interface é escrita de forma genérica, independente da linguagem que efetivamente o serviço está implementado. Isto também é um fator que possibilita portabilidade.

UDDI

A especificação define como um possível cliente pode obter informações de forma a escolher um serviço Web a se conectar. Partes destas informações são obtidas com o download do WSDL do serviço que se deseja acessar. Os registros UDDI podem ser

públicos, privados ou semi-privados. Esses registros possuem informações sobre serviços Web e seus respectivos arquivos WSDL.

Por padrão serviços Web trabalham sem a definição de uma forma segura de criptografia definida pelo SOAP. De modo que necessita de outros meios para tornar os dados sigilosos onde ocorrem transações que precisam de confidencialidade.

A diferença dos serviços Web para os serviço de Grade é a possibilidade de guardar o estado de um objeto. Os serviços de grade implementam uma especialização de serviços Web chamado Web Service Resource Framework. A WSRF define que os serviços podem ser *statefull*, ou seja, clientes podem manipular valores de objetos que estão em outra máquina de forma transparente.

Os serviços de grade proporcionam mais de uma maneira de tornar as transações seguras. Como o uso do protocolo HTTPS ou criptografia do corpo de mensagens SOAP com arquitetura de Chave Privada.

4 -Descrição da solução

4.1 -Arquitetura

A arquitetura em que se baseia a solução de SAD aqui descrita é a AGRADE (Arquitetura baseada em Grades Computacionais para Apoio a Decisões Estratégicas). Proposta pelo Professor Alexandre de Assis Bento Lima, a arquitetura possui o Serviço de Cubo (SC), e o Registro de Cubos (RC). A primeira versão da AGRADE está definida na figura 4.1, abaixo.

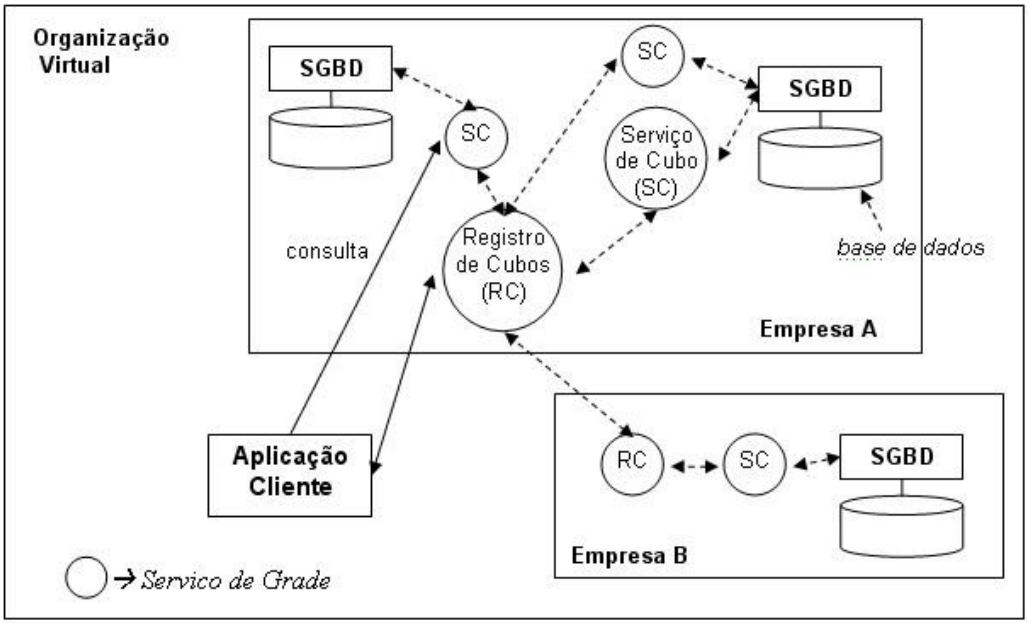


Figura 4.1 - AGRADE - Primeira Versão

A figura 4.1 apresenta as fontes de dados armazenadas em bases de dados controladas por Sistemas de Gerência de Banco de Dados (SGBD). Estas fontes de dados comumente estão organizadas de forma dimensional e compõem o Armazém de Dados (Data Warehouse) de uma empresa. Quando estão organizadas desta forma, as fontes de dados são chamadas de “cubos de dados”. Cubos de Dados são a principal fonte de alimentação de SADs. Um cubo contém dados a respeito de um assunto específico. As bases de dados podem possuir um ou mais cubos. Cada “Serviço de Cubo” (SC) pode representar um ou mais cubos de dados. Embora seja ideal que a base de dados esteja

organizada de forma dimensional, isto não é obrigatório, sendo possível acessar fontes que possuem outros modelos de dados, como o relacional. Esta abstração é uma importante característica do Serviço de Cubo.

Cada cubo mapeado no Serviço de Cubo (SC) deve ser registrado no Registro de Cubos (RC), como indica a figura 4.2. O Registro de Cubos opera como um catálogo, com referências para cada cubo de cada Serviço de Cubo. Na primeira versão da arquitetura, foi proposto que “o RC de uma empresa pode se registrar junto ao RC de outra, possibilitando a integração de cubos entre empresas”. Entretanto, em nossa implementação da pseudo arquitetura contamos com a possibilidade de haver somente um Registro de Cubos para todos os Serviços de Cubo.

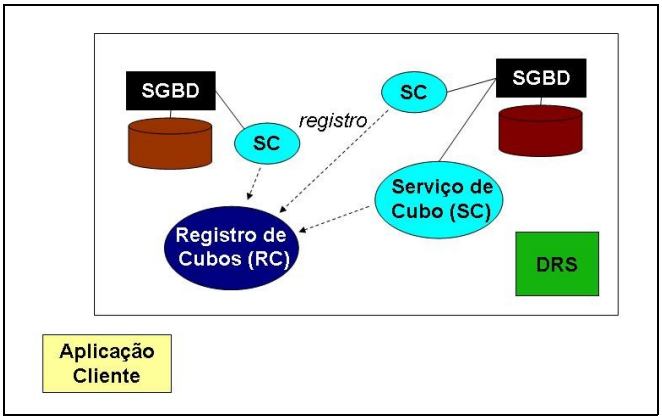


Figura 4.2 – Rigistro de cubos no RC

Quando uma aplicação cliente deseja acessar uma fonte de dados ela deve primeiramente consultar o catálogo de cubos disponíveis (figura 4.3). A listagem de cubos ativos é obtida no Registro de Cubos (figura 4.4). A aplicação seleciona um cubo disponível (figura 4.5) e recebe uma referência deste (figura 4.6), o que lhe permitirá submeter consultas (figura 4.7) diretamente sobre o mesmo.

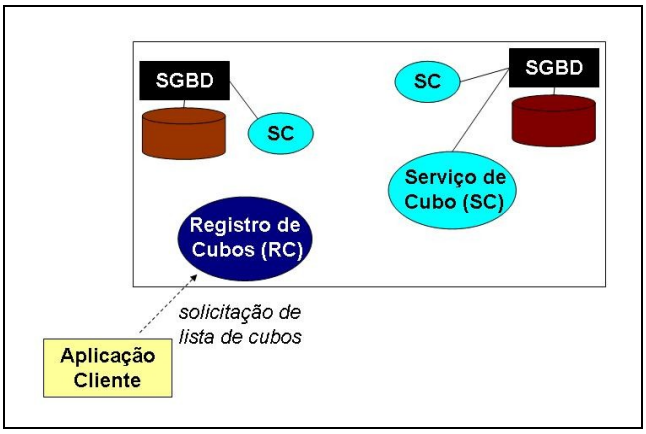


Figura 4.3 – Cliente solicita lista de cubos ao RC

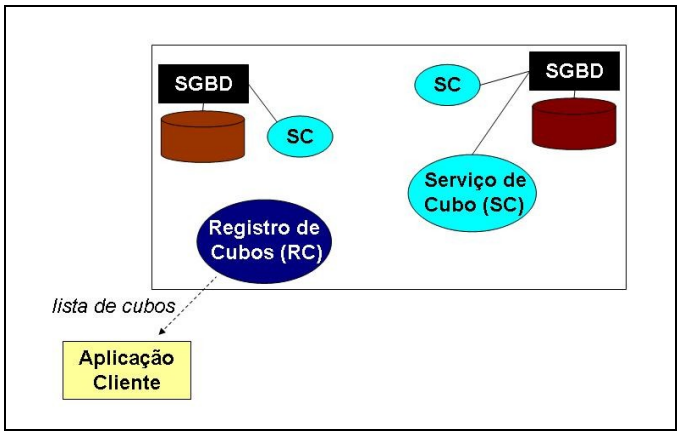


Figura 4.4 – Registro de Cubos envia lista para Aplicação Cliente

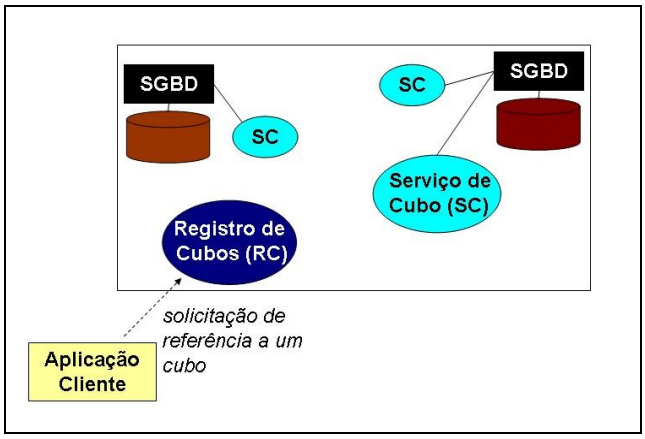


Figura 4.5 – Solicitação de referência a determinado cubo

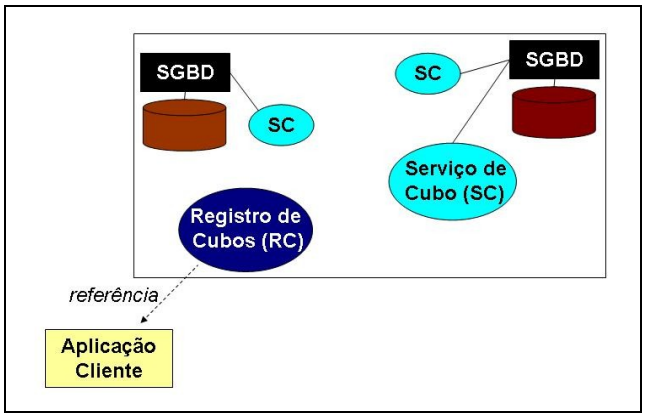


Figura 4.6 – Cliente obtém referência a um cubo no RC

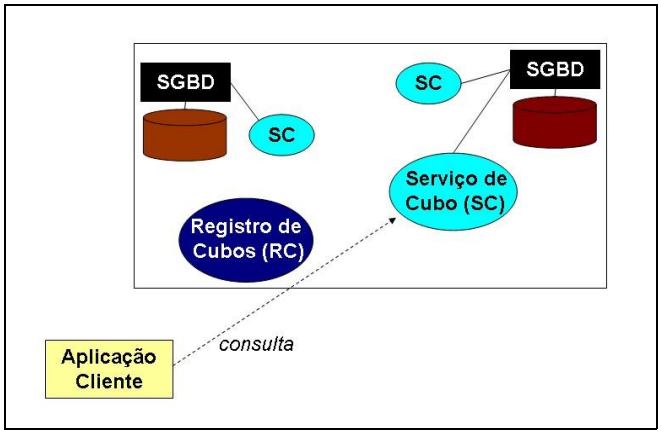


Figura 4.7 – Solicitação de execução de consulta

O Data report Service (DRS) é o serviço responsável por criar consultas dinâmicas do tipo ad-hoc, baseado em parâmetros fornecidos por um usuário leigo, através de um cliente (figura 4.8). O usuário informa ao serviço DRS sobre qual cubo deseja efetuar a consulta passando a referência obtida no RC. O DRS passa a se conectar ao SC para receber os metadados do cubo em questão (figura 4.9). Usando as condições estabelecidas pelo usuário e de posse dos metadados disponíveis, o DRS produz dinamicamente uma consulta no formato SQL (figura 4.10) que pode ser interpretada por todos os SGBDs relacionais.

