UNIVERSIDADE METODISTA DE PIRACICABA

FACULDADE DE CIÊNCIAS EXATAS E DA NATUREZA

MESTRADO EM CIÊNCIA DA COMPUTAÇÃO

AUTOMATIZAÇÃO DO PROCESSO DE CRIAÇÃO DE VISÕES PARA MODELAGEM DE DATA WAREHOUSE

RICARDO ALEXANDRE NEVES

ORIENTADORA: PROFª. DRª. MARINA TERESA PIRES VIEIRA

PIRACICABA, SP
2009

UNIVERSIDADE METODISTA DE PIRACICABA

FACULDADE DE CIÊNCIAS EXATAS E DA NATUREZA

MESTRADO EM CIÊNCIA DA COMPUTAÇÃO

AUTOMATIZAÇÃO DO PROCESSO DE CRIAÇÃO DE VISÕES PARA MODELAGEM DE DATA WAREHOUSE

RICARDO ALEXANDRE NEVES

ORIENTADORA: PROFª. DRª. MARINA TERESA PIRES VIEIRA

Dissertação apresentada ao Mestrado em Ciência da Computação, da Faculdade de Ciências Exatas e da Natureza, da Universidade Metodista de Piracicaba – UNIMEP, como requisito para obtenção do Título de Mestre em Ciência da Computação.

PIRACICABA, SP
2009

AUTOMATIZAÇÃO DO PROCESSO DE CRIAÇÃO DE VISÕES PARA MODELAGEM DE DATA WAREHOUSE

AUTOR: RICARDO ALEXANDRE NEVES

ORIENTADORA: PROFª. DRª. MARINA TERESA PIRES VIEIRA

Dissertação de Mestrado apresentada em 19 de fevereiro de 2009, à Banca Examinadora constituída dos Professores:

_____

Profª. Drª. Marina Teresa Pires Vieira
FACEN - UNIMEP

_____

Prof. Dr. Luiz Camolesi Junior
CESET - UNICAMP - Campinas

_____

Profª. Drª. Cristina Dutra Aguiar Ciferri
ICMC - USP – São Carlos

Dedico este trabalho a

Deus por ter me abençoado durante todos os momentos;

À minha esposa Geise, por compreender minhas ausências.

AGRADECIMENTOS

À professora Profª.Marina pela orientação, compreensão, incentivo e dedicação dispensados ao desenvolvimento deste trabalho;

Aos professores Ricardo Ciferri (UFSCar) e Cristina Ciferri (USP – São Carlos) por suas valiosas contribuições durante o trabalho;

Ao amigo Eduardo Fernando Mendes por seu apoio, alegria e companheirismo no decorrer de nosso período de estudos;

Ao amigo Adriano dos Santos Fernandes pelas valiosas contribuições na implementação do protótipo.

“Não se mede o valor de um homem pelas suas roupas

ou pelos bens que possui.

O verdadeiro valor do homem é seu caráter,

pensamentos e a nobreza de seus ideais.”

Charles Spencer Chaplin

RESUMO

Um *Data Warehouse* geralmente armazena um grande volume de dados, cuja carga é proveniente de informações transacionais. O acúmulo dessas informações, com o passar do tempo, dificulta a obtenção de informações para a tomada de decisão. A materialização de visões é uma técnica comumente utilizada no projeto físico de um *Data Warehouse*, tornando persistentes visões pré-computadas, para que possam responder, de forma eficiente às consultas submetidas de acordo com a necessidade dos usuários. Deste modo, muitas pesquisas nesta linha de trabalho são realizadas, tanto no meio acadêmico, quanto no meio comercial. As pesquisas apresentam, em sua maioria, abordagens que minimizam o problema da seleção de visões a materializar, sob determinado foco de interesse. Este trabalho apresenta uma abordagem que automatiza a seleção de visões a materializar, para *Data Warehouses*, levando em consideração parâmetros do banco de dados transacional e as consultas dos usuários, identificadas na fase de definição do projeto conceitual do *Data Warehouse*, em consequência dos requisitos de análise dos usuários.

PALAVRAS-CHAVE: *Data Warehouse*, Visões Materializadas, Seleção de Visões a Materializar.

ABSTRACT

Data Warehouses stores usually a large volume of data from transitional information. The growing volume of information, over time, turns the data manipulation difficult for decision-making. Materialized Views is a technique frequently used in physical Data Warehouse design keeping persistent pre-computed views in order to efficiently answer to requests of users. Many researches about this topic have been made as well by academic as by business areas. The great majority of the approaches minimize the problem of views selection to materialize under particular focus of interest. The present work introduces an approach that automates the process of view selection to materialize for data warehouses, taking into account the parameters of the transactional database, and the user queries. These queries are identified in the design definition phase of the Data Warehouse as user requirements analysis.

KEYWORDS: Data Warehouse*,* Materialized Views, Materialized Views Selection.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE EQUAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

DW	*Data Warehouse.*
ODBC	*Open Data Base Connectivity.*
JDBC	*Java Database Connectivity.*
ETL	*Extract Transform Load.*
OLAP	*On-Line Analytical Processing.*
SGBD	*Sistema Gerenciador de Banco de Dados.*
WAVE	*Warehouse And View Engineering.*
WAND	*Warehouse Integrated Designer.*
SQL	*Structured Query Language.*
IDE	*Integrated Development Environment.*
SDK6	*Standard Developer Kit Versão 1.6*
SVM	Módulo de Seleção de Visões.
RAM	*Random Access Memory.*

LISTA DE TABELAS

1. INTRODUÇÃO

As empresas enfrentam, atualmente, uma crescente concorrência, pontecializada pela abertura de mercados competidores cada vez mais promissores. Este fato estimula a competitividade entre as empresas, levando-as a repensar e a reestruturar seus modelos de negócio. Recorrendo à reengenharia de processos, automação e informatização, por meio dos sistemas de informação, desenvolveram-se sistemas de informação capazes de monitorar variáveis de produção, estimando indicadores mais precisos e em tempo real, para a tomada de decisão. Deste modo, um conjunto de ferramentas e mecanismos para monitoração e controle de processos estão disponíveis por meio dos sistemas de *Data Warehousing,* existentes no mercado (MARTINS; BELO; NOVAIS, 2005).

Um *Data Warehouse* (*DW*) contém múltiplas visões, as quais são acessadas por consultas. Uma das mais importantes decisões em projetos de *Data Warehouse* é a seleção de visões materializadas com o propósito de implementar um sistema de tomada de decisões eficiente (ZHANG; YAO;YANG, 1999).

Em *Data Warehouse,* as duas técnicas mais utilizadas para reduzir o custo de processamento de consultas *OLAP* (Online Analytical Processing) são as visões materializadas e os índices (BELLATRECHE; KARLAPALEM; SCHNEIDER, 2000).

A técnica de materialização de visões utiliza o conceito da antecipação do cálculo (total ou parcial) das consultas, de forma a minimizar o impacto do tempo de resposta no desempenho global do sistema. A otimização dos processos do processamento de consultas, utilizando a técnica de materialização de visões, divide-se essencialmente em três áreas de estudo: a seleção de um conjunto de visões para materializar, a utilização de um

conjunto de visões materializadas e a manutenção de um conjunto de visões materializadas (MARTINS; BELO; NOVAIS, 2005).

O presente trabalho está vinculado à área de estudo que compreende a seleção de um conjunto de visões para materializar no ambiente do *Data Warehouse* e atua diretamente na fase de projeto do *DW*.

Lima (2006) desenvolveu uma ferramenta para a geração do modelo dimensional para *Data Warehouse*. Com base nesta ferramenta, foi dado continuidade com a construção de um módulo para seleção de visões a materializar, objetivo principal deste trabalho.

1.1. MOTIVAÇÃO

A principal motivação para o desenvolvimento de uma nova proposta de seleção de visões para materializar, em *Data Warehouses*, é decorrente da necessidade de se ter uma abordagem de seleção de visões relativamente simples, para que seja aplicada em tempo de projeto. A maioria das soluções para a seleção de visões a materializar, apresentada na literatura, trata a seleção das visões de maneira complexa e sempre após o projeto estar implantado, usando como base estatísticas de uso do *Data Warehouse*.

Há inúmeros estudos apresentados na literatura sobre a técnica de materialização de visões, envolvendo grandes empresas como IBM, Microsoft e Oracle, que podem ser citados, tratando-se de pesquisas direcionadas ao aperfeiçoamento de seus produtos. Alguns destes trabalhos como (ZILIO ET AL.; 2004), (AGRAWAL ET AL.; 2004), (ZHOU ET AL.; 2007) e (HOBBS; 2005) discutem a técnica de materialização sob vários aspectos de interesse.

A motivação também se aplica sob o foco do trabalho de Lima (2006), pois a partir deste trabalho, foi possível o desenvolvimento e a aplicação da técnica de seleção de visões proposta, considerando o contexto de projeto do *DW*, tornando a ferramenta de Lima (2006) ainda mais completa e fácil, para uso do projetista de *Data Warehouse*.

1.2. OBJETIVOS DO TRABALHO

Os objetivos deste trabalho estão focados na busca de uma nova solução para seleção de visões para materializar, acompanhando a concepção do modelo dimensional.

Em linhas gerais, este trabalho contribui com um novo algoritmo para a seleção de visões a materializar. O algoritmo é aplicado no Módulo de Seleção de Visões, responsável por automatizar o processo de seleção de visões, objetivo principal deste trabalho.

Para que o objetivo pudesse ser alcançado foi utilizada, como instrumento de validação, a ferramenta de Lima (2006), que passou por uma etapa de refinamentos. Tais refinamentos foram necessários para a obtenção dos pré-requisitos para que as visões pudessem ser geradas e submetidas à seleção.

1.3. ESTRUTURA DO TRABALHO

Este trabalho está organizado da seguinte forma: o capítulo 2 apresenta os conceitos relacionados à materialização de visões para *DW*, assim como a análise das principais abordagens, de modo a contemplar as políticas de materialização de visões e os algoritmos utilizados para tal finalidade. O capítulo 3 apresenta um estudo realizado sobre o trabalho de Golfarelli, Maniezzo e Rizzi (2004), por se tratar da abordagem que mais se assemelha ao trabalho realizado. O capítulo 4 apresenta uma visão geral da Ferramenta para Geração do Modelo Dimensional para *Data Warehouse*, conforme abordagem de Lima (2006), instrumento de validação da abordagem de seleção de visões a materializar. Nesse capítulo são também abordados alguns conceitos básicos sobre modelagem dimensional, os quais são aplicados à ferramenta. No capítulo 5 são tratados: o módulo de seleção de visões, o controle de hierarquias, a nova arquitetura da ferramenta para suportar a nova abordagem e seus refinamentos. No capítulo 6 são apresentados os experimentos realizados. E, por fim, no capítulo 7 são

apresentadas as conclusões, bem como as contribuições deste trabalho e de trabalhos futuros.

2. MATERIALIZAÇÃO DE VISÕES PARA DATA WAREHOUSE

A materialização de visões é um dos problemas clássicos em um *Data Warehouse* (VALLURI; VADAPALLI; KARLAPALEM, 2002). A materialização de visões é a técnica comumente empregada em projetos de *Data Warehouse* em busca da minimização de custos, tais como: custo de resposta das consultas e o custo de manutenção das visões.

Segundo Teodoratos e Simitsis (2006), alguns dos fatores considerados para a materialização de visões são as restrições de espaço e tempo de processamento. Segundo Gupta (1997), o espaço em disco, o tempo de processamento e custo de manutenção são restrições que justificam não materializar todas as possibilidades de visões, pois tais restrições devem ser consideradas.

O problema da seleção de um conjunto apropriado de visões para materializar é uma das mais importantes decisões de projeto em um *DW* (GUPTA, 1997).

Existem muitas pesquisas e abordagens focadas no assunto da seleção de visões a materializar para *Data Warehouse*. Algumas das abordagens mais recentes têm buscado soluções avançadas, por meio de algoritmos híbridos, como é o caso da abordagem de Broukra, Nascer e Bouroubri (2007). Outras abordagens têm aperfeiçoado a aplicação da seleção de visões para grafos digiridos, de acordo com Gupta e Mumick (2005).

Este capítulo não tem por finalidade esgotar as abordagens da literatura, e sim estudar as abordagens que apresentam políticas eficientes para resolver o problema da seleção de visões para materializar. Tem também, por objetivo, discutir os resultados apresentados por essas abordagens e os algoritmos utilizados, segundo os custos e/ou restrições que cada abordagem considera e que possa contribuir para este trabalho.

2.1. REPRESENTAÇÃO DAS VISÕES PARA UM DATA WAREHOUSE

Em Harinarayan et al. (1996) são apresentados conceitos para representação das visões em *Data Warehouse*, os quais são definidos a seguir:

- As relações de dependência nas consultas: Se considerarmos duas consultas Q1 e Q2, Q1 $\preceq$ Q2, se e somente se, Q1 pode ser respondida usando somente os resultados da consulta Q2, então Q1 é dependente de Q2. A exemplo, na Figura 2.1, a consulta (part) pode ser respondida usando os resultados da consulta (part, customer), formalmente representada por:

  (part) $\preceq$ (part, customer)

- Notação *lattice*: *Lattice* é definido como um conjunto de elementos (consultas ou visões) L e sua relação de dependência, representada por $\langle L, \preceq \rangle$. Para elementos ***a*** e ***b*** de um lattice $\langle L, \preceq \rangle$, **a** $\preceq$ **b** significa que: **a** $\prec$ **b** e que **a** $\neq$ **b**. Os ancestrais e descendentes de um elemento do lattice $\langle L, \preceq \rangle$ são definidos como segue:

  ancestral (a) = {*b* | *a* $\preceq$ b}

  descendente (a) = {*b* | b $\preceq$ *a*}

Os ascendentes imediatos de um determinado elemento ***a*** no *lattice* pertencem a um conjunto chamado de ***next(a)***. Formalmente, $\text{next}(a) = \{\text{b} \mid \text{a} \prec b, \nexists c, a \prec c, c \prec b\}$.

Diagrama *lattice*: Gráfico em que os elementos do *lattice* são nós e há uma ligação de ***a*** (inferior) para ***b*** (superior), se somente se, ***b*** está em ***next(a).*** Assim, para quaisquer dois elementos *lattice* ***x*** e ***y,*** o diagrama tem um caminho que vai do superior para o inferior, a partir de **y** para **x** e somente se **x** $\preceq$ **y**. Como exemplo, veja o diagrama *lattice* de um conjunto de visões, também na Figura 2.1.

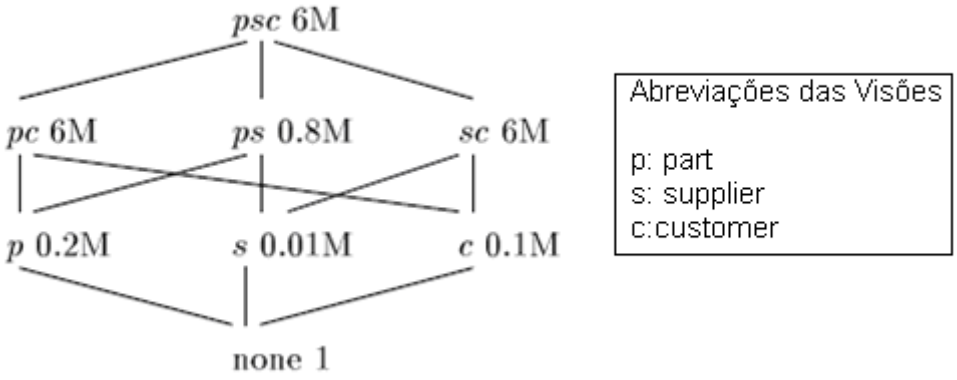


Figura 2.1 – Diagrama *Lattice* de um Conjunto de Visões
(HARINARAYAN ET AL.; 1996)

- Hierarquia: As Hierarquias introduzem as dependências de consultas para que se possam determinar quais delas materializar. Em linhas gerais, as dimensões de um cubo[1] de dados consistem em mais de um atributo. Desse modo, tais dimensões são organizadas como hierarquias desses atributos. Um exemplo simples para melhor entendimento das hierarquias pode ser observado na organização do tempo, levando-se em conta as dimensões: dia, semana, mês e ano. Veja na Figura 2.2 como o tempo pode ser organizado em hierarquias, e de que maneira é estabelecida a relação de dependência entre as dimensões.

