UNIVERSIDADE FEDERAL DE PERNAMBUCO

GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
CENTRO DE INFORMÁTICA

2004.2

FRAGMENTAÇÃO VERTICAL DE DATA WAREHOUSE EM TERMOS DE MEDIDAS NUMÉRICAS: UM ALGORITMO BÁSICO.

TRABALHO DE GRADUAÇÃO
EM
BANCO DE DADOS

Aluno – Marcelo Victor Calado de Sousa Costa.
Orientador – Fernando da Fonseca de Souza.

17 de Março de 2005.

UNIVERSIDADE FEDERAL DE PERNAMBUCO

GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
CENTRO DE INFORMÁTICA

2004.2

MARCELO VICTOR CALADO DE SOUSA COSTA

FRAGMENTAÇÃO VERTICAL DE DATA WAREHOUSE EM TERMOS DE MEDIDAS NUMÉRICAS: UM ALGORITMO BÁSICO.

ESTE TRABALHO FOI APRESENTADO À GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO DO CENTRO DE INFORMÁTICA DA UNIVERSIDADE FEDERAL DE PERNAMBUCO COMO REQUISITO PARCIAL PARA OBTENÇÃO DO GRAU DE BACHAREL EM CIÊNCIA DA COMPUTAÇÃO.

ORIENTADOR: PROF. DR. FERNANDO DA FONSECA DE SOUZA

17 de Março de 2005.

Dedico,

Aos meus pais
Ubiratan e Tânia

Aos meus irmãos
Phelipe e Gustavo

AGRADECIMENTOS

Aos meus pais, Ubiratan e Tânia, pelo amor, carinho, apoio, compreensão e permanente incentivo.

Aos meus irmãos, Phelipe e Gustavo, pelos momentos de compreensão em que abriram mão de utilizar nosso computador para realização do trabalho.

A Fernando da Fonseca de Souza, pela amizade e constante disponibilidade para orientação do trabalho elaborado.

A Artur Luis do Nascimento, pelos estudos e colaboração na elaboração do trabalho.

A Márcio, Tiago e Saulo, integrantes do grupo de pesquisa RISCTEC, do qual faço parte, pelas orientações na elaboração da implementação gráfica dos grafos de derivação.

A Cristina Dutra de Aguiar Ciferri e seu orientando Diogo pela disponibilidade em oferecer ajuda.

E, finalmente, a todos aqueles que contribuíram direta ou indiretamente para a realização deste trabalho.

Marcelo Victor Calado de Sousa Costa

RESUMO

Um *data warehouse* consiste em uma coleção de dados orientada por assuntos, integrada, variante no tempo, e não volátil, dando suporte à tomada de decisão. Dados centralizados demais podem resultar em perda de disponibilidade e queda de desempenho das consultas, daí surge a necessidade de fragmentação e conseqüentemente de elaboração de algoritmos, minimizando o tempo de processamento dos aplicativos que atuam sobre esses fragmentos. Grandes organizações necessitam trabalhar com medidas numéricas de forma veloz para que possam estar sempre à frente de seus concorrentes. Assim, este trabalho de graduação propõe a elaboração de um algoritmo em pseudocódigo, baseado nos conceitos de grafos de derivação, para fragmentação vertical de um *data warehouse* em termos de medidas numéricas.

ABSTRACT

A data warehouse is a data collection oriented by subject, integrated, variant at time, and persistent, giving support to decision making processes at business. Data excessively centralized can result in lost of reliability and degrade the performance of queries, with rises the necessity of fragmentation and consequently development of algorithms to minimize the processing time of applications that work at these partitions. Organizations need to work with numerical attributes fastest in order to stay always ahead of their competitors. This document presents an elaboration of an algorithm in pseudo code, based in derived graphs definitions, to vertical partitioning of a data warehouse in function of numerical attributes.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE QUADROS

1. INTRODUÇÃO

O *data warehouse* nasceu a partir do reconhecimento da importância do valor da informação nas organizações. Segundo Inmon [INMON_95], a definição clássica de *data warehouse* é: "uma coleção de dados orientada por assuntos, integrada, variante no tempo, e não volátil, que tem por objetivo dar suporte aos processos de tomada de decisão". Assim, o *data warehouse* forma uma base de dados que permite efetuar um tratamento adequado da informação, o qual pode habilitar a descoberta e exploração de tendências empresariais importantes.

Atualmente, os diversos sistemas de informação das corporações estão literalmente "distribuídos", de forma que os seus dados são armazenados em diferentes cidades, regiões e até mesmo países e, possivelmente, utilizando sistemas de gerenciamento de banco de dados diversos, dificultando o entendimento da corporação como um todo. Desta forma, a maioria dos sistemas é instalada com uma visão local e seu principal propósito é resolver um problema singular e isolado, tal como folha de pagamento, planejamento de manufatura e análise de resultado financeiro. Embora esta abordagem não esteja necessariamente incorreta, começa a apresentar problemas quando uma visão cruzada dos dados é necessária para o entendimento da dinâmica da situação. A integração dessas informações dispersas é um dos desafios do *data warehouse*. Porém, a simples integração dos dados não é suficiente, visto que após a coleta, os dados devem ser analisados para determinar sua significância. Falhas na implementação de sistemas e métodos para análise desses dados colocarão a empresa em desvantagem diante de um mercado que está cada vez mais competitivo.

Em contrapartida, dados extremamente centralizados podem resultar em perda de disponibilidade e queda de desempenho das consultas. Daí surge a necessidade de fragmentação do *data warehouse*, seja utilizando-se de algoritmos de particionamento[1] horizontal, vertical ou até mesmo mistos. De acordo com Özsu e Valduriez [ÖZSU_99], o objetivo da fragmentação vertical é particionar uma relação em um conjunto de relações menores, de tal modo que muitos dos aplicativos do usuário possam atuar apenas sobre um fragmento. É com esta motivação que surge a necessidade de desenvolvimento de algoritmos para realizar uma fragmentação vertical em um ambiente de *data warehouse*.

Neste contexto, um esquema de fragmentação vertical deve minimizar o tempo de processamento dos aplicativos do usuário que atuam sobre esses fragmentos [ÖZSU_99]. Baseado nisto é que se pode identificar a necessidade de algoritmos para fragmentação vertical.

Sabe-se que as grandes organizações necessitam trabalhar com medidas numéricas de forma veloz para que possam estar sempre à frente de seus concorrentes. A fragmentação vertical dos dados numéricos faz elevar a velocidade com que esses dados são manipulados, visto que eles podem estar em *sites*[2] de alta performance, além do fato que em conjunto de dados poderiam ser acessados paralelamente. Além disso, rotinas de segurança mais robustas podem ser implementadas com uma complexidade menor nestes dados, já que eles representam porções do todo.

É com todas estas motivações que surge a necessidade para o desenvolvimento de um algoritmo voltado a resolver o problema da fragmentação vertical de um *data warehouse* em termos de medidas numéricas. O algoritmo

[1] Fragmentação.
[2] Localidades.

desenvolvido será baseado nos conceitos de grafos de derivação, de propagação das dimensões sendo fragmentadas aos vértices do grafo e de fragmentação ou reconstrução de agregações. Sendo assim uma maneira adequada para o armazenamento dos dados do *data warehouse* em diferentes níveis de agregação, conforme cita Ciferri [CIFERRI_02A]. Além disso, o algoritmo proposto aqui constitui uma das fundamentações para o sistema WebD²W, que consiste em um ambiente de *data warehousing* distribuído e será explicado em tópicos mais adiante. Todo processo de particionamento apresentado pelo algoritmo é realizado pelo componente de distribuição da arquitetura deste ambiente, mais especificamente pelo módulo de fragmentação.

1.1. Estruturação do Trabalho

Além deste capítulo introdutório, o trabalho é composto por mais cinco capítulos. O capítulo 2 tem por objetivo descrever esclarecer o tema *data warehouse* os principais conceitos relacionados. O capítulo 3 apresenta o sistema WebD²W, descrevendo todo seu funcionamento, arquitetura e componentes. O capítulo 4 apresenta algoritmos de fragmentação vertical e toda base teórica para entendimento do algoritmo proposto, além do detalhamento do algoritmo desenvolvido e exemplos de aplicação. O capítulo 5 cita as contribuições do trabalho elaborado e as propostas para trabalhos futuros. Por fim, o capítulo 6 exibe as referências bibliográficas utilizadas para elaboração do trabalho de graduação.

2. DATA WAREHOUSE

A cada dia o volume de dados disponíveis é maior e a necessidade de transformar esses dados em informações tem sido um grande desafio para as empresas. São cada vez maiores as pressões para prover informações de melhor qualidade para tomada de decisões, em formatos que sejam de fácil acesso e manipulação [WANG_98].

Antes de esclarecer o que se trata um *data warehouse*, é preciso fazer uma distinção entre *data warehouse* e *data warehousing*, embora a maioria da literatura trate os dois termos de maneira idêntica. Sempre que ocorrerem referências ao termo *data warehouse* neste trabalho, trata-se de sua tradução literal, que significa "armazém de dados", e consiste na base de dados que oferece o suporte aos usuários no processo de tomada de decisão. Já as referências à *data warehousing* devem ser tratadas como todo o ambiente do *data warehouse*, que engloba o próprio *data warehouse*, suas arquiteturas, algoritmos e ferramentas, usuários, componentes para realização de consulta, entre outros.

Um *data warehouse*, como descrito anteriormente, pode ser definido como uma coleção de dados orientada por assunto, integrada, variante no tempo, e não volátil, que tem por objetivo dar suporte aos processos de tomada de decisão [INMON_95].

Orientado por assunto porque contém informações sobre temas específicos importantes para o negócio da empresa. Em uma instituição financeira, por exemplo, existem aplicações como empréstimo, poupança e cartão de crédito. O

data warehouse é organizado em volta de assuntos principais tais como cliente, vendedor, produto, atividade e marketing.

É integrado, pois contém dados em estado uniforme, ou seja, existe uma consistência entre nomes, unidades das variáveis, e demais dados. A integração é uma das mais importantes características do *data warehouse* e ao mesmo tempo uma das mais complexa. Como exemplo clássico, tem-se as mais diversas representações correspondentes ao campo sexo nos diversos provedores de informações. Cada aplicação pode determinar sua maneira de representação, como "M" e "F", "Masculino" e "Feminino" e "1" (masculino) e "0" (feminino). Entretanto, o *data warehouse* deve ser capaz de gerar uma única representação para este atributo.

Tem a característica de ser variante no tempo, já que os dados do *data warehouse* devem ser exatos em algum momento do tempo. Esta é uma característica básica diferente dos sistemas operacionais onde os dados devem ser exatos no momento do acesso [INMON_94].

Ele também não apresenta volatilidade de seus dados, pois permite apenas a carga inicial dos dados e consultas a estes dados. Esta é outra característica que difere o *data warehouse* dos sistemas operacionais que permitem inclusões, exclusões e atualização na base registro por registro em tempo real [INMON_97].

Os dados nas organizações costumam estar distribuídos. Isto dificulta todo o processo de transformar os dados em informações úteis no processo de suporte à decisão. Segundo Singh [SINGH_97], as empresas normalmente não sofrem de falta de dados, mas de uma abundância e redundância de dados inconscientes. Assim, a utilização da tecnologia de *data warehouse* vem sendo apontada como uma

maneira de ordenar esses dados, podendo assim transformá-los em informações proveitosas.

O *data warehouse* vem oferecendo às organizações uma maneira flexível e eficiente de obter informações que os gestores necessitam nas rotinas empresariais, dando suporte à função de apoio à decisão.

2.1. Data Warehouse Versus Banco de Dados Operacional

O ambiente de *data warehousing* caracteriza-se por ser um ambiente informacional, isto porque sua base de dados sofre mais freqüentemente operações de leitura, enquanto que os bancos de dados operacionais das organizações, como os localizados em suas filiais, caracterizam-se por serem ambientes operacionais, pois além das consultas, sofrem também operações de inserção, remoção e atualização com grande freqüência.

No ambiente operacional, manipula-se um volume grande de transações que geralmente são simples, pequenas e acessam poucos registros por vez. Já no ambiente informacional, manipula-se um baixo volume de transações que são longas, complexas e acessam muitos registros, necessitando muitas vezes realizar funções de junção e agregação [INMON_96].

Em questão de quantidade de armazenamento, um *data warehouse* geralmente chega a armazenar terabytes de informações, enquanto que as bases de dados operacionais costumam armazenar megabytes, chegando às vezes à gigabytes

Uma outra diferença marcante está na granularidade com que os dados são armazenados. No ambiente operacional, temos uma maior granularidade, de

forma que mais detalhes acerca dos dados são armazenados, enquanto que no ambiente informacional existe uma menor granularidade e menos detalhes são armazenados. Como exemplo de alta granularidade temos vendas por dia e de baixa granularidade vendas por ano.

Por fim, os usuários dos dois sistemas são bem distintos. Enquanto que em uma base de dados operacionais milhares deles realizam a entrada de dados, em um *data warehouse*, gerentes, diretores, entre outros, chegando a no máximo algumas centenas de usuários, utilizam o sistema para auxiliá-los nos processos de tomada de decisão. O tempo de resposta que cada um desses ambientes leva para trazer o resultado de volta ao usuário é bem perpendicular. Em um ambiente operacional, geralmente bastam alguns segundos, enquanto que em um *data warehouse* levá-se horas, pois as operações geralmente manipulam um grande volume de dados.

2.2. Uma Breve Análise das Necessidades das Corporações

Hoje em dia uma organização precisa utilizar toda informação disponível para criar e manter vantagem competitiva. Sai na frente àquela organização que consegue tomar decisões corretas e rápidas. Com esta importante tarefa nas mãos, profissionais responsáveis por decisões estratégicas tais como executivos, gerentes e analistas, exigem dos sistemas de suporte à decisão mais recursos para análise. Surge assim o *data warehouse*, com a idéia de integrar os dados internos e externos de uma organização em uma estrutura única permitindo uma melhor utilização dos dados pelos analistas, gerentes e executivos.

Em um mundo capitalista, onde as organizações buscam cada vez lucros maiores, o *data warehouse* surge com a promessa de aumentar a produtividade a partir de informações aparentemente desconexas.