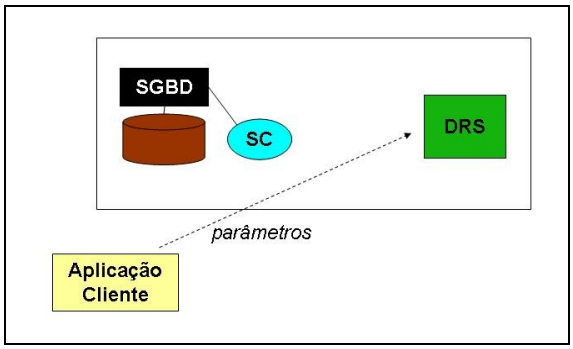


Figura 4.8 – Envio de parâmetros ao DRS por meio de um cliente

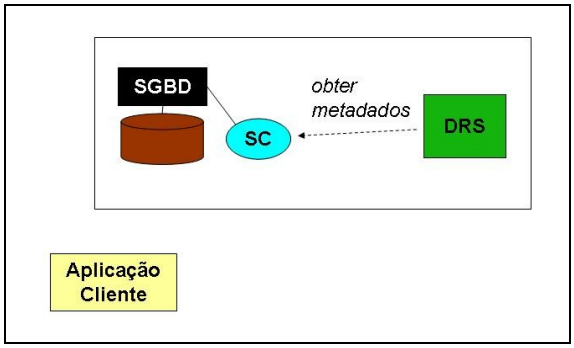


Figura 4.9 – DRS obtém metadados do cubo selecionado

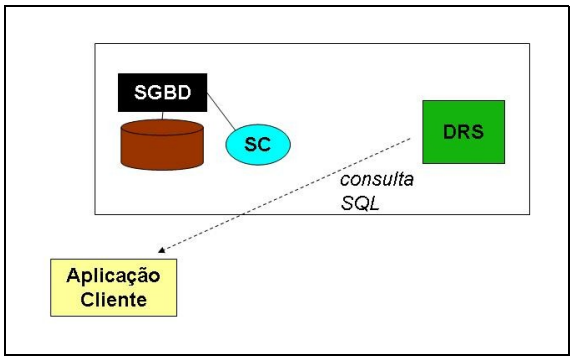


Figura 4.10 – Cliente obtém consulta no formato SQL

A Aplicação cliente por fim obtém do serviço DRS a consulta ah-doc e a submete ao cubo (figura 4.7). O cubo, por sua vez, que se encontra num determinado SC é responsável por submeter a consulta ao SGBD que manipula a base de dados correspondente. O resultado é enviado então ao cliente que solicitou a execução da consulta (figura 4.11).

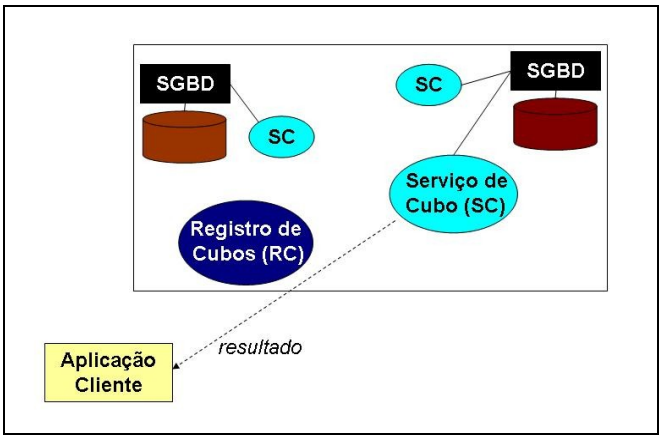


Figura 4.11 – Resultado enviado para cliente que solicitou execução de consulta

4.1.1 - Serviço de Cubo (SC)

O Serviço de Cubo tem papel chave na arquitetura. Ele supre os outros serviços e clientes com metadados das bases de dados que podem ser acessadas, ou seja, seus cubos ou assuntos. Oferece o único meio de efetivamente submeter consultas a uma fonte de dados e obter um resultado, pois somente o cubo possui a configuração necessária para se comunicar com o SGBD correspondente.

O SC possui um ou mais cubos que fornecem informações sobre seus respectivos bancos de dados. O Serviço de Cubo mantém informações tais como as tabelas disponíveis e seus respectivos atributos e relacionamentos. As informações sobre os atributos incluem, por exemplo, o tipo, tamanho e precisão. Sobre as ligações são informados os atributos que são necessários para se estabelecer um relacionamento entre duas relações. Como dito, a arquitetura está proposta para atender aos requisitos de SAD, o que torna importante a compatibilidade com o modelo dimensional de dados. Isto determina que cada cubo ou assunto possua apenas uma tabela de fatos e as demais são chamadas de dimensões. Desta forma o SC informa se cada tabela, é caracterizada como fato ou dimensão.

O Serviço de Cubo (SC) utiliza um banco de dados para manter os metadados dos cubos que suporta. Ao entrar em operação, o SC carrega em memória cada cubo e seus respectivos metadados.

Quando uma aplicação cliente deseja efetuar operações de pesquisa sobre uma determinada base de dados, estas têm de ser intermediadas pelo Serviço de Cubo. O Cliente define uma operação e espera obter um resultado. A operação desejada é enviada ao cubo que por sua vez a repassa para o Sistema de Gerência de Banco de Dados

responsável pela fonte de dados que se deseja acessar. O SGBD processa a consulta sobre a base correta e envia o resultado para o SC que, finalmente repassa o resultado para o cliente e fecha o ciclo. Quando um cliente acessa o serviço de catálogo, descobre quais cubos estão ativos e quais fontes de dados podem ser acessadas, mas não recebe as configurações de como chegar diretamente ao SGBD da fonte pretendida. As informações que ficam restritas ao cubo e seu SC são o endereço IP do servidor, nome, usuário, e senha da base de dados.

4.1.2 - Registro de Cubos (RC)

O Registro de Cubos é o maestro da arquitetura. Ele é o responsável por indicar quais cubos estão ativos e fornecer uma referência para que sejam acessados. Diferente de como opera o SC, que mantém um registro permanente a respeito de seus cubos, o Registro de Cubos (RC) não guarda metadados de forma permanente. O RC trabalha com informações mantidas apenas em memória volátil. Seu objetivo não é listar todos os cubos que estiveram ativos, mas sim os que se encontram ativos no momento em que um cliente requisita a lista de cubos.

Quando o Registro de cubos entra em operação ele se encontra vazio sem nenhuma referência para cubo. Ele fica aguardando até que um cubo que passe a entrar em operação comunique a ele este fato. Um cubo que entra em operação passa a se comunicar com o RC em busca de ser adicionado na lista de cubos ativos. Quando um cubo é incluído na lista, o RC passa a guardar uma referência deste para que clientes possam posteriormente se conectar diretamente a este determinado cubo. Essa referência é criada baseada em informações fornecidas pelo próprio cubo quando é incluído no RC.

O RC dá a cada cubo um prazo de validade para seu registro. Isto é feito para que se possa manter o controle dos cubos que ainda estão ativos e dos cubos que interromperam seu serviço. Isso permite que o RC informe às aplicações clientes quais cubos estão realmente ativos. Cada cubo precisa enviar uma mensagem para o RC que informe que ainda se encontra em operação antes que o tempo se esgote. Caso um cubo não envie a confirmação antes do tempo limite isso não implica em sua remoção completa do catálogo. Ele deixa de ser listado para clientes que requisitem a lista de cubos ativos.

Entretanto, caso o cubo envie a mensagem de atividade mesmo depois do prazo, ele passará a ser elencado como um cubo disponível. Isto é importante, pois um cubo pode levar mais tempo para se comunicar devido a diversos fatores como problemas de rede.

Quando o SC retira um cubo de sua lista e ele deixa de operar, o RC é informado disso e o retira imediatamente de sua lista.

4.1.3 - Data Report Service (DRS)

O Serviço Cubo e o Registro de Cubo oferecem o meio de se disponibilizar e descobrir fontes de dados. O DRS ao mesmo tempo que é um serviço atua como cliente do Serviço de Cubo a fim de obter metadados dos cubos. O objetivo primário do DRS é suportar a elaboração de consultas ad-hoc comuns em sistemas OLAP. Estas consultas tem como característica serem elaboradas para atenderem a uma necessidade específica de informação, que não se poderia prever.

A aplicação cliente envia para o DRS as opções, comparações e restrições, enfim tudo o que foi definido pelo usuário. Para cada novo atributo que passa a fazer parte das definições, o DRS envia uma mensagem para o Serviço de Cubo para obter todos os metadados referentes a este novo objeto. Estes metadados incluem, como já citado, o tipo de dado, tamanho máximo e precisão. O DRS coleta também informações a respeito das ligações da tabela que o atributo faz parte. Usando estes metadados e as opções definidas pelo usuário, o DRS tem condições de gerar uma consulta que pode ser submetida a qualquer banco de dados relacional. Esta consulta SQL é enviada por fim ao cubo correspondente, no SC, que se encarregará de se conectar ao SGBD que controla a base de dados que se deseja consultar. A aplicação cliente deve solicitar ao DRS a consulta no formato SQL. De posse da consulta SQL deve enviá-la ao cubo para que este execute a pesquisa sobre a sua fonte de dados.

O serviço DRS possibilita que mais de uma consulta seja definida simultaneamente sem que as definições de uma interfira em outra. De forma que apenas uma instância deste serviço é necessária para atender a toda demanda de criação de consultas.