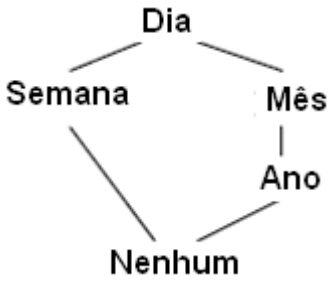


Figura 2.2 – Hierarquia de Atributos de Tempo
Adaptado de (HARINARAYAN ET AL.; 1996)

Em Forlani, Ciferri e Ciferri (2006), foi utilizado o grafo de derivação ou *lattice* para a representação de diferentes níveis de agregação, mediante o uso do Benchmark *TPC-H*, representado em um esquema estrela adaptado, onde *Lineitem* e *Partsupp* são tabelas de fatos que contêm medidas numéricas de

[1] Nome de uma estrutura dimensional em uma plataforma de banco de dados de processamento analítico on-line (*OLAP*) ou multidimensional, originalmente referindo-se a três dimensões (Kimball, 2002).

interesse. A definição do *TPC-H* foi adaptada para organizar um *DW* em diferentes níveis de agregação, por meio de dois *lattices* G1 e G2, *Lineitem* e *Partsupp* respectivamente. A Figura 2.3 representa os *lattices* que consideram as hierarquias *part* (p) → *brand* (b) → *MRGR* (m) e *supplier* (s) → *nation* (n) → *region* (r) para as dimensões *Part* e *Supplier*.

Por definição, segundo Forlani, Ciferri e Ciferri (2006), um *lattice* é um par (V,E) de conjuntos disjuntos de vértices V e arestas E. V(G) representa um conjunto de agregações (i.e., visões), enquanto E(G) representa um conjunto de relações de dependência entre estas agregações. Cada vértice do grafo agrega medidas numéricas sobre as dimensões presentes naquele vértice, sendo nomeado com base na granularidade do atributo de cada uma dessas dimensões. Os vértices de um grafo de derivação seguem as mesmas características do *lattice* de visões de Harinarayan et al. (1996).

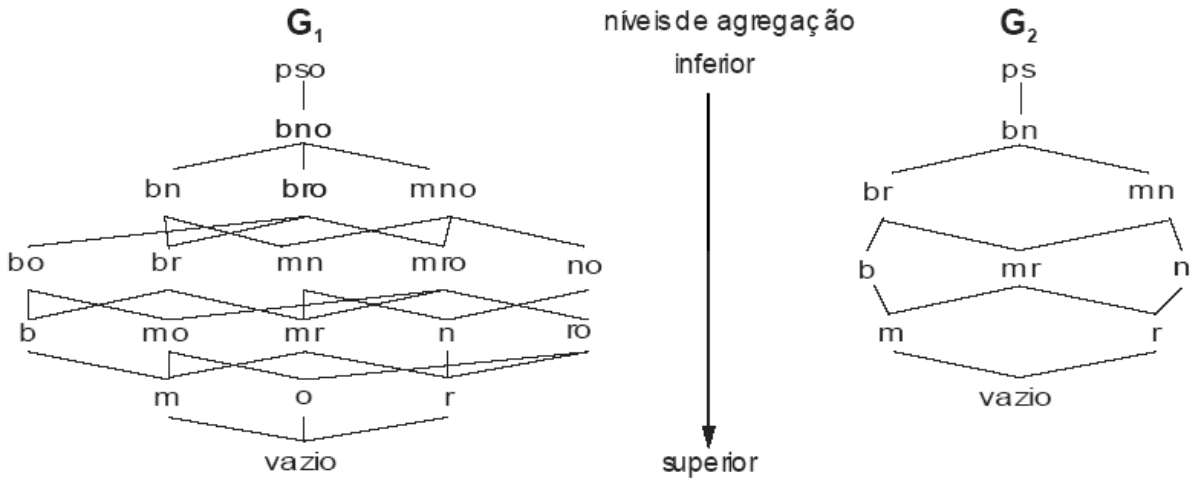


Figura 2.3 – Grafos de Derivação para *Lineitem* (G_1) e *Partsupp* (G_2)
(FORLANI; CIFERRI; CIFERRI, 2006)

2.2. ABORDAGEM DE GUPTA

Na abordagem de Gupta (1997), foi desenvolvido um *framework* teórico para o problema geral da seleção de visões em um *DW*. A heurística apresentada tem, por objetivo, otimizar o tempo de resposta das consultas para alguns casos especiais de cenários de *data warehouse*, onde a formulação do problema é dada por expressões *AND-DAG* e *ANDOR-DAG*. A Figura 2.4 exemplifica as expressões *AND-DAG* e *ANDOR-DAG*.

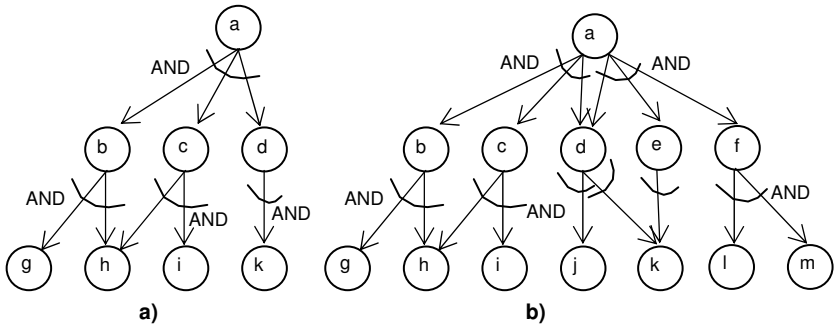


Figura 2.4 – a) *AND-DAG* - b) *ANDOR-DAG* (GUPTA, 1997)

As definições para as expressões *AND-DAG* e *ANDOR-DAG*, segundo Gupta (1997) e Gupta e Mumick (2005) são apresentadas a seguir:

- *AND-DAG* para uma consulta ou uma visão V é um grafo acíclico dirigido, que tem suas ligações base como caminhos (sem arestas de saída) e o nó V como uma origem (sem arestas de entrada). Se um nó "u" ou visão tem arestas de saída para os nós V1,V2,...Vk, então todas as visões V1,V2,...Vk são necessárias para o cálculo de "u". Essa dependência é indicada pelo desenho de um semicírculo, chamado de arco *AND*, diretamente nas arestas (u,V1), (u,V2)... (u,Vk). Cada um dos arcos AND tem um operador[2] e um custo associado a ele. Esse custo é incluso durante o cálculo de u (dependência) para V1,V2...Vk.

- *ANDOR* para uma visão ou consulta V é um grafo acíclico dirigido com V, como origem, e suas ligações base, como caminhos. Cada nó está associado com um ou mais arcos *AND* e cada ligação, um subconjunto dessas arestas de saída. Como na definição anterior, cada arco *AND* tem um operador e um custo chamado de custo de consulta associado a ele. Mais que um arco, um nó *AND* mostra múltiplas formas de calcular esse nó.

Gupta (1997) apresenta algoritmos para atender os cenários *AND-DAG* e *ANDOR-DAG*. Os algoritmos foram estendidos para uso de um conjunto de índices associados com cada visão. O benefício utilizado nessa abordagem considera um conjunto **C** de visões dispostas em grafo **G** e um conjunto **M** como um

[2] O operador associado ao arco *AND* é uma função que envolve operadores como: junção, união, agregação, etc.

conjunto de visões selecionadas. Dessa forma, o benefício é expresso por: $B(C,M)$. A definição da expressão deste benefício é apresentada pela equação: $\tau(G,M) - \tau(G_1 M \cup C)$, onde τ é uma função definida pela equação 1:

$$\tau(G,M) = \sum_{i=1}^{k} f_{Q_i} Q(Q_i,M) + \sum_{i=1}^{m} g v_i U(V_i,M) \qquad (1)$$

$Q(Q_i,M)$ denota o custo de resposta da consulta Q_i (também em um nó de G) utilizando um conjunto **M** de visões materializadas em um dado grafo **G**.
$U(V_i,M)$ é o custo de manutenção de uma visão V_i na presença de um conjunto de visões materializadas **M** e f_{Q_i} é a frequência da consulta Q_i dado um conjunto **M.**

Equação 1 - Função τ: Cálculo do Benefício das Visões do Grafo G (GUPTA, 1997)

O benefício por unidade de espaço para um conjunto **M** é expresso pela equação 2:

$$B(C,M) / S(C) \qquad (2)$$

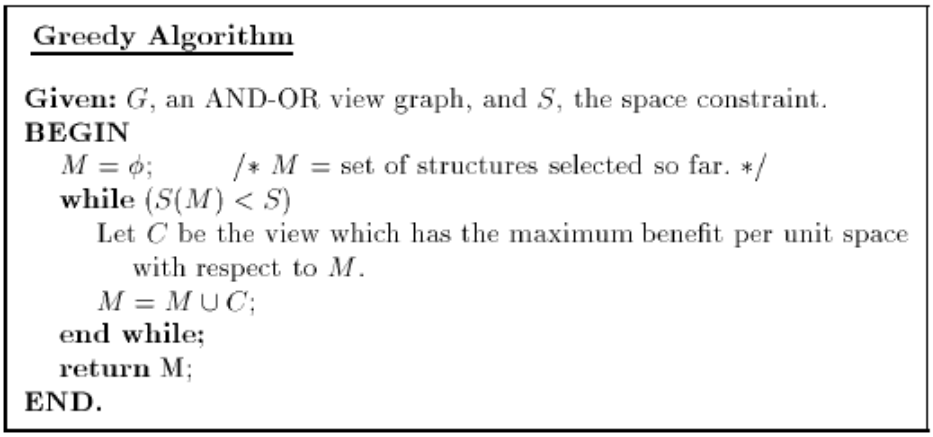
$S(C)$ é o espaço ocupado por uma visão em **C**.
$B(C,\varnothing)$ é chamado de benefício absoluto em **C**.

Equação 2 – Benefício por Unidade de Espaço (GUPTA, 1997)

Para a seleção de visões no *AND view graph* é apresentada uma heurística, a princípio, sem custo de atualização. Neste caso, é apresentado o algoritmo *greedy,* ilustrado na Figura 2.5, que seleciona em cada iteração o máximo de benefício por unidade de espaço. Logo após, é apresentada a heurística chamada de *greedy-interchange*, tratando-se de um algoritmo que começa com a solução produzida pelo *greedy,* que melhora a solução por permuta de uma visão, já selecionada por uma visão não selecionada.

O *Inner-Level Greedy* é um algoritmo que trabalha com a seleção de um subconjunto de **C,** que consiste em uma visão e alguns de seus índices selecionados, de alguma maneira, pelo *greedy*, ou um índice simples cuja visão já

tenha sido selecionada em um dos estágios anteriores. Este algoritmo é utilizado também para contemplar o *AND view graph*.



Greedy Algorithm

Given: G, an AND-OR view graph, and S, the space constraint.
BEGIN
  $M = \phi$; /* M = set of structures selected so far. */
  while $(S(M) < S)$
    Let C be the view which has the maximum benefit per unit space with respect to M.
    $M = M \cup C$;
  end while;
  return M;
END.

Figura 2.5 – Algoritmo Greedy (GUPTA, 1997)

Em se tratando do *OR view graph*, outro cenário especial em um *DW*, são apresentados algoritmos que resolvem o problema para este caso, sem custo de atualização de visões. São apresentados como parte do *framework* o algoritmo *AO-Greedy* e o algoritmo *Multi-level Greedy*, tratando-se de uma modificação do *AO-Greedy,* para melhorar o tempo de execução, como forma de garantir o desempenho.

2.3. ABORDAGEM DE CHAN, LI E FENG

Chan, Li e Feng (1999) fazem a avaliação dos custos e benefícios envolvidos em cada visão materializada, para uma determinada companhia “R”. A metodologia utilizada para análise de custo é uma adaptação do algoritmo *greedy*, apresentado por Gupta (1997), e aplicado a um estudo de caso a partir de um conjunto real de dados. No estudo de caso apresentado por Chan, Li e Feng (1999) foram aplicados os esquemas estrela e *snowflake* e, baseado na análise de custo, foram selecionados, para otimização, o custo total das consultas, o custo de manutenção e o custo para armazenamento. A consulta, a manutenção e armazenamento são calculados em termos de blocos de dados de tamanho B (equivalente ao tamanho de 2 KB). A seguir são apresentados os cálculos referentes

à avaliação dos custos e benefícios[3] para cada visão materializada, na análise realizada por Chan, Li e Feng (1999):

- O custo total das consultas envolvendo as operações de seleção, agregação e junção para processar "r" consultas do usuário, entre cada intervalo de tempo de atualização, é expresso pela equação 3:

$$Total(Cqr) = \sum_{i=1}^{r} fqi * Cq(qi) \qquad (3)$$

Cqr : custo de r consultas
qi : consulta
fqi : intervalo de tempo de atualização
Cq :custo da consulta

Equação 3 - Custo Total das Consultas (CHAN; LI; FENG, 1999)

- O custo total de manutenção para as "j" visões materializadas em um *DW* é calculado pela equação 4:

$$Total(Cm) = \sum_{i=1}^{j} fui * Cm(Vi) \qquad (4)$$

Cm : custo de manutenção
fui : intervalo de tempo de atualização (***fui*** = 1 para este estudo de caso – intervalo de tempo fixo)
Vi : visão re-computada

Equação 4 - Custo de Manutenção (CHAN; LI; FENG, 1999)

- O custo de armazenamento de uma visão em termos de bloco de dados B é expresso pela equação 5:

[3] Termo utilizado para expressar "alguma vantagem" de custo proporcionada pelo uso das visões materializadas.

$$Cstore = (Vi) = S(Vi) \qquad (5)$$

Cstore : custo de armazenamento
S : espaço

Equação 5 - Custo de Armazenamento (CHAN; LI; FENG, 1999)

- Para determinar o conjunto ótimo **M** de visões materializadas, o benefício "***Net(Bi)***" é calculado pelas equações 6 e 7.

$$Net(Bi) = \{\sum_{n=1}^{m} fq(Vru) * [Ct(Vru \leftarrow Vai) - Ct(Vru \leftarrow Vi)]\} - Cm(Vi) - Cstore(Vi) \quad (6)$$

m : visão materializada
fq : frequência de consulta
Vru : visão analisada
Vai : visão antecessora
Ct : Custo total

Equação 6 - Conjunto Ótimo e Benefício (CHAN; LI; FENG, 1999)

$$\eta i = Net(Bi) / S(Vi) \qquad (7)$$

ηi : benefício do custo por unidade de espaço ocupada por uma visão materializada
S(Vi) : espaço para visões

Equação 7 - Benefício Net(Bi) (CHAN; LI; FENG, 1999)

Se todas as visões são materializadas, o desempenho da consulta pode ser otimizada, entretanto requer um custo mais alto de manutenção e armazenamento. Para um *DW* com limitação de espaço de armazenamento e uma pequena janela (intervalo de tempo) de manutenção, materializa-se poucas visões sumarizadas, as quais devem possuir um grande benefício de ηi, a fim de reduzir o tempo de resposta de acordo com o uso do benefício, onde a quantia de espaço e custo de manutenção possam ser atendidos.

Na situação de um *DW* que possa ser levado à *manutenção off-line*[4] de visões, e possa ter um espaço em disco muito grande disponível para o armazenamento de visões materializadas, armazena-se o conjunto ótimo de visões materializadas, podendo assim minimizar a consulta e manutenção alcançando um bom desempenho na consulta (CHAN; LI; FENG, 1999).

2.4. ABORDAGEM DE KOTIDIS E ROUSSOPOULOS

Em Kotidis e Roussopoulos (2001) são discutidos dois algoritmos aplicados ao gerenciamento dinâmico de visões, de acordo com a arquitetura proposta pelo sistema *DynaMat.* Os algoritmos utilizados são conhecidos como *"replace algorithm"* e *"refinePlan".* Em se tratando do *"replace algorithm"*, o algoritmo atua especificamente na *View Pool* (espaço em disco dedicado para armazenar agregações computadas no *DynaMat*).

Se houver espaço disponível, os resultados das consultas de entrada são sempre armazenados, porém se a *View Pool* estiver cheia, o algoritmo de substituição *"replace algorithm"* é empregado, recebendo como entrada o estado atual da *View Pool,* o novo resultado computado para o fragmento e a restrição de espaço.

O fragmento armazenado é substituído somente se a *goodness* (i.e., métrica utilizada para definir a preferência de fragmentos a serem armazenados) do novo fragmento apresentar um menor resultado que o fragmento candidato à substituição.

O algoritmo *refinePlan* parte de um plano inicial de atualização e substitui fragmentos na *View Pool*. Como primeiro passo, o algoritmo descarta todos os fragmentos onde a estimativa de custo é maior que W (janela de manutenção)

Os fragmentos podem ser atualizados, de acordo com o algoritmo, a uma complexidade $O(|V|^2)$. Porém, em muitos casos, somente uma pequena fração dos resultados armazenados é descartada. A métrica *goodness* é computada em

[4] Manutenção decorrente de informações estatísticas (cálculos realizados sobre os metadados mantidos pelo DW que deverão conter todas as informações que são monitoradas online).

tempo constante, de acordo com o custo de espaço e substituições dos fragmentos, definidos na equação 8:

$$O(k_2 * |V| * O(1)) + O(k_2 |V|)^6 = O(K_2 |V|) \qquad (8)$$

V : número de fragmentos na View Pool
K₂ : fragmento substituído
O : complexidade de atualização dos fragmentos

Equação 8 – Métrica de Goodness (KOTIDIS; ROUSSOPOULOS, 2001)

2.5. ABORDAGEM DE VALLURI, VADAPALLI E KARLAPALEM

Valluri, Vadapalli e Karlapalem (2002) apresentam um algoritmo baseado na relevância de visão e o algoritmo *VRDS* (*View Relevance Driven Selection*) para selecionar visões em ambiente de *DW*.