Com ele, os executivos podem obter, de modo imediato, respostas para as perguntas mais exóticas e, com isso, tomar decisões com base em fatos, não em meras intuições ou especulações. A seguir são descritos alguns exemplos de sucesso apresentados em [CAMPOS_99]:

- A Wal-Mart Stores, uma das maiores redes de varejo dos Estados Unidos descobriu, em seu gigantesco *data warehouse*, que a venda de fraldas descartáveis estava associada à de cerveja. Em geral, os compradores eram homens, que saíam à noite para comprar fraldas e aproveitavam para levar algumas latas de cerveja para casa. Os produtos foram postos lado a lado e como resultado a organização conseguiu aumentar a venda de fraldas e cervejas;

- O banco Itaú, pioneiro no uso de *data warehouse* no Brasil, costumava enviar mais de 1 milhão de malas diretas, para todos os seus correntistas, entretanto, no máximo 2% deles respondiam às promoções. Atualmente, o banco tem armazenado toda a movimentação financeira de seus três milhões de clientes nos últimos dezoito meses. A análise desses dados permite que cartas sejam enviadas apenas a quem tem maior chance de responder. Com isso, a taxa de retorno subiu para 30% e a conta do correio foi reduzida a um quinto;

- A Sprint, um dos líderes no mercado americano de telefonia de longa distância, desenvolveu, com base no seu *data warehouse*, um método capaz de prever com 61% de segurança se um consumidor trocaria de companhia telefônica dentro de um período de dois meses. Com um marketing agressivo, conseguiu evitar a deserção de 120 mil clientes e uma perda de cerca de 35 milhões de dólares em faturamento;

- Uma outra empresa de telefonia, a Telefônica, detectou, ao implantar seu *data warehouse*, que quatro grandes clientes empresariais eram responsáveis por mais da metade das chamadas de manutenção, e que um deles estava prestes a abandonar os serviços. A telefônica fez reparos imediatos, convenceu o cliente a ficar e manteve uma receita anual de 150 milhões de dólares;

- O Serpro, órgão responsável pelo processamento dos dados do governo federal, já investiu 2 milhões no seu projeto de *data warehouse*, desenvolvido com a Oracle. Só consolidou 5% de suas informações, mas já é possível fazer em cinco minutos cruzamentos de dados que antes demandavam quinze dias de trabalho.

Do ponto de vista financeiro o *data warehouse* pode-se resumir a uma única palavra: produtividade. Isto é, ganho de tempo e, conseqüentemente, de dinheiro, com qualquer informação acessível aos executivos no momento e no formato que eles determinarem.

Porém, esta tecnologia tem uma custo elevado. Primeiro, é preciso hardware para guardar e acessar com eficiência um volume astronômico de informações. O maior *data warehouse* do mundo, do Wal Mart, foi expandido pela NCR para conter 23 trilhões de caracteres, mais de 60.000 anos de jornal diário. Esse tempo equivale a 10 vezes toda a história humana, desde o surgimento da escrita [CAMPOS_99]. O hardware é a parte mais cara e deve ser planejado desde o início para comportar futuras expansões. Uma das característica destes verdadeiros armazéns de dados é que, ao contrário dos bancos de dados operacionais, as informações não são voláteis.

Além das máquinas, vários programas são necessários para manter um *data warehouse* em funcionamento. Por exemplo, para extração e transformação dos dados armazenados nas diversas fontes. Esses dados, por sua vez, são acessados por programas especiais de banco de dados, feitos por empresas como Oracle ou Microsoft. Toda essa estrutura cria duas dificuldades: o preço e a necessidade de realizar a integração de tudo. Quem se encarrega disso, em geral, são as consultorias, ou grandes empresas integradoras, como Unisys, IBM e NCR.

Finalmente, há software para a consulta e elaboração de relatórios. Um novo nicho de mercado poderia ser criado com o desenvolvimento de software para distribuição e, conseqüentemente, fragmentação do *data warehouse*, que utilizariam algoritmos como o descrito mais adiante para solucionar a fragmentação vertical dos dados.

A instalação do *data warehouse* pode levar anos pois não é um produto, é um processo. Envolvendo uma mudança cultural no modo como os executivos se relacionam com os computadores e com a informação.

Uma das maiores vantagens do *data warehouse* para as empresas é que mantém um quadro único e coerente das informações ao longo da empresa, uma única versão da verdade. Com ele, executivos passam a ter uma idéia muito mais precisa do papel de seus departamentos e do que é essencial ao negócio. Um exemplo disso é a Kaiser. Toda madrugada as informações são atualizadas no *data warehouse*. Assim, quando o executivo chega de manhã, tudo está disponível em seu computador.

2.3. Arquitetura do Ambiente de Data Warehousing

Atualmente, os ambiente de *data warehousing* das organizações seguem uma arquitetura típica ilustrada na figura 1.

Esta arquitetura demonstra os processos para criação, utilização e manutenção de todo o ambiente de *data warehousing* centralizado, além de estabelecer a forma com que as consultas dos usuários ao sistema são tratadas. A funcionalidade dos principais componentes desta arquitetura será descrita nos tópicos seguintes.

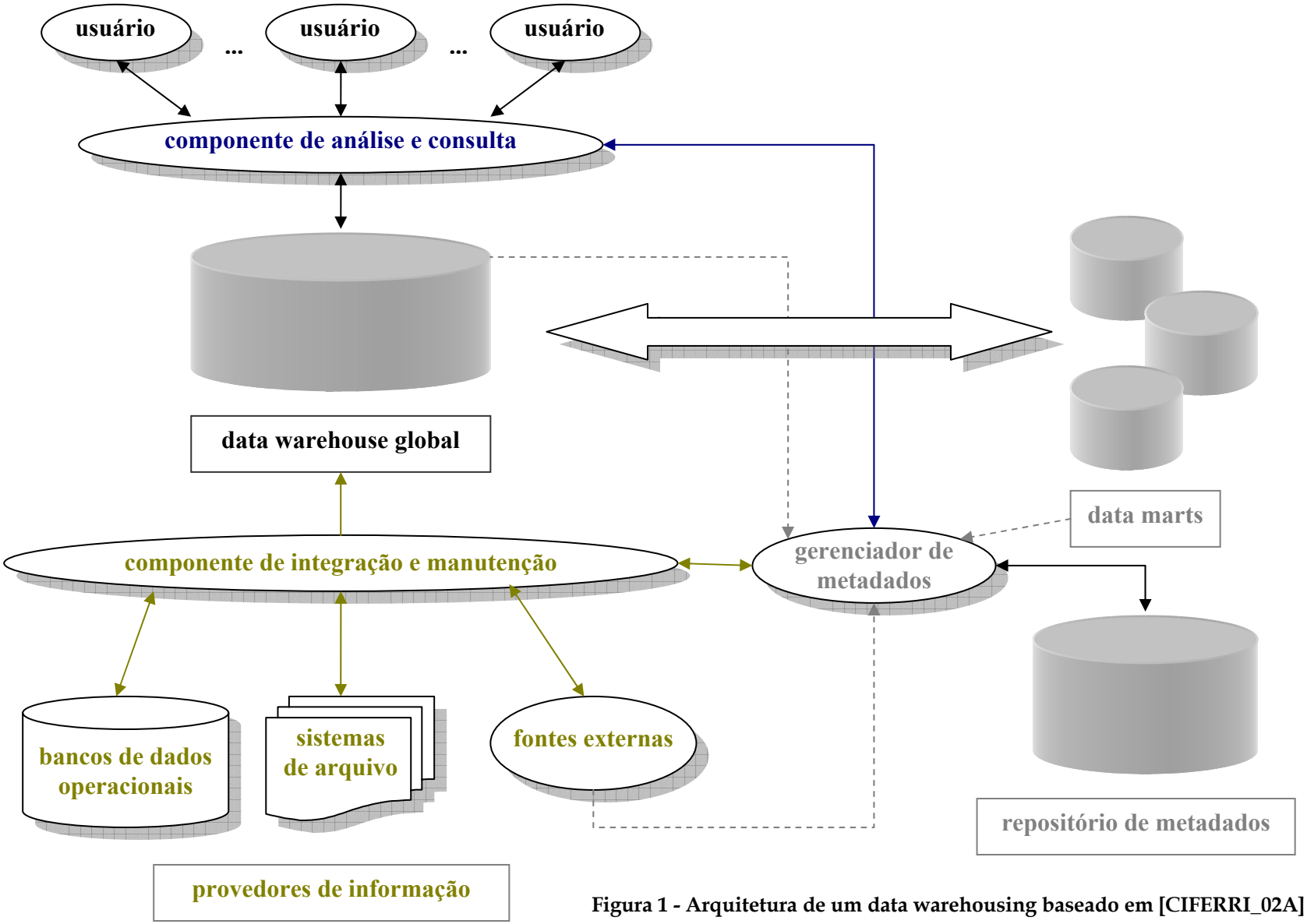

Figura 1 - Arquitetura de um data warehousing baseado em [CIFERRI_02A]

2.3.1. Componente de Integração e Manutenção

A primeira etapa na construção de um *data warehouse* é realizar a integração dos dados contidos nos mais diversos provedores de informação. De forma que na maioria das vezes é preciso manipular dados de diferentes fontes, como banco de dados relacionais, banco de dados orientados a objetos, sistemas de arquivo e, até mesmo, de páginas *web*. Isto torna este componente um dos mais complexos na elaboração de um *data warehouse*, sendo responsável por todo o processo de carregamento dos dados, que envolve: extração, tradução, limpeza e integração dos dados.

A extração dos dados consiste em todo o acesso aos provedores de informações e recolhimento dos dados contidos neles. Para isso, utilizam-se vários protocolos diferentes de comunicação com as diversas bases de dados. Em seguida é preciso fazer a tradução destes dados, pois a sua representação pode não estar adequada àquela que será utilizada pelo *data warehouse*. Assim, diferentes representações como HTML e XML devem ser convertidas para a representação adequada.

Após a extração e tradução dos dados, é preciso realizar a limpeza dos dados, isto é, precisa-se garantir que os dados que serão armazenados no *data warehouse* estejam corretos e obedeçam a um padrão de qualidade. Por exemplo, durante um processo de integração, uma organização pode simplesmente decidir por não armazenar os CPF inválidos (como '01234') dos clientes que foram inseridos em uma unidade da organização que não fazia a verificação do CPF.

Por fim, como etapa final do processo de carregamento, deve-se fazer a integração dos dados. Segundo Ciferri [CIFERRI_02], em um *data warehouse* existe

uma alta probabilidade de existir várias cópias da mesma informação em diferentes provedores, representadas da mesma maneira ou não, ou informações correlatas armazenadas em diferentes provedores que são inconsistentes entre si e esta diversidade de representações está relacionada, em primeiro lugar, ao fato de que as aplicações representadas por estes provedores foram criadas independentemente, de forma não coordenada e sem considerar que um dia seriam integradas e depois, porque a modelagem de cada aplicação varia de acordo com diversos fatores, tais como as necessidades, os objetivos e o universo de atuação de cada aplicação. Esses fatores determinam entidades próprias a cada domínio, e por fim, diferentes analistas podem possuir diferentes percepções do mundo real, originando, muitas vezes, modelos distintos para uma mesma aplicação.

Como exemplos para o que foi dito acima, um determinado registro de um cliente pode ser caracterizado por seu "nome", "CPF", "sexo" e "endereço" em uma localidade A e por "nome", "CPF", "sexo" e "Identidade" em uma localidade B. Neste registro, o nome da mesma pessoa poderia estar sendo representado por "Marcelo Costa" em um sítio e por "Marcelo V. C. S. Costa", no outro, devendo o processo de integração perceber que se trata do mesmo registro. Além disso, a representação do atributo "sexo" destes registros podem estar definidos como "M" e "Masculino", respectivamente. E por fim, as modelagens de uma entidade de mesmo nome, como "doutor" podem representar diferentes semânticas para representação, como um professor doutor, ou um médico.

Além de ser responsável por todo o processo de armazenamento, este componente realiza também a atualização e a expiração dos dados. A atualização consiste em estar, de tempos em tempos, verificando se os dados nos provedores de informações foram atualizados e propagando isto para o *data warehouse* global da empresa.

O algoritmo de fragmentação proposto neste trabalho não traria melhorias nas rotinas realizadas por este componente.

2.3.2. Componente de Análise e Consulta

Estando o *data warehouse* criado, é preciso oferecer rotinas para que se possa usufruir de sua principal funcionalidade que é dar suporte aos processo de tomadas de decisão através de consultas à base de dados. Esta interação é realizada através dos aplicativos que oferecem o acesso aos dados de maneira simples, através de interfaces amigáveis. Estas consultas executadas pelos usuários sofrem um processo de otimização, para que sejam executadas de forma a obter uma melhor performance.

Através da fragmentação vertical dos dados do *data warehouse*, estas consultas dos diversos usuários poderiam ser otimizadas para atuarem sobre estes fragmentos isoladamente, e assim, poderiam ser executadas paralelamente, gerando um ganho em desempenho, algo sempre procurado no desenvolvimento de sistemas computacionais. Para melhor exemplificar a fragmentação vertical dos dados em termos de medidas numéricas, um determinado gerente da organização precisa consultar informações relativas à vendas de determinada marca de seus produtos, que ficariam armazenadas em uma localidade, enquanto que outro gerente pode estar buscando pela quantidade de produtos em estoque de determinada marca, que estariam armazenadas em uma outra localidade diferente, possibilitando assim a execução paralela das duas consultas sobre os fragmentos gerados em função das medidas numéricas vendas e quantidade.

2.3.3. Data Marts

Um *data mart* é um pequeno *data warehouse* que fornece suporte à decisão de um pequeno grupo de pessoas [INMON_95]. Apesar de utilizar um conjunto de dados restrito em relação ao *data warehouse*, apresenta as mesmas características oferecidas por este, ou seja, uma coleção de dados orientada por assunto, integrada, variante no tempo, e não volátil, que tem por objetivo dar suporte aos processos de tomada de decisão.

Apresenta um custo inferior para o seu desenvolvimento em relação ao do *data warehouse* e pode ser visto como um conjunto de soluções particionadas em termos das áreas de negócios das organizações, de forma que se pode ter *data marts* relativos a áreas como marketing, vendas e finanças. Sua construção pode seguir duas abordagens distintas: *top-down* e *bottom-up*.

Na abordagem *top-down*, os *data marts* são criados a partir do *data warehouse*, isto é, primeiramente constrói-se o *data warehouse* global, para só depois construir-se os *data marts*. Esta abordagem é defendida por Bill Inmon [INMON_97], considerado um dos precursores do *data warehouse*. Já na abordagem *bottom-up*, que tem como seu principal defensor Ralph Kimball [KIMBALL_99], os *data marts* devem ser construídos de forma a representarem as diversas áreas setoriais das organizações e, após a criação deles, parte-se para a construção do *data warehouse* que deverá representar o conjunto de todas áreas das organizações.

2.3.4. Gerenciador e Repositório de Metadados

Constituem a estrutura do *data warehouse*, fornecendo embasamento para todo o processo de criação, manutenção, acesso e gerenciamento do ambiente de *data warehousing*.

O gerenciador de metadados cuida de todo o acesso ao repositório de metadados por parte dos outros componentes, como o de consulta, que precisa conhecer previamente a estrutura do *data warehouse* para gerenciar as suas transações de forma adequada.

Já o repositório de metadados contém o esquema dos provedores de informação, regras de mapeamento utilizadas para a solução de problemas de heterogeneidade existentes entre os dados dos diversos provedores de informação que participam do ambiente, informações relacionada à linguagem de dados, dados sobre a freqüência das consultas e sobre o desempenho do sistema [CIFERRI_02A].