4.1.4 - Aplicação Cliente

Em nosso projeto a aplicação cliente se encontra dentro de um Container Web java(ex. Tomcat), e embora pudesse se encontrar implementada de forma tradicional como aplicativo em janelas de interface gráfica(Desktop). Ou ainda em console de linhas de comando em modo texto. Em todos estes casos é possível criar clientes que se comunicam da mesma forma com os serviços, ou seja, enviar e receber as mesmas mensagens. Esta flexibilidade se dá graças às características do ambiente de desenvolvimento e se estende também à plataforma usada pelo cliente.

O cliente inicialmente precisa descobrir quais são os cubos que estão disponíveis. Para isso entra em contato com o RC a fim de conhecer os cubos ativos. O RC verifica os cubos que tem seu tempo de vida válido e responde ao cliente sua solicitação.

O próximo passo é escolher um cubo a submeter consultas. Para consultar o cubo escolhido em busca de seus metadados é preciso usar a referência a ele, obtida do RC. Através desta referência o cliente pode solicitar diretamente ao cubo todos os metadados que ele tem disponível.

Agora que tem metadados disponíveis o cliente pode passar a montar a consulta. Para isso começa solicitando ao DRS que crie uma nova consulta vazia. O cliente envia sucessivamente as condições para o serviço DRS que passa a registrá-las. Ao final do envio de todas as restrições e condições, o cliente solicita ao DRS que transforme todos o parâmetros enviados em uma consulta SQL que pode ser submetida ao cubo. De posse dessa consulta gerada pelo DRS, o cliente faz uma nova solicitação ao cubo para que execute a consulta sobre sua fonte de dados.

O cliente recebe do SC que mantém o cubo consultado uma resposta à consulta submetida. Esta resposta contendo o resultado da pesquisa não está formatada para ser exibido em alguma forma específica. Pode ser exibida no console, janela, página da internet ou exportada para XML ou qualquer outro formato, ou mesmo sofrer novos processamentos de cálculos e filtros.

4.2 - MODELAGEM

Nesta seção apresentaremos a modelagem que implementamos, tanto dos Serviços quanto do Cliente e indicaremos os pacotes e as classes que compõem a implementação. Daremos mais atenção às classes que suportam as funcionalidades que julgamos as principais de cada serviço e fazemos sucinta descrição das demais. Sem mais delongas iniciemos o estudo dos componentes.

4.2.1 -Modelagem do Serviço de Cubo

O serviço de Cubo possui dois pacotes, org.cube.service.impl e org.cube.service.impl.metadados. O pacote org.cube.service.impl contém a classe que recebe as mensagens remotas e a classe de Controle. O pacote org.cube.service.impl.metadados possui as classes que representam os metadados de uma base de dados modelada com o modelo dimensional. A figura 4.2 apresenta o diagrama de classes do serviço.

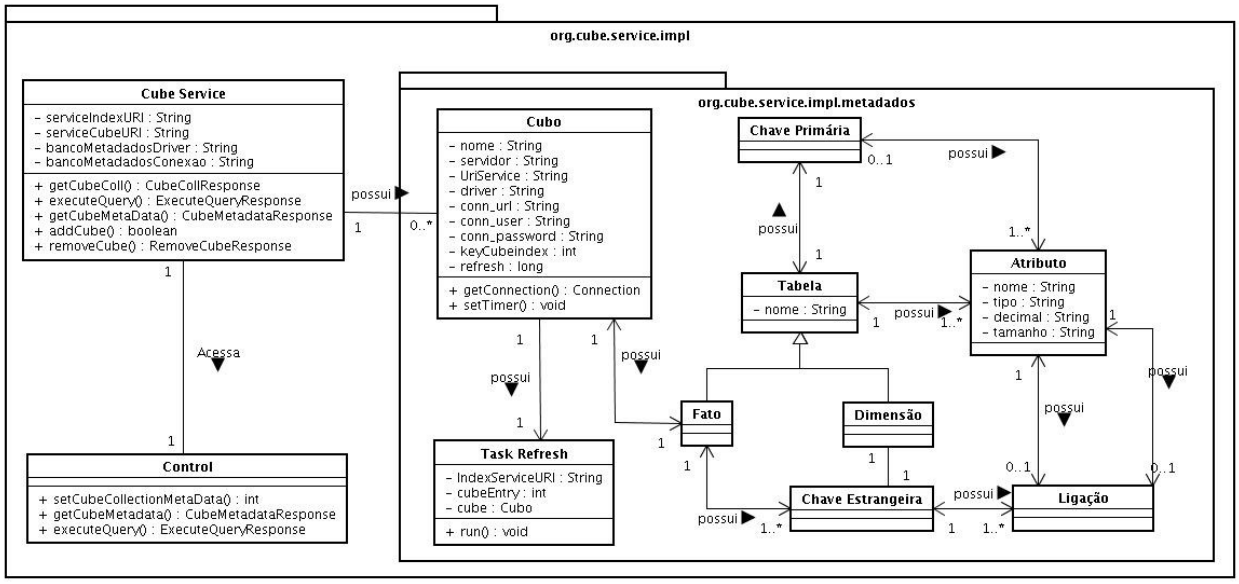


Figura 4.2 - Diagrama de Classes do Serviço de Cubo

Agora vamos descrever as classes do pacote org.cube.service.impl.metadados:

Cubo

A classe Cubo contém um nome que é o assunto a que se referem seus dados e serve para sua identificação. Esta classe possui uma ligação com a classe Fato que é o ponto central na definição do esquema estrela.

Uma instância da classe Cubo possui informações necessárias para se estabelecer uma conexão com o SGBD que mantém sua base de dados. Com a chamada ao método getConnection() é possível obter uma nova conexão ativa com o banco de dados. Não há um sistema de gerenciamento de conexões que implemente um pool de conexão.

A classe possui a funcionalidade de criar uma tarefa que se comunica periodicamente com o serviço RC para manter ativa sua entrada no catálogo de cubos. Esta tarefa repetitiva é iniciada invocando setTimer() que instancia um objeto do tipo TaskRefresh.

Task Refresh

Extende a classe TimerTask e em seu método run() envia a mensagem refreshCube() para o RC atualizar o tempo de validade do cubo.

Tabela

Representa uma relação no banco de dados e é a classe genérica dos tipos Fato e Dimensão. Possui relacionamento com Chave Primária e Atributo.

Fato

Representa a tabela de Fatos na modelagem dimensional. Estende o tipo Tabela. Usa relacionamento um para muitos com Chave Estrangeira, de forma a recuperar objetos que representam Dimensões.

Dimensão

Representa tabelas de Dimensão na modelagem dimensional. Estende o tipo Tabela. Usa relacionamento um para um com Chave Estrangeira, de forma a recuperar o objeto que representa o fato.

Atributo

A Classe Atributo representa o objeto que leva seu nome em um esquema de banco de dados. Há um relacionamento do tipo muitos para um com Tabela que indica a Tabela que possui o Atributo. Possui relacionamento com Chave Primária quando uma instância compõe a chave primária de sua relação. Quando uma instância de Atributo deve fazer parte da condição de junção entre Dimensão e Fato, um relacionamento entre Atributo e Ligação deve existir.

Ligação

Classe que define quais atributos de Fato e Dimensão se relacionam. Indica um atributo de Fato e outro de Dimensão que estão vinculados para compor a associaçõa entre as tabelas. Caso a condição de junção exija que mais de um par de atributos estejam relacionados, outra instância de Ligação será necessária para representar este relacionamento.

Chave Estrangeira

Permite que haja o relacionamento entre fato e dimensão. Através do relacionamento com uma ou mais instâncias de Ligação consegue configurar a condição de junção.

Chave Primária

Configura a chave primária de uma relação, seja ela Fato ou Dimensão. Pussui um ou mais atributos.

Passemos agora a abordar o pacote org.cube.service.impl que possui as seguintes classes:

Cube Service

Esta classe implementa os métodos definidos no arquivo WSDL(Web Service Description Language) de configuração do serviço. Estes métodos podem ser acessados remotamente.

Quando entra no ar o serviço não possui instância de nenhum cubo. Ao iniciar, o Container de Web service cria um objeto da classe CubeService. A partir deste momento inicia-se a recuperação dos metadados de cada cubo cadastrado na base de metadados. Para descobrir qual é seu servidor e criar uma conexão com este banco de dados é preciso acessar um arquivo de configuração que contém estas informações. O caminho deste arquivo está configurado em uma variável de ambiente chamada CUBO_CONF.

Neste momento o SC pode aguardar solicitações de clientes que busquem informações de algum cubo e executar consultas sobre suas respectivas bases de dados.

Destacamos o método getCubeMetaData(). Este método acessa os objetos que representam os metadados de um cubo para obter as informações disponíveis sobre ele. Este método recebe como parâmetro um número inteiro que representa o identificador do cubo. Com este identificador é possível descobrir de qual é o cubo o cliente deseja descobrir os metadados. Os metadados disponíveis são enviados como resposta através do tipo CubeMetaDataResponse. Este tipo é um Stubs, gerado dinamicamente, que se baseia nas definições do arquivo WSDL que permite a definição de tipos de dados complexos.

O SC também é o responsável por acessar a base de dados que ele representa. Ou seja, por meio dele os clientes podem submeter consultas e obter o resultado num formato fácil de tratar. O SC recebe a mensagem remota executeQuery() que tem com parâmetros o identificador do cubo e também a consulta que se deseja submeter no formato SQL. O SC obtém do cubo a conexão e submete a consulta. O resultado é transformado para que seja devolvido ao cliente no formato definido no WSDL. O tipo de retorno é executeQueryResponse.

Control

Apesar do nome sugerir controlador do padrão de projetos em três camadas modelo, visão e controle, não é exatamente dessa forma que o tipo controle está implementado. Ele oferece serviços ou funções para as outras classes. Ele implementa os métodos executeQuery() e getCubeMetaData(). Estes são invocados pela classe CubeService quando ela recebe mensagens de mesmo nome. O método

setCubeCollectionMetaData() desta classe acessa a base de metadados e cria em memória os cubos encontrados com seus respectivos metadados. Ele é chamado quando o container cria uma instância de CubeService e chama seu construtor padrão. Ao chamar registerCubeIndexService() da classe Control, uma TaskRefresh registra o cubo no RC, caso ele ainda não esteja cadastrado.

4.2.2 - Modelagem do Registro de Cubos

A figura 4.3 mostra a modelagem com o diagrama de classe da UML do serviço Registro de Cubos.