Foi utilizado o algoritmo *VRDS* para obter um conjunto ótimo de visões para materializar, utilizando um plano de processamento de consultas. É possível a execução de um plano com todas as consultas para formar um Grafo Acíclico Dirigido *AND (DAG AND).* Todos os possíveis planos juntos formam o Grafo Acíclico Dirigido *AND-OR*. Os grafos acíclicos *DAG AND* e *DAG AND-OR* podem ser vistos na Figura 2.6.

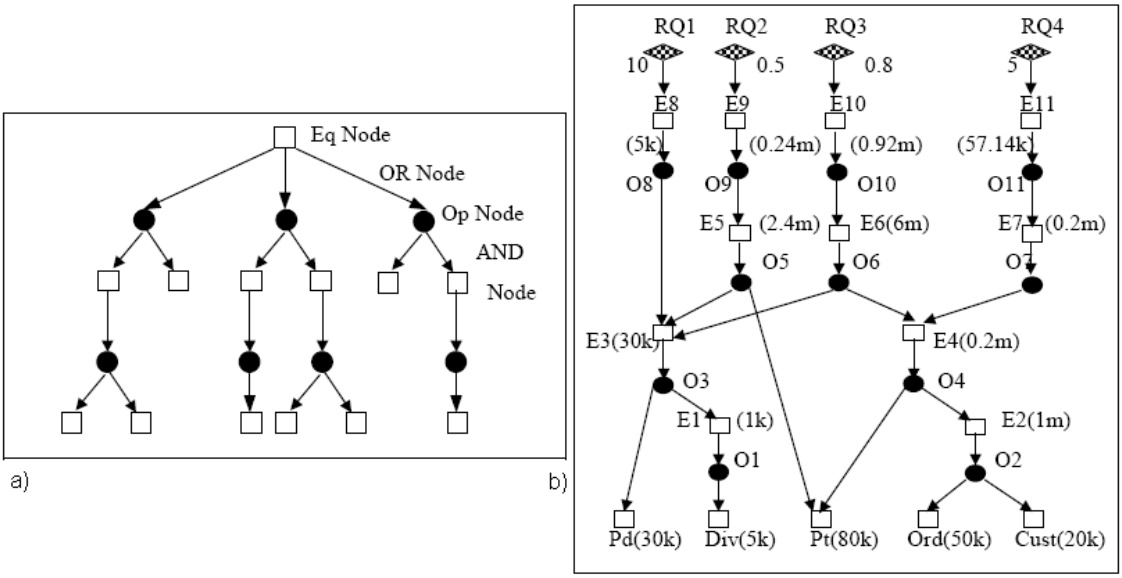


Figura 2.6 – Grafos a) *DAG AND* b) *DAG AND-OR*
(VALLURI; VADAPALLI; KARLAPALEM, 2002)

Após obter um plano *DAG AND* ótimo, é selecionado um subconjunto de visões para materializar, baseado na relevância da visão, que indica como a presença de uma visão em um conjunto afeta o benefício de outras visões, assim como afeta o custo de processamento das consultas e o custo de manutenção da atualização.

Para a definição da relevância de visão de um par de visões são usadas as matrizes a seguir:

BaseView Matrix: Essa matriz define o relacionamento entre a visão e a tabela base. É uma matriz (b x n). O elemento da matriz **BV[Bi][Nj] = 1** se a tabela de base Bi é filha do nó Nj. Caso contrário, **BV[Bi][Nj] = 0.**

Connectivity Matrix: A matriz (n x n) é definida se o nó ∈ N, e está conectado a outro nó ∈ N. O elemento da matriz **Conn[Ni][Nj] = 1** se o nó Ni é filho de Nj ou vice versa. Caso contrário, **Conn[Ni][Nj] = 0**.

QueryUsage Matrix: A matriz (rq x n) define se uma consulta é somente leitura e se acessa um nó ∈ N.

UpdateBaseUsage Matrix: Essa matriz é definida se uma consulta de atualização acessar a tabela base. O elemento da matriz **UBU [UQi][Bj] = 1** se consulta de atualização UQi acessar a tabela base no nó Nj. Caso contrário, **UBU [UQi][Bj] = 0**.

UpdateFreq Matrix: Essa matriz (uq x n) define a frequência que cada consulta atualizada acessa o nó ∈ N. O elemento da matriz **UF [UQi][Nj] = Frequência de consulta UQi**, se **UU[UQi][Nj] = 1**. Caso contrário, **UU[UQi][Nj] = 0**.

QCost (**Vi, M**) = Custo da formação de um nó **Vi** na presença do conjunto de visões materializadas em **M**.
UCost (**Vi, M**) = Custo de manutenção do nó **Vi** na presença do conjunto de visões materializadas em **M**.

Direct child Matrix: Essa matriz define quais as direções dos descendentes de um nó. Todos os nós que são conectados diretamente a um nó são chamados de

descendentes diretos. O elemento da matriz **directchild [Vi][Vj] = 1** se **Vj** é o descendente direto de **Vi**. Caso contrário, **directchild [Vi][Vj] = 0**.

Flags Matrix: Essa matriz (n x n) define qual(is) visão(s) é (são) materializada(s). Após toda iteração, a matriz da visão de relevância e a matriz de flags são recomputadas. A Tabela 1 apresenta quatro valores de *flag.* Os valores dos *flags* representam se uma ou mais visões particulares serão materializadas.

Tabela 1 – Matriz de *Flags*

Flags [Vi][Vj]	**Ação**
0	Ambos Vi e Vj são para ser materializados
1	Somente Vi é para ser materializado
2	Somente Vj é para ser materializado
3	Nenhuma visão será materializada

(VALLURI; VADAPALLI; KARLAPALEM, 2002)

A partir dos resultados colhidos dos experimentos realizados por Valluri, Vadapalli e Karlapalem (2002) foi observado que, quando a frequência de atualização de consultas é muito alta, o algoritmo *greedy* (GUPTA, 1997) não adquire um conjunto ótimo de visões materializadas. Porém, o algoritmo *VRDS* seleciona um conjunto de maneira que o custo seja minimizado.

Os autores citam que o algoritmo *greedy* (GUPTA, 1997) seleciona as visões que têm menos frequência de atualização. Foi verificado que o algoritmo *VRDS* seleciona um conjunto de visões de maneira que o custo total seja minimizado.

Valluri, Vadapalli e Karlapalem (2002) observaram que o algoritmo *greedy* de Gupta (1997) seleciona, relativamente, menos visões que o algoritmo *VRDS,* para todo espaço avaliado. O algoritmo *VRDS* consegue selecionar visões de tamanho pequeno, importantes na redução do custo total de processamento, porém, aumenta a margem de custo de atualização, proporcionando assim, uma diminuição do custo de processamento das consultas.

A representação completa e o cálculo da matriz de relevância de visão são transparentes ao usuário final, e podem ser incorporados a uma ferramenta de projeto para *DW*.

2.6. ABORDAGEM DA IBM

A abordagem de Zilio et. al. (2004) descreve um algoritmo para determinar, simultaneamente, um conjunto ótimo de visões materializadas, definidas nesta abordagem por *MQT* (*Materialized Query Tables*), e de índices para uma *workload,* considerando a restrição de espaço em disco.

A ideia chave da abordagem é estender o uso do otimizador de consulta do banco de dados, para sugerir objetos candidatos (*MQTs* e índices), e assim, avaliar o benefício e o custo dos objetos candidatos. O otimizador foi estendido também para explorar a otimização de múltiplas consultas com a finalidade de descobrir subexpressões comuns entre as consultas definidas como candidatas por uma *workload*.

A abordagem proposta pelos autores melhora o custo-benefício, em razão das *MQTs* recomendadas pelo otimizador, porém a solução final é encontrada pelo algoritmo *ADD_COMBINE*.

É permitida que uma boa faixa de *MQTs* seja selecionada e mantida. Em particular, a abordagem suporta todas as atualizações das *MQTs* (quando atualizadas periodicamente pelo usuário) e atualizações imediatas (atualizadas quando as tabelas base são atualizadas).

O otimizador estima o custo de execução individual das consultas, obtendo informações, automaticamente colhidas, para avaliação do otimizador. As informações obtidas são:

- Características do banco de dados incluindo o esquema, estatísticas como as cardinalidades de tabelas e colunas, e outras características do modelo físico, como índices, particionamento, definições de visões e visões materializadas.

- Informações da configuração do sistema, incluindo recursos estimados de processamento, a latência, taxa de transferência e o tempo de acesso no disco etc.

O DB2 *Designer Advisor*, por meio do módulo de geração de *MQT* candidatas, define, inicialmente, para uso do algoritmo, um conjunto de *MQT* candidatas, as quais são geradas de acordo com os métodos a seguir:

- Usa as próprias consultas fornecidas na *workload* como candidatas *MQT*;
- Usa as visões não materializadas, definidas pelo usuário como candidatas *MQT*;
- Utiliza a técnica de otimização de múltiplas consultas para a busca de subexpressões entre as múltiplas consultas.

De acordo com Zilio et.al (2004), o algoritmo de seleção das *MQTs* obtém todos os objetos por ordem decrescente de peso, definido pelo benefício dividido pelo custo, onde cada valor alto de peso sugere que uma *MQT* ou um índice seja um bom candidato para a seleção. O benefício da *MQT* é a soma sobre todas as consultas, da frequência da consulta, tempo de execução para uma consulta e/ou atualização. Formalmente o peso *w(A)* de uma *MQT* A ou de um índice é definido pela equação 9.

$$w(A) = \frac{\sum_{q \in Q} f(q) * B(q)}{D(A)} \qquad (9)$$

Q : conjunto das consultas da workload para uso de *MQT* A;
$f(q)$ **:** frequência da consulta q;
$B(q)$: desempenho para uma consulta q;
$D(A)$: espaço em disco consumido por A.

Equação 9 – Cálculo de Peso para uma *MQT* ou índice (ZILIO ET AL.; 2004)

A estimativa de desempenho para a execução de cada consulta q é definida pela equação 10.

$$B(q) = \mathrm{E}(q) - M(q) \qquad (10)$$

$\mathrm{E}(q)$: custo estimado de execução de q na *workload*;
$M(q)$: custo estimado do plano resultante para q.

Equação 10 – Estimativa de Desempenho (ZILIO ET AL.; 2004)

Quanto mais índices e *MQTs* são introduzidos em cada iteração do algoritmo, melhor o desempenho. Devido às atualizações, o desempenho fica sujeito a uma penalidade por manutenção, em cada expressão de atualização, seja por índices ou atualizações imediatas de *MQTs* ou por atualizações periódicas de *MQT*.

O custo de manutenção associado com uma atualização imediata da *MQT* pode ser determinado de duas maneiras:

- Pelo uso estimado do número de linhas para todas as *MQTs*, a frequência estimada e o número de linhas atualizadas, para cada instrução *SQL* (*Update, Insert e Delete*) e para o cálculo do custo de atualização bruto;

- Para cada *MQT* e instrução *SQL* atualizada, determina-se a diferença estimada de tempo para cada instrução de atualização, entre as *MQTs* usadas e as não usadas.

Para cada atualização completa da *MQT* A, é adicionada, ao benefício de "A", uma penalidade C(A) para toda materialização de "A", com a frequência de g(A). A configuração padrão para g(A) é "1", incluindo efetivamente o custo da menor materialização de "A", durante o intervalo de tempo que está sendo otimizado. Assim, o peso para a atualização completa de *MQT* A inclui a manutenção, estimada pela equação 11.

$$W(A) = \frac{\sum_{q \in Q} f(q) * B(q)}{D(A)} - g(A) * \mathrm{C}(A) / D(A) \qquad (11)$$

$g(A)$: frequência de atualização
$\mathrm{C}(A)$: penalidade por atualização

Equação 11 – Cálculo de Manutenção de uma *MQT* (ZILIO ET AL.; 2004)

2.7. ABORDAGEM DE YOUSRI, AHMED E MAKKY

De acordo com a abordagem de Yousri, Ahmed e Makky (2005), são propostos novos algoritmos para seleção de visões para materializar. Dois objetivos são considerados: o primeiro propõe uma abordagem para resolver o problema da seleção de visões materializadas, considerando otimizações *multi-query* e a otimização do processo de manutenção. Como segundo objetivo, propõe o uso de uma estratégia simples que reduz o espaço de pesquisa para o problema de seleção de visão e reduz também a complexidade de tempo para linear, ao invés de uma complexidade de tempo quadrática.

Na primeira abordagem, três algoritmos são propostos baseados na mesma estratégia de busca, porém, diferentes quanto ao uso da estratégia de manutenção. Os algoritmos propostos são: o *IRVSA* (I*ncremental Recomputation strategy View Selection Algorithm),* o *IVSA (Incremental Strategy View Selection Algorithm*) e o *RVSA (Recomputation strategy View Selection Algorithm)*. A segunda abordagem apresenta o algoritmo *IMDVSA (Recomputation strategy Materialized Descendants View Selection Algorithm).*

O cálculo do benefício da visão depende do cálculo do custo total formado por dois componentes: o custo total da consulta referente a todas as consultas e o custo de manutenção total de todas as visões materializadas.

O benefício da visão **v** dado **M** como um conjunto de visões materializadas é calculado de acordo com a equação 12:

$$benefit(v \mid M) = TotalCost(M) - TotalCost(M \cup \{v\}) \qquad (12)$$

$TotalCost(M)$: é o custo total quando **M** é o conjunto de visões materializadas

$TotalCost(M \cup \{v\})$: é o custo total quando a visão **v** é adicionada ao conjunto **M**.

Equação 12 – Benefício v Dado o conjunto M (YOUSRI; AHMED; MAKKY, 2005)

O cálculo do custo total de **M**, para que **M** seja o conjunto de visões materializadas é expresso pelas equações 13, 14, 15 e 16:

$$TotalCost(M) = TotalMCost(M) + TotalQCost(Q \mid M) \qquad (13)$$

$TotalMCost(M)$: representa o custo total de manutenção para o conjunto **M**, calculado pela equação 14.

$TotalQCost(Q \mid M)$: é o custo total da consulta para o conjunto **Q** de consultas submetidas ao DW, dado que **M** é o conjunto de visões materializadas como mostra a equação 15.

Equação 13 – Custo Total do Conjunto M de Visões
(YOUSRI; AHMED; MAKKY, 2005)

$$TotalMCost(M) = \sum_{v \in M} (f_u(v) * mntc(v \mid M)) \qquad (14)$$

$\mathbf{f_u}$: frequência de atualização de **v**;

$mntc(v \mid M)$ é o custo de manutenção de **v** dado **M** de acordo com a equação 16.

Equação 14 – Custo de Manutenção de v (YOUSRI; AHMED; MAKKY, 2005)

$$TotalQCost(Q \mid M) = \sum_{q \in Q} (f_q(q) * c(q \mid M)) \qquad (15)$$

$f_q(q)$: a frequência de representar uma consulta q;

$c(q \mid M)$ é o custo da consulta **q** dado **M** como mostra a equação 15.

Equação 15 – Custo Total da Consulta para Q (YOUSRI; AHMED; MAKKY, 2005)

$$mntc(v \mid M) = min(imntc(v \mid M), rmntc(v \mid M)) \qquad (16)$$

$rmntc(v \mid M)$:custo de re-computação da atualização de **v** dado **M** como mostra a equação 17.

$imntc(v \mid M)$: custo de atualização incremental de **v** dado **M** como mostra a equação 18.

Equação 16 – Custo de Manutenção de v dado M (YOUSRI; AHMED; MAKKY, 2005)

Custo de re-computação da atualização da visão **v** é expresso pela equação 17:

$$rmntc(v \mid M) = c(v \mid M) + matc(v) \qquad (17)$$

$c(v \mid M)$: custo de computação de "v" dado M conforme a equação 19 e $matc(v)$ é o custo de materialização de **v**.

Equação 17 – Custo de Re-Computação (YOUSRI; AHMED; MAKKY, 2005)

O custo de atualização incremental de uma visão **v** para um número **"n"** de vértices na qual depende de **v** é expresso pela equação 18:

$$imntc(v \mid M) = \begin{cases} \sum_{k=1}^{5} (c(\delta_v^k \mid M) + mergec(\delta_v^k)) & n = 2 \\ c(\delta_v \mid M) + mergec(\delta_v) & n = 1 \end{cases} \quad (18)$$

n=2 quando uma operação de união é considerada e ***n=1*** quando uma operação de seleção ou uma projeção são consideradas.