2.4. Data Warehouse em Alta

Após ter definido os conceitos que envolvem um *data warehouse*, é importante que se saiba que estes ambientes estão em alta na atualidade, isto porque vêm sofrendo um crescimento vertiginoso em todos os sentidos:

- **No volume de dados que são manipulados:** A cada dia, um volume maior de dados é manipulado pelos *data warehouses*, isto se deve principalmente a necessidade da organizações de estarem tratando com mais dados, além do

fato de o *data warehouse* ter a característica de ser não volátil. Muitas vezes, o envelhecimento dos dados, que é retirar uma parte mais antiga dos dados e armazená-los em fitas de backup de alta capacidade, pode prejudicar o processo de tomada de decisão;

- **Na quantidade de serviços e produtos oferecidos:** Cada vez mais produtos de software, processamento de hardware e mão-de-obra especializada são oferecidos ao mercado a partir *data warehouse*. Oferecendo assim um melhor custo-benefício para implantação destes;

- **Na sua escolha pelas empresas de tecnologia da informação:** Está se tornando bastante comum a escolha de oferecer soluções para ambientes de *data warehousing* por parte das empresas de TI, isto porque elas estão enxergando esse nicho de mercado como sendo muito lucrativo e com grandes possibilidades de crescimento;

- **Na quantidade de usuários que utilizam estas aplicações:** É cada vez maior o total de usuários que percebem no *data warehouse* uma ferramenta de auxílio aos seus processos de tomada de decisão. Gerando assim uma demanda pela construção de ambientes de *data warehousing* em suas organizações.

Com todo esse crescimento, é importante que uma metodologia de desenvolvimento visando aumentar o desempenho desses sistemas seja desenvolvida, visto que, com maior número de acessos aos dados do *data warehouse*, as consultas dos usuários podem perder desempenho. E isto não é tolerado pelas grandes organizações, que para estarem sempre a frente de seus concorrentes precisam ter acessos aos dados de maneira veloz e com isso, realizar tomadas de decisões estratégicas. Conforme diz o sábio ditado popular: "tempo é

dinheiro". E qual seria a solução para isso? Criar ambientes de *data warehousing* distribuídos, como o WebD²W, que será tratado detalhadamente em tópicos futuros.

2.5. Ambiente de Data Warehousing Distribuído

Existem diversas vantagens que um ambiente assim poderia trazer em termos de melhorias para as organizações que trabalham com ambientes de *data warehousing*. Primeiramente, um ambiente desse poderia aumentar a disponibilidade[3] dos dados do *data warehouse*. Através da fragmentação dos dados, conseqüentemente, sua disponibilidade estaria sendo aumentada, pois os dados seriam distribuídos em diversos *sites*.

Um ambiente distribuído poderia também aumentar a disponibilidade do acesso aos dados do *data warehouse*, de forma que a utilização de técnicas de replicação de seus fragmentos tornariam mais disponível o acesso aos dados pelos usuários do *data warehouse*.

Poderia oferecer um aumento de desempenho durante o processamento de consultas. Como os dados estariam distribuídos, as suas consultas não mais iriam sobrecarregar o *data warehouse* global do sistema, já que diversas consultas dos usuários poderiam ser executadas em suas próprias unidades de distribuição caso possível, ou através de junções e agregações dos dados distribuídos. Assim, o gargalo no *data warehouse* global seria diminuído, e menos transações esperariam para serem executadas.

[3] É a probabilidade de um dado estar continuamente disponível em um dado intervalo de tempo.

Além disso tudo, poderia também oferecer suporte a um grande número de usuários através da utilização da *web* como meio. Assim, um grande número de usuários poderiam realizar suas consultas e conseqüente acesso aos dados através da Internet.

Entretanto, é preciso que um ambiente de *data warehousing* distribuído e integrado na *web* supra algumas necessidades. Ele deve, por exemplo, garantir as consistências intra *site* (da própria unidade de distribuição) e entre *sites* (entre as unidades de distribuição).

É importante que esses ambientes garantam também as transparências na manipulação dos dados, isto é, as transparências de fragmentação, replicação e localização. Na transparência de fragmentação, o usuário não precisa saber a forma em que o dado está fragmentado, ou melhor, ele não precisa nem saber se o dado está fragmentado no ambiente de *data warehousing* distribuído. Na transparência de replicação, o usuário não precisa saber se existem réplicas dos dados no ambiente de *data warehousing* distribuído, de forma que ao realizar uma consulta ele não deve saber se está acessando o dado ou uma réplica dele. E por fim, na transparência de localização o usuário não precisa saber em que *site* se encontra o dado no ambiente de *data warehousing* distribuído.

2.6. Segurança de um Data Warehouse Fragmentado

O ambiente de *data warehousing* contém informações valiosas e muitas vezes confidenciais da organização. Desta forma, ele necessita que algum mecanismo de segurança seja implementado, evitando que usuários sem autorização acessem os dados do *data warehouse*.

Entretanto, segundo Priebe e Pernul [PRIEBE_00] ambientes de *data warehousing* enfrentam um conflito de segurança, já que se por um lado a principal meta destes ambientes é tornar todos os dados necessários acessíveis aos usuários dos sistemas de suporte à decisão da maneira mais fácil possível, por outro, os dados destes ambientes são muito valiosos, gerando a demanda por medidas de segurança efetivas.

Segundo Singh [SINGH_97], para contornar este problema os dados podem ser criptografados e só poderão ser acessados por quem tiver a chave de decodificação. Em um ambiente de *data warehousing* distribuído, apenas seria necessário implementações de tais procedimento nos fragmentos que requeressem uma maior proteção, diminuindo assim os custos relativos à implantação de mecanismos de segurança. Porém, caso uma grande parte dos fragmentos necessite de um procedimento de segurança refinado ocorreria uma maior complexidade em garantir segurança, o que poderia elevar os custos do projeto do ambiente de *data warehousing* distribuído.

Também é preciso garantir a segurança física dos dados, através de rotinas de backup, de armazenamento remoto, de replicação e hardware tolerante a falhas, protegendo assim a perda dos dados devido a falhas de energia ou mau funcionamento do hardware, por exemplo. A realização de backup é uma tarefa importante mais ao mesmo tempo tende a ser um problema para os administradores de *data warehouse*, pois a quantidade de dados a serem salvos durante o procedimento de backup, geralmente, é muito grande, podendo consumir muito tempo. Através da utilização de um *data warehouse* fragmentado a tarefa dos administradores seria facilitada, visto que eles poderiam atuar de maneira particular sobre cada um destes fragmentos.

2.7. Ambiente de Data Warehousing Virtual

O ambiente de *data warehousing*, além de apresentar a estrutura convencional, pode apresentar uma abordagem virtual. Nela, os dados não ficam armazenado de forma persistente no *data warehouse*, ou seja, são carregados em memória à medida que os usuários solicitam as consultas. São boas soluções quando a integração dos dados contidos nos provedores de informação puder ser feita de forma simples, porém, quando as informações contidas nos provedores de informação necessitarem de rotinas de carregamento mais rebuscadas, essa abordagem não é aconselhável, pois degradaria o desempenho. Sua principal vantagem é a ausência de rotinas de manutenção da consistência, uma vez que à medida que os dados contidos nas bases de dados operacionais forem atualizados, as consultas executadas pelos usuários já retornaram o dado atualizado, enquanto que no *data warehouse* convencional, esta atualização só seria propagada em um momento oportuno.

O esquema de um ambiente de *data warehousing* virtual é exibido na figura 2, onde o controle de acesso aos dados é feito através de um componente denominado mediador, cuja função consistem em processar as consultas dos usuários e adequá-las em diferentes linguagens de consulta para que possam ser aplicadas aos diversos provedores de informações. Os dados obtidos podem estar representados de diferentes formas. Então, após obter todos resultados consultados é necessário também um tratamento sobre estes dados antes de retorná-los aos usuários.

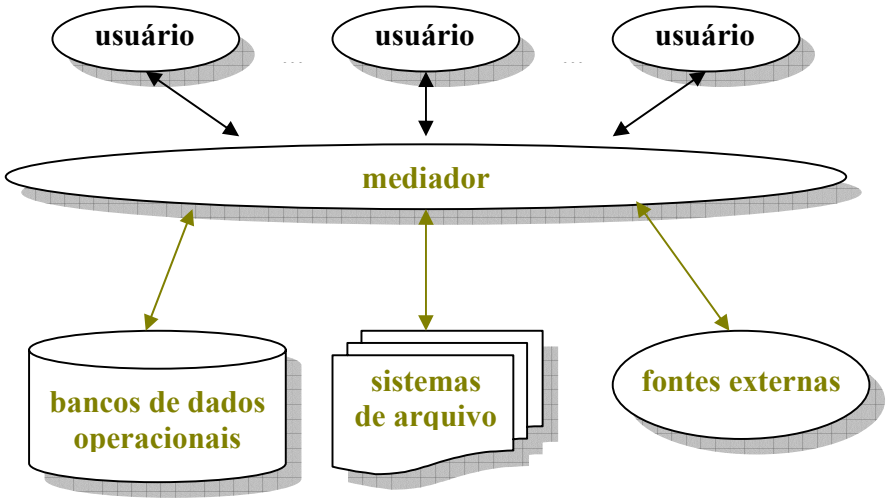

Figura 2 - Esquema de um data warehousing virtual

2.8. Modelo de Dados Multidimensional

Em um ambiente de *data warehousing*, as análises efetuadas pelos usuários de SSD[4] representam, de maneira geral, requisições multidimensionais aos dados do *data warehouse*, as quais têm por objetivo a visualização dos dados segundo diferentes perspectivas, denominadas de dimensões [CIFERRI_02A].

É de fundamental importância entender o conceito de modelagem multidimensional, uma vez que representa a maneira como os dados do *data warehouse* podem ser agregados de diferentes formas.

Para entender este conceito, nada melhor do que um exemplo. Considere-se as dimensões região, filial, mês e vendas. Uma consulta agregando estas entidades pode ser visualizada na figura 3, de forma que se está buscando o somatório das

[4] Sistemas de Suporte à Decisão.

vendas no mês 2, da filial 1, da região 5. Como resultado terem-se um cubo de dados tridimensional, já que a visão[5] vendas está sendo consultada por três diferentes perspectivas.

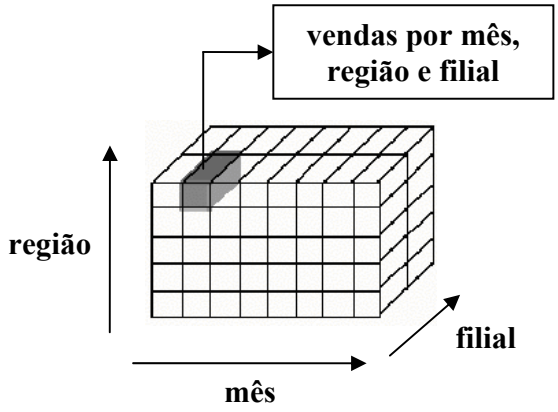

Figura 3 - Cubo de dados tridimensional

Caso a agregação ficasse restrita para visualizar as vendas no mês 2 e região 5, o resultado seria apenas a face do cubo, que resultaria no somatório das vendas daquele período e região, como pode ser visto na figura 4. Resultando assim em um cubo de dado bidimensional, pois a visão vendas é consultada sob duas dimensões, região e mês. Caso a visão fosse consultada sob mais de três perspectivas, o cubo de dados resultante seria n-dimensional, onde n representa o número de dimensões analisadas.

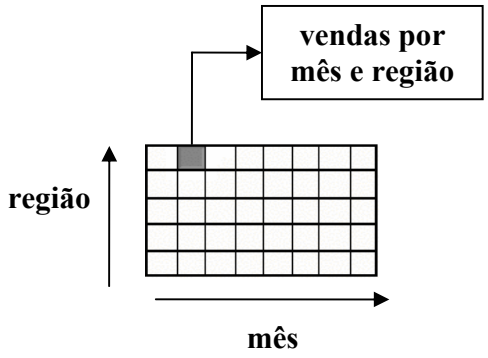

Figura 4 - Cubo de dados bidimensional

[5] Dado que o usuário analisa para dar suporte aos seus processos de tomada de decisão.

3. O SISTEMA WEBD²W

O sistema WebD²W (*Web Distributed Data Warehousing*) consiste em um ambiente de *data warehousing* distribuído cliente-servidor que visa não somente a distribuição dos dados do *data warehouse*, mas também o acesso distribuído a esses dados usando a tecnologia *web* como infra-estrutura [CIFERRI_02B].

Desenvolvido com o intuito de suprir as necessidades citadas em tópicos anteriores, este sistema apresenta uma arquitetura de um ambiente de *data warehousing* distribuído, porém enfatiza todo o processo de fragmentação, principal responsável pelo aumento da disponibilidade dos dados. Este processo de fragmentação é realizado pelo componente de distribuição, e é nele onde ficará situado o algoritmo de particionamento vertical dos dados proposto neste trabalho.

Para que se possa entrar em detalhes da implementação do algoritmo, é importante que seja apresentada toda a arquitetura do sistema e seus componentes, para que se possa construir uma visão clara do funcionamento deste ambiente de *data warehousing,* representado na figura 5.

3.1. Usuários do Sistema

O sistema é composto por cinco classes de usuários:

- Projetista do *data warehouse* global - É responsável por todo o projeto do *data warehouse* global, devendo estabelecer a estrutura dos dados e a forma com que os dados serão integrados dos diversos provedores de informação;

- Administrador do *data warehouse* global - Realiza a administração do *data warehouse* global, sugerindo modificações quando necessárias e controles de segurança, desempenho, entre outros;

- Projetista do *data warehouse* distribuído - Deve ser o responsável pela construção do ambiente distribuído, estabelecendo a metodologia a ser utilizada baseada em seu requisitos através do módulo de componente de distribuição, determinando assim a maneira como os dados serão fragmentados. O desenvolvimento de um algoritmo para realizar a fragmentação vertical dos dados facilita todo o processo de distribuição elaborado pelo projetista do *data warehouse* distribuído;

- Administrador do *data warehouse* distribuído - Fornece otimizações ao componente de consulta do ambiente distribuindo e deve manter a consistência entre as unidades de distribuição e o *data warehouse* global;

- Usuários finais - Esta classe de usuário é a principal beneficiada com as vantagens oferecidas pelo ambiente distribuído e são os responsáveis pela realização das consultas que vão auxiliá-los nos processos de tomada de decisão.

3.2. Arquitetura

A arquitetura proposta no sistema WebD²W é um pouco semelhante à arquitetura de um ambiente de *data warehouse* comum, porém, a inclusão dos componentes de distribuição e manutenção do ambiente distribuído faz com que os outros componentes apresentem modificações em sua estrutura interna, apresentando assim, comportamentos diversos do padrão, resultando uma maior complexidade no desenvolvimento de cada componente. Entretanto, algoritmos, como o proposto aqui, podem agilizar o processo de elaboração dos componentes. Na figura 5 encontra-se o esquema da arquitetura deste sistema.