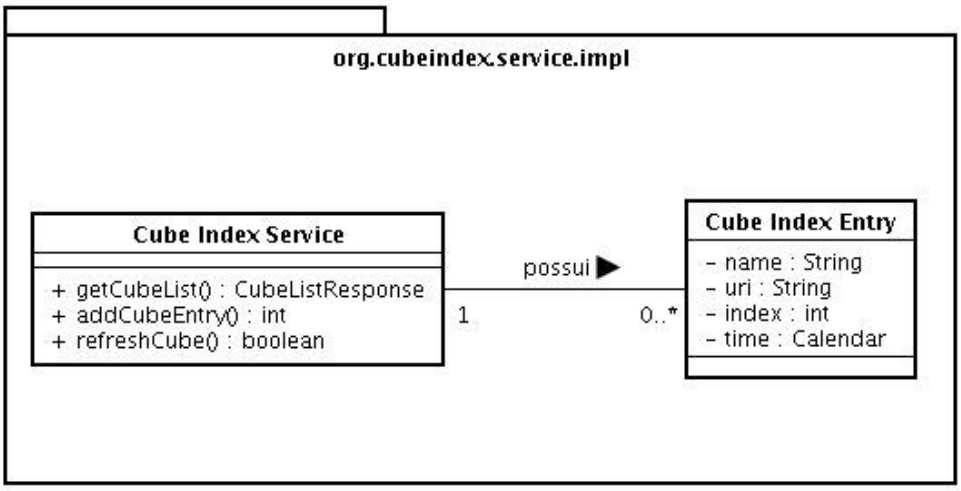


Figura 4.3 - Modelagem do serviço de Cubos

O Registro de Cubos possui o pacote org.cube.service.impl. Suas classe são as seguintes:

Cube Index Entry

Cada objeto criado deste tipo representa um cubo ativo em um determinado SC. Este tipo mantém uma referência ao cubo. Com esta referência, um cliente pode enviar

mensagens remotas ao cubo. Também possui o nome do cubo e seu tempo de máximo de atividade. Este tempo deve ser atualizado constantemente e é feito quando um cubo envia a mensagem refreshCube() para o RC.

Cube Index Service

Esta classe implementa os métodos que podem ser chamados remotamente pelos clientes do serviço RC.

Um cubo que entrou em atividade envia um pedido de inclusão na lista de cubos ativos chamando addCubeEntry(). Nele é criada uma instância de Cube Index Entry para representar este cubo que está ativo. Neste momento é atribuído um identificador e configurado o tempo de vida para esta entrada no catálogo. Neste momento o cubo que solicitou sua inclusão na lista deve chamar o método refreshCube() antes que sua entrada expire. Ao enviar esta mensagem remota deve informar o identificador que recebeu quando fez a solicitação de registro.

Quando um aplicativo cliente deseja enviar consultas a uma base de dados dimensional deve antes descobrir quais bases estão disponíveis. Ao Enviar a mensagem getCubeList() ao RC e examinar sua resposta o cliente pode descobrir quais cubos podem ser acessados. Com este método a classe Cube Index Service verifica quais cubos estão cadastrados e com tempo de vida ativo, e então envia a resposta ao solicitante.

4.2.3.Modelagem do DRS

Como o nome sugere este é o principal serviço de nosso projeto. Com ele um cliente consegue gerar dinamicamente um consulta no formarto SQL que pode ser executada por um SGBD Relacional. Aplicações tradicionais oferecem consultas pré-definidas, onde o que muda são os parâmetros inseridos pelo usuário. Atributos recuperados e relações, assim como seus relacionamentos, são definidos uma vez em “tempo de programação” e não mais alterado. O que existe então é uma inflexibilidade quando se está criando relatórios. Ferramentas de SAD procuram dar mais flexibilidade aos gerentes por permitir que os objetos acessados sejam definidos dinamicamente. Estes objetos incluem quais atributos recuperar e tabelas a acessar. Além da escolha dinâmica

dos objetos, estas ferramentas também garantem flexibilidade no agrupamento dos dados por condições e outras restrições sobre cada campo.

Este serviço permite que usuários criem consultas que acabam sendo convertidas para a SQL. Para representar todas as definições escolhidas por um usuário a ferramenta manipula seis cláusulas da SQL: select, from, where, group by, having e order by.

O DRS possui dois pacotes, org.drs.service.impl e org.drs.service.impl.consulta. O pacote org.drs.service.impl contém a classe que recebe as mensagens remotas. O pacote org.drs.service.impl.consulta possui as classes que representam as consultas definidas pelos usuários.

A figura 4.4 apresenta o diagrama de classes do DRS.

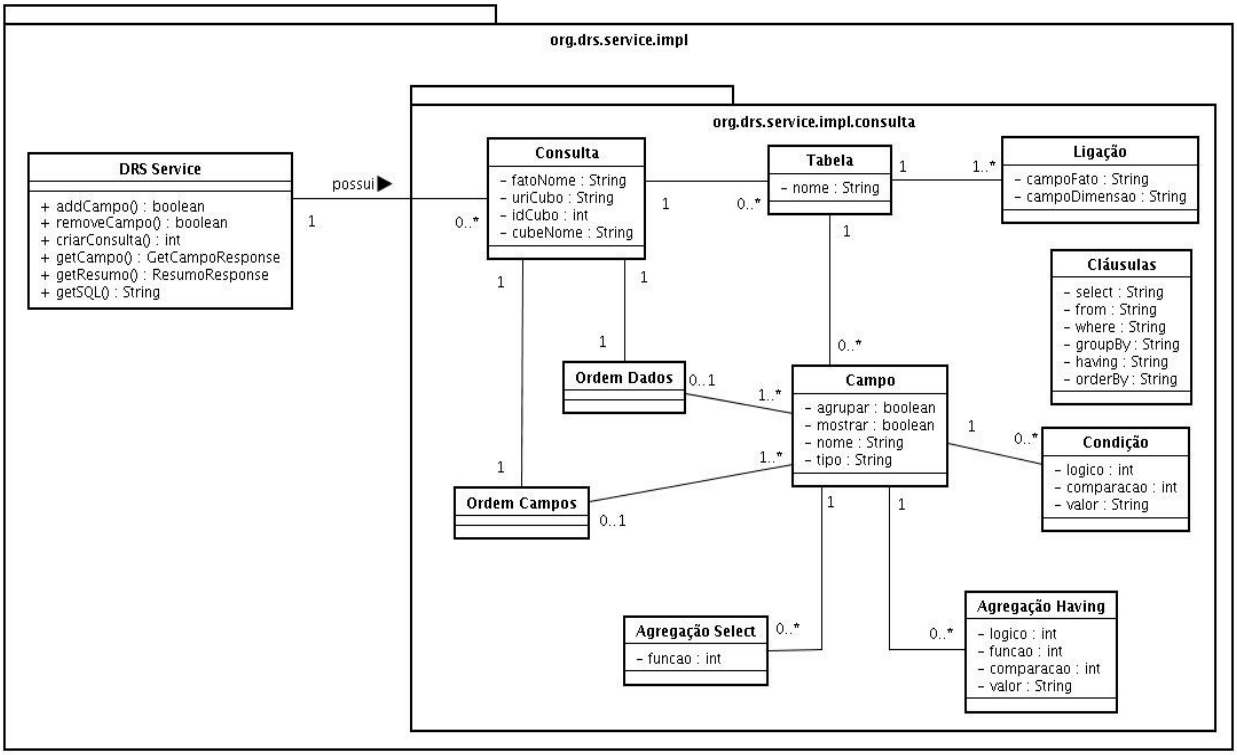


Figura 4.4 - Diagrama de classes do DRS

O pacote org.drs.service.impl.consulta possui as classes:

Cláusulas

Essa classe representa as cláusulas da SQL de uma consulta. Para cada cláusula há um atributo do tipo literal, String. Os objetos que fazem parte da consulta incluem seus valores nas cláusulas correlatas. Um atributo, por exemplo, que deve ser mostrado no resultado deve incluir seu nome na cláusula select. Uma comparação pode ser incluída na cláusula having e assim por diante.

Cada objeto que faz parte da consulta deve possuir o método getClausulas() onde recebe como parâmetro um objeto do tipo Clausula e faz a devida inserção de valores na cláusula correta. Os tipos Tabela, Campo, Condicao, AgregacaoHaving, AgregacaoSelect, Ligacao possuem esse método. Quando o método getClausulas() de todos os objetos forem chamados, então faltará apenas um passo para termos uma consulta SQL. O método getSQL() faz algumas verificações na sintaxe e caso esteja correta os atributos que representam as cláusulas são concatenados gerando uma consulta no formato SQL.

Tabela

Representa uma relação no banco de dados. Caso seja uma dimensão haverá um relacionamento com Ligacao que indicará quais atributos fazem parte da junção com a tabela de fatos. Esta ligação é usada automaticamente, quando o usuário define atributos de uma ou mais dimensões, para fazer a junção entre as dimensões e a tabela de fatos. Uma Tabela pode ter vários Campos.

Campo

Representa um atributo no banco de dados. Possui alguns atributos e relacionamentos que indicam *todas* as definições, restrições e escolhas feitas pelo usuário.

Ligação

Possui dois atributos que indicam qual atributo de uma dimensão está ligada a qual atributo do fato.

Condição

Define as condições que um atributo estabelecidas para o atributo. Estas devem ser atendidas quando a consulta é processada para que um registro possa aparecer no resultado, ou ser encaminhado para ser processado no próximo passo de verificações que o SGBD realiza ao gerar a saída de uma consulta.

Agregação Select

Representa as agregações que aparecem na cláusula select de uma consulta SQL.

Agregação Having

Representa as agregações que aparecem na cláusula having de uma consulta SQL.

Ordem Dados

Representa a ordem que os dados devem aparecer no resultado. A manipulação deste tipo afeta diretamente a cláusula order by.

Ordem Campos

Representa a ordem que os atributos devem aparecer no resultado. A manipulação deste tipo afeta diretamente a cláusula select.

Consulta

Representa uma consulta e possui informações sobre o cubo ao qual está elaborando a pesquisa. Sabe qual o nome da tabela de fatos. Em seu método getSQL() faz algumas modificações no estado da instância de Clausulas que recebe como parâmetro e devolve o objeto retornado por getSQL() dessa instância.

Vamos passar agora a considerar a classe que se encontra no pacote org.drs.service.impl:

DRS Service

Esta classe implementa os métodos que podem ser chamados remotamente que estão descritos no WSDL do serviço.

O DRS possibilita que vários aplicativos clientes criem e configurem consultas ao mesmo tempo. Para passar a usar o serviço, um aplicativo cliente precisa criar uma nova consulta. Isto é feito quando a mensagem criarConsulta() é enviada. Este método retorna um identificador numérico da consulta para que o cliente possa fazer referência a ela nas próximas solicitações ao serviço.

A consulta passa a ser configurada quando cada atributo é incluído. Para se adicionar um novo atributo à consulta, o cliente deve enviar a mensagem addCampo(). O parâmetro passado é do tipo AddCampo, que possui referência a Atributo. O tipo atributo é definido no WSDL usando-se a definição XML para tipos, W3C XML Schema. A figura 4.5 apresenta o tipo Atributo definido na configuração do serviço.