Equação 18 – Custo de Atualização Incremental (YOUSRI; AHMED; MAKKY, 2005)

O custo de computação de uma visão **v** é expresso pela equação 19:

$$c(v \mid M) = \begin{cases} Localc(v) + \sum_{u \in S(v)} childc(v \mid M) \ \ if \ \ S(v) \neq \varnothing \\ commc(v) \ if \ S(v) = \varnothing \ (v \ is \ a \ base \ relation) \end{cases} \quad (19)$$

$S(v)$: conjunto de vértices tendo arcos apontando para **v** (filhos de **v**)

$Localc(v)$: custo de operação local de **v**

$childc(v \mid M)$: custo de computação de child **v** como mostra a equação 20

$commc(v)$: custo de comunicação associado à transferência de dados entre o *DW* e o local da base relacional de **v** em ordem para computar os pais de **v**

$S(v) \neq \varnothing$: sempre existe quando alcança o nó folha de um *MVPP*[5] *(Multiple View Processing Plan).* Mais detalhes sobre a estrutura *MVPP* são mostrados na Figura 2.7.

Equação 19 – Custo de Computação da Visão v (YOUSRI; AHMED; MAKKY, 2005)

O Custo de computação *Child* é expresso pela equação 20:

$$childc(v \mid M) = \begin{cases} reuse(v) & if \ v \in M \\ c(v \mid M) & if \ v \notin M \end{cases} \quad (20)$$

$reuse(v)$: custo de reuso do resultado de uma visão materializada **v**.

$c(v \mid M)$: custo de computação de **v** conforme a equação 19.

Equação 20 – Custo de Computação *Child* (YOUSRI; AHMED; MAKKY, 2005)

[5] Estrutura construída pela combinação de planos ótimos de consulta de usuários no *Data Warehouse*. Maiores esclarecimentos são mostrados na Figura 2.7.

Custo de consulta da visão **v** que é resultado final para a consulta **q**, sendo que o custo para a consulta **q** é expresso pela equação 21.

$$c(q \mid M) = \begin{cases} Access(v) \ if \ v \in M \\ c(v \mid M) \ \ if \ v \notin M \end{cases} \qquad (21)$$

$Access(v)$: custo de acesso aos blocos de **v**.

$c(v \mid M)$:custo de computação de **v**, dado **M** de acordo com a equação 19.

Equação 21 – Custo da Visão v para Consulta q (YOUSRI; AHMED; MAKKY, 2005)

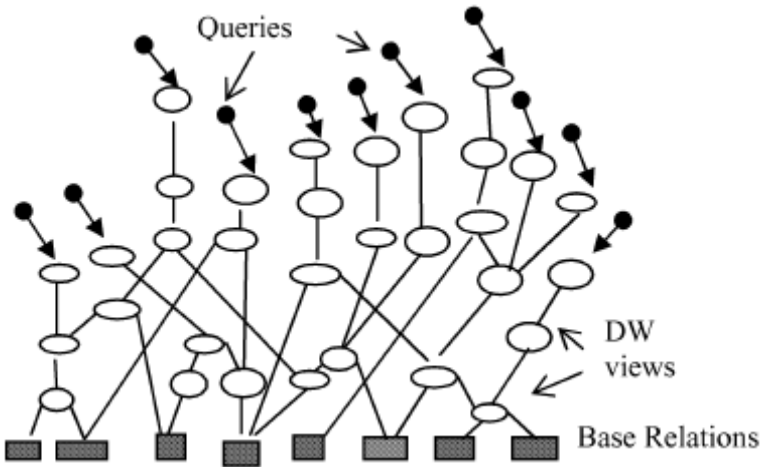


Figura 2.7 – Estrutura MVPP (YOUSRI; AHMED; MAKKY, 2005)

2.8. ABORDAGEM DE MARTINS, BELO E NOVAIS

Em Martins, Belo e Novais (2005) são comparados os comportamentos de dois algoritmos: o *BPUS (Benefit Per Unit of Space)* de *Gupta (1997)* e o A* de Gou, Yu e Lu (2006), ambos os algoritmos de busca exaustiva, ou seja, determinísticos. O algoritmo *BPUS* utiliza grafos acíclicos orientados para representar as visões e consultas, sendo que cada nó desse grafo ***G*** representa uma operação de agregação, seleção, projeção ou junção. Em cada iteração do algoritmo são selecionadas as visões, com o máximo de benefício por unidade de espaço.

O cálculo do benefício de uma visão consiste na diferença dos custos de processamento dos conjuntos de visões materializadas, com e sem a

visão. O custo total de um conjunto de visões materializadas é dado pela equação 22:

$$B(v,M) = \tau(G,M) - \tau(G,M \cup \{v\}) \qquad (22)$$

B : Benefício, ***V*** : Visão,
M : Conjunto de visões selecionadas (após cada iteração),
G : Grafo e τ : Custo de processamento.

Equação 22 – Custo Total de um Conjunto de Visões Materializadas (MARTINS; BELO; NOVAIS, 2005)

O benefício da unidade de espaço é calculado através do quociente entre o benefício apresentado pela visão ***v*** e seu espaço ***S(v)***, expresso pela equação 23:

$$BPUS(v,M) = \frac{B(v,M)}{S(v)} \qquad (23)$$

$S(v)$: Espaço
v : Visão
B : Benefício
M : Conjunto de visões selecionadas

Equação 23 – Benefício por Unidade de Espaço (MARTINS; BELO; NOVAIS, 2005)

O algoritmo A* apresentado na abordagem de Martins, Belo e Novais (2005) é utilizado para seleção de visões, sob a restrição de tempo de manutenção, e pode ser estendido para uso da restrição de espaço. São utilizados grafos acíclicos dirigidos, como forma de representação do problema. Para um dado cubóide ***G***, conforme Figura 2.8, o algoritmo A* expande uma árvore de busca binária denominada *TG*. A seguir é descrita a maneira como o algoritmo lida com o conjunto de visões, conforme apresentado em (MARTINS BELO; NOVAIS, 2005). Cada nó é do tipo ***(Nx,Mx)***, denotado por ***x=(Nx,Mx)***, onde ***Nx*** é o conjunto de visões visitadas e ***Mx*** é o conjunto de visões selecionadas para materialização $(Mx \subseteq Nx)$. É seguida a ordem de inserção predeterminada ***(v1,v2,...,vn)***, onde cada visão v_i é considerada para materialização no i-ésimo nível de TG. O descendente esquerdo de ***x*** é definido por $l(x) = (N_x \cup \{v_{v+1}\}, M_x)$, o que significa que a visão recém inserida $\boldsymbol{v_{i+1}}$ não será materializada. O descendente

direito ***x*** é definido por $r(x) = (N_x \cup \{v_{i+1}\}, M_x \cup \{v_{i+1}\})$, e nesse caso, a visão recém inserida v_{i+1} será materializada. Finalmente, no nível mais baixo da árvore de busca $N_x = V$, significando que todas as visões do cubóide ***G*** foram consideradas ou não para materialização (***V*** denota o conjunto de visões ou nós em ***G***). O benefício de expandir o nó em ***x*** é a soma de duas funções ***g(x)*** e ***h(x)***. A primeira calcula o benefício adquirido e a segunda o benefício estimado a materializar ***x***. O processo de construção da árvore binária de busca A* cobre todo o espaço de 2^n soluções possíveis. Isso assegura que o algoritmo contempla a solução ótima. A complexidade do algoritmo A* situa-se entre ***O(n)*** e $O(2^n)$ dependendo da qualidade da função estimadora ***h(x)***. O pior caso do algoritmo A* tem comportamento exponencial, portanto apresenta um pior desempenho em relação ao algoritmo *BPUS*.

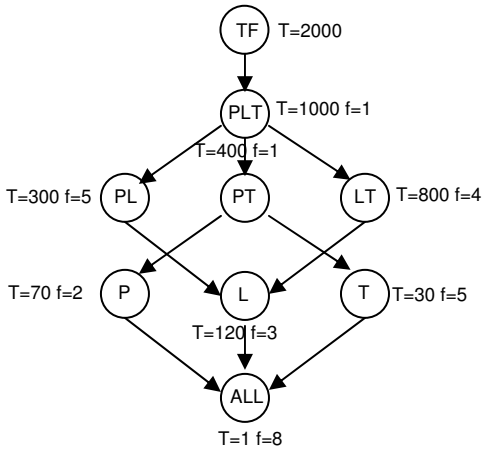


Figura 2.8 – Cubóide G (MARTINS; BELO; NOVAIS, 2005)

De acordo com os resultados obtidos por experimentos realizados por Martins, Belo e Novais (2005), mostrou-se que o algoritmo *BPUS* tem melhor comportamento que A*, devido à forma que foi concebido. Sua concepção lógica permite-lhe implementar uma visão extra, desrespeitando a restrição de espaço. Contudo, depois de modificado de forma a restringir-se ao parâmetro de espaço disponível, foi verificado que seu comportamento, em termos de custo de processamento, é semelhante ao de seu concorrente, o A*. Segundo Martins, Belo e Novais (2005) o A* apresenta ser mais rápido para encontrar a solução.

2.9. ABORDAGEM DE THEODORATOS E SIMITSIS

Theodoratos e Simitsis (2006) tratam o problema da decisão de quais visões serão materializadas em um *DW*, de modo a satisfazer a minimização das funções de custo, tais como: custo de avaliação da consulta, custo de manutenção da visão, custo operacional, ou satisfazer uma restrição orientada a usuário ou orientada a sistema. No caso das restrições orientadas a sistema são consideradas a restrição de espaço, a restrição de custo de manutenção da visão, a automanutenibilidade e a resposta de consultas, utilizando, exclusivamente, as visões materializadas. As restrições orientadas a usuário são divididas em:

- Restrição de atualidade dos dados: É considerado o tempo decorrido entre a resposta da consulta do usuário e o tempo da alteração mais recente ocorrida da relação fonte.

- Restrição de tempo de resposta da consulta: Estabelece o tempo necessário para executar uma determinada consulta utilizando as visões materializadas do *DW*, porém sem ultrapassar o limite máximo definido pelo usuário.

Theodoratos e Simitsis (2006) propõem resolver o problema da seleção de visões realizando tarefas como: busca por espaço como alternativa para um conjunto de visões materializadas e a utilização de algoritmos para seleção de um conjunto ótimo de visões ou quase ótimo para o espaço buscado. A abordagem proposta por Theodoratos e Simitsis (2006) pode ser utilizada para diferentes tipos de consultas, com foco principal nas consultas aplicadas sobre o esquema estrela.

2.10. CONSIDERAÇÕES FINAIS

Este capítulo apresentou algumas das diversas abordagens existentes na literatura sobre políticas para a seleção de visões a materializar, para *Data Warehouses*, consideradas relevantes, diante das referências encontradas durante as pesquisas.

Os assuntos discutidos nesse capítulo são relevantes no embasamento do presente trabalho. Dessa forma, é exibido a seguir, na Tabela 2, o resumo das abordagens estudadas, de acordo com a linha do tempo. As

abordagens de Martins, Belo e Novais (2005) e Theodoratos e Simitsis (2006) não estão entre as abordagens contidas, na Tabela 2, por não apresentarem uma solução nova para a materialização de visões, e sim a discussão de abordagens existentes.

Algumas das abordagens citadas tratam o custo de manutenção de visões. Isso significa que, tais abordagens são utilizadas para a materialização de visões durante o uso do *DW*. A abordagem apresentada no presente trabalho atua na fase de projeto do *DW*, de modo a selecionar visões para serem utilizadas na implantação do *DW*.

Tabela 2 – Tabela Resumo das Abordagens

Autores	**Algoritmo**	**Política Utilizada**
(GOU; YU; LU, 2006)	*A* Search.*	• Espaço em disco.
(YOUSRI; AHMED; MAKKY, 2005)	*IRVSA,IVSA, RVSA, IMDVSA.*	• Frequência das consultas; • Frequência da atualização das visões; • Espaço em disco.
(ZILIO ET. AL.; 2004)	*ADD_COMBINE.*	• Frequência das consultas; • Espaço em disco; • Otimização de consultas; • Uso de índices; • Custo de Manutenção de consultas e índices.
(GOLFARELLI; MANIEZZO; RIZZI, 2004)	*VFP (Vertical Fragmentation Problem):* Faz menção que utiliza um algoritmo da literatura para seleção das visões (Não cita qual algoritmo).	• Não trabalha com política própria de seleção de visões. • Otimização de espaço (fragmentação);
(VALLURI; VADAPALLI; KARLAPALEM, 2002)	*VRDS (View Relevance Driven Selection).*	• Espaço em disco
(KOTIDIS; ROUSSOPOULOS, 2001)	*Replace algorithm, refinePlan algorithm.*	• Espaço em disco; • Tempo de atualização;
(CHANG; LI; FENG, 1999)	*Greedy algorithm (adapted).*	• Espaço em disco; • Tempo de atualização; Frequência das consultas.
(GUPTA, 1997)	*Greedy algorithm, Greedy-Interchange Algorithm* e *Inner-Level Greedy Algorithm.*	• Espaço em disco; • Uso de Índices.
(HARINARAYAN; RAJARAMAN; ULLMAN, 1996)	*Greedy algorithm*	• Espaço em disco.

Além das abordagens apresentadas nesse capítulo, vêem surgindo trabalhos mais recentes sobre materialização de visões em *Data Warehouse*. Nesse sentido podem ser citadas as abordagens: (BOUKRA; NACER; BOUROUBI, 2007) e (ZHOU ET AL.; 2007).

O próximo capítulo faz menção ao trabalho de Golfarelli, Maniezzo e Rizzi (2004), por discutir e aplicar conceitos que se aproximam da abordagem proposta neste trabalho.

3. ABORDAGEM DE GOLFARELLI, MANIEZZO E RIZZI

Este capítulo tem, por objetivo, apresentar um estudo sobre o trabalho de Golfarelli, Maniezzo e Rizzi (2004), por se tratar da abordagem que mais se assemelha ao trabalho realizado. Os autores da abordagem não tratam especificamente da seleção de um conjunto ótimo de visões, conforme as abordagens apresentadas no capítulo 2, e sim da técnica de fragmentação vertical. Por sua vez, a fragmentação vertical consiste na minimização do tempo de processamento das consultas, submetidas a uma *workload*, levando em consideração o custo dessas consultas, as visões candidatas e os fragmentos gerados.

A abordagem de Golfarelli, Maniezzo e Rizzi (2004) apresenta melhorias nas funções de custo e no algoritmo de fragmentação vertical, propostos anteriormente em Maio, Golfarelli e Rizzi (2000). Para dar suporte a tal metodologia, foi implementado um protótipo, a ferramenta *WanD*, de acordo com Golfarelli, Rizzi e Saltarelli (2002). Por este motivo, a *WanD* passa a ser também um tópico importante a ser apresentado neste capítulo.

3.1. ABORDAGEM CONCEITUAL

O trabalho de Golfarelli, Maniezzo e Rizzi (2004) investiga como responder a uma *workload* (carga de consultas), por meio da materialização de visões em fragmentos verticais. As dimensões do cubo são atributos que representam possíveis dimensões para análise. Em uma visão, o conjunto de dimensões determina as medidas, sendo que cada dimensão pode ser associada a uma hierarquia de atributos por meio da qual é descrita.

A abordagem proposta, pelos autores, tem por meta unificar duas ou mais visões, em um único fragmento, ou até mesmo particionar uma visão em dois ou mais fragmentos.

De acordo com Golfarelli, Maniezzo e Rizzi (2004), novas contribuições são apresentadas, pois outras abordagens não sugerem como pode

ser determinada uma boa fragmentação. Dessa forma, é apresentado o problema da fragmentação, sob nova formulação, e formalizado por meio da programação linear que, dada uma *workload*, uma restrição global de espaço em disco para materialização e uma função de estimativa de custo por execução, de cada consulta, em cada fragmento, produz-se uma ótima fragmentação.

Um esquema de cubo $\mathscr{C}$ é definido por um conjunto de dimensões, ***Gby(*$\mathscr{C}$*)***, e por um conjunto de medidas, ***Meas(*$\mathscr{C}$*)***. Tipicamente, cada dimensão é associada a um conjunto de atributos que a descreve. Esses atributos são organizados em uma árvore dirigida, chamada hierarquia, onde a raiz da árvore é a dimensão, cujos arcos representam dependências funcionais.

Um esquema multidimensional $\mathscr{D}$ é definido por um conjunto de esquemas de cubo. Como exemplo, é apresentado um esquema multidimensional de acordo com o *TPC-H*. Este exemplo inclui dois esquemas de cubo, Item (IT) e Fornecedor_Peça (FP), os quais descrevem a composição de ordens de remessa e fornecimento de uma companhia. Na Figura 3.1 **IT** e **FP** são caracterizados por hierarquias, onde as dimensões são diferenciadas por marcações em negrito.

- ***Gby(IT)*** = {peça, pedido, data_envio, modo_envio, flag_retorno, status, data_execução, data_recebimento};
- ***Meas(IT)*** = {preço_unit, quant, preço_est, desconto, preço_desc, debito, taxa};
- ***Gby(FP)*** = {peça, fornecedor, data};
- ***Meas(FP)*** = {quant_disp, custo_forn}.

Peça→ container; **Peça**→ marca → MFRG; **Peça** → tipo; **Peça** → tamanho.

Pedido→ prioridade_envio; **Pedido**→ estado_ped; **Pedido**→ cliente→país_cliente→ região_cli; **Pedido**→data_ped → mês_pedido→ quadrimestre_pedido→ ano_ped.