Esta arquitetura propõe uma abordagem *top-down* na construção do ambiente de *data warehousing* distribuído, isto é, primeiramente constrói-se o *data warehouse* global para só depois serem criadas as unidades de distribuição. Essas unidades de distribuição geradas podem ser vistas ou não como sendo semanticamente equivalentes a *data marts*, isto vai depender dos critérios utilizados como base para a distribuição dos dados [CIFERRI_02A]. Caso seja utilizada uma abordagem semântica para dividir os dados, como "dados de marketing", "dados de vendas", entre outros, esses dados são equivalentes a departamentos das organizações, e assim são considerados *data marts*. Já se a abordagem para distribuição utilizada for outra, como aumento da performance, ou segurança, não importando a maneira com que a semântica será utilizada, essas unidades de distribuição resultantes não serão equivalentes a *data marts*.

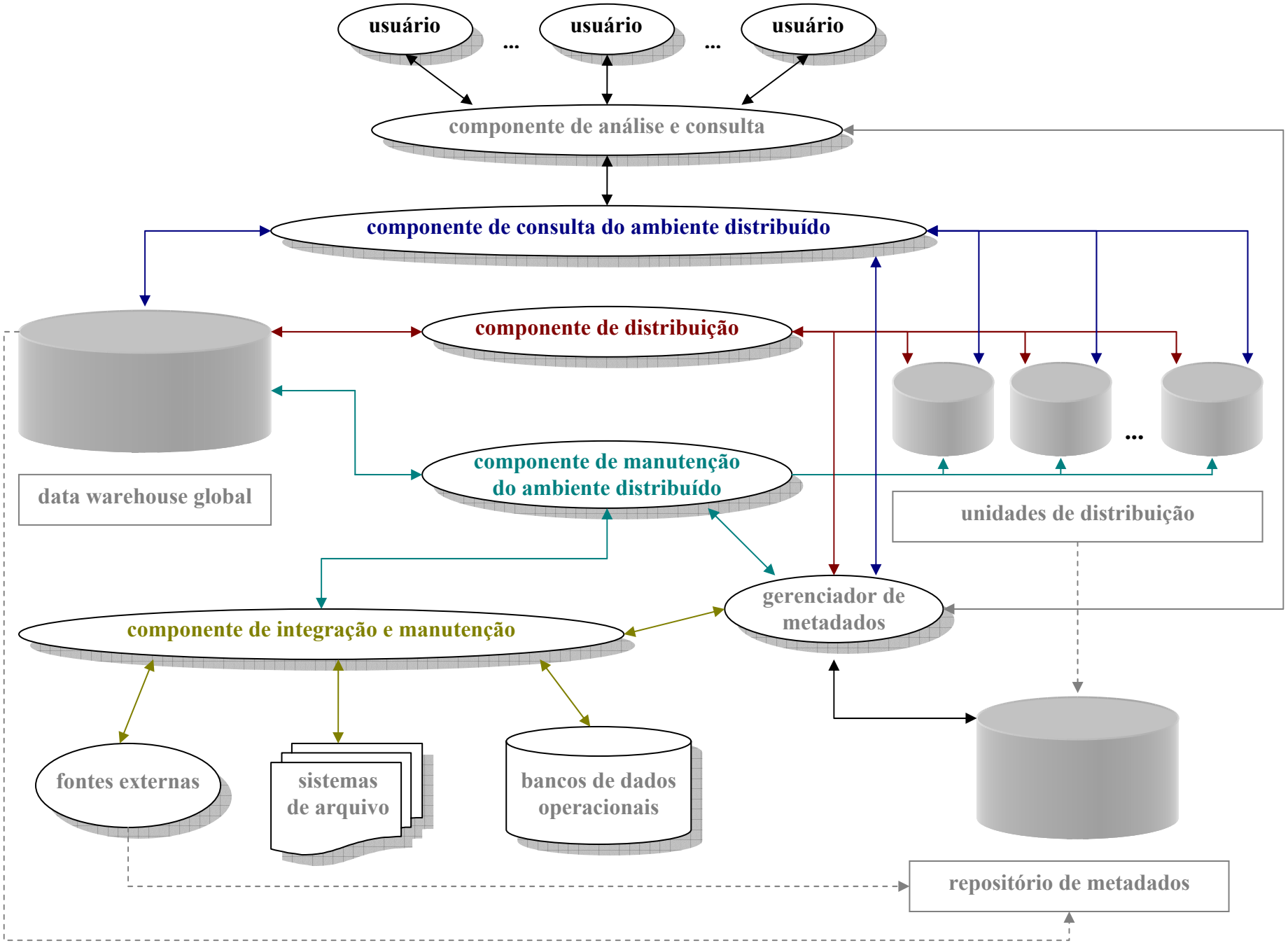

Figura 5 - Arquitetura do Sistema WebD2W baseado em [CIFERRI_02A]

Durante os estudos desenvolvidos para elaboração deste trabalho, cogitei a possibilidade de remover o *data warehouse* global, pois apresentaria algumas vantagens como a simplificação do componente de manutenção do ambiente distribuído, visto que qualquer dado modificado em uma das unidades de distribuição deve ser propagado para o *data warehouse* global, e com sua eliminação, apenas seria necessário propagar os dados atualizados entre suas réplicas. Além disso, surgiria a possibilidade de união entre o componente de distribuição com o de integração e manutenção, de forma que uma única rotina precisaria ser implementada ao invés de duas.

Porém, estas vantagens não foram suficientes frentes aos diversos fatores para não se fazer isso. O *data warehouse* global representa o banco de dados mais atualizado do sistema, servindo assim de fundamentação para adição de novas unidades de distribuição sugeridas pelo administrador do *data warehouse* distribuído quando necessárias. Além disso, ele contém os dados já previamente extraídos, traduzidos, filtrados e integrados. Sabe-se que o carregamento dos dados é um processo de atividades extremamente caras, complexas e demoradas, de forma que caso fosse realizada por cada unidade de distribuição em particular, este processo se tornaria bem mais penoso, sendo necessárias assim diversas rotinas, tanto quanto fossem a quantidade de unidades de distribuição a serem geradas.

Para completar, sabe-se que as consultas OLAP[6] podem manipular de centena a milhares de registros durante suas execuções, e por isso elas são longas e complexas, e por demandarem uma grande quantidade de operações de varredura, junção e agregação [CIFERRI_02A]. Caso não existisse o *data warehouse* global, essas consultas poderiam ocasionar um grande volume de dados trafegando pela rede, já que elas teriam que ser realizadas em sub-consultas nas

[6] *On-Line Analytical Processing*. Consultas analíticas típicas de ambientes de data warehousing.

unidades de distribuição, e depois seria necessário uma junção e agregação desses dados, possibilitando assim um congestionamento na rede.

3.3. Componentes da Arquitetura

A seguir serão apresentados todos os cincos componentes que integram o sistema WebD²W, que são: componente de distribuição, componente de análise e consulta, componente de consulta do ambiente distribuído, componente de manutenção do ambiente distribuído e o componente de integração e manutenção.

O objetivo e funcionamento de cada um deles serão explicados. Entretanto, será dada uma ênfase maior ao componente de distribuição, pois é nele que estão situados os algoritmos responsáveis por fragmentar os dados.

3.3.1. Componente de Distribuição

Este componente visa aumentar a disponibilidade dos dados do *data warehouse*. Isto pode ser alcançado através de uma distribuição dos dados contidos no *data warehouse* global em diversas unidades de distribuição, dando assim suporte a um grande número de usuários, de forma que os dados contidos no *data warehouse* global são fragmentados, utilizando técnicas, metodologias e algoritmos como o citado neste trabalho. Em seguida, estes dados podem ser replicados e depois alocados em diferentes *sites*. Para realizar todo este processo, o componente de distribuição é composto por quatro módulos: requisitos, fragmentação, alocação e carga.

A seguir, através da figura 6, será apresentado o esquema da estrutura deste componente.

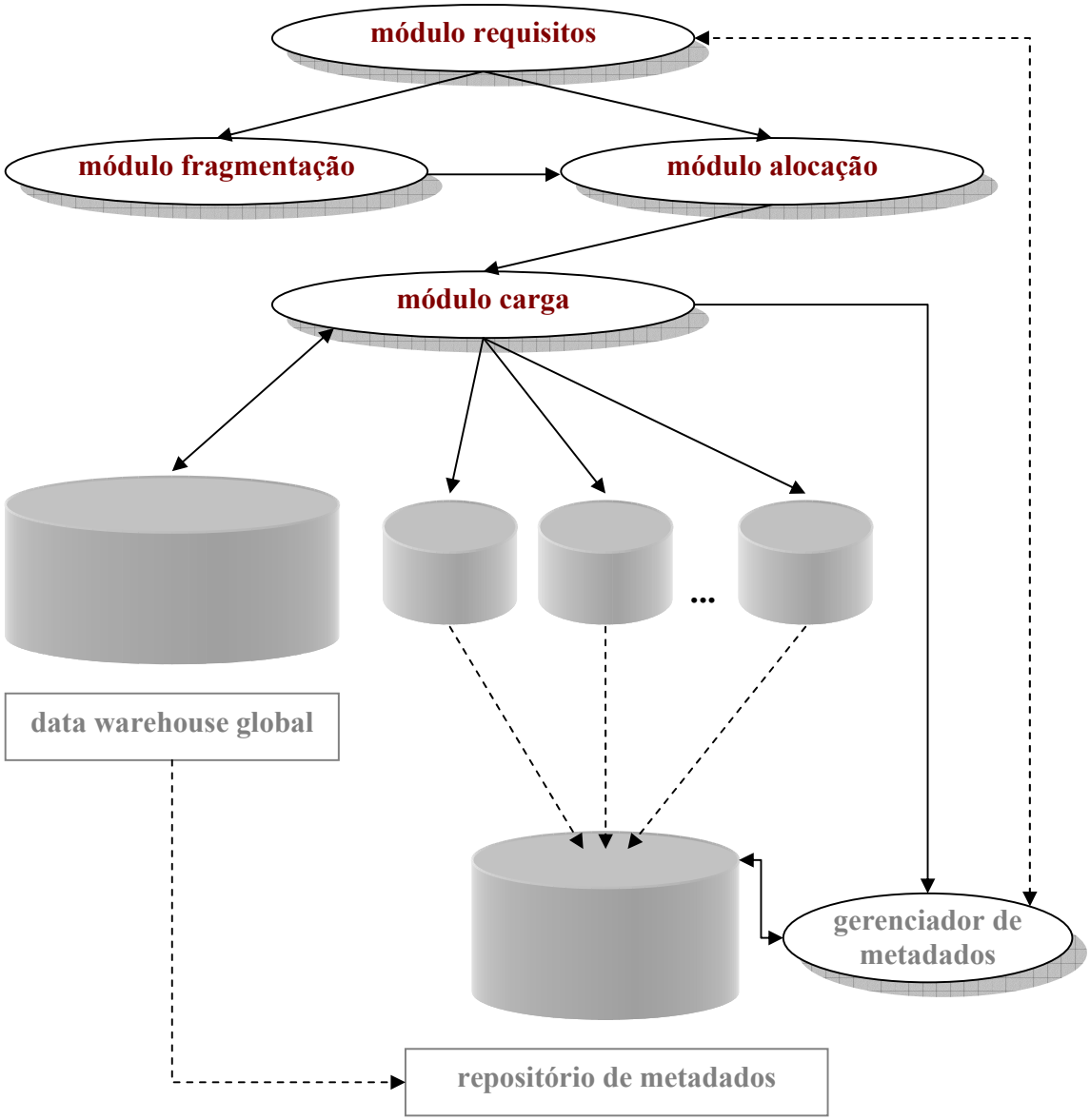

Figura 6 - Componente de distribuição

3.3.1.1. Módulo Requisitos

Responsável pela identificação de um conjunto de critérios que deve ser utilizado como base pelo projetista do *data warehouse* distribuído para definição de restrições a serem aplicadas aos processos de fragmentação, replicação e alocação dos dados do *data warehouse* global [CIFERRI_02A]. Assim, é preciso que o projetista tenha um conhecimento antecipado da carga de trabalho. Entretanto, um grande problema é que essa carga geralmente é dinâmica e inconstante. Por isso, é de extrema importância não só uma análise atual, mas também uma projeção do cenário futuro. Como requisitos para o processo de fragmentação e alocação tem-se que obter informações acerca dos sistemas computacionais, da rede de comunicação, da carga de trabalho e do banco de dados.

As informações dos sistemas computacionais requerem um conhecimento dos *sites* que serão utilizados pelo ambiente distribuído para alocação das unidades de distribuição: características como nível de segurança oferecido, capacidade de processamento de hardware e armazenamento de dados, softwares utilizados, entre outros. Deve-se obter também informações da rede de comunicação, como largura de banda da rede (pois é importante colocar dados muito acessados em *sites* de grande largura de banda para melhor desempenho) segurança da rede, como utilização de firewalls[7], entre outros.

Já as informações a respeito da carga de trabalho consistem em saber qual porção de dados é acessada mais freqüentemente para que uma análise sobre eles seja traçada e possam resultar em uma fragmentação e alocação adequadas, e que realmente aumentem a disponibilidade dos dados consideravelmente. E por fim

[7] Pontos de conexão entre duas redes não confiáveis que permitem que a comunicação entre elas seja monitorada e segura.

informações relativas ao banco de dados, como as estruturas e características do banco de dados de maneira geral.

3.3.1.2. Módulo Fragmentação

Após serem estabelecidos os requisitos, é preciso dar continuidade ao processo de distribuição. O objetivo deste componente é particionar os dados do *data warehouse* global em subconjuntos desses dados usando metodologias e algoritmos de fragmentação que podem ser horizontal[8], vertical e mista, que utiliza as duas técnicas anteriores, mais sempre uma por vez, ou se fragmenta horizontalmente para depois fazer o particionamento vertical, ou o contrário. Segundo Özsu e Valduriez [ÖZSU_99], mesmo para bancos de dados relacionais, não existem algoritmos integrados que fragmentam uma relação global em um conjunto de fragmentos, alguns dos quais decompostos horizontalmente e outros decompostos verticalmente.

Este módulo fará uso do algoritmo aqui proposto para gerar os fragmentos verticais que servirão de entradas para o módulo de alocação. De acordo com [ÖZSU_99], o objetivo da fragmentação vertical é particionar uma relação em um conjunto de relações menores, de tal modo que muitos dos aplicativos do usuário possam atuar apenas sobre um fragmento.

3.3.1.3. Módulo Alocação

Tem o objetivo de identificar quais *sites* e, conseqüentemente, quais unidades de distribuição, devem armazenar quais porções de dados de *data warehouse* global [CIFERRI_02A]. Porções de dados estas já fragmentadas pelo

[8] Particionamento dos dados segundo um critério estabelecido. Cada fragmento gerado representa tuplas da relação original.

módulo anterior. Este módulo também deve estabelecer quais os fragmentos resultantes do módulo fragmentação que devem ser replicados de forma a aumentar a disponibilidade e eficiência. Entretanto é preciso ter cuidado ao se replicar dados, pois pode aumentar a manutenção dos dados, principalmente no caso deles serem constantemente atualizados.