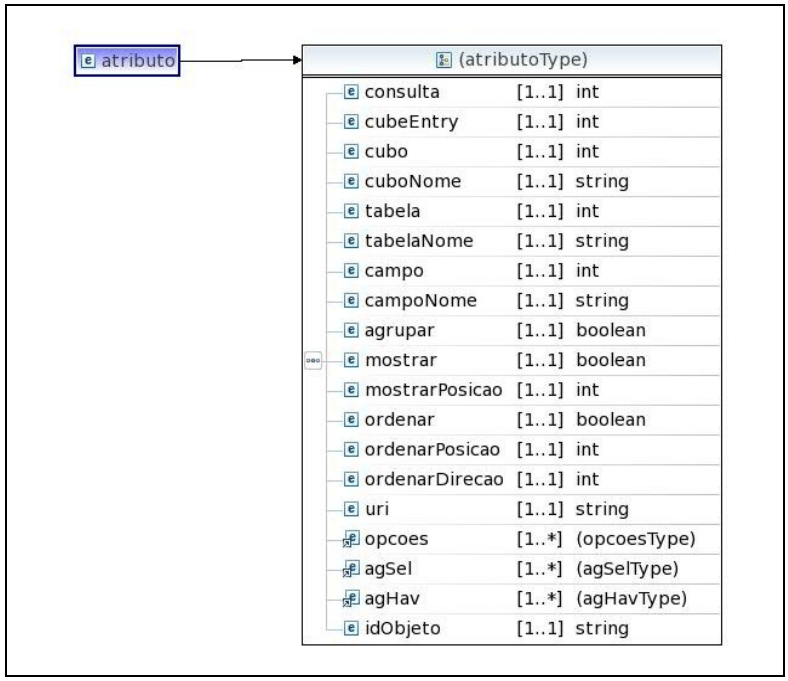


Figura 4.5 - Tipo Atributo Definido no WSDL do serviço DRS

Quando o DRS recebe a mensagem addCampo() é verificado se o atributo que se deseja adicionar já encontra-se incluído na consulta. Quando não está presente é chamado

o método remoto getCubeMetaData() do SC com o intuito de recuperar informações que serão usadas no momento de se transformar a consulta em SQL.

Este método é chamado também quando se deseja alterar alguma informação de um atributo, seja ela um atributo booleano ou objeto relacionado com vários parâmetros, uma coleção. O método addCampo() verifica todas as opções que foram definidas que se encontram no objeto Atributo, que veio como parâmetro, e as coloca no objeto do topo Campo Adequado.

O cliente pode solicitar informações de um atributo que faz parte da consulta chamando getCampo().

O método remoto getResumo() permite que um cliente obtenha uma lista de todos os atributos que de alguma forma fazem parte da consulta.

Invocando removeCampo(), um campo e toda a sua configuração é retirada da consulta. Quando a SQL é gerada este campo não aparecerá mais em nenhuma cláusula.

Finalmente o cliente obtém a consulta no formato SQL enviando a mensagem getSQL() para o serviço DRS. Ele chama getSQL() da consulta que por sua vez chama getSQL() da cláusula, que então devolve a SQL para o cliente. O cliente eventualmente executará esta consulta sobre a base de dados do cubo enviando executeQuery() para o SC, por sua vez devolverá um resultado.

Assim Fechamos o ciclo de descoberta de metadados, na definição da consulta e envio de uma mensagem contendo um SQL para o SGBD que manipula a base de dados.

Agora iremos considerar a interface cliente desenvolvida como aplicativo WEB.

4.3.Interface Cliente

Iremos apresentar a ferramenta que desenvolvemos com o objetivo de exemplificar as funcionalidades que os serviços podem oferecer. Iremos demonstrar o uso da ferramenta e ao mesmo tempo informar algumas chamadas aos serviços que executa.

A aplicação cliente se encontra disponível através de páginas de internet. Esta aplicação foi desenvolvida com a tecnologia JSP. Inicialmente ela não trabalhava com servlets, que são melhores para atuar como controladores. Neste trabalho fizemos um

esforço consciente para implementar novas funções usando este recursos, e encaminhando a aplicação para usar o padrão de projeto MVC2.

Quando um usuário se conecta à ferramenta, ela cria automaticamente uma nova consulta no serviço de cubos. A ferramenta envia a mensagem remota criarConsulta() ao serviço DRS. A identificação da consulta criada é exibida ao usuário num alerta da página, como mostrado na figura 4.6.

Figura 4.6 - Exibição da Consulta criada no DRS

A ferramenta exibe do lado esquerdo do browser a lista de cubos ativos obtida do Registro de Cubos. A ferramenta exibe os cubos ativos depois de receber a resposta a chamada do método getCubeList() do RC. Podemos ver os cubos ativos na figura 4.7.

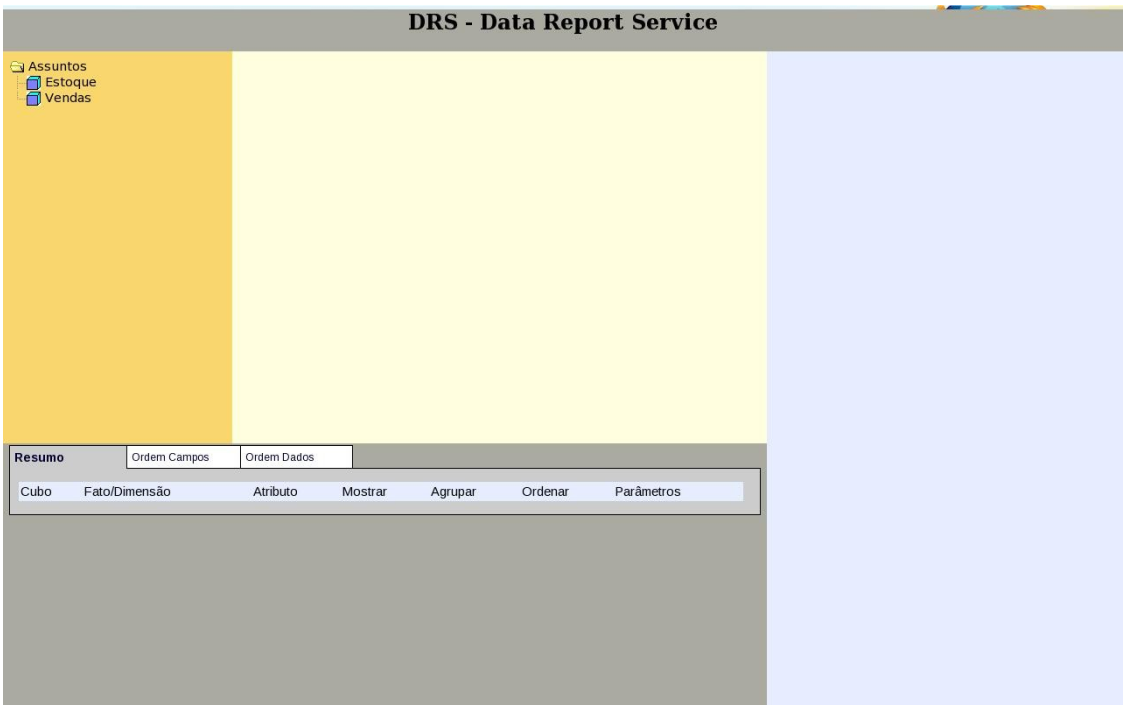


Figura 4.7 - Cubos Ativos no lado esquerdo da página

Ao selecionar um assunto, a ferramenta busca metadados do cubo e os apresenta, conforme a figura 4.8. A ferramenta envia a mensagem getCubeMetaData() para o SC a fim de recuperar as informações daquele cubo. Os metadados estão organizados hierarquicamente numa estrutura de árvore. O nó raiz representa o fato. No primeiro nível encontramos as dimensões e os nós folha da tabela de fatos, que são atributos desta relação. As dimensões também possuem filhos que são seus atributos.

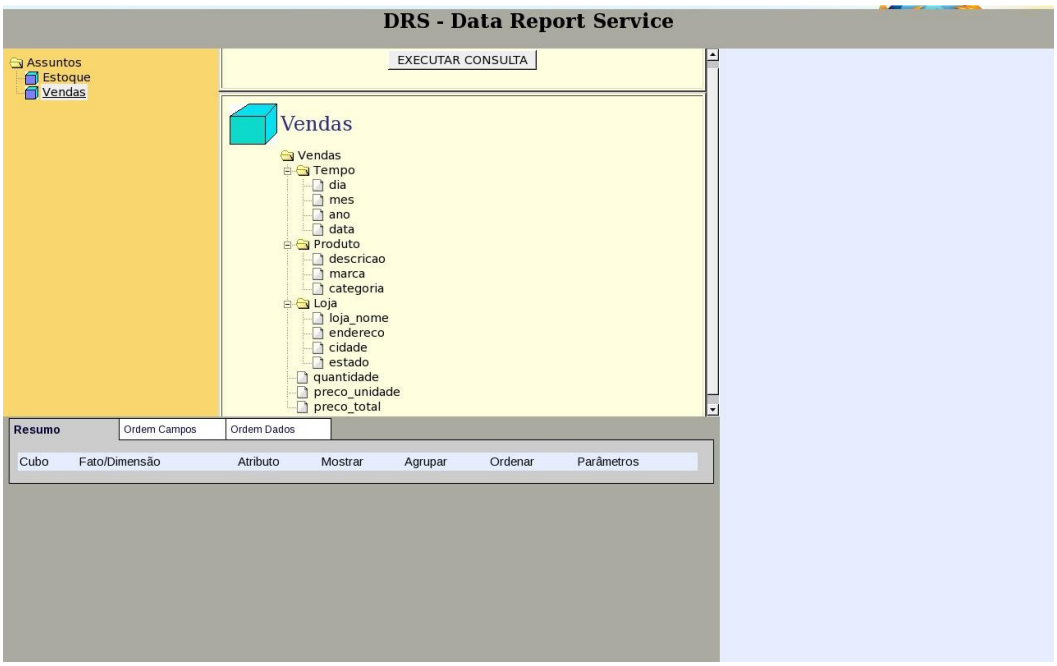


Figura 4.8 - Exposição dos Metadados de um Cubo

Escolhendo um nó folha o usuário passa a editar os parâmetros para um atributo específico. Do lado direito da janela o usuário pode visualizar e editar as restrições para um campo específico. Estas definições afetarão diretamente no resultado final, ou seja, afetará a consulta gerada no formato SQL. O interessante é que o usuário pode escolher aleatoriamente os atributos, quer sejam de fato ou uma dimensão, não se preocupando com os relacionamentos necessários entre as tabelas. Os relacionamentos que são necessários para atender à demanda do usuário são detectados automaticamente pelo serviço DRS e incluídos na consulta SQL gerada. A figura 4.9 apresenta as opções que o usuário tem a sua disposição para fazer, sobre um atributo.