Fornecedor→ país_forn → região_forn.

Data→ mês→ quadrimestre→ ano.

Figura 3.1 – Hierarquias dos Esquemas IT (Item) e FP(Fornecedor_Peça)
Adaptado de (GOLFARELLI; MANIEZZO; RIZZI, 2004)

3.1.1. CARGA DE CONSULTAS (WORKLOAD)

Em princípio, uma *workload* para um ***MD*** (Multidimensional Database, ou base de dados multidimensional) significa um conjunto de consultas comumente formuladas neste ***MD***. São consideradas, para um ***MD***, classes de consultas ***GPSJ*** (***G****eneralized* ***P****rojection/****S****election/****J****oin*, ou Projeção/Seleção/Junção Generalizada) definidas por uma seleção executada, sobre uma projeção generalizada, sobre uma seleção, sobre uma junção.

Formalmente: Dada uma consulta ***q***, denota-se por ***Gby(q)***, ***Meas(q)***, ***Sel(q)***$\in$ [0...1] e ***Freq(q)*** $\in$ [0...1], respectivamente, o conjunto agrupado de ***q***, as medidas que ele retorna, sua seletividade e, por fim, sua frequência dentro da *workload*.

Dado um esquema de cubo $\mathscr{C}$, ***G***($\mathscr{C}$) denota o conjunto dos conjuntos agrupados de todas as possíveis consultas em $\mathscr{C}$, onde a consulta ***q*** em $\mathscr{C}$ é bem-formada ***Meas(q)*** $\subseteq$ ***Meas(***$\mathscr{C}$***)*** e ***Gby(q)*** $\in$ $\mathbf{G}_{\mathscr{C}}$.

3.1.2. VISÕES CANDIDATAS

Os autores assumem, nessa abordagem, que um conjunto de visões candidatas já foi determinado para cada esquema de cubo envolvido na *workload*. ***Cand***($\mathscr{C}$) denota o conjunto do agrupamento dos conjuntos de visões candidatas determinado por um esquema de cubo $\mathscr{C}$***.***

Para cada esquema de cubo toda medida deve ser materializada em seu conjunto grupado primário (isto é, em sua forma não agregada), assim: ***Gby(***$\mathscr{C}$***)*** $\in$ ***Cand***($\mathscr{C}$) para todo $\mathscr{C}$***.***

3.1.3. MATERIALIZAÇÃO DE FRAGMENTOS DE VISÕES

Na abordagem de Golfarelli, Maniezzo e Rizzi (2004) uma visão (i.e.,mais adequadamente chamada de fragmento) é caracterizada por seu conjunto

agrupado e por seu conjunto de medidas, pertencentes ao esquema de cubo no esquema multidimensional.

Dado um esquema multidimensional $\mathcal{D}$, um fragmento $\mathcal{F}$ em $\mathcal{D}$ é definido por um conjunto agrupado ***Gby(*$\mathcal{F}$*)*** e por um conjunto de medidas ***Meas(*$\mathcal{F}$*)***, onde ***Gby(*$\mathcal{F}$*)*** $\in$ $\mathcal{G}_{\mathcal{C}_k}$ para cada $\mathcal{C}_k \in \mathcal{D}$, tal que ***Meas(*$\mathcal{F}$*)*** $\cap$ ***Meas (*$\mathcal{C}_k$*)*** $\neq$ ***0.*** Em termos de implementação relacional, cada fragmento $\mathcal{F}$ corresponde a uma tabela fato, tendo os atributos em ***Gby(*$\mathcal{F}$*),*** como chave, e as medidas em ***Meas(*$\mathcal{F}$*)***, como atributos não chave.

Exemplo: A tabela fato para o fragmento $\mathcal{F}$, onde ***Gby***($\mathcal{F}$)***=*** {Peça} e Meas($\mathcal{F}$) = {quant_disp, quant} tem o esquema: **Item_F** {peça, quant_disp, quant}, e é definido pela consulta, de acordo com a Figura 3.2.

```
SELECT  P.Peça, SUM(PS.quant_disp), SUM(IT.quant)
FROM    ITEM AS IT, Peça AS P, FORNECEDOR_PEÇA AS FP
WHERE   IT.Peça = P.Peça
AND     PS.Peça = P.Peça
GROUP BY P.Peça
```

Figura 3.2 – Consulta Exemplo
Adaptado de (GOLFARELLI; MANIEZZO; RIZZI, 2004)

3.1.4. PROBLEMA DA FRAGMENTAÇÃO VERTICAL (PFV)

Dada uma *workload* e as medidas de cada esquema de cubos em $\mathcal{D}$, pode-se particionar, em subconjuntos (minitermos), todas as medidas em um minitermo, podendo ser solicitadas juntas por pelo menos uma consulta, porém não poderão aparecer de forma isolada em nenhuma consulta. Os autores determinam que termos são conjuntos de medidas de diferentes esquemas de cubos recursivamente definidos, onde:

- Minitermo é um termo;

- A união de dois termos, incluindo as medidas solicitadas para a mesma consulta, também é um termo.

Exemplo: Para as consultas ***q1***,***q2*** e ***q3*** dadas na Figura 3.3 têm-se os minitermos mt_1, mt_2, mt_3, mt_4 e o termo ***T***:

- $mt_1 =$ {preço_desc, debito, taxa}
- $mt_2 =$ {preço_unit, preço_est, desconto}
- $mt_3 =$ {quant_disp}
- $mt_4 =$ {quant}
- $T =$ { $mt_1, mt_2, mt_3, mt_4, mt_2 \cup mt_4, mt_3 \cup mt_4$, ***Meas(IT), Meas(FP)*** }

Meas(q1) = {preço_desc, debito, taxa} e ***Gby(q1)*** = {marca, fornecedor}

Meas(q2) = {preço_unit, quant, preço_est, desconto} e ***Gby(q2)*** = {peça, fornecedor, data_envio}

Meas(q3) = {quant, quant_disp} e ***Gby(q3)*** = {peça, fornecedor}.

Figura 3.3 – Exemplos de Consultas (q1,q2 e q3)
Adaptado de (GOLFARELLI; MANIEZZO; RIZZI, 2004)

O problema da fragmentação vertical (***PFV***) pode ser modelado por expressão linear, sobre dois conjuntos de variáveis binárias 0 -1, $\{x_{ijk}\}$ e $\{y_{jk}\}$, onde os índices de ***X*** e ***Y*** correspondem aos índices das consultas $q_i \in Q$, do conjunto agrupado $g_j \in G$, e dos termos $t_k \in T$, respectivamente. O valor de $\{y_{jk}\} = 1$ significa que o fragmento será criado, fazendo uso da visão candidata ***j*** e a medida do termo k. Então, será criado um fragmento $\mathscr{F}_{jk}$. Quando o valor de $\{x_{ijk}\} = 1$ significa que a consulta ***i*** será respondida utilizando o fragmento $\mathscr{F}_{jk}$. Em outras palavras, a matriz de fragmentação bidimensional denota a fragmentação e a matriz tridimensional indica em qual fragmento uma consulta deve ser respondida.

Dentre as matrizes de fragmentação obtidas, encontra-se, entre as fragmentações possíveis, uma fragmentação que otimiza o tempo de resposta das consultas, obtidas por meio de uma função de custo, a ser definida na próxima

seção. Na Figura 3.4 segue um exemplo considerando as consultas da Figura 4.3, os termos T, as visões candidatas de (IT) e de (FP).

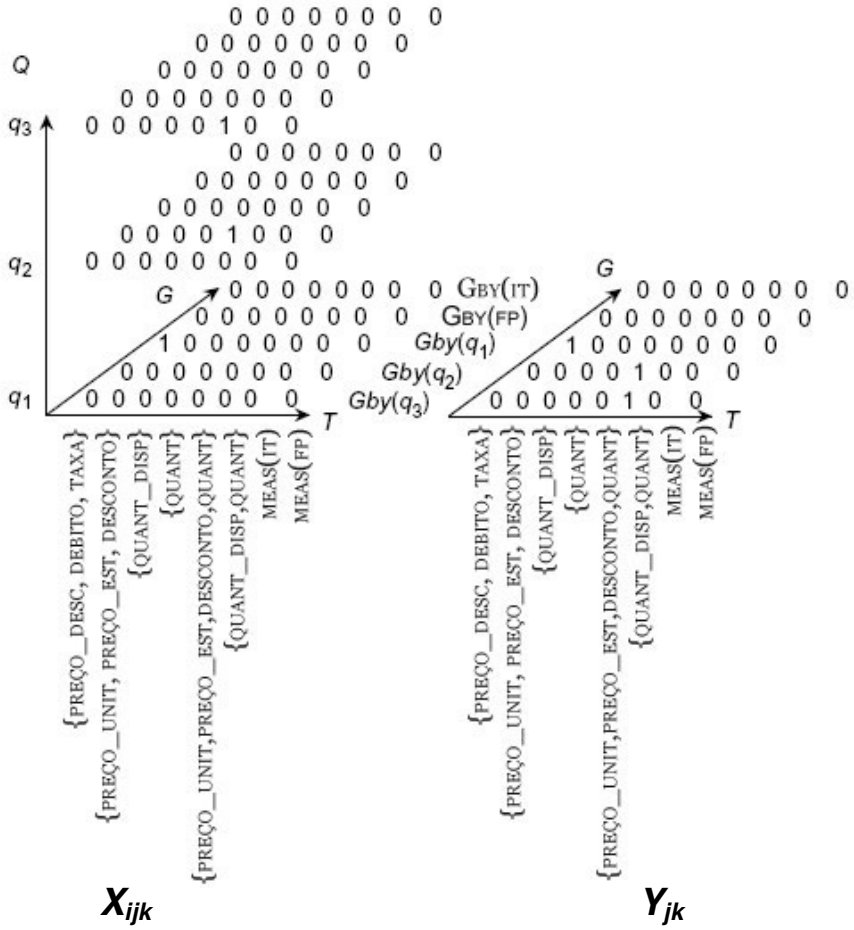


Figura 3.4 – Matriz (*q1,q2 e q3*)
Adaptado de (GOLFARELLI; MANIEZZO; RIZZI, 2004)

3.1.5. A FUNÇÃO DE CUSTO

Golfarelli, Maniezzo e Rizzi (2004) propõem uma função de custo baseada no número de páginas de disco onde estão armazenadas as tuplas de interesse para uma dada consulta.

Seja q_i uma consulta que requer pelo menos uma medida no fragmento $\mathcal{F}_{jk}$, definido por Gby($\mathcal{F}_{jk}$) = g_j e Meas($\mathcal{F}_{jk}$) = t_k. Seja Freq(q_i) frequência de q_i .O número de tuplas de $\mathcal{F}_{jk}$ que devem ser acessadas, a fim de atender q_i, é:

*Sel(**q**$_i$)* * *Card*($\mathcal{F}_{jk}$), onde *Card*($\mathcal{F}_{jk}$) é a cardinalidade de $\mathcal{F}_{jk}$. O número total de páginas, no qual $\mathcal{F}_{jk}$ é contido, é representado pela equação 24:

$$\left\lceil \frac{b_{jk} = \text{Card}(\mathcal{F}_{\text{jk}})}{\beta_{\text{jk}}} \right\rceil \qquad (24)$$

Equação 24 – Número Total de Páginas (GOLFARELLI; MANIEZZO; RIZZI, 2004)

Onde $\boldsymbol{\beta_{jk}}$ é o número de tuplas por página de disco para $\mathcal{F}_{jk}$. O número de páginas esperado de $\mathcal{F}_{jk}$ ($\boldsymbol{c_{ijk}}$) para atender à consulta qi, ponderado pela frequência, pode ser estimado de acordo com a equação 25:

$$\text{c}_{\text{i jk}} = \text{Freq}(\text{q}_{\text{i}}) * \Phi(\text{Sel}(\text{q}_{\text{i}}) * \text{Card}(\text{F}_{\text{jk}}), \text{b}_{\text{jk}}) \qquad (25)$$

Equação 25 – Número de Páginas de Fjk (cijk)
(GOLFARELLI; MANIEZZO; RIZZI, 2004)

O custo de execução da consulta ***i*** na fragmentação denotada por $\boldsymbol{X}$= {$\boldsymbol{X_{ijk}}$}, usando a notação de conjunto de índices da Tabela 3, é estimado como o número total de páginas de disco, as quais deverão ser acessadas a fim de recuperar todas as medidas em $\boldsymbol{q_i}$, veja a equação 26:

$$Cost_i(x) = \sum_{j \in \mathcal{G}_i} \sum_{k \in Q \mathcal{F}_{ijk}} c_{ijk} X_{ijk} \qquad (26)$$

Equação 26 - O custo de execução da consulta *i*
(GOLFARELLI; MANIEZZO; RIZZI, 2004)

Onde $\boldsymbol{X_{ijk}}$=1, se as medidas em Meas($\boldsymbol{q_i}$) $\cap$ $\boldsymbol{t_k}$ são lidas a partir de $\boldsymbol{F_{jk}}$, e $\boldsymbol{X_{ijk}}$= 0 em caso contrário. Finalmente, o custo total para a *workload* **Q** é representado pela equação 27:

$$Cost(x) = \sum_{i \in \mathcal{Q}} Cost_i(x) = \sum_{i \in \mathcal{Q}} \sum_{j \in \mathcal{G}_i} \sum_{k \in Q \mathcal{F}_{ij}} c_{ijk} Xi_{jk} \qquad (27)$$

Equação 27 – Custo total para a workload **Q**
(GOLFARELLI; MANIEZZO; RIZZI, 2004)

Tabela 3 – Notação para o Conjunto de Índices para a Formulação de *PFV*

$\mathcal{Q}$	Conjunto de índices das consultas em Q
$\mathcal{G}$	Conjunto de índices do conjunto agrupado em G
$\mathcal{T}$	Conjunto de índices do termo em T
$\mathcal{M}$	Conjunto de índices das medidas em $\bigcup_{\mathcal{C} \in \mathcal{D}}$
$\mathcal{G}_i \subseteq \mathcal{G}$	Conjunto de índices do conjunto agrupado útil para uma consulta $i \in \mathcal{Q}$
$\mathcal{M}i \subseteq \mathcal{M}$	Conjunto de índices das medidas em Meas($\boldsymbol{q}_i$)
$\mathcal{T}_i \subseteq \mathcal{T}$	Conjunto de índices dos termos no conjunto agrupado $j \in \mathcal{G}$
${}_Q\mathcal{T}_{ij} \subseteq \mathcal{T}$	Conjunto de índices de termos em uso por uma consulta $i \in \mathcal{Q}$ no conjunto agrupado $j \in \mathcal{G}_i$
${}_M\mathcal{T}_{js} \subseteq \mathcal{T}$	Conjunto de índice de termos em uso para medidas $s \in \mathcal{M}$ no conjunto agrupado $j \in \mathcal{G}$

(GOLFARELLI; MANIEZZO; RIZZI, 2004)

3.2. WanD: Uma Ferramenta Case para Projeto de Data Mart

Golfarelli, Rizzi e Saltarelli (2002) propõem o protótipo *WanD* (*Warehouse Integrated Designer*) para dar suporte à metodologia para fragmentação vertical, aplicada à materialização de visões. Esse protótipo auxilia o projetista a desenvolver o projeto de um *data mart*.

Inicialmente, a ferramenta carrega o modelo conceitual, de forma semi-automática, por meio de uma conexão ODBC, estabelecida entre a *WanD* e um esquema de dados de um banco operacional. Carregado o modelo conceitual é a definida uma *workload* com o conjunto de consultas formuladas pelo projetista.

Em seguida obtém-se o volume de dados, o que compreende na estimativa de cardinalidade das visões agregadas obtidas do modelo conceitual, necessário para a aplicação correta da técnica de materialização de visões. A partir deste ponto define-se o projeto lógico, onde o mesmo é transportado graficamente para tabelas relacionais, de acordo com o esquema estrela.

A ferramenta determina o melhor conjunto de agregações a ser materializado, levando em consideração o esquema fato, uma *workload* e seu

volume de dados, em função da minimização do tempo de resposta das consultas. Os dados são materializados em fragmentos verticais, armazenados em subconjuntos de medidas no esquema fato.

A etapa da *WanD* que compreende o projeto físico ocorre a partir de um esquema lógico incluindo os fragmentos materializados, uma *workload*, um volume de dados e a definição das restrições de espaço em disco para indexação. Então, é definido o conjunto de índices para minimizar o custo de execução da *workload* em função do espaço em disco. A política de materialização baseia-se também em regras para otimização, capaz de determinar um plano de execução para cada consulta e um modelo de custo para a comparação de diferentes planos.

E, por fim, a *WanD* produz a documentação do projeto de *data mart* incluindo: o projeto do esquema fato, o glossário dos atributos e medidas, a *workload*, o volume de dados, e as instruções *SQL* para criar e popular o *data mart*. As demonstrações do protótipo *WanD*, apresentadas na próxima seção, utilizam a base de dados sintética *benchmark TPC-H.* Tais demonstrações fornecidas pelos autores, estão de acordo com a metodologia desenvolvida e detalhada nas seções anteriores deste capítulo.