De acordo com Elmasri e Navathe [ELMASRI_00], o problema de alocação consiste em um problema de otimização muito complexo. Já que em adição aos custos de manutenção, o problema de alocação também deve considerar os custos de armazenamento dos fragmentos nos *sites*, os custos associados ao processamento das consultas e os custos de transmissão [CIFERRI_02A].

3.3.1.4. Módulo Carga

Após estabelecer a fragmentação dos dados e sua respectiva alocação, é necessário realizar o carregamento inicial dos dados das unidades de distribuição, tarefa esta executada pelo módulo carga. Essa transferência pode ser feita via *web* ou de forma indireta quando o volume de dados inviabiliza o uso da rede. Este componente pode também realizar o carregamento dos dados quando novas unidades de distribuição são criadas.

3.3.2. Componente de Consulta

Seu principal objetivo é aumentar a disponibilidade de acesso aos dados do *data warehouse* e para isso apresenta duas perspectivas na realização das consultas:

- Acesso local - A consulta do usuário será realizada na própria unidade de distribuição em que ele se encontra. A verificação se a consulta pode ser realizada localmente é feita através do acesso ao gerenciador de metadados;

- Acesso global ao ambiente distribuído - A consulta do usuário será realizada ou nas diversas unidades de distribuição ou no *data warehouse* global. Deve combinar custos de comunicação com gargalos de acesso aos dados. Isto é, não se pode sobrecarregar consultas ao *data warehouse*, e nem usar excessivamente os componentes de distribuição, para evitar uma sobrecarga na rede. Por isso, este componente deve fazer um gerenciamento acerca das consultas realizadas.

3.3.3. Componente de Manutenção

Visa manter a consistência dos dados do *data warehouse* distribuído, isto é, tanto a consistência dos dados do *data warehouse* global, quanto a consistência dos dados das unidades de distribuição.

3.3.4. Componente de Integração

Realiza a integração dos dados recolhidos das diversas fontes de informação, fazendo todo o processo de extração, tradução, limpeza, filtragem e integração dos dados. Trata-se do componente mais complexo da arquitetura. É preciso que se consiga uma granularização adequada dos dados, de forma que não se armazene dados demais, pois o custo de capacidade é alto e pode prejudicar o desempenho, mas também não se pode resumir demais os dados, pois isto poderia prejudicar o processo de tomada de decisão pelos gerentes e executivos das organizações. Deve-se fazer um planejamento bem estruturado, pois falhas na implementação deste componente podem levar o projeto do *data warehouse* ao fracasso e colocar a empresa em desvantagem diante de um mercado que está cada vez mais competitivo.

4. ALGORITMOS DE FRAGMENTAÇÃO VERTICAL

Nesta seção será apresentado o algoritmo motivador deste trabalho. Este algoritmo faz parte integrante do componente de distribuição, mais especificamente do módulo de fragmentação, da arquitetura do sistema WebD²W.

O algoritmo aqui proposto deve ser utilizado para fragmentação vertical de *data warehouses* e baseia-se no conceito de grafos de derivação, já que segundo Ciferri [CIFERRI_02] está é uma maneira apropriada de representação dos dados de um *data warehouse*. Porém, antes de apresentar o algoritmo, é preciso que se esclareçam alguns conceitos como o de fragmentação vertical, grafos de derivação e medida numérica.

4.1. Fragmentação Vertical

Uma definição de fragmentação vertical mais detalhada é que uma dada relação R produz fragmentos R^1, R^2,... R^n, de forma que cada um deles contém um subconjunto de atributos de R, bem como a chave primária de R [MUTHURAJ_92]. Isto é, além de todas relações possuírem a mesma chave primária, cada relação tem seu próprio conjunto de atributos, de forma que caso seja necessário acessar todos os dados da relação original, uma operação de junção pode ser efetuada sobre cada um de seus fragmentos.

O principal benefício gerado pela fragmentação vertical é que transações distintas que precisem acessar os dados da relação original sem sobreposição,

podem executar em paralelo [RA_90]. De forma que cada fragmento contém atributos que são semanticamente relacionados de alguma maneira.

Uma fragmentação vertical deve obedecer às regras de corretude, que são completude, disjunção e reconstrução. Uma estratégia de fragmentação vertical é completa se todo atributo da relação global original pode ser encontrado em algum fragmento da relação. A regra de reconstrução é satisfeita se a partir dos fragmentos gerados pode-se obter novamente a relação original, na fragmentação vertical isto é feito por uma junção dos fragmentos utilizando a chave primária. Por fim, a regra de disjunção estabelece que ao se realizar a junção dos dados estes não podem ser repetidos. Entretanto esta última regra não se aplica ao pé da letra à fragmentação vertical, uma vez que a chave primária precisa ser replicada em todos os fragmentos para garantir a reconstrução. Dessa forma, excluindo a chave primária, a disjunção é garantida se nenhum item de dado é encontrado em mais de um fragmento. As regras de corretude são verificadas no algoritmo proposto.

Os Quadros 1, 2 e 3, exemplificam uma fragmentação vertical simples em uma base de dados relacional. De forma que a primeira representa o esquema global e as outras duas os respectivos fragmentos.

Quadro 1 - Esquema simples de uma tabela a ser fragmentada

COD_CLIENTE	NOME_CLIENTE	FONE_CLIENTE
0000001	Marcelo Victor Calado	3434.4343
0000002	Fernando da Fonseca	3131.3232

Quadro 2 - Fragmento Vertical 1

COD_CLIENTE	NOME_CLIENTE
0000001	Marcelo Victor Calado
0000002	Fernando da Fonseca

Quadro 3 - Fragmento Vertical 2

COD_CLIENTE	FONE_CLIENTE
0000001	3434.4343
0000002	3131.3232

Em uma fragmentação horizontal, uma condição deveria ser imposta, como por exemplo, nome ≤ 'f' e nome > 'f', que resultariam nos fragmentos dos Quadros 4 e 5.

Quadro 4 - Fragmento horizontal com restrição nome > 'f'

COD_CLIENTE	NOME_CLIENTE	FONE_CLIENTE
0000001	Marcelo Victor Calado	3434.4343

Quadro 5 - Fragmento horizontal com restrição nome ≤ 'f'

COD_CLIENTE	NOME_CLIENTE	FONE_CLIENTE
0000002	Fernando da Fonseca	3131.3232

4.2. Algoritmos que Abordam a Fragmentação Vertical

Vários algoritmos de particionamento vertical tem sido sugeridos, embora muitos deles não tenham aplicabilidade para ambientes de *data warehousing*. Para o desenvolvimento do algoritmo aqui proposto, foi preciso realizar um estudo dos algoritmos já existentes acerca deste assunto na literatura.

Hofer e Severance [HOFER_75] medem a afinidade entre pares de atributos e tentam alocar atributos semanticamente semelhantes em pares utilizando algoritmos de BEA (*Bond Energy Algorithm*). Hammer e Niamir [HAMMER_79] usam uma abordagem de análise de custo e uma heurística[9] para chegar a um particionamento vertical através da abordagem *bottom-up*. Já Navathe [NAVATHE_84] estende a pesquisa através do BEA e propõe um algoritmo de particionamento vertical em duas fases. Em seguida, Cornell e Yu [CORNELL_87] aplicam o trabalho realizado por Navathe no design físico de um banco de dados relacional. Ceri, Pernici e Wiederhold [CERI_89] também estendem o trabalho de Navathe, mas consideram esta técnica como apenas uma técnica de "divisão", e então adicionam a técnica de "conquista", formando o já conhecido método para resolução de problemas "dividir para conquistar".

Todos os algoritmos citados acima baseiam-se em técnicas heurísticas para criação dos fragmentos, e utilizam como entrada uma matriz de utilização de atributos (*Attribute Usage Matrix* - AUM). Esta matriz indica quais atributos são utilizados por quais transações, e da informação de freqüência de cada consulta, e isso é relevante no momento de execução do algoritmo, para que ele possa gerar fragmentos que aumentem a disponibilidade dos dados. Os algoritmos mais

[9] Método de perguntas e respostas para encontrar a solução de vários problemas.

recentes focam uma outra abordagem de entrada, que é a matriz de afinidade dos atributos (*Attribute Affinity Matrix* - AAM), que é uma derivação da AUM. Nela, é indicada a afinidade entre pares de atributos da relação, que é determinada pela freqüência de acesso a esses pares.

Em adição aos algoritmos de fragmentação vertical, diversos autores propuseram técnicas de alocação dos fragmentos. Uma proposta que poderia ter aplicabilidade ao algoritmo aqui proposto é a técnica criada por Zahn, que se baseia no conceito de grafos de derivação [ZAHN_95].

4.3. Grafos de Derivação

De acordo com Diestel [DIESTEL_97], um grafo G (V, A) é definido como o par de conjuntos V e A, de forma que V representa o conjunto não vazio dos vértices do grafo e A o conjunto de arestas do grafo. Um grafo de derivação apresenta a característica de ser orientado, isto é, acíclico, ou seja, ele não possui ciclos assim com não possui arestas múltiplas, ou seja, existe uma única aresta entre os mesmos dois vértices do grafo.

Antes de definir o conceito de grafo de derivação é preciso que se apresente o conceito de *lattice*. Um *lattice* de visões agregadas é definido através das seguintes propriedades [HARINARAYAN_96]:

- Existe uma ordenação parcial ├ entre as agregações no *lattice*. Para as visões agregadas X e Y, X ├ Y se e somente se X pode ser determinada usando somente os resultados de Y. Em outras palavras, a agregação X é dependente da agregação Y;

- Existe uma visão topo, da qual cada outra visão agregada é dependente. Mais especificamente, esta visão é utilizada para derivar qualquer outra visão no *lattice*;

- Existe uma visão completamente agregada vazio que pode ser calculada a partir de qualquer outra visão no *lattice*. A visão vazio representa a visão completamente agregada.

Um grafo de derivação G consiste em um grafo orientado no qual V(G) representa um conjunto de visões agregadas (medidas numéricas agregadas), ao passo que A(V) representa um conjunto de relações de dependência ├ entre essas visões agregadas [CIFERRI_02A]. Assim, se a aresta A ∈ G representa uma relação de dependência entre dois vértices X e Y, ambos ∈ G, de forma que X ├ Y, então vértice inicial(A) = Y e o vértice terminal(A) = X.

Para solidificar o conceito apresentado, a figura 7 mostra um grafo de derivação representado pelas dimensões m, r e v, que poderia representar, marca, região e vendas. Através da figura, pode-se perceber também que o nível de agregação aumenta a medida em que se deriva o grafo original.

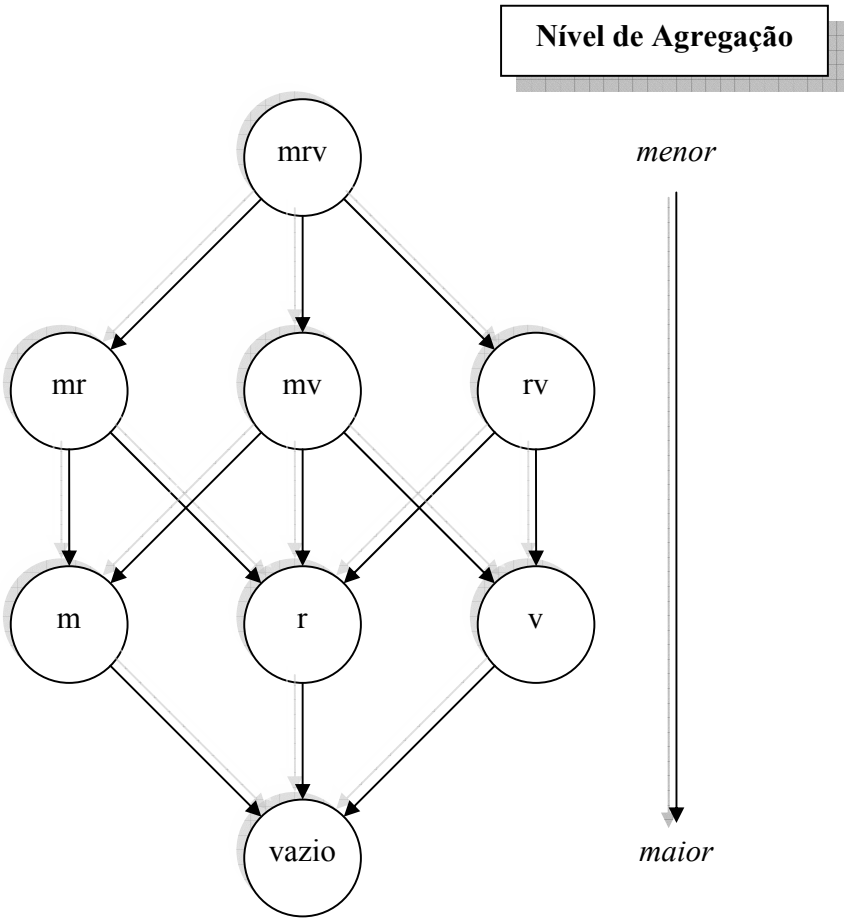

Figura 7 - Grafos de Derivação para as dimensões mrv

4.4. Medida Numérica

De acordo com Ciferri [CIFERRI_02A], uma medida numérica pode ser vista como uma função das suas dimensões correspondentes, representando, desta forma, um valor no espaço multidimensional.

Uma medida numérica pode representar, por exemplo, média de valor de um determinado produto por região e filial. Quanto mais restrições, menos

agregados os dados estarão. De forma que a quantidade de produtos por região é mais agregada do que a quantidade de produtos por região e cidade, pois este representa um conjunto maior de dados, fornecendo uma resposta mais detalha à consulta do usuário. As medidas numéricas podem ser consideradas aditivas, semi-aditivas e não-aditivas.

São consideradas aditivas quando a medida numérica pode ser somada em todas suas dimensões. Como exemplo, pode-se citar a medida numérica vendas, já que ela pode ser somada em função do mês, da filial, da região, e assim sucessivamente, assim se teria diferentes agregações que seriam a soma das vendas dos atributos que tivessem o mesmo valor. Para ficar mais claro de como funciona essa agregação seja o esquema simplificado mostrado no Quadro 6.

Quadro 6 - Esquema simplificado das dimensões de um data warehouse

Mês	Filial	Região	Vendas
Janeiro	Filial A	Nordeste	200
Janeiro	Filial C	Sudeste	150
Janeiro	Filial B	Nordeste	540
Janeiro	Filial B	Sudeste	190
Fevereiro	Filial A	Nordeste	300
Fevereiro	Filial C	Sudeste	410
Fevereiro	Filial B	Nordeste	270
Fevereiro	Filial B	Sudeste	520

De forma que uma agregação de vendas por mês resultaria na soma das vendas agrupadas por mês. A representação da agregação de uma medida numérica aditiva pode ser vista no Quadro 7.