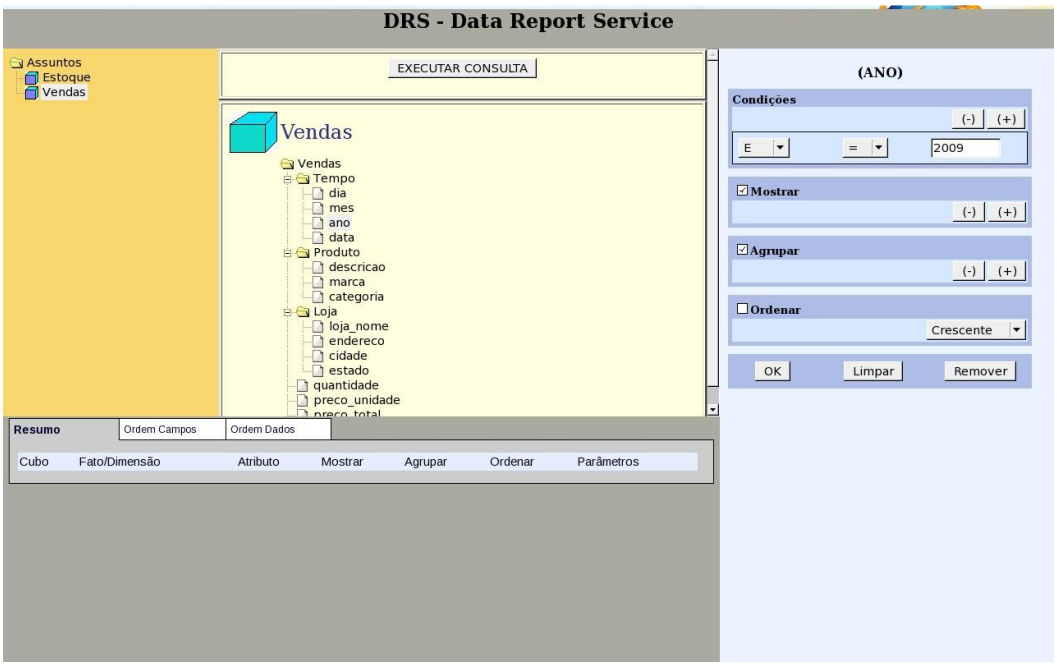


Figura 4.9 - Opções para definir sobre um Atributo

Na parte inferior é mostrado o resumo dos atributos escolhidos e as opções definidas sobre ele. A primeira aba Resumo contém estas informações. A figura 4.10 apresenta um resumo das opções definidas de uma consulta.

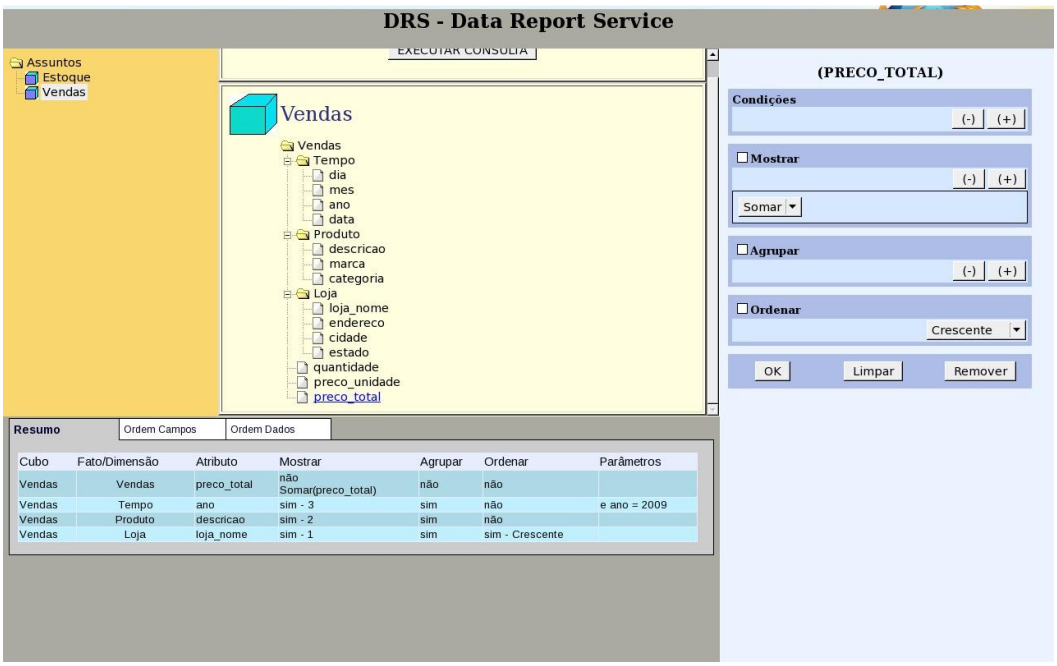


Figura 4.10 - Resumo de uma consulta na parte inferior

Ainda na parte inferior da janela temos outra aba, Ordem Campos. Nesta aba o usuário pode ver e editar a ordem de uma lista de atributos. A ordem desta lista define a ordem com que aparecerão os atributos no resultado da consulta, portanto afeta a cláusula select da da consulta em SQL. Os objetos na lista podem ser movidos com o uso do mouse usando drag and drop, baseada em DHTML. A figura 4.11 nos mostra a aba Ordem Campos.

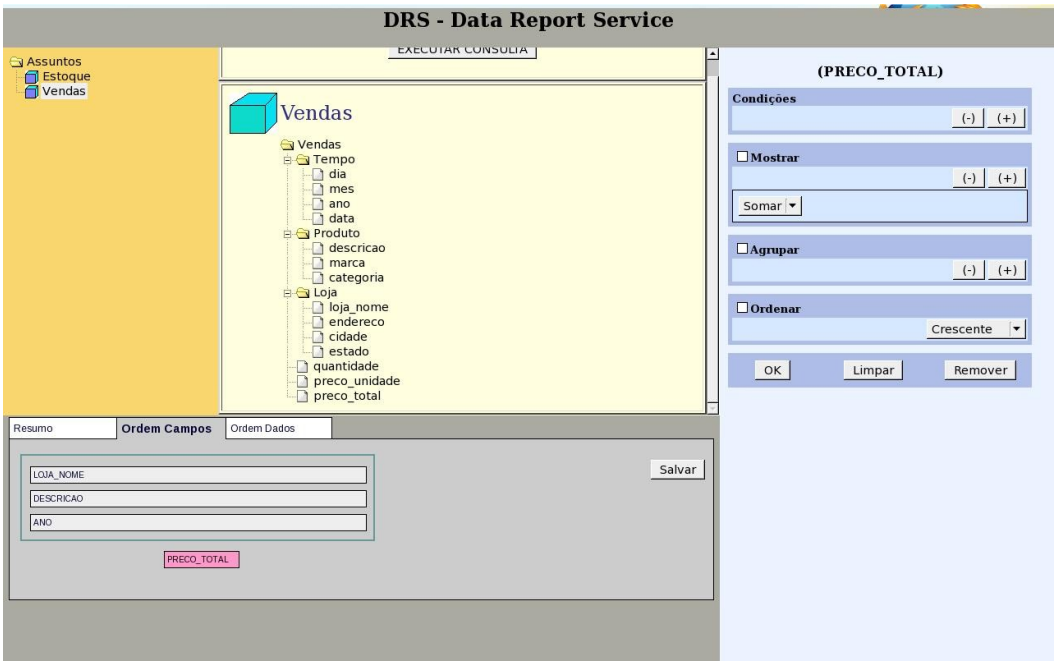


Figura 4.11 - Ordem que os campos aparecerão no resultado

Outra aba tem função semelhante a Ordem Campos, é a aba Ordem Dados. Diferente da Ordem Campos esta permite, como o nome sugere, estabelecer ordenação dos dados que fazem parte do resultado de uma consulta. Na figura 4.12 podemos observar a aba Ordem Dados.

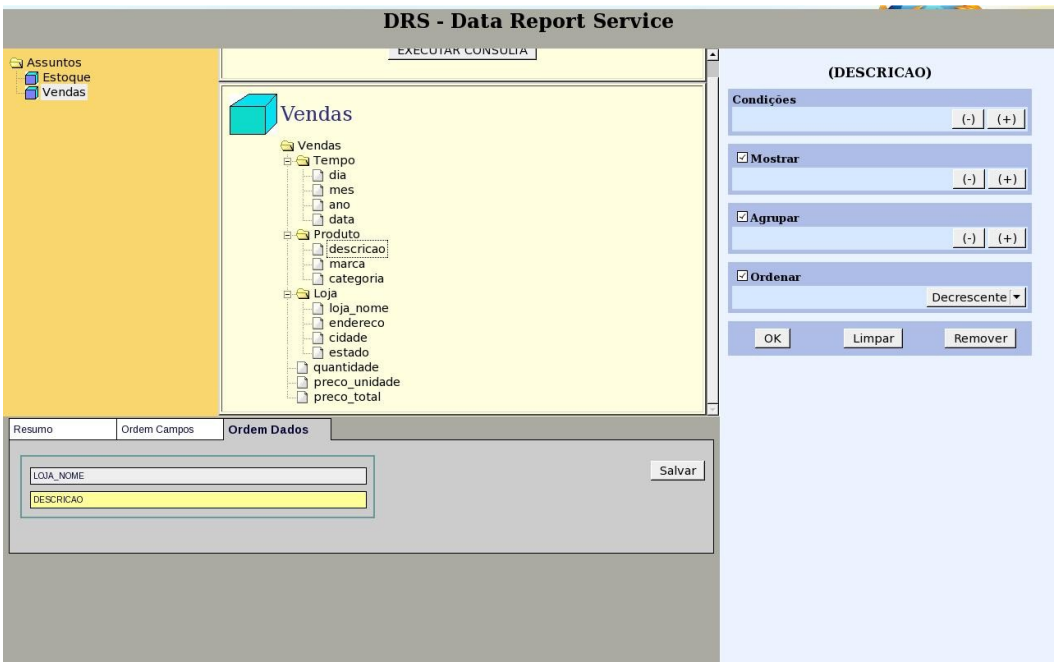


Figura 4.12 - Aba Ordem Dados para ordenação do resultado de uma consulta

O suário passa a iniciar o processo de execução de consulta quando clica no botão "Executar Consulta" no topo da página. Este evento faz com que uma nova janela seja aberta para que seja escolhida como a ferramenta tratará o resultado da consulta que será executada. Na figura 4.13 vemos as duas opções disponíveis no momento.