3.2.1. ARQUITETURA E DEMONSTRAÇÃO DA WAND

A arquitetura funcional do protótipo da *WanD*, apresentada na Figura 3.5, foi desenvolvida para contemplar as funcionalidades de aquisição do esquema relacional, que descreve o banco de dados operacional; de obtenção do modelo conceitual; de definição da *workload*; de aquisição do volume de dados; da obtenção do modelo lógico; de obtenção do modelo físico; e da produção de documentação.

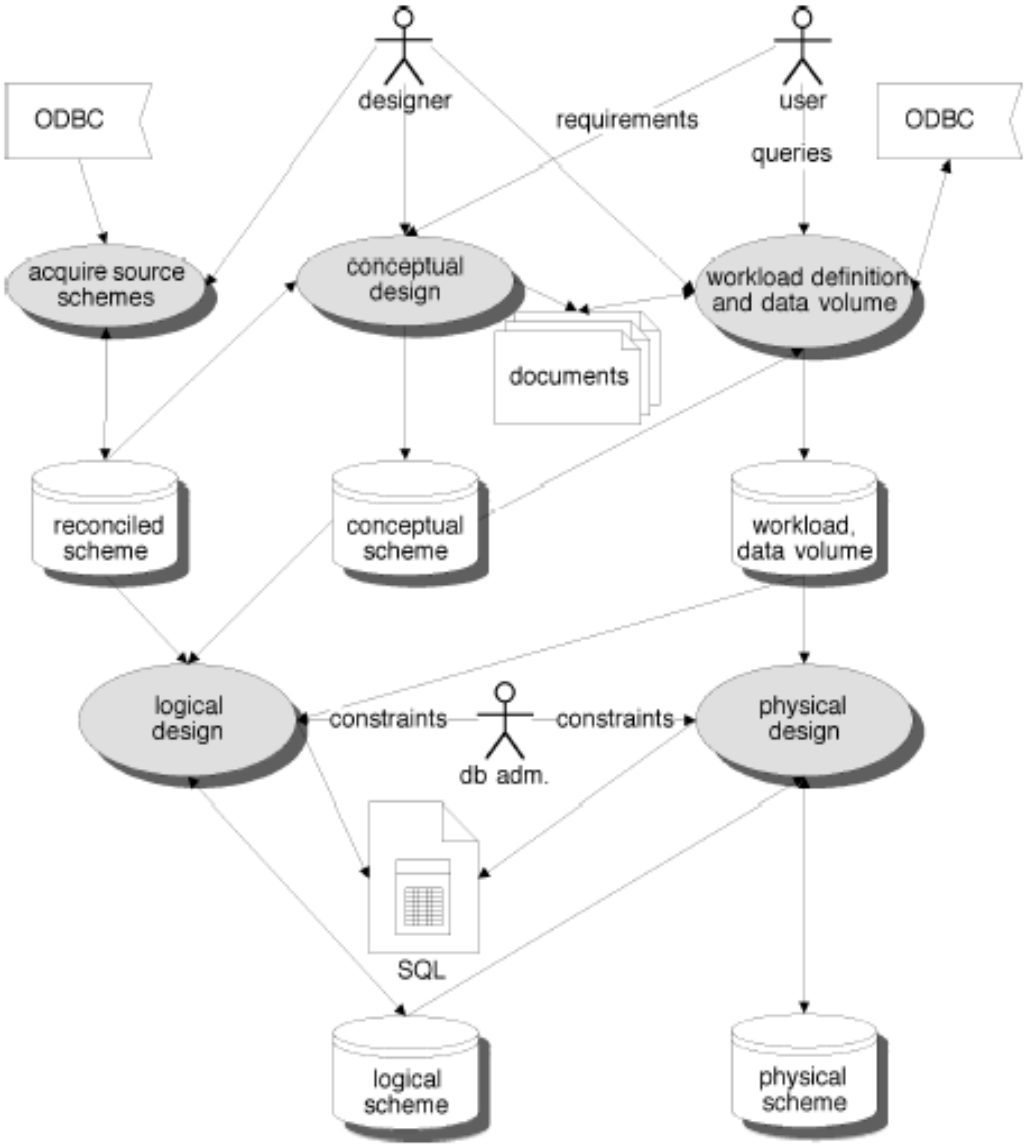


Figura 3.5 – Arquitetura da *WanD* (GOLFARELLI; RIZZI; SALTARELLI, 2002)

A interação do projetista, assim como suas funcionalidades, são apresentadas nas principais telas do protótipo a seguir. Na Figura 3.6 é mostrado, como saída, o conjunto de esquemas fato, representado por um esquema para cada fato de interesse. A notação usada para representar os esquemas fato é apresentada em (GOLFARELLI; MAIO; RIZZI, 1998).

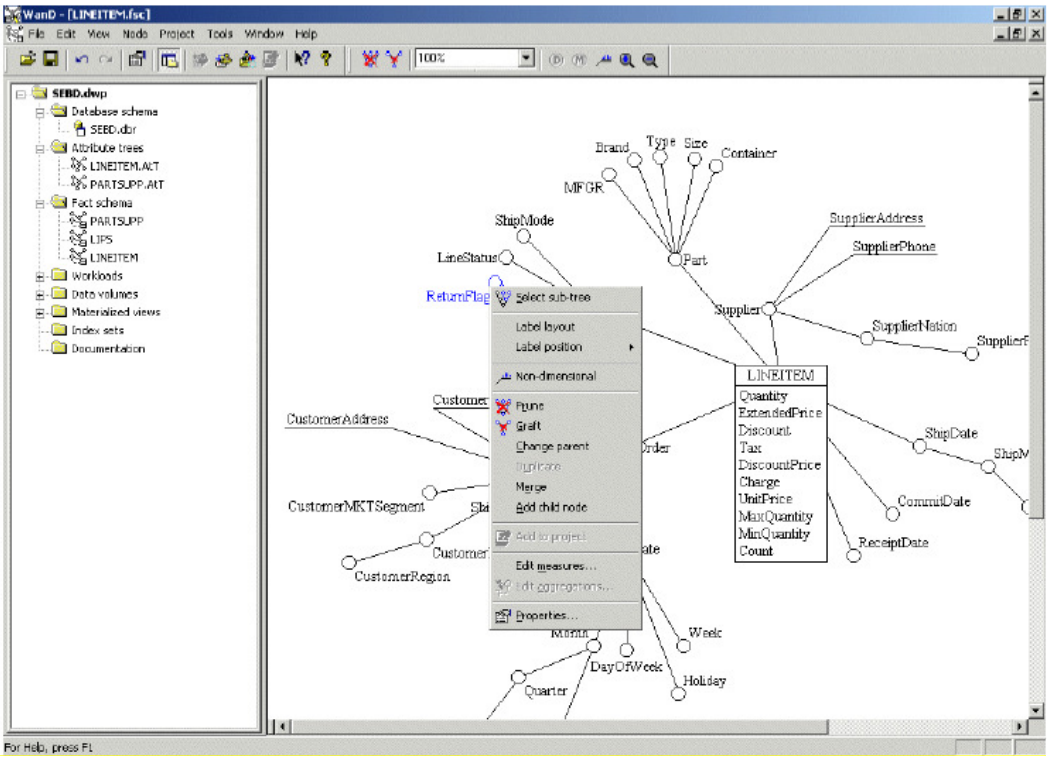


Figura 3.6 – Conjunto de Esquemas Fato (GOLFARELLI; RIZZI; SALTARELLI, 2002)

A Figura 3.7 exibe a aquisição de uma *workload* e a extração do volume de dados a partir de uma conexão *ODBC* no banco de dados.

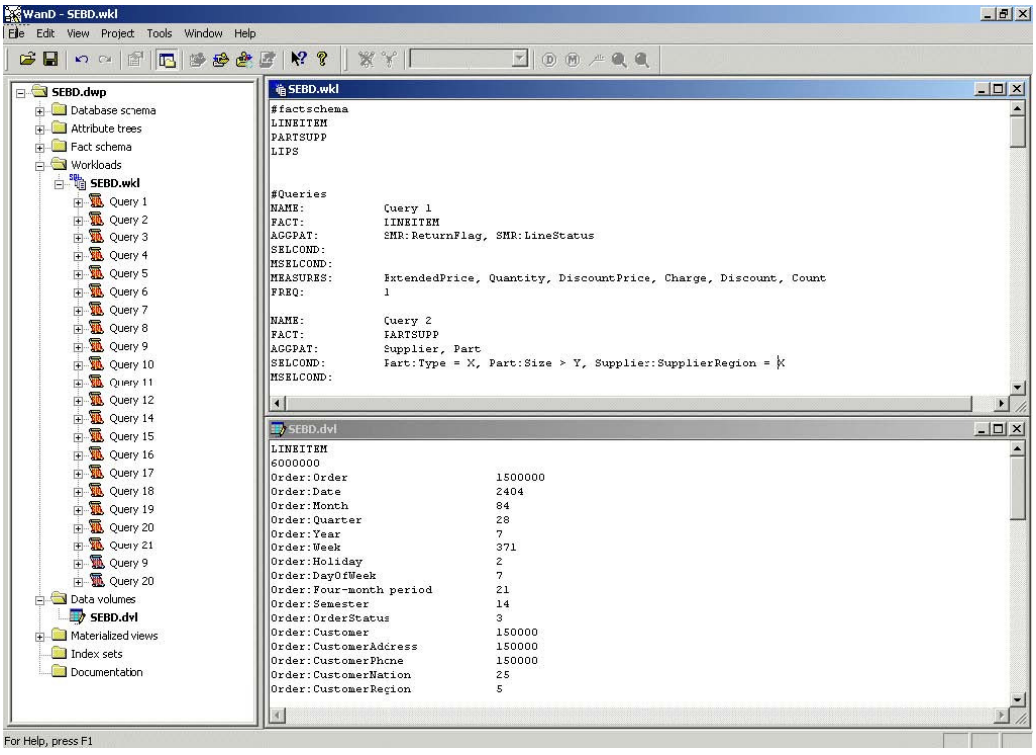


Figura 3.7 – Adquirindo um *workload*/ Volume de dados
(GOLFARELLI; RIZZI; SALTARELLI, 2002)

A próxima tela, exibida na Figura 3.8, apresenta o script *SQL* para gerar e popular o *data mart*.

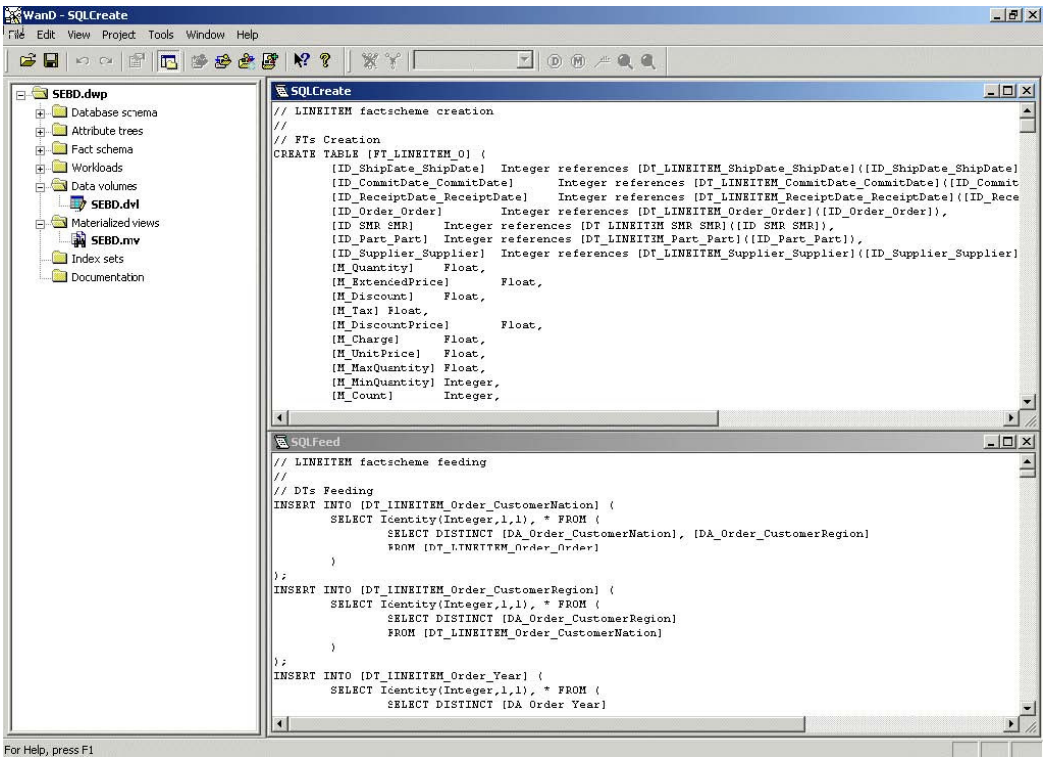


Figura 3.8 – SQL para Criar e Popular o *Data Mart*
(GOLFARELLI; RIZZI; SALTARELLI, 2002)

3.3. CONSIDERAÇÕES FINAIS

Neste capítulo foram apresentados conceitos de materialização de visões e fragmentação vertical para a minimização de custos de processamento de consultas no *DW*, segundo a abordagem de Golfarelli, Maniezzo e Rizzi (2004). Foi apresentada também, a aplicação da metodologia desenvolvida, por meio do protótipo da ferramenta *WanD*, de acordo com Golfarelli, Rizzi e Saltarelli (2002).

O próximo capítulo apresenta conceitos sobre modelagem dimensional, sobre a construção do modelo de dados, e uma visão geral da Ferramenta para Geração do Modelo Dimensional para *Data Warehouse*, proposta por Lima (2006).

4. FERRAMENTA PARA GERAÇÃO DO MODELO DIMENSIONAL PARA DW

Este capítulo apresenta a ferramenta para Geração do Modelo Dimensional para *Data Warehouse* (LIMA, 2006), em que se aplica a validação dos conceitos apresentados no decorrer deste trabalho. O principal objetivo dessa ferramenta é auxiliar o projetista na tarefa de construção do modelo de dados para *DW*, cujos propósitos gerais são: minimizar as dificuldades e os erros que podem ocorrer na concepção do modelo, decorrentes da inabilidade na construção da modelagem dimensional e na falta de conhecimento da metodologia de modelagem a ser adotada pelo projetista.

No decorrer deste capítulo são apresentados conceitos de modelagem dimensional, de construção do modelo de dados, e uma visão geral da ferramenta sob os seguintes aspectos: metodologia utilizada, os módulos, as principais telas e os recursos que a ferramenta disponibiliza ao projetista.

4.1. CONCEITOS BÁSICOS DA MODELAGEM DIMENSIONAL

Segundo Kimball (1998) e Machado (2000), apud Lima (2006, pag.9), a modelagem dimensional consiste em uma técnica do projeto lógico, de um banco de dados, em que os dados são apresentados de forma padronizada, possibilitando acessos com bom desempenho. A técnica é composta por três elementos básicos: fatos, dimensões e medidas (variáveis).

De acordo com Kimball (2002), a tabela de fatos é a tabela em que as medições numéricas de desempenho da empresa são armazenadas, onde o termo “fato” representa a medição de um negócio.

Em uma tabela de fatos, uma linha corresponde a uma medição. Uma medição é uma linha na tabela de fatos. Todas as medições em uma tabela de fatos devem estar alinhadas a uma mesma granularidade (KIMBALL, 2002).

As tabelas de dimensão são pontos de entrada para a tabela de fatos, cujos atributos são eficazes para produzir recursos de slicing e dicing, os quais

compreendem, respectivamente, “separação e combinação”. Desse modo, representam a interface do usuário com o *Data Warehouse*. As dimensões contêm descritores textuais de acordo com o contexto da empresa. Os atributos das tabelas de dimensão desempenham um papel fundamental no *DW*, pois são a origem de praticamente todas as restrições e rótulos de relatórios interessantes. Esses atributos são fundamentais para o entendimento e uso do *Data Warehouse*. Um atributo de dimensão pode ser descrito de forma resumida ou até mesmo por uma descrição mais longa (que contenha de 30 a 50 caracteres) (KIMBALL, 2002).

Segundo Kimball (1998), o modelo estrela é composto por uma tabela de fatos, que é a entidade central, e um conjunto de dimensões posicionadas ao seu redor, formando uma estrela, conforme ilustrado na Figura 4.1.