Quadro 7 - Resultado de uma agregação vendas-mês

Mês	Vendas
Janeiro	1080
Fevereiro	1500

Já a agregação vendas por filial traria um resultado diferente para o somatório da medida numérica agregado pela filial, representado no Quadro 8.

Quadro 8 - Resultado de uma agregação vendas-filial

Filial	Vendas
Filial A	500
Filial B	1520
Filial C	560

As medidas numéricas semi-aditivas, assim como as aditivas podem ser somadas. A diferença consiste no fato de que esta adição não pode ser feita para todas as dimensões. Isto se deve porque para algumas dimensões não tem sentido fazer o somatório da medida numérica, ou até mesmo porque esta soma poderia resultar num valor incorreto em situações que poderiam ser contabilizadas mais de uma vez.

Por fim, tem-se as medidas numéricas não-aditivas, isto é, não podem ser somadas, pois não teria sentido algum realizar o somatório de seus valores por qualquer que fosse a dimensão escolhida. Como exemplo clássico, tem-se medida numérica preço unitário, de forma que qualquer que fosse a sua agregação, como por mês, por região ou por filial, o resultado não poderia ser sua soma. Medidas numéricas não-aditivas geralmente utilizam outras funções de agregação, como média, valor máximo, ou uma outra fórmula mais complexa previamente definida pela organização.

4.5. Fragmentação Vertical em Termos de Medidas Numéricas

Neste tópico será apresentado o algoritmo que é o objetivo principal deste trabalho e que realizará a fragmentação vertical de um *data warehouse* em termos de medidas numéricas baseados no conceito de grafos de derivação.

Na fragmentação vertical se uma relação possui n atributos que não são chaves primárias, o número de possíveis fragmentos é igual a B(n), onde B é o n-ésimo número de Bell[10]. Para valores muito grandes B(n) se aproxima de n^n. Com isso, torna-se inviável uma procura por soluções ótimas de fragmentação vertical, sendo necessário recorrer à investigação de heurísticas. Duas são as abordagens heurísticas propostas para fragmentação vertical: grupamento e repartição.

A técnica de grupamento consiste em começar com tantos fragmentos quanto forem os atributos e a cada passo reuni-los em fragmentos um pouco maiores, até que algum critério seja satisfeito, como no caso do algoritmo aqui proposto, agrupar as medidas numéricas. Por outro lado, a técnica de repartição consiste em iniciar com a relação inteira e particioná-la baseando-se no comportamento de acesso das aplicações.

Nas próximas seções serão apresentadas as entradas do algoritmo proposto, denominado de algoritmo de fragmentação vertical em termos de medidas numéricas (FV-MN), seu pseudocódigo e exemplo de aplicação.

[10] Número de possibilidades em que um grupo de n elementos podem ser particionados em subconjuntos não vazios.

4.5.1. Entradas do Algoritmo FV-MN

Entre as entradas necessárias para execução dos procedimentos do algoritmo, tem-se:

- Um grafo de derivação (G) que represente o esquema do *data warehouse* global a ser fragmentado;
- Conjunto de medidas numéricas a serem fragmentadas;
- As funções de agregação (F_{agr}) que serão usadas para realizar a agregação das medidas numéricas.

Onde os vértices do grafo são representados por $V(G) = \{v^1, v^2, v^3, ..., v^n\}$, $n \in N^*$, e as arestas por: $A(G) = \{a^1, a^2, a^3, ..., a^m\}$, $m \in N$. O vértice v^1 representa a agregação derivante total de todas as dimensões do *data warehouse*, apresentando assim um menor nível de agregação, enquanto que os próximos representam os descendentes, que são mais agregados. O vértice v^n representa o vértice vazio, isto é, dados totalmente agregados.

Tomando como exemplo o grafo da figura 7:

V(G) = {mrv, mr, mv, rv, m, r, v, vazio}
A(G) = {mrv_mr, mrv_mv, mrv_rv, mr_m, mr_r, mv_m, mv_v, rv_r, rv_v,
m_vazio, v_vazio, r_vazio, vazio}

O conjunto de medidas numéricas é representado por: $CMN = \{m^1, m^2, ... m^w\}$, onde $w \in N^*$. Esta entrada pode ser implícita ao algoritmo, de forma que caso um conjunto vazio de medidas numéricas seja utilizado como entrada, o algoritmo utilizará técnicas heurísticas para determinar quais dados da agregação derivante

total de G são medidas numéricas, e aplicará a função de agregação a todas as medidas numéricas encontradas para serem particionadas verticalmente.

O conjunto de funções de agregação (F_{agr}) representará a forma com que cada medida numérica será agregada: CF = $\{F_{agr}(m^1), ..., F_{agr}(m^w)\}$. Esta função pode ser soma, média, valor máximo e mínimo, e são representados respectivamente por: $F_{agr(m)} \in \{SUM, AVG, MAX, MIN\}$

4.5.2. Pseudocódigo

A seguir é apresentado o funcionamento do algoritmo em alto nível, em uma representação utilizando pseudocódigo.

Algoritmo FV-MN

```
// entradas necessárias para execução do algoritmo.
inputs:

➡ G(V,A);
   // o grafo de derivação a ser fragmentado, onde:
   // os vértices são: V(G) = {v^1, v^2,..., v^n}, onde n ∈ N*.
   // as arestas são: A(G) = {a^1, a^2, ..., a^m}, onde m ∈ N.
   // o vértice v^1 representa a agregação derivante
   // total de todas as dimensões do data warehouse.

➡ CMN = {m^1, m^2, ... m^w}, onde w ∈ N*;
   // o conjunto de medidas numéricas a serem fragmentadas.
   // caso este conjunto seja vazio, o algoritmo buscará
   // por medidas numéricas na agregação derivante total
   // de G para preencher o conjunto.

➡ CF = {F_agr(m^1), ..., F_agr(m^w)}, onde w ∈ N*;
   // representa as funções de agregação a serem aplicadas
   // sobre cada medidas numéricas que será fragmentada.
   // Caso o conjunto de medidas numéricas seja vazio,
   // uma única função de agregação deve ser passada como
   // parâmetro dentro do conjunto de funções.
```

```
// execução do algoritmo.
begin:
    // inicialização dos possíveis domínios para
    // medidas numéricas.
    DOMINIO_MEDIDAS_NUMÉRICAS ← {SMALLINT, INTEGER, DOUBLE,
                                 FLOAT, REAL, BIT, DECIMAL}
    // em princípio analisa-se o vértice inicial v¹.
    v ← v¹;
    // caso o conjunto de medidas numéricas seja vazio, o
    // algoritmo fará uma varredura em todas as dimensões
    // em busca de medidas numéricas.
    if conjunto de medidas numéricas for vazio, CMN = {} do
      // para cada dimensão de v, verifique se contém
      // medidas numéricas.
      while existirem dimensões em v do
         // uma variável temporária domínio é criada
         // instanciada pelo domínio da respectiva
         // dimensão de v.
         domínio ← domínio da dimensão i de v;
         // verifica se o domínio do respectivo domínio é
         // uma medida numérica.
         if domínio ∈ DOMINIO_MEDIDAS_NUMÉRICAS do
           // insere a dimensão ao conjunto
           // de medidas numéricas.
           adicione dimensão i à CMN;
         // fim do if.
         end if
      // fim do while.
      end while
    // fim do if.
    end if
```

```
// para cada medida numérica m, o algoritmo realiza
// todo o processo de fragmentação vertical
// para aquela medida, incluindo a geração
// dos grafos de reconstrução que contém os
// diferentes níveis de agregação da medida numérica.
while existirem medidas numéricas em CMN do
  // é criado um fragmento para cada
  // medida numérica de forma a representar um
  // fragmento vertical contendo as dimensão da
  // agregação derivante total de G sem a
  // presença das outras medidas numéricas.
  fv^k ← v fragmentado em função de m^k;
// fim do while.
end while

// para cada fragmento gerado, instanciar este
// a um vértice e associá-lo à um grafo de
// derivação G^k.
while existirem medidas numéricas em CMN do
  // é criado um vértice de forma a representar as
  // as dimensões de v que não representam medidas
  // numéricas e a medida numérica do fragmento em
  // questão.
  instanciar v^k com fv^k;
  associar v^k à V^k(G^k);
// fim do while.
end while
```

```
// após a criação dos fragmentos verticais é preciso
// entrar no processo de reconstrução, de forma que os
// vértices obtidos pelos fragmentos devem ser agregados
// de todas as formas a fim de se obter as diferentes
// combinações de agregação em função das respectivas
// dimensões e que incluam a medida numérica em questão.
while existirem vértices fragmentados v^k do
  // cria-se conjunto de vértices derivados do grafo
  // fragmentado a partir do vértice v^k.
  CV ← adicione os vértices que representam as
       combinações das dimensões de v^k onde a
       medida numérica m^k está presente;
     // enquanto existirem vértices no conjunto
     // agregado.
     while existirem vértices em CV do
       // associa o vértice v^k como ancestral ao
       // vértice agregado em questão.
       associe v^k como ancestral direto do vértice v^z do
       conjunto de vértices CV;
       // insere os valores no vértice em questão de
       // com seu ancestral v^k e a função de agregação
       // da medida numérica m^k, representada por
       // F_agr(m^k).
       determinar os valores a serem armazenados em v^z
       de acordo com seu ancestral v^k e com F_agr(m^k);
       // associa o vértice agregado ao
       // Grafo fragmentado correspondente, G^k.
       associar v^z à V^z(G^k);
     // fim do while.
     end while
```

```
      // entra em recursão, refazendo todo o procedimento
      // de criação do conjunto de vértices que representam
      // a combinação das dimensões, só que isto para cada
      // descendente, e assim sucessivamente, até que não
      // existam mais descendentes.
      repeat  refaça o procedimento de criação do conjunto
              de vértices derivados para cada descendente
              do conjunto CV até que não existam mais
              descendentes;

    // fim do while.
    end while

    // após fragmentar e agregar os dados é preciso realizar
    // o processo criar as relações das dependências entre
    // as agregações geradas, ou seja, G^1,... G^k,... G^w.
    while existirem agregações G^k do
      // cria a aresta e associa ao grafo em questão.
      crie uma aresta a^r, de forma idêntica a a^k e
      associe a A^k(G^k);
    // fim do while.
    end while

// fim do algoritmo.
end.
```

// saídas após execução do algoritmo.

outputs:

➡ $CG = \{G^1(V^1, A^1),...,G^j(V^k, A^k),...,G^k(V^w, A^w)\}$, $1 \leq k \leq w$;

// conjunto de n grafos de derivação

// fragmentados de acordo com

// suas medidas numéricas e agregados em

// diferentes níveis de agregação de suas dimensões.

// cada um destes grafos pode ser visto como

// uma representação de uma visão, de forma que

// as tabelas das unidades de distribuição podem

// ser geradas através do esquema de representação

// dos grafos resultantes e seriam preenchidas nos

// mais diversos níveis de agregação de seus atributos.

4.5.3. Exemplo de Aplicação do Algoritmo

Para esclarecer o funcionamento do algoritmo será mostrado todo o processo para realizar a fragmentação vertical em um *data warehouse* bastante simplificado utilizando o algoritmo aqui proposto.

Supondo que em uma determinada organização os seguintes assuntos são modelados: produto, região e tempo. E as seguintes medidas numéricas são utilizadas para assessorar o processo de tomada de decisão pelos departamentos de estoque e vendas da organização: quantidade e vendas, em função de todas as possíveis agregações, como vendas por filial em um determinado tempo, ou quantidade em estoque por produtos e por filial.

Considerando que a utilização do sistemas de informação responsável por oferecer suporte às consultas para tomadas de decisão é constantemente alta, de forma que está ocorrendo uma queda do desempenho das consultas, pois elas têm que acessar a mesma base de dados, o *data warehouse*.

A solução para este problema é oferecida pelo algoritmo de fragmentação vertical em termos de medidas numéricas, o FV-MN. Através dele, os dados são fragmentados em função de suas medidas numéricas, de forma que seriam gerados tantos fragmentos quanto forem a quantidade de medidas numéricas.

Neste exemplo, em princípio, dois fragmentos seriam gerados, que representariam os grafos com a distribuição dos dados em dois fragmentos de acordo com sua medida numérica e que poderiam ser alocados em cada departamento, assim como suas agregações que são apresentadas mais adiante. No grafo simplificado da figura 8 é esquematizada a estrutura resultante da

fragmentação em função da medida numérica, de forma que as dimensões produto, região, tempo, vendas e quantidade são representadas respectivamente por p, r, t, v, q. Os descendentes marcados com a cor verde são os vértices fragmentados, de forma que os marcados em vermelho não correspondem a fragmentos gerados pelo algoritmo e são descartados.

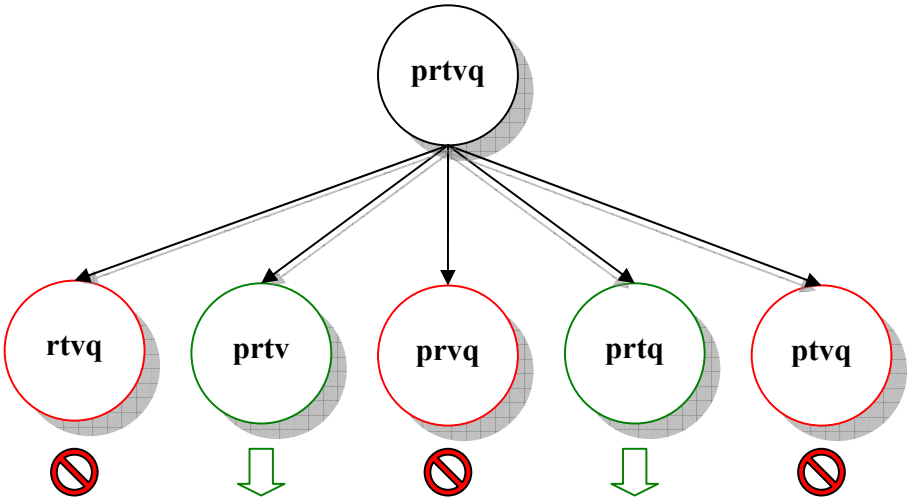

Figura 8 - Grafos de derivação simplificado para as dimensões prtvq

Como entrada do algoritmo teria-se o grafo mostrado na figura 8, as medidas numéricas, vendas e quantidade, e suas respectivas funções de agregações, neste exemplo considerar a soma dos valores.

Após estabelecer os vértices fragmentados é preciso agregar os valores em suas diversas formas. Isto é feito através do preenchimento das informações presentes em um de seus ascendentes e agregando-se a medida numérica de acordo com a função estabelecida como entrada, neste caso, soma. Nas figuras 9 e 10 são apresentados os grafos de derivação para os dois fragmentos gerados e quais vértices são reconstruídos para a medida numérica vendas e quantidade, respectivamente.