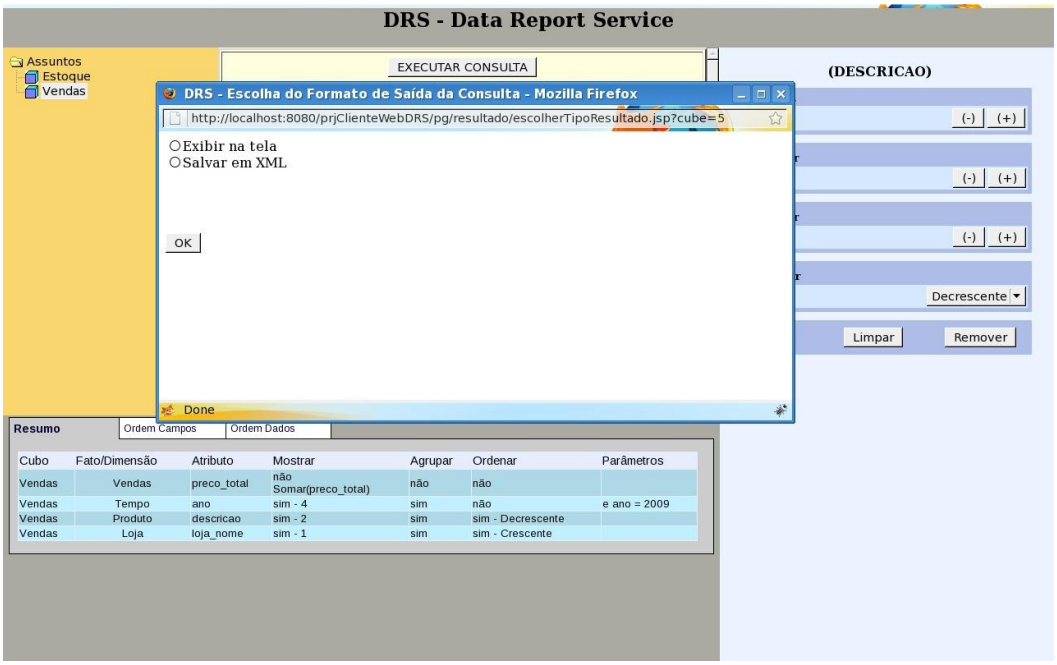


Figura 4.13 - Janela para escolha da saída do resultado para o usuário

Caso escolha a opção Exibir na tela o resultado será processado e exibido na mesma janela em HTML conforme a figura 4.14 nos mostra. Vemos somar_numero, descricao e contar_descricao que são as colunas retornadas e respectivamente 140, primeira linha e 2 é uma tupla do resultado. Na cor azul mais abaixo vemos SQL Elaborada. Esta foi a consulta obtida do serviço DRS quando solicitado pela mensagem getSQL(). Esta mesma consulta foi enviada ao serviço SC como parâmetro da chamada executeQuery(). O resultado foi processado e vemos abaixo o resultado.

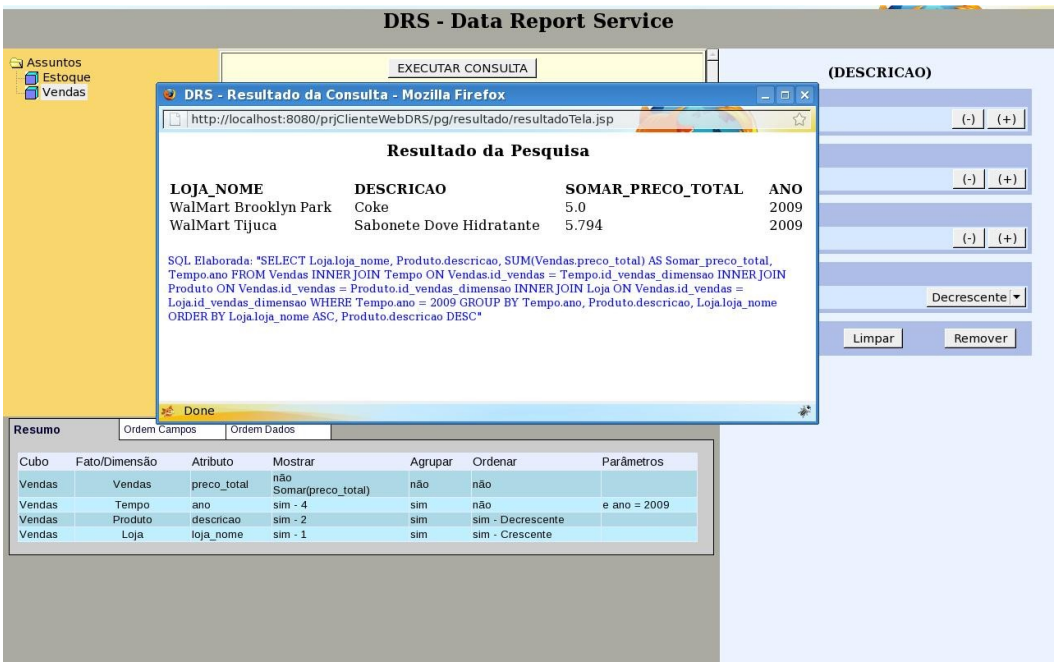


Figura 4.14 - Resultado de consulta exibido na tela como HTML

O usuário pode escolher receber o resultado no formato XML escolhendo a opção Salvar em XML na escolha da saída do resultado. O processo é similar ao da exibição na tela o que muda é o processamento que o cliente faz do resultado, transformando-o em XML com auxílio do arcabouço JDOM. A figura 4.15 nos mostra que o usuário pode escolher salvar o resultado da consulta em seu computador ou outro meio persistente.

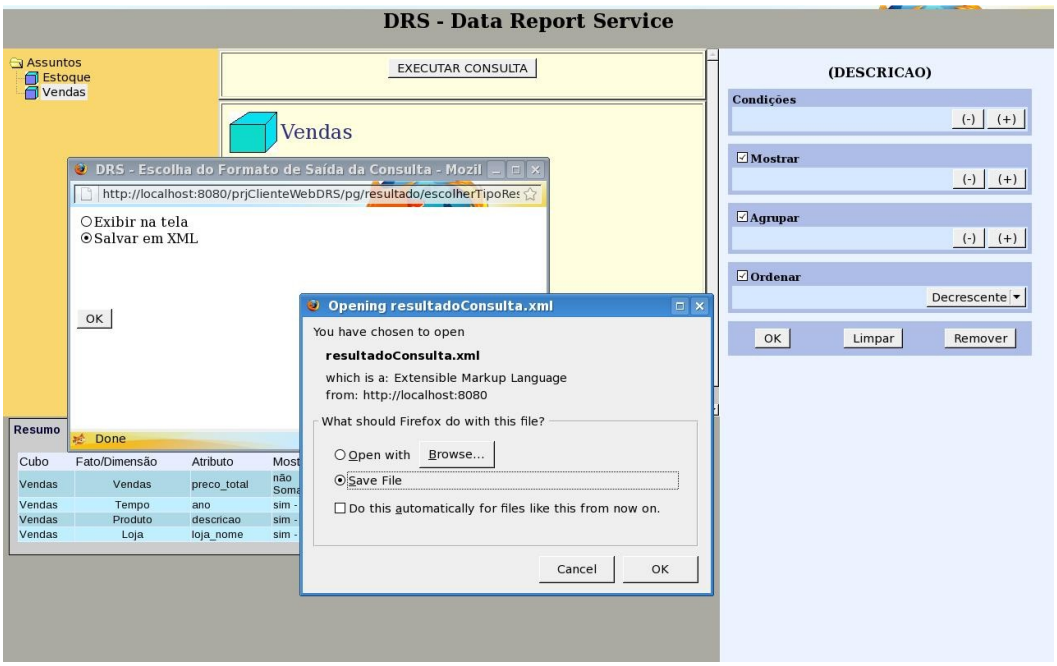
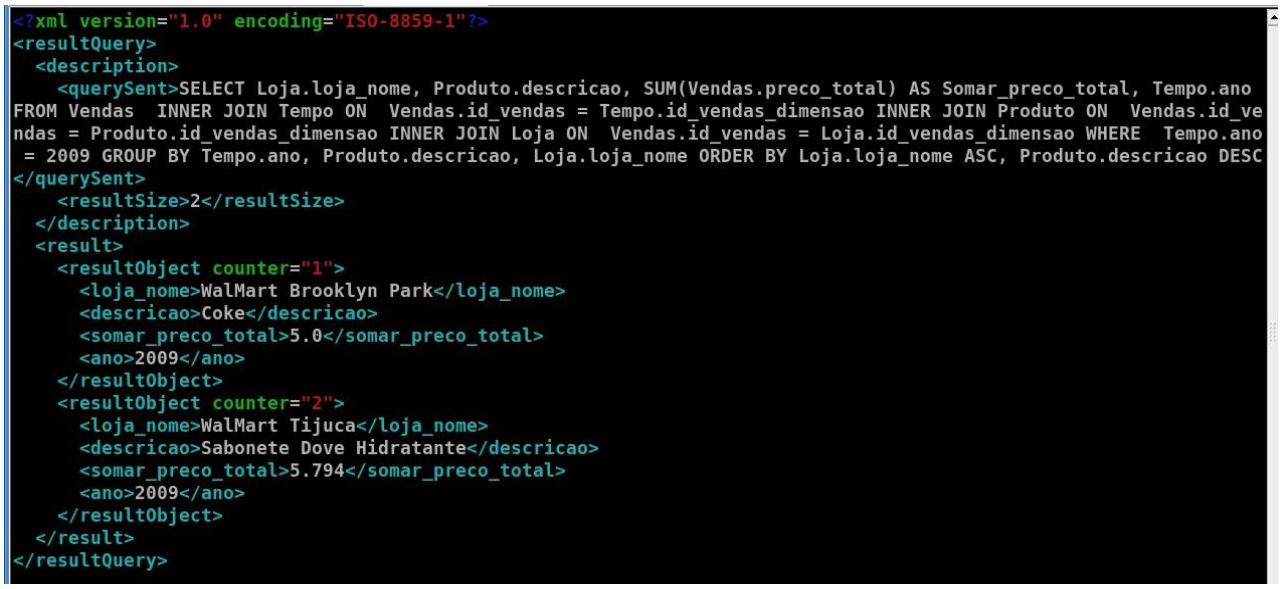


Figura 4.15 - Usuário pode salvar o Resultado no formato XML

Este arquivo contendo o resultado da consulta está exemplificado na figura 4.16. O arquivo está sendo visualizado com o editor VIM do Linux, que possui facilidades como mudança de cores na visualização de arquivos XML. Vemos em description a consulta SQL que produziu o resultado e a quantidade de tuplas retornadas. Em result temos o resultObject que representa uma única tupla com seus campos(field) e respectivos valores. Cada fild possui o atributo columnName que indica o nome da coluna e dentro da tag o valor para o campo.