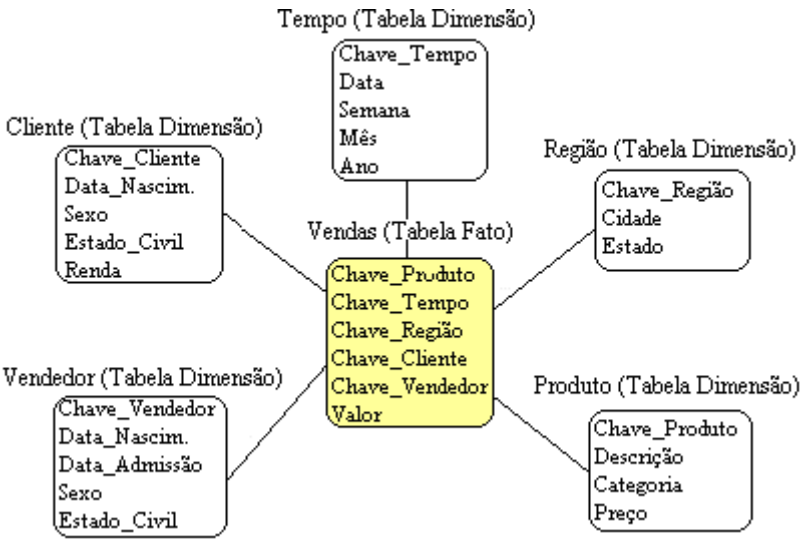


Figura 4.1 – Modelo Estrela Adaptado de (LIMA, 2006)

As vantagens assim como as desvantagens do uso do modelo estrela são relacionadas abaixo:

- Vantagens: O tempo de resposta é rápido porque a desnormalização reduz a quantidade de junções que uma consulta precisa processar; o banco de dados é projetado da mesma forma a que o usuário está acostumado a pensar e a usar os dados; simplifica o entendimento e a navegação dos metadados pelos desenvolvedores e usuários finais; possibilita o uso de diversas ferramentas para acesso aos dados, pois vários produtos existentes no mercado exigem o modelo estrela; é flexível, ou seja, permite a inclusão de novos elementos de dados e

alterações que ocorrem no projeto, sem implicar em alterações nas ferramentas de consultas e aplicações (SINGH, 1998).

- Desvantagens: Devido à desnormalização, pode ocupar muito espaço em disco; a existência de dimensões muito grandes pode afetar o desempenho (SINGH, 1998).

4.2. CONSTRUÇÃO DO MODELO DE DADOS

Segundo Lima (2006), há várias abordagens a serem consideradas para a construção do modelo de dados para *Data Warehouse*, sendo que cada uma dessas abordagens apresenta uma metodologia para a construção do modelo. Assim, independente da abordagem que o projetista do *DW* possa adotar, o mesmo deve ter conhecimento das etapas ou fases da metodologia apresentada pela abordagem, para que seja gerado o modelo de dados. Dessa forma, Lima (2006) dá ênfase para as abordagens de Ralph Kimball e de Bill Inmon, as quais são brevemente comentadas a seguir.

Kimball (1998) sugere quatro processos para a construção do modelo:

- Escolha do processo de negócio a ser modelado;
- Escolha da granularidade (nível de detalhe) do processo de negócio;
- Escolha das dimensões;
- Escolha dos fatos mensuráveis (medidas).

Inmon (1997) sugere a construção de um modelo de dados intermediário, através do modelo de dados corporativo da empresa, e a transformação deste no modelo de dados do *DW*. Para a construção desse modelo intermediário, devem ser realizadas as seguintes atividades:

- Remover os atributos puramente operacionais;
- Adicionar um elemento de tempo à estrutura de chaves do *DW*;
- Adicionar dados derivados;
- Criar artefatos de relacionamentos;

- Ajustar os diferentes níveis de granularidade;
- Agrupar dados comuns que estão em diferentes tabelas;
- Criar *arrays* de dados;
- Separar os atributos de acordo com a frequência de atualização.

Gallas (1999) e Jukic (2006) fazem algumas considerações sobre as abordagens de Kimball e Inmon. Jukic (2006) apresenta uma visão concisa e imparcial sobre metodologias que podem ser utilizadas na construção de um projeto de modelagem de *Data Warehouse*. O autor estabelece uma comparação entre as abordagens de *Inmon* e *Kimball*, as quais têm sido tópico de debate durante anos.

De acordo com Jukic (2006), a abordagem de Kimball é a maneira mais simples e rápida de se criar um *DW*, se as estruturas dimensionais de toda a organização puderem atender à análise dos requisitos necessários. Entretanto, se for necessário adicionar outras análises de dados analíticos, às análises já armazenadas à estrutura dimensional, então a abordagem de Inmon, prevalece como o método mais eficiente. Jukic (2006) menciona que é muito comum ocorrer comparativos mencionando desvantagens de Inmon ou Kimball. Sobre a abordagem de Inmon é dito que a fase de modelagem requer um alto nível de especialização e uma considerável antecipação ao compromisso. Por outro lado, a crítica mais comum sobre Kimball é a falta de foco nas dificuldades, concentrando-se primeiramente nos negócios individuais, usuários ou grupos de usuários. Na abordagem de Inmon é dito que se gasta mais tempo na modelagem, devido ao fato de que se pode utilizar diversas maneiras para criar o modelo.

Os fatores prioritários a serem considerados, independente da abordagem, são: levar em conta que um projeto de *DW* envolve um considerável investimento; o processo *ETL (Extração Transformação e Carga)* é um fator que consome tempo; o sucesso do projeto, na maioria das vezes, depende da fase de modelagem (JUKIC, 2006).

Baseado no estudo das abordagens de Kimball, Inmon entre outras, Lima (2006) propõe uma abordagem centrada nos requisitos de análise, levando o projetista a definir os dados necessários para atender cada requisito, por meio do banco de dados operacional, que é base para o desenvolvimento do *DW*. Tal

abordagem é viabilizada através da ferramenta para Geração do Modelo Dimensional para *DW* apresentada a seguir.

4.3. METODOLOGIA PARA GERAÇÃO DO MODELO DIMENSIONAL DA FERRAMENTA

A metodologia desenvolvida para a geração do modelo dimensional utiliza um assistente, de acordo com as funcionalidades divididas em módulos, com funções bem definidas. A proposta de Lima (2006) separa a ferramenta em três módulos: Geração de Dicionário de Metadados de Bancos de Dados Operacionais, Cadastro de Expressões e Modelagem do *Data Warehouse*. A arquitetura dos módulos e a maneira que os mesmos funcionam são apresentadas na Figura 4.2.

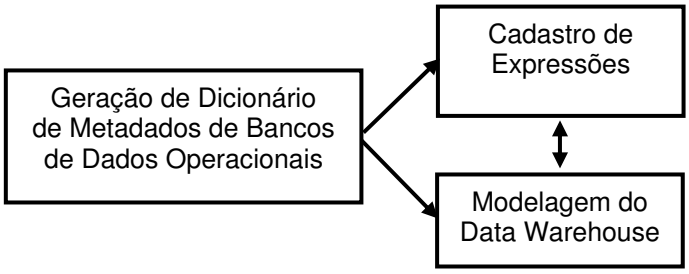


Figura 4.2 - Módulos da Ferramenta (LIMA, 2006)

A metodologia adotada pela ferramenta considera o esquema estrela para a apresentação do modelo gerado para o *DW*. A ideia principal do processo de modelagem para o *DW*, segundo os conceitos implementados na ferramenta, faz menção: ao conhecimento do negócio e às informações inseridas na ferramenta através dos requisitos de usuário. Essas informações são baseadas em um assunto específico o qual se deseja projetar. A ferramenta trabalha com apenas uma base de dados por vez. O sistema gerenciador de banco de dados – (SGBD), disponível para uso, trata-se do *Firebird*. A ferramenta, em sua concepção, não teve por objetivo o uso de múltiplas bases de dados ou, até mesmo, trabalhar com bases de dados distribuídas ou paralelas.

Um assistente é utilizado para guiar o projetista na geração do modelo dimensional, com o intuito de facilitar a compreensão das informações do usuário, ou seja, os requisitos, e também proporcionar que o processo de

modelagem seja seguido corretamente. O assistente conduz o processo de geração do modelo, de modo que seja inserido um requisito por vez.

O modelo dimensional gerado é exibido em forma de relatório. De posse desse relatório, o projetista implementa o *Data Warehouse* no SGBD que melhor atende à necessidade do cliente.

4.4. DESCRIÇÃO DOS MÓDULOS DA FERRAMENTA

O módulo de Geração de Dicionário de Metadados de Banco de Dados Operacionais é o responsável por conectar no banco operacional e retirar dele as informações de estrutura, tais como: nome das tabelas, atributos das colunas, chaves primárias, chaves estrangeiras e restrições. A ferramenta manipula os metadados de acordo com a estrutura dos dados fonte, isto é, dados operacionais, e os armazena como metadados de *DW*, após a geração do modelo dimensional.

O processo de conexão com o banco de dados operacional necessita de atributos obrigatórios, como o caminho que se encontra o arquivo pelo qual se deseja extrair as informações, o usuário, e a senha de acesso e também qual o SGBD a ser usado. No protótipo da ferramenta desenvolvida é possível, somente, a conexão com o SGBD *Firebird* até a versão 1.5 ou a utilização de arquivos do SGBD *Interbase*. Na Figura 4.3 é mostrada a visualização da tela de configurações de cadastro dos bancos de dados.

O módulo de Cadastro de Expressões é responsável por definir funções ou operadores e associá-los a uma coluna específica de uma tabela, escolhida para compor o modelo de dados dimensional, durante o processo no assistente. Podem ser cadastradas expressões e, a cada uma delas, atribuída uma descrição para identificação e uma abreviação correspondente. Na Figura 4.4 é mostrada a tela do cadastro de expressões.

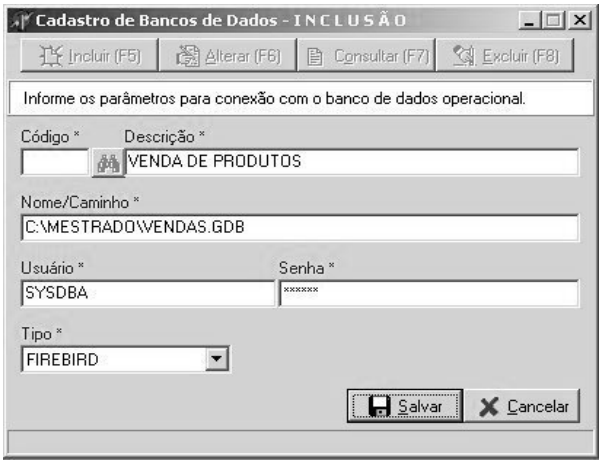


Figura 4.3 – Módulo de Cadastro dos B.Ds Operacionais (LIMA, 2006)

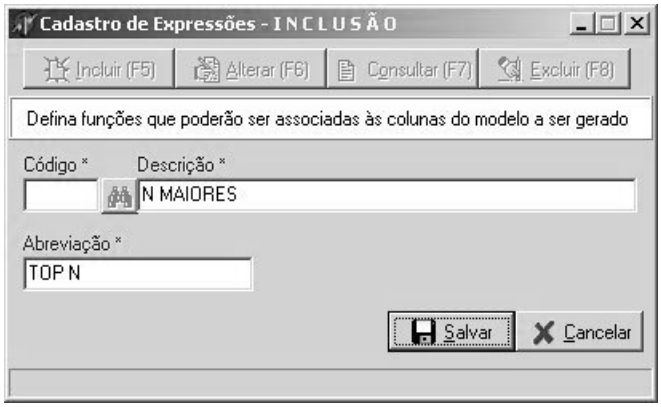


Figura 4.4 – Módulo de Cadastro de Expressões (LIMA, 2006)

O módulo de Modelagem do *Data Warehouse* tem, por objetivo, conduzir a construção do modelo de dados dimensional, fazendo uso de um assistente, o qual orienta o projetista de forma gradativa, de acordo com os passos a serem seguidos, conforme a metodologia adotada para a ferramenta. Para que esse processo possa ser realizado, é necessário ter em mãos os requisitos de negócio, os assuntos definidos e o cadastro das expressões. Os fatores temporais responsáveis pela dimensão “Tempo” do modelo dimensional podem ser cadastrados no assistente, durante a inserção dos requisitos de negócio. Em razão do assistente trabalhar com um requisito por vez, as chances de erro no modelo diminuem. Durante cada inserção de um requisito de negócio, as tabelas, as colunas e expressões envolvidas são escolhidas para atender o requisito em questão. Na Figura 4.5 é mostrada a representação do processo funcional do módulo de Modelagem de *Data Warehouse*.

Os três primeiros processos estão agrupados em etapas sequenciais que são executadas pelo assistente. O quarto processo é uma etapa independente que o projetista executa após ter tratado todos os requisitos.

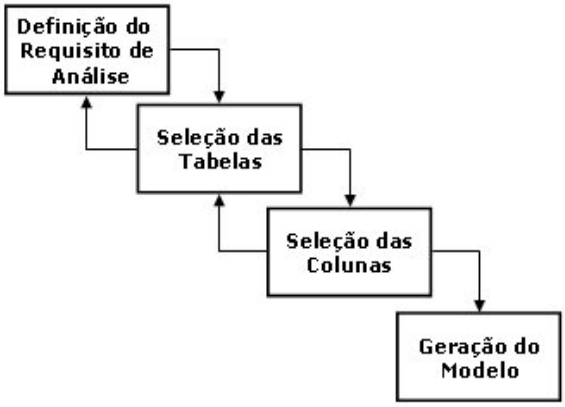


Figura 4.5 – Processo Funcional do Módulo de Modelagem (LIMA, 2006)

A fim de facilitar a visualização do funcionamento do processo funcional do módulo de modelagem, conforme a metodologia de Lima (2006), o algoritmo para geração do modelo dimensional é descrito de acordo com a Figura 4.6.

Algoritmo: Geração do Modelo Dimensional
Entrada: o assunto de análise.
Saída: modelo dimensional.

Algoritmo {Geração do Modelo Dimensional}
1. Ler o assunto de análise
2. Identificar a tabela de fatos e as dimensões
3. Identificar as colunas da tabela de fatos e das dimensões
4. Criar a dimensão tempo
5. Definir a chave primária para as dimensões identificadas
6. Definir as chaves estrangeiras na tabela de fatos
7. Salvar o modelo gerado no banco de dados
8. Exibir o modelo gerado
fim algoritmo

Figura 4.6 – Algoritmo para Geração do Modelo Dimensional (LIMA, 2006)

4.5. ARMAZENAMENTO DOS DADOS NA FERRAMENTA

Os dados obtidos a partir da execução de cada módulo são armazenados na estrutura do banco de dados da ferramenta, conforme a Figura 4.7.

Em um primeiro momento, a ferramenta trabalha com dados de bancos operacionais, ou seja, metadados obtidos através da conexão no banco de

dados do cliente. O projetista utiliza esses metadados durante os passos realizados pelo assistente da ferramenta, para a geração do modelo dimensional, levando em consideração os requisitos de negócio. Além dos metadados, são armazenados os dados em operação, pelo assistente e, em seguida, os dados resultantes, ou seja, o modelo dimensional é armazenado como metadados de *DW*.

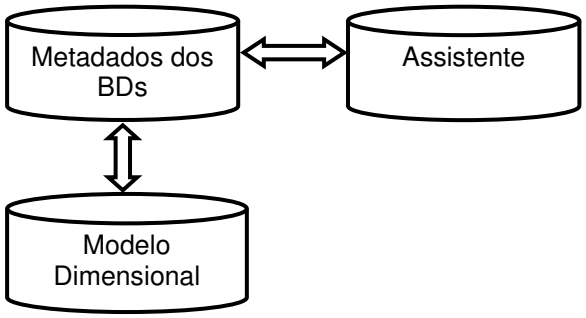


Figura 4.7 - Armazenamento dos dados na Ferramenta (LIMA; VIEIRA, 2006)

4.6. Demonstração do Uso da Ferramenta para Geração do Modelo Dimensional

Como primeiro passo para uso da ferramenta, deve-se cadastrar um ou mais bancos de dados operacionais do cliente, através da interface de cadastro de banco de dados operacionais da ferramenta.

A interface de cadastro da ferramenta permite trabalhar com os gerenciadores de bancos de dados *Firebird* ou *Interbase*, conforme dito anteriormente. Cadastram-se os dados necessários para a conexão ao banco operacional e em seguida o dicionário de metadados pode ser gerado para iniciar o processo através do assistente. O tradicional exemplo de vendas será utilizado para demonstrar o uso do assistente. O modelo de entidade e relacionamento, assim como os requisitos que satisfazem esse modelo de negócio são mostrados na Figura 4.8.