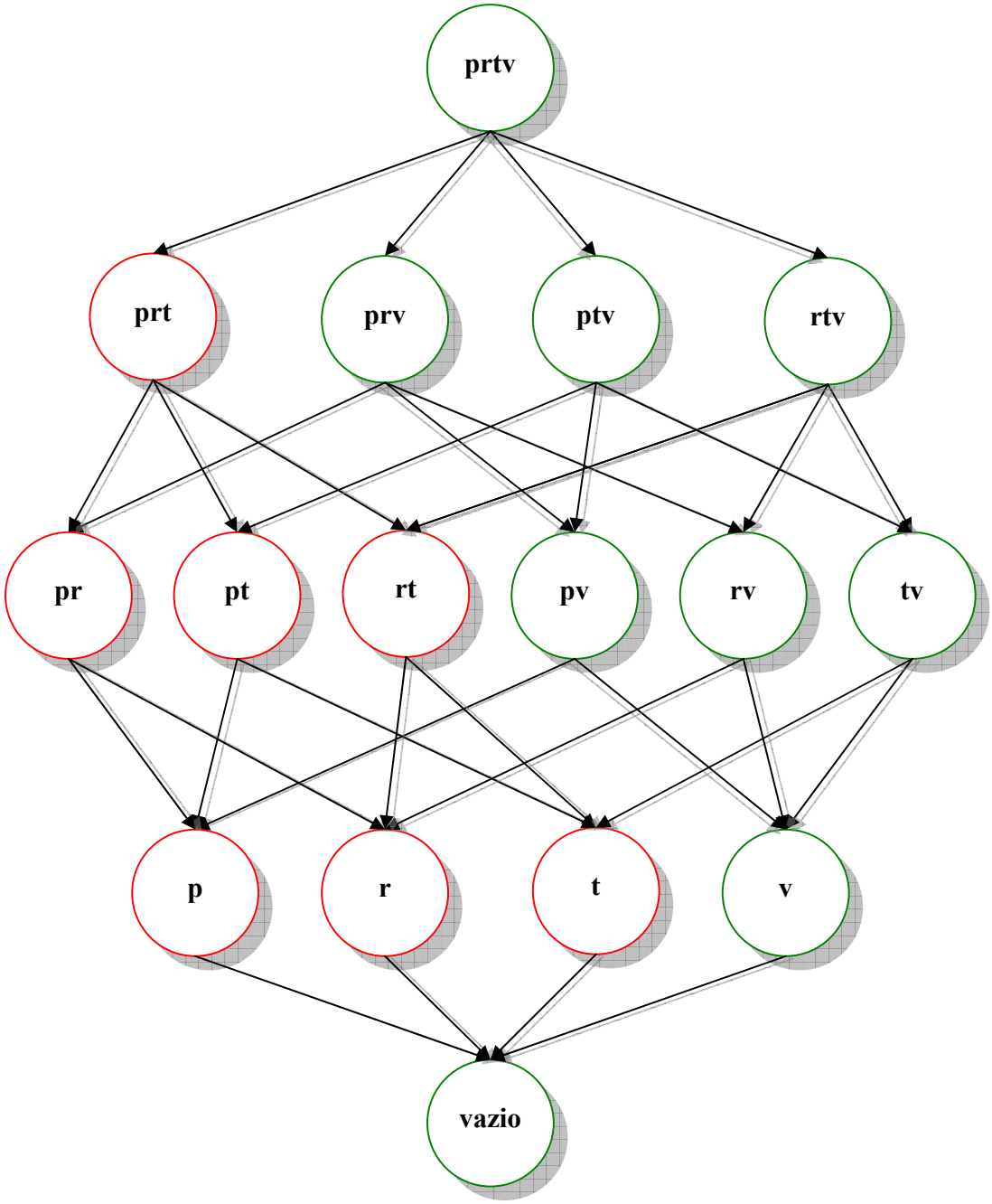

Figura 9 - Grafos de derivação para o fragmento vertical prtv

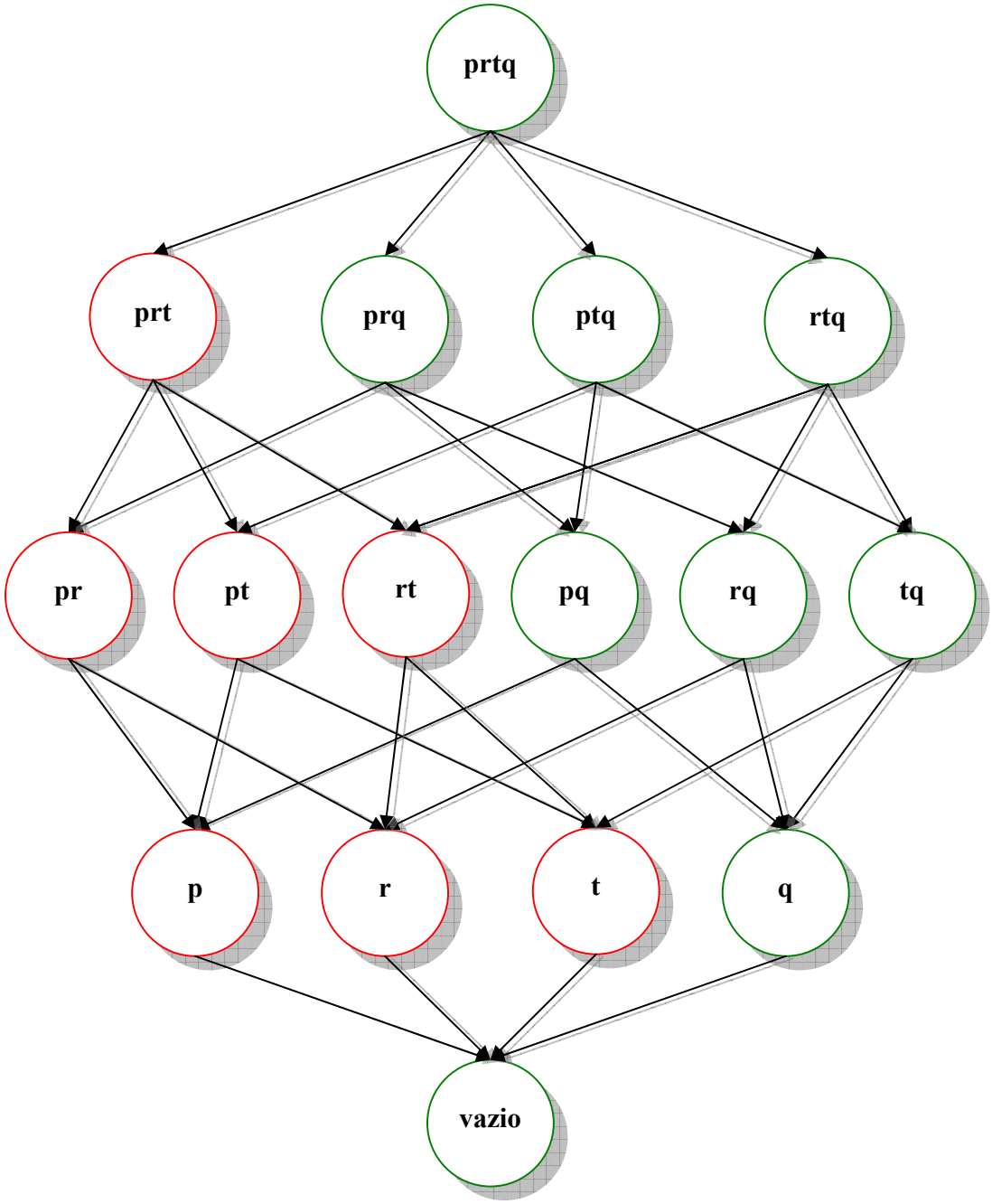

Figura 10 - Grafos de derivação para o fragmento vertical prtq

O vértice vazio de cada grafo gerado nas figuras 9 e 10 representam, respectivamente, a agregação total sobre o somatório total das vendas da organização e o somatório da quantidade total em estoque, isto é, apenas uma tupla[11] é representada nessa entidade gerada. Uma outra agregação resultante, por exemplo, é a do vértice rv, que representa o total de vendas agrupados por região. Todas agregações em cor verde são reconstruídas, isto é, são criadas as tabelas que armazenarão os dados e as rotinas de inserções, enquanto que as de vermelho não, pois não representam agregações em termos da medida numérica. Exemplos de agregação podem ser vistos nos Quadros 6, 7 e 8, na seção Medida Numérica.

Como saída do algoritmo neste exemplo, tem-se os dois grafos fragmentados e que são apresentados nas figuras 9 e 10.

4.5.4. Implementação do Algoritmo Proposto

Embora não fosse prevista nenhuma implementação do algoritmo proposto neste trabalho, foi resolvido ir além dos estudos teóricos e apresentar uma solução prática da utilização do algoritmo de fragmentação vertical aqui proposto.

Através da linguagem de programação *JAVA*, utilizando a ferramenta de desenvolvimento *JBuilder*[12] 8, foi desenvolvido um aplicativo com interface gráfica amigável, de forma a facilitar sua utilização por parte dos usuários. As entradas são definidas pelo usuário e caracterizam-se nas dimensões do *data warehouse* e o conjunto de medidas numéricas a serem fragmentados e suas respectivas funções de agregação.

[11] Conjunto de dados correspondente a uma linha de uma tabela no modelo relacional.
[12] Ferramenta de desenvolvimento em *Java* da *Borland*.

O esquema global do grafo de derivação gerado para o mesmo exemplo da seção 3.5.3 é representado na figura 11, que representa a GUI[13] do aplicativo de fragmentação vertical. Onde os grafos verdes representam àqueles que o algoritmo cria os scripts das tabelas para armazenamento dos dados e seus respectivos comandos "INSERT" para povoamento dos dados em seus diferentes níveis de agregação através de cláusulas "GROUP BY".

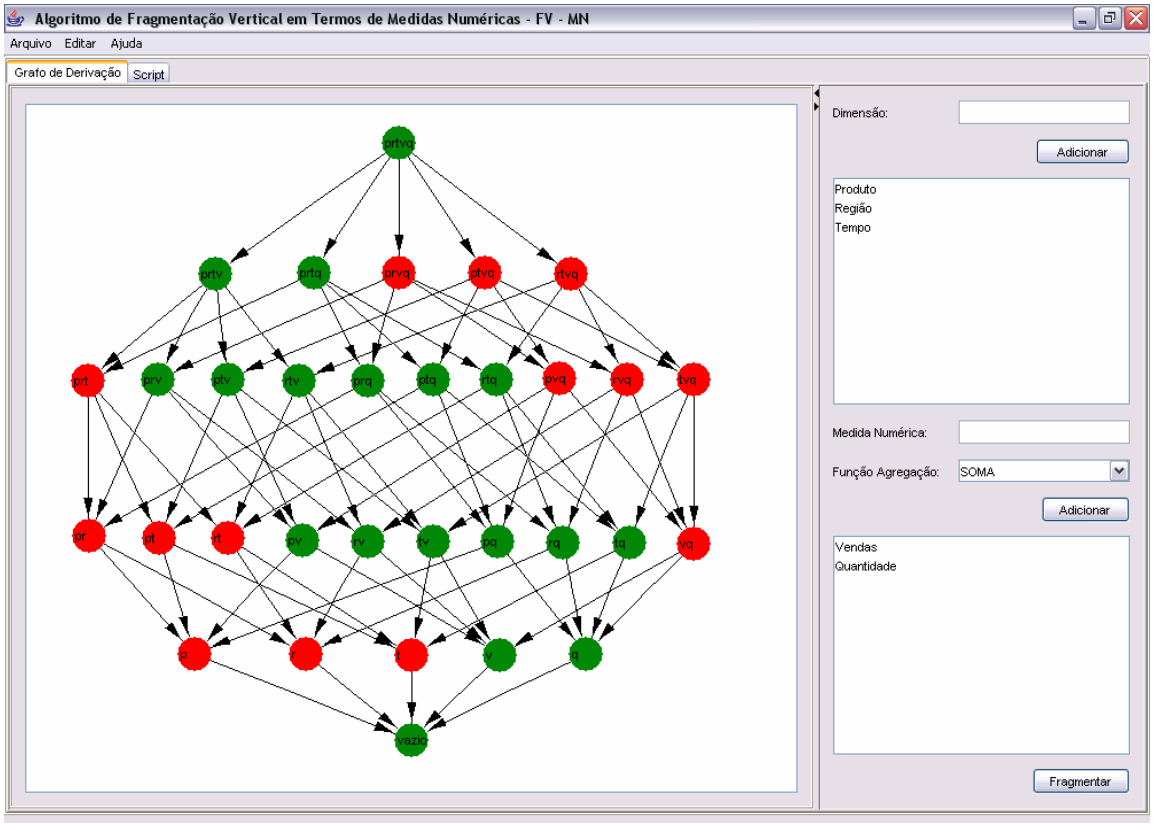

Figura 11 - GUI da geração do grafo pelo aplicativo de fragmentação vertical

Na parte direita do aplicativo pode-se visualizar as entradas para o algoritmo, que são passadas pelo usuário, de forma que quatro funções de agregação são possíveis para cada medida numérica: SOMA, MÉDIA, VALOR MÁXIMO E VALOR MÍNIMO. Para cada dimensão e medida numérica, o

[13] *Graphical User Interface*. Interface gráfica.

programa utiliza seu caractere inicial como sigla para poder gerar o grafo e as tabelas de forma mais concisa. O algoritmo em questão não permite que seja realizada a rotina de fragmentação vertical se não existirem pelo menos duas medidas numéricas, pois não há sentido em fragmentar algo que já representa uma partição em termos de medidas numéricas.