A consulta que usamos como exemplo foi elaborada para responder a seguinte pergunta hipotética: “Qual foi o total vendido de cada produto, em cada loja, no ano de 2009?”. A consulta elaborada acessa todas as Dimensões e o Fato Vendas para atender a esta demanda. Também aplicamos ordenação nos dados de forma crescente e decrescente.



```
<?xml version="1.0" encoding="ISO-8859-1"?>
<resultQuery>
  <description>
    <querySent>SELECT Loja.loja_nome, Produto.descricao, SUM(Vendas.preco_total) AS Somar_preco_total, Tempo.ano
FROM Vendas  INNER JOIN Tempo ON  Vendas.id_vendas = Tempo.id_vendas_dimensao INNER JOIN Produto ON  Vendas.id_ve
ndas = Produto.id_vendas_dimensao INNER JOIN Loja ON  Vendas.id_vendas = Loja.id_vendas_dimensao WHERE  Tempo.ano
 = 2009 GROUP BY Tempo.ano, Produto.descricao, Loja.loja_nome ORDER BY Loja.loja_nome ASC, Produto.descricao DESC
</querySent>
    <resultSize>2</resultSize>
  </description>
  <result>
    <resultObject counter="1">
      <loja_nome>WalMart Brooklyn Park</loja_nome>
      <descricao>Coke</descricao>
      <somar_preco_total>5.0</somar_preco_total>
      <ano>2009</ano>
    </resultObject>
    <resultObject counter="2">
      <loja_nome>WalMart Tijuca</loja_nome>
      <descricao>Sabonete Dove Hidratante</descricao>
      <somar_preco_total>5.794</somar_preco_total>
      <ano>2009</ano>
    </resultObject>
  </result>
</resultQuery>
```

Figura 4.16 - Arquivo XML Contendo resultado de Consulta e visualizado no VIM

Assim observamos como o cliente usa os três serviços para descobrir as fontes de dados disponíveis e seus metadados, escolher as opções disponíveis para seus campos, enviar uma consulta para o cubo e processar o resultado.

5 -Conclusão

Neste projeto apresentamos os conceitos relacionados a inteligência de negócio e uma ferramenta deste escopo, o DRS. Mostramos como funcionam serviços de grade e partimos então para análise dos serviços que compõem o DRS. Durante o projeto houve o nosso crescimento no que diz respeito ao aprofundamento do conhecimento de tecnologias e o aprendizado de novas. Como futuros desenvolvedores, consideramos positiva a experiência que obtivemos nesse trabalho ao documentarmos os serviços e sua arquitetura em um nível mais alto.

5.1 -Trabalhos Futuros

Verificaremos na continuação do projeto a viabilidade de se implementar subconsultas da SQL para que os relatórios sejam mais precisos. Também pretendemos implementar consultas que envolvam mais de um cubo de dados. Será preciso então atualizar a aplicação cliente para que faça uso das novas possibilidades que os serviços permitirão. Além disto, implementaremos a metodologia AJAX na recriação do cliente web, assim como JSF.

Referências Bibliográficas

INMON, W.H., HACKARTHORN, R. D. (1997) “Como usar o data warehouse”. Rio de Janeiro: IBPI Press.

FOSTER, I., KESSELMAN, C., NICK, J. M., AND TUECKE, S. *The Physiology of the Grid. An Open Grid Services Architecture for Distributed Systems Integration* Open Grid Service Infrastructure WG, Global Grid Forum, June 22, 2002.

MORALES, Aran Bey Tcholakian. Sistemas de Apoio a Decisão: Negócios Inteligentes na WEB. Disponível em:< http://www2.stela.ufsc.br/aran/sad/sad_aula4.htm> Acessado em: 20 set. 2009.

ALVES, Marcos. Modelagem Dimensional:Granularidade. Disponível em: < http://servidor3.fes.br/disciplinas/tpd/tabd/Modelagem_Multidimensional.pdf>. Acessado em: 20 set. 2009.

POTTS, S.; KOPACK, M. Teach Yourself Web Services in 24 hours. [S.l.]: Sams, 2003.

Anexo I – Configuração dos serviços da arquitetura

Scripts

Para agilizar os testes há os scripts *s* e *deploygars*. O script *s*, chamado pelo usuário dono do diretório, está dentre dos projetos dos serviços e gera o arquivo GAR correspondente. O *deploygars* deve ser executado pelo usuário globus porque coloca e tira serviços do *container* por meio de *globus-deploy-gar* e *globus-undeploy-gar*. Padrão de utilização:

Script *s*:

$ ./s <ação>

Ação:

-1 gera o arquivo GAR.

-2 compila classes clientes do serviço

-3 executa clientes.

-4 itens 2 e 3.

ex: Gerar GAR.

$ ./s 1

Script *deploygars*:

$ ./deploygars <serviço><ação>

Serviço:

-1 Cube

-2 DRS

-3 CubeIndex

Ação:

-1 Pára Globus Container e *undeploy* do serviço

-2 Faz *deploy* do serviço e inicia Globus Container

-3 *Undeploy* do serviço

-4 *Deploy* do serviço

ex: Parar container e tirar DRS da lista de serviços gerenciados.

$ ./deploygars 21

ex: Colocar CubeIndex na lista e iniciar o container.

$ ./deploygars 32

Variável de ambiente e Arquivo de Configuracão

A variável de ambiente $CUBO_CONF deve ampontar diretamente para o arquivo *cubo.conf*. A estrutura do arquivo cubo.conf é a seginte:

cubeindexservice:*uri_cubeindex* (caso possua um cliente que acessa o cubeIndex)

cubeservice:*uri_cube* (caso possua um serviço Cube)

drsservice:*uri_drs* (caso possua um cliente que acessa o DRS)

bancometadadosdriver:*classe_driver* (classe do banco de dados que mantém os registros dos metadados dos cubos. Variável acessada pelo serviço Cube)

bancometadadosconexao:*url_database* (url de conexão com o banco de dados que mantém os registros)

Exemplo:

cubeindexservice:http://192.168.0.2:8443/wsrf/services/cubeIndex

cubeservice:http://192.168.0.2:8443/wsrf/services/cube/Cube

drsservice:http://192.168.0.2:8443/wsrf/services/cube/DRS

bancometadadosdriver:org.postgresql.Driver

bancometadadosconexao:jdbc:postgresql://eingrid005.unigranrio.br:5432/cubo

Repositório de Metadados

O serviço cubo de em uma máquina da grade mantém em um banco de dados os cubos cadastrados nele. Este banco de dados pode ter qualquer nome. A froma de acesso a ele e a classe do seu driver devem ser declarados no arquivo cobo.conf citado acima. Deve haver um usuário do banco de dados chamado *globus* e com a senha *globus* para que o serviço possa acessá-lo. O esquema do banco de dados é o seguinte.

```
CREATE TABLE cubo
(
idcubo integer NOT NULL,
nome character varying(50),
servidor character varying(60),
conexao_url character varying(200),
conexao_usuario character varying(50),
conexao_senha character varying(50),
conexao_driver character varying(50),
tempo_refresh integer,
CONSTRAINT pk_cubo_idcubo PRIMARY KEY (idcubo)
)
ALTER TABLE cubo OWNER TO globus;

CREATE TABLE tabela
(
idtabela integer NOT NULL,
nome character varying(50),
CONSTRAINT pk_tabela_idtabela PRIMARY KEY (idtabela)
)
ALTER TABLE tabela OWNER TO globus;

CREATE TABLE fato
(
idfato integer NOT NULL,
idcubo integer NOT NULL,
CONSTRAINT pk_fato_idfato PRIMARY KEY (idfato),
CONSTRAINT fk_fato_idfato FOREIGN KEY (idfato)
```

```
REFERENCES tabela (idtabela) MATCH SIMPLE
ON UPDATE RESTRICT ON DELETE RESTRICT,
CONSTRAINT fk_fato_idcubo FOREIGN KEY (idcubo)
REFERENCES cubo (idcubo) MATCH SIMPLE
ON UPDATE RESTRICT ON DELETE RESTRICT
)
ALTER TABLE fato OWNER TO globus;

CREATE TABLE dimensao
(
iddimensao integer NOT NULL,
CONSTRAINT pk_dimensao_idimensao_idtabela PRIMARY KEY (iddimensao),
CONSTRAINT fk_dimensao_idtabela FOREIGN KEY (iddimensao)
REFERENCES tabela (idtabela) MATCH SIMPLE
ON UPDATE RESTRICT ON DELETE RESTRICT
)
ALTER TABLE dimensao OWNER TO globus;

CREATE TABLE chaveestrangeira
(
idchaveestrangeira integer NOT NULL,
idfato integer NOT NULL,
iddimensao integer NOT NULL,
CONSTRAINT pk_chaveestrangeira_idchaveestrangeira PRIMARY KEY (idchaveestrangeira),
CONSTRAINT fk_chaveestrangeira_dimensao FOREIGN KEY (iddimensao)
REFERENCES dimensao (iddimensao) MATCH SIMPLE
ON UPDATE RESTRICT ON DELETE RESTRICT,
CONSTRAINT fk_chaveestrangeira_fato FOREIGN KEY (idfato)
REFERENCES fato (idfato) MATCH SIMPLE
ON UPDATE RESTRICT ON DELETE RESTRICT
)
ALTER TABLE chaveestrangeira OWNER TO globus;

CREATE TABLE atributo
(
idatributo integer NOT NULL,
nome character varying(50),
tipo character varying(50),
tamanho smallint,
precisao smallint,
```

```
idtabela integer NOT NULL,
chaveprimaria boolean NOT NULL,
CONSTRAINT pk_atributo_idatributo PRIMARY KEY (idatributo),
CONSTRAINT fk_atributo_idtabela FOREIGN KEY (idtabela)
REFERENCES tabela (idtabela) MATCH SIMPLE
ON UPDATE RESTRICT ON DELETE RESTRICT
)
ALTER TABLE atributo OWNER TO globus;

CREATE TABLE ligacao
(
idligacao integer NOT NULL,
idchaveestrangeira integer NOT NULL,
idatributofato integer NOT NULL,
idatributodimensao integer NOT NULL,
CONSTRAINT pk_ligacao_idligacao PRIMARY KEY (idligacao),
CONSTRAINT fk_ligacao_chaveestrangeira FOREIGN KEY (idchaveestrangeira)
REFERENCES chaveestrangeira (idchaveestrangeira) MATCH SIMPLE
ON UPDATE RESTRICT ON DELETE RESTRICT,
CONSTRAINT fk_ligacao_estrangeiro FOREIGN KEY (idatributofato)
REFERENCES atributo (idatributo) MATCH SIMPLE
ON UPDATE RESTRICT ON DELETE RESTRICT,
CONSTRAINT fk_ligacao_primario FOREIGN KEY (idatributodimensao)
REFERENCES atributo (idatributo) MATCH SIMPLE
ON UPDATE RESTRICT ON DELETE RESTRICT
)
ALTER TABLE ligacao OWNER TO globus;
```

Ao fazer o mapeamento do diagrama de classes para o modelo relacional das classes de metadados do serviço Cube (4.2.1) implementamos a classe Chave Primária, do pacote org.cube.service.impl.metadados, como um atributo booleano da relação Atributo.