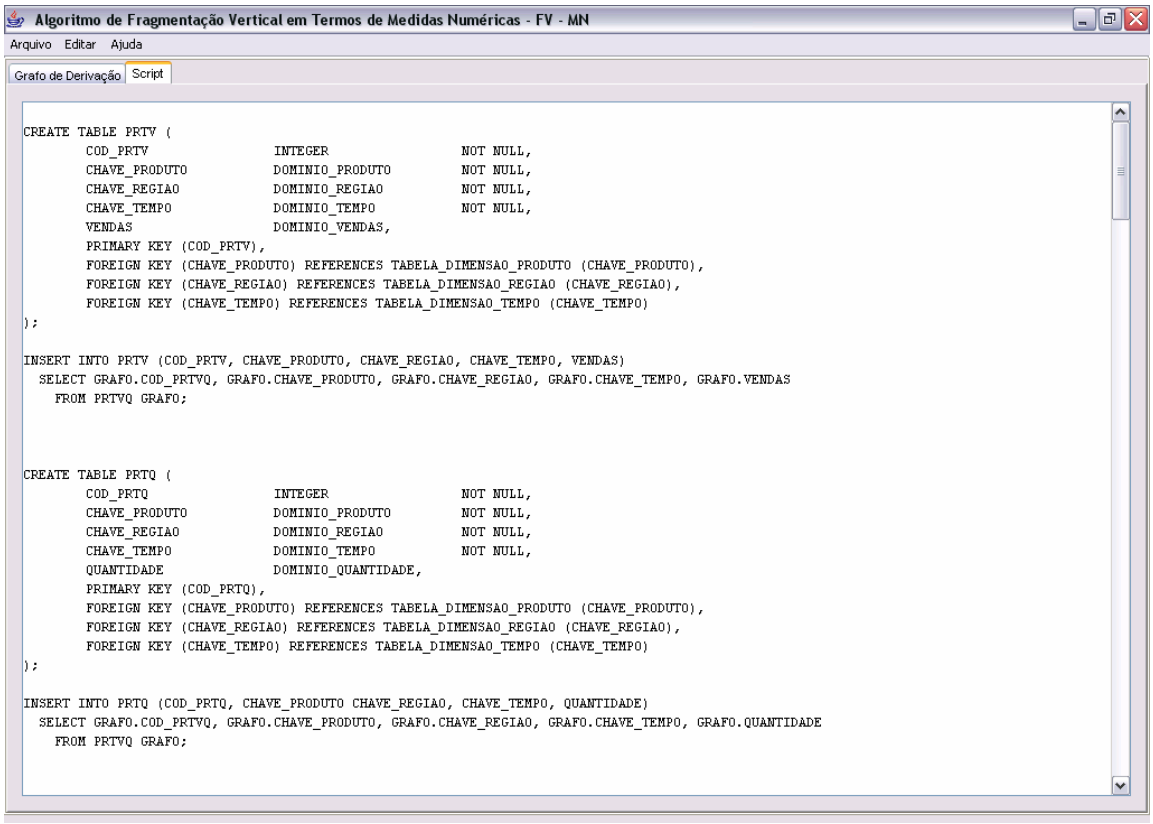

Figura 12 - GUI do script dos vértices fragmentados pelo algoritmo FV-MN

A figura 12 ilustra o trecho do script gerado pelo algoritmo em função dos vértices fragmentados, no caso dois, prtv e prtq. E, finalmente na figura 13 é apresentado o trecho do script gerado para algumas das agregações do fragmento prtq, que são: soma da quantidade em estoque agregado por produto e região, soma da quantidade em estoque agregado por produto e tempo e o início da

entidade que representa a soma da quantidade em estoque agregado por região e tempo.

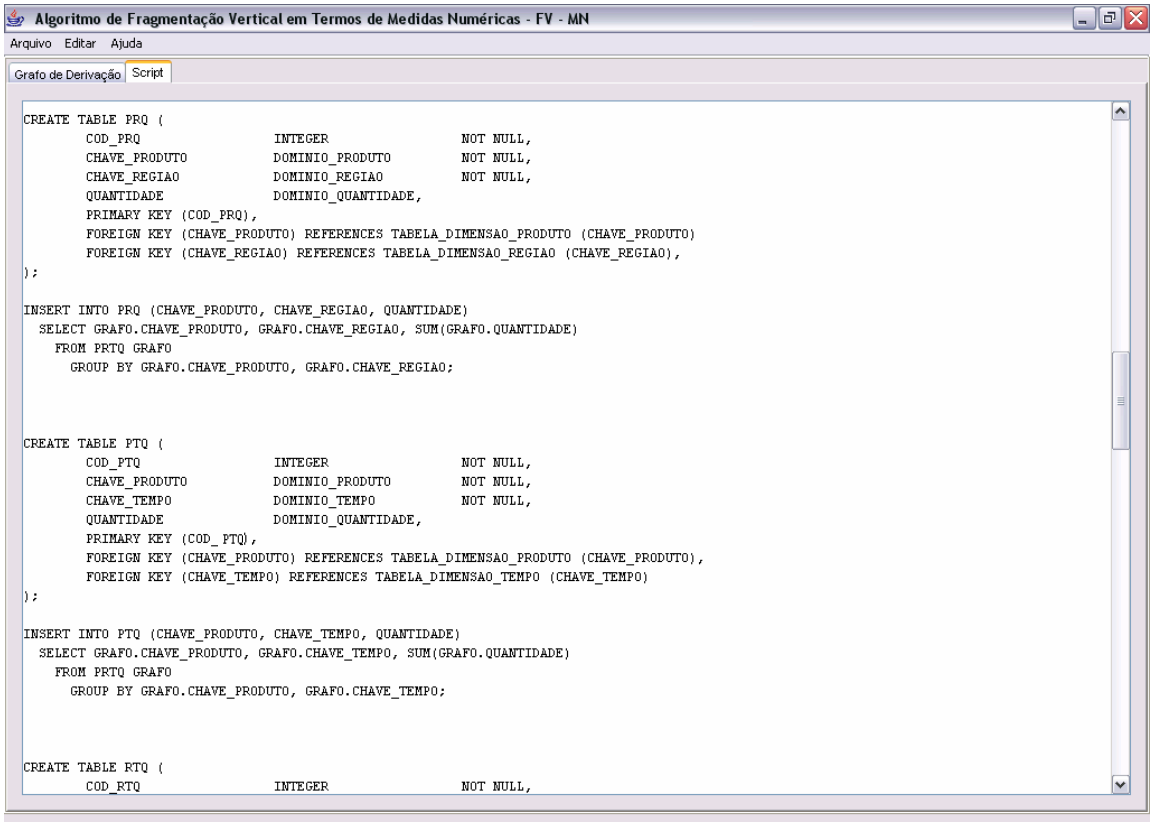

Figura 13 - GUI do script dos vértices agregados pelo algoritmo FV-MN

4.5.5. Vantagens e Desvantagens Proporcionadas pelo Algoritmo

O algoritmo aqui proposto apresenta como principal vantagem o aumento da disponibilidade dos dados, visto que as porções de dados criadas pelo algoritmo poderiam ser acessadas paralelamente e em diferentes níveis de agregação. Assim, as partições resultantes poderiam estar localizadas em diferentes *sites* da organização, representado os diversos departamentos e

elevando o desempenho das consultas agregadas a cada medida numérica relativa àquela área da empresa.

Uma outra vantagem oferecida é a segurança, uma vez que os fragmentos gerados contendo medidas numéricas confidenciais, como lucro da organização, poderiam ser alocadas em *sites* com um controle rigoroso de segurança a um custo menor.

A desvantagem apresentada pelo algoritmo consiste no fato de que ele não leva em consideração a freqüência de acesso aos dados, pois se leva em consideração apenas uma abordagem semântica, isto pode resultar muitas vezes na criação de fragmentos que deveriam estar unidos, como por exemplo, as medidas numéricas preço e desconto, que dão suporte à tomada de decisões principalmente do departamento de vendas das organizações, levando a uma possível degradação do desempenho.

5. CONCLUSÃO

A tecnologia de *data warehouse* mostra-se muito interessante para empresas que possuem grandes volumes de dados gerados e acumulados durante sua existência e necessitam recuperar estes dados de uma forma que eles possam auxiliar os administradores destas empresas a tomarem decisões estratégicas rapidamente e com segurança.

Entretanto, devido ao crescente volume de dados manipulados e aumento do número de consultas em função do crescimento de usuários, os ambientes de *data warehousing* podem estar perdendo em desempenho.

Para contornar este problema pode-se fragmentar os dados para aumentar sua disponibilidade. Chega a parecer uma utopia distribuir dados que foram integrados de diversos provedores de informações, mas esta solução tem se mostrado adequada.

Para implementar ambientes de *data warehousing* assim, novos desafios surgiram, como a criação de metodologias e algoritmos para fragmentação destes dados. Foi esse contexto que motivou a elaboração deste trabalho: a criação de um algoritmo para fragmentação vertical de um *data warehouse* em termos de medidas numéricas e baseado nos conceitos de grafos de derivação, algo inovador.

O resultado alcançado foi além do esperado, pois além de toda abordagem teórica envolvida no algoritmo, foi possível também realizar uma implementação simplificada da utilização do algoritmo, de forma que o usuário deve informar as dimensões existentes no *data warehouse*. Uma implementação mais complexa

poderia ser feita utilizando-se o esquema do *data warehouse* como entrada para que as dimensões fossem obtidas de maneira mais dinâmica.

5.1. Trabalhos Futuros

O algoritmo proposto baseou-se em técnicas de grupamento, que consistem em começar com tantos fragmentos quanto forem os atributos e a cada passo reuni-los em fragmentos um pouco maiores, até que algum critério seja satisfeito. No caso do algoritmo aqui proposto, agrupar as medidas numéricas.

A técnica de repartição que consiste em iniciar com a relação inteira e particioná-la baseando-se no comportamento de acesso das aplicações, de forma que dados altamente acessados seriam particionados em fragmentos diferentes, seria uma outra forma adequada para fragmentação de um *data warehouse*. Em primeiro lugar por ser mais apropriada a uma abordagem de projeto *top-down*, e em segundo lugar por existir uma probabilidade maior da solução ótima estar mais próxima da relação como um todo, do que de seus atributos individualmente. Desta forma, o desenvolvimento de um algoritmo de fragmentação vertical utilizando esta técnica seria uma boa proposta de trabalho futuro.

Também poderia-se focar um novo trabalho em criar metodologias para alocação dos fragmentos gerados ou no desenvolvimento de algoritmos para realizar a fragmentação mista dos dados.

6. REFERÊNCIAS BIBLIOGRÁFICAS

[BABAD_97] BABAD, M., ***A record and file partitioning model***, Janeiro, 1997, consultada em 09/12/2004, disponível em <http://www.acm.org>.

[CAMPOS_99] CAMPOS, M. L., FILHO, A. V. R., ***Data warehouse***. Universidade Federal do Rio de Janeiro, Março, 1999.

[CERI_83] CERI, S., Navathe, S. B., WEIDERHOLD, G., ***Distribution Design of Logical Database Schemas***, IEEE trans. em SW engg. Vol SE-9, nº .4, p 487 - 503, Julho 1983, consultada em 11/12/2004, disponível em <http://www.ieee.org>.

[CERI_89] CERI, S., PERNICI, S., WEIDERHOLD, G., ***Optimization Problems and Solution Methods in the Design of Data distribution***, Information Sciences, Vol 14, nº 3, pág. 261-272, 1989.

[CIFERRI_02A] CIFERRI, C. D. A., ***Distribuição dos Dados em Ambientes de Data Warehousing: O Sistema WebD²W e Algoritmos Voltados à Fragmentação Horizontal dos Dados***, Universidade Federal de Pernambuco, 2002.

[CIFERRI_02B] CIFERRI, C. D. A., SOUZA, F. F., ***Focusing on Data Distribution in the WebD²W System***. In Proceedings of the 4th International Conference on *Data Warehousing* and Knowledge Discovery, França, Setembro de 2002, volume 2454 of Lecture Notes in Computer Science, Springer, 2002.

[COME_01] COME, C., ***Contribuição ao Estudo da Implementação de Data Warehousing: Um Caso no Setor de Telecomunicações***, Universidade de São Paulo, 2001.

[CORNELL_87] CORNELL, D., YU P., ***A Vertical Partitioning Algorithm for Relational Databases***. Proc. Third International Conference on Data Engineering, 1987, pág. 30-35.

[DIESTEL_97] DIESTEL, R., ***Graph Theory***, Springer, Verlag, Nova Iorque, Inc., EUA, 1997, ISBN 0-387-98211-6.

[ELMASRI_00] ELMASRI, R., NAVATHE, S. B.**,** ***Fundamentals of Database Systems***. Addison-Wesley Publishing Company, Menlo Park, CA, EUA, 3ª edição, 2000. ISBN 0-201-54263-3.

[HAMMER_79] HAMMER, M., NIAMIR, B., ***A heuristic approach to attribute partitioning. In Proceedings,*** ACM SIGMOD Int. Conf. on Management of Data, Nova Iorque, 1979, consultada em 09/12/2004, disponível em <http://www.acm.org>.

[HARINARAYAN_96] HARINARAYAN, V., RAJARAMAN, A., ULLMAN, J.D., ***Implementing Data Cubes Efficiently***, Proceedings of the 1996 International Conference on Management of Data, volume 25, SIGMOD Record, pág. 205-216, ACM Press, Junho, 1996, consultada em 20/02/2005, disponível em <http://www.acm.org>.

[HOFER_75] HOFER, J., SEVERANCE, D., ***The Uses of Cluster Analysis in Physical Database Design***, In Proc. 1st International Conference on VLDB, Framingham, MA, 1975, pág. 69 - 86.

[INMON_97] INMON, W. H., ***Como Construir o Data Warehouse***, Rio de Janeiro, Campus, 1997.

[INMON_95] INMON, W. H., ***Data Mart ≠ Data Warehouse***, DM Review, 1995, consultada em 04/11/2004, disponível em <http://www.dmreview.com>.

[INMON_94] INMON, W. H., HACKATHORN, R. D., ***Using the data warehouse***, Nova Iorque, 1994.

[INMON_96] INMON, W. H., WILEY, J., ***Building the Data Warehouse***, 1996, ISBN 0471141615;

[KIMBALL_99] KIMBALL, R., ***Divide and Conquer***, 1999, consultada em 19/01/2005, disponível em <http://www.rkimball.com>.

[MUTHURAJ_92] MUTHURAJ, J., ***A formal approach to the vertical partitioning problem in distributed database design***, Universidade da Flórida, 1992.

[NAVATHE_84] NAVATHE, S., CERI, S., WIEDERHOLD, G., DOU, J., ***Vertical Partitioning Algorithms for Database Design,*** ACM Transactions on Database Systems, Vol. 9, nº 4, Dezembro, 1984, consultada em 10/12/2004, disponível em <http://www.acm.org>.

[NAVATHE_89] NAVATHE, S., RA, M., ***Vertical Partitioning for Database Design: A Graphical Algorithm.*** ACM SIGMOD, Portland, Junho, 1989, consultada em 10/12/2004, disponível em <http://www.acm.org>.

[ÖZSU_99] ÖZSU, M.T., VALDURIEZ P., ***Principles of Distributed Database Systems***, pág. 108-165, Upper Saddle River, New Jersey, USA. Prentice-Hall, 1999. ISBN 0-13-659707-6.

[PRIEBE_00] PRIEBE, T., PERNUL, G., ***Towards OLAP Security Design: Survey and Research Issues***. In Proceeding of the 3rd International Conference on *Data Warehousing* and OLAP, Washington, EUA, Novembro, 2000.

[RA_90] RA, M., ***Data Fragmentation and Allocation Algorithms For Distributed Database Design.*** Ph.D Dissertation, Database R & D Center, Universidade da Flórida, Gainesville, 1990.

[SINGH_97] SINGH, H., ***Data Warehousing: Concepts, Technologies, Implementations, and Management***, Upper Saddle River, NJ, Prentice Hall, 1997.

[WANG_98] WANG, C. B., ***Techno Vision II***, São Paulo, Makron Books, 1998.

[ZAHN_95] ZAHN, C., ***Graph-theoretical methods for detecting and describing Gestalt Clusters***. IEEE Transactions on Computers C 20, 68-86, Fevereiro, 1995, consultada em 11/12/2004, disponível em <http://www.ieee.org>.