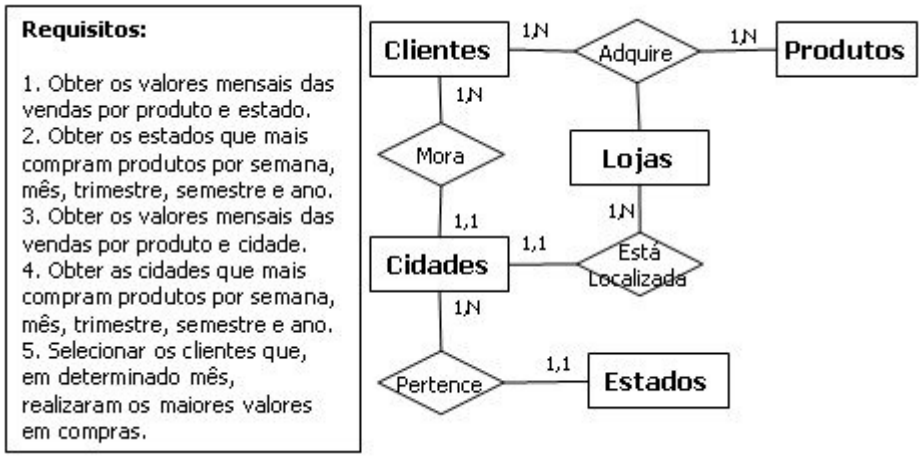


Figura 4.8 – Modelo de Entidade e Relacionamento de Vendas (LIMA, 2006)

A definição do requisito, do assunto e a escolha do fator de tempo fazem parte da primeira etapa das três a serem cumpridas de acordo com a Figura 4.9. Durante a etapa 1, pode ser adicionado um novo assunto ou um novo fator de tempo.

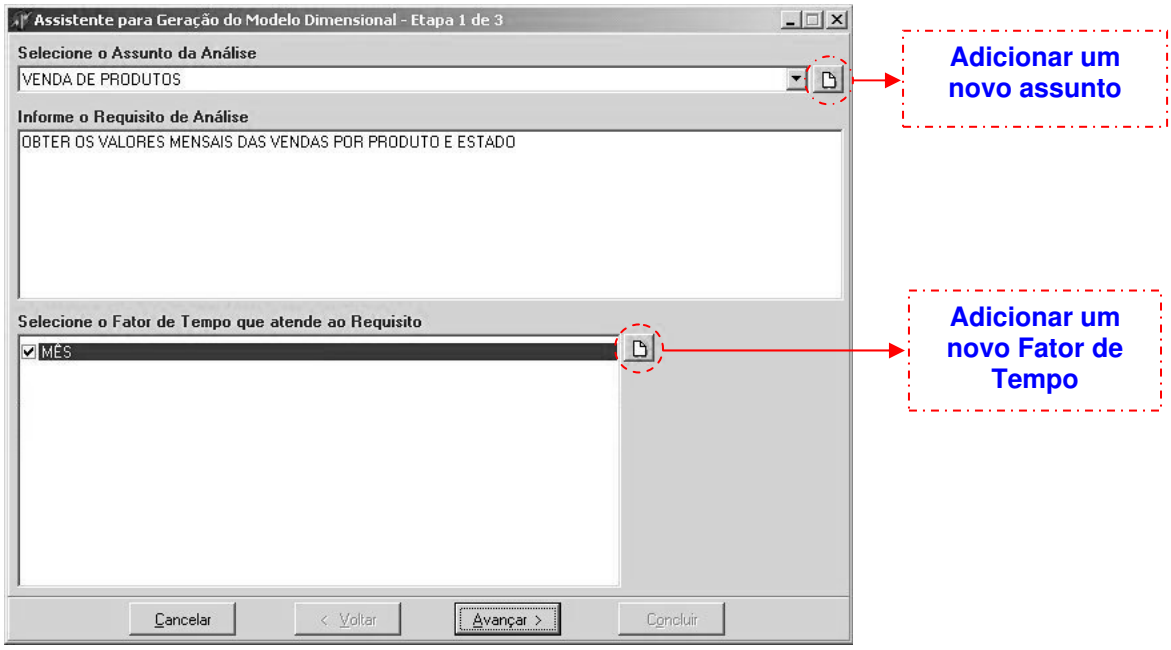


Figura 4.9 – Etapa 1 de 3 para geração do Modelo Dimensional (LIMA, 2006)

Dando sequência ao exemplo de vendas, a segunda etapa consiste na escolha das tabelas para atender ao requisito. Como a tabela Estado não possui relacionamento com a tabela ItensVendidos, ou seja, não possui nenhuma coluna na tabela ItensVendidos que identifique o estado, é necessário identificar através de

quais tabelas os valores das vendas serão obtidos para atender a esse requisito. A tabela Estado possui relacionamento com as tabelas Clientes e Lojas; o projetista deve informar se os valores das vendas são obtidos de acordo com o estado do cliente ou o estado da loja que vendeu o produto. Veja a Figura 4.10.

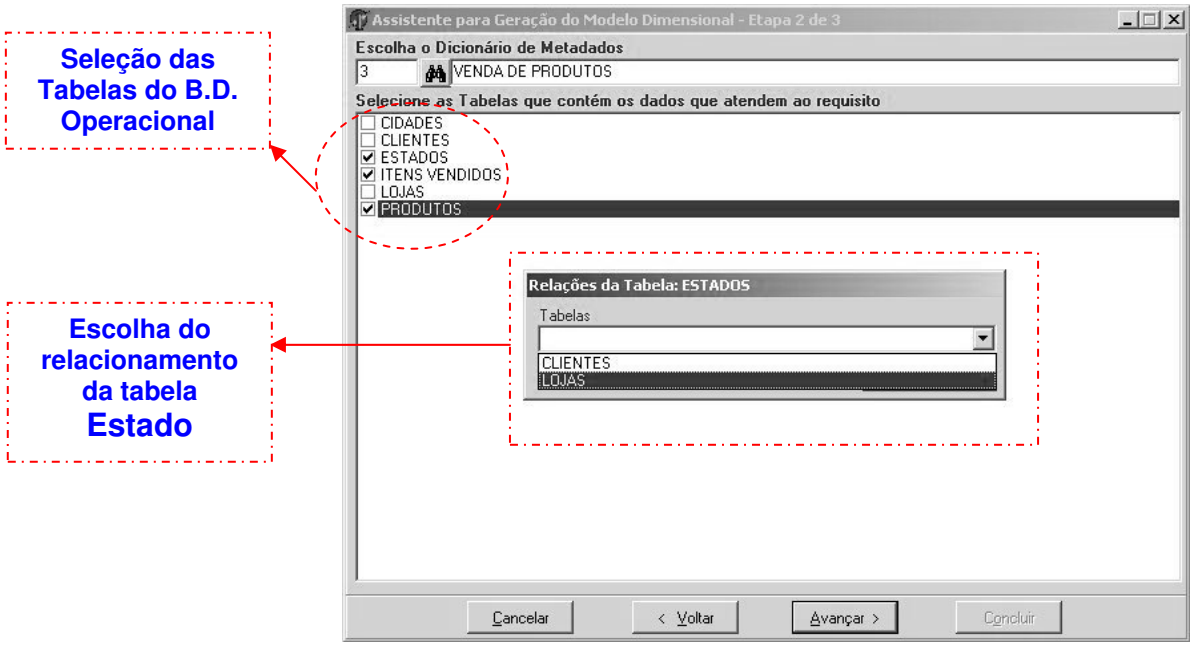


Figura 4.10 – Etapa 2 de 3 para geração do Modelo Dimensional (LIMA, 2006)

Como terceiro passo devem ser escolhidas as colunas necessárias para atender o requisito de análise. As colunas são originadas do Dicionário de Metadados através das tabelas selecionadas pelo projetista, na etapa anterior. Nesse momento, além de escolher as colunas do Dicionário de Metadados, o projetista também pode utilizar as funções cadastradas no módulo de expressões. Para visualizar a tela que apresenta tal situação, veja o exemplo exposto na Figura 4.11.

Os três passos apresentados pelo assistente são necessários para que cada requisito seja adicionado ao modelo dimensional e devem ser repetidos até que tenham sido satisfeitas as necessidades. Dessa maneira, é encerrada a fase de tratamento dos requisitos e faz-se necessário executar o módulo de geração do modelo dimensional, cujo algoritmo foi apresentado na Figura 4.6.

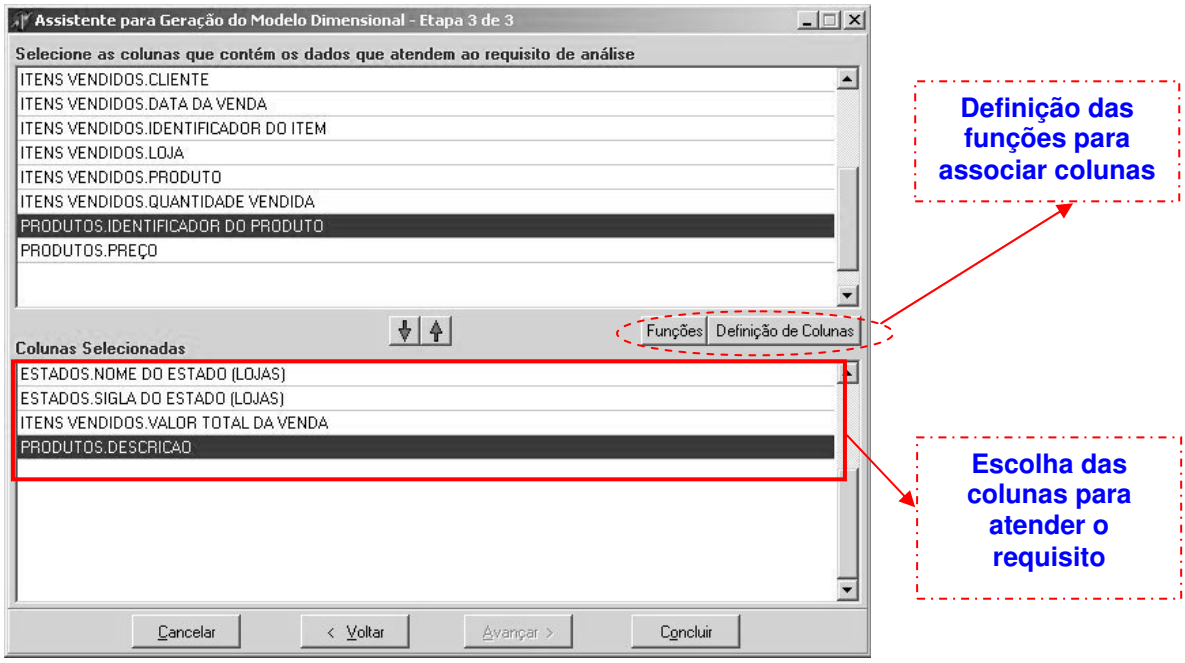


Figura 4.11 – Etapa 3 de 3 para geração do Modelo Dimensional (LIMA, 2006)

E, finalmente, o modelo gerado pode ser visualizado e impresso por meio de um relatório, conforme a Figura 4.12.

LISTAGEM DO MODELO GERADO
ASSUNTO: VENDA DE PRODUTOS

Data: 04/01/2006
Hora: 21:04
Pág.: 1

DADOS DO MODELO

Esquema: ESTRELA *Data/Hora da Geração:* 04/01/2006 / 20:59:25

REQUISITOS

OBTER OS VALORES MENSAIS DA VENDAS POR PRODUTO E ESTADO
OBTER OS ESTADOS QUE MAIS COMPRAM PRODUTOS POR SEMANA, MÊS, TRIMESTRE, SEMESTRE E ANO
OBTER OS VALORES MENSAIS DAS VENDAS POR PRODUTO E CIDADE
OBTER AS CIDADES QUE MAIS COMPRAM PRODUTOS POR SEMANA, MÊS, TRIMESTRE, SEMESTRE E ANO
OBTER OS MAIORES CLIENTES POR MÊS, SEGUNDO O VALOR DA VENDA

MODELO

FATO ITENS VENDIDOS (ITENS_VENDIDOS)

Coluna	Descrição	Tipo
PK_LOJAS	LOJAS	FK
PK_PRODUTOS	PRODUTOS	FK
PK_CLIENTES	CLIENTES	FK
PK_TEMPO	DIMENSÃO TEMPO	FK
QUANTIDADE	QUANTIDADE VENDIDA	
VALOR_TOTAL	VALOR TOTAL DA VENDA	

DIMENSÃO CLIENTES (CLIENTES)

Coluna	Descrição	Tipo
PK_CLIENTES	CHAVE PRIMÁRIA	PK
NOME_1	NOME DA CIDADE	
NOME_2	NOME DO CLIENTE	

Figura 4.12 – Visualização do modelo Dimensional (LIMA, 2006)

4.7. REFINAMENTOS DA FERRAMENTA

A ferramenta de Lima (2006) foi aperfeiçoada, tanto nas funcionalidades, quanto na tecnologia de sua construção, para atender aos propósitos deste trabalho. Assim sendo, surgiu a nova versão da ferramenta e que daqui em diante será conhecida como *WAVE* (***W**arehouse **A**nd **V**iew **E**ngineering*).

Foi desenvolvida uma nova versão da ferramenta *WAVE* fazendo uso da tecnologia Java, em razão dos novos módulos de seleção de visões e controle de hierarquias. Na nova versão são refinadas as funcionalidades da ferramenta proposta por Lima (2006), sem descaracterizar a versão original. Os refinamentos aplicados foram: a reorganização do conteúdo nas telas do assistente do modelo dimensional, a possibilidade de conexão com outros SGBDs (*Oracle* e *Mysql*) e a integração do controle de hierarquia no modelo dimensional. Na Figura 4.13, são mostrados os processos do módulo de modelagem a partir dos refinamentos.

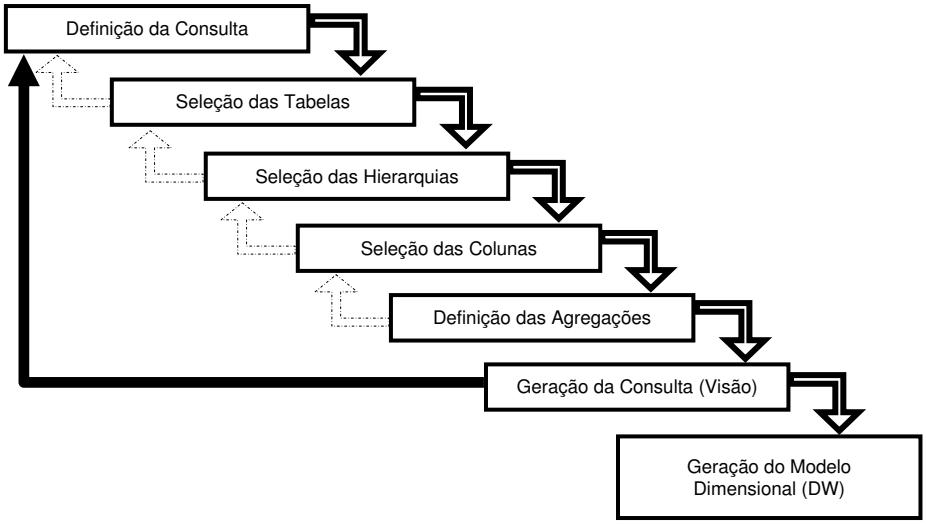


Figura 4.13 – Processos do Módulo de Modelagem – *WAVE*

4.8. RECURSOS COMPUTACIONAIS

Os recursos computacionais utilizados para o desenvolvimento da nova versão da ferramenta *WAVE* estão relacionados abaixo:

- Linguagem de programação: Java utilizando o *SDK6* (*Standard Developer Kit)*;

- Ambiente de desenvolvimento: *IDE* (*Integrated Development Environment*) - Eclipse Europa – Versão 3.3.0;

- Banco de dados *Firebird* versão 2.0;

Na conexão da *WAVE* com novos SGBDs *Mysql* e *Oracle* recomenda-se utilizar as seguintes versões relacionadas a seguir:

- SGDB *Mysql* Versão 5 ou superior;

- SGBD *Oracle* versão 10g *Enterprise* ou versão *Express Edition* 10g.

4.9. CONSIDERAÇÕES FINAIS

Neste capítulo foram apresentados os fundamentos, o funcionamento da ferramenta para Geração do Modelo Dimensional para *Data Warehouse* segundo a abordagem de Lima (2006), bem como os refinamentos para a nova versão.

O conteúdo apresentado nesse capítulo mostrou que a ferramenta auxilia no processo de geração do modelo do *DW*, e que também, para uso dela, o projetista não precisa ter conhecimento do processo para a geração de um modelo de dados dimensional. Dessa forma, o projetista pode ser conduzido pelos processos de geração do modelo dimensional, de forma interativa, fazendo uso de um assistente.

5. MÓDULOS DE SELEÇÃO DE VISÕES (SVM) E CONTROLE DE HIERARQUIAS

O foco principal a ser tratado, neste capítulo, é o problema que consiste na decisão de quais visões devem ser materializadas, em um *DW*. É sabido que existem diversas propostas para seleção de visões, para materializar para *Data Warehouse*, na literatura. O capítulo 2 apresentou algumas dessas abordagens. Cada uma delas apresenta uma proposta para seleção de um conjunto de visões que, de alguma forma, irá contribuir para a otimização do processamento das consultas em um *DW*. A seleção do conjunto ótimo de visões depende de restrições que podem ou não ser consideradas, no contexto da aplicação. Porém, o conjunto de visões obtido, por meio da seleção, tem, por objetivo, alcançar a melhoria no desempenho do tempo de processamento das consultas. Um dos fatores considerado relevante, pela literatura, é o espaço em disco ocupado pelas visões, restrição que o projetista do *DW* deve sempre considerar.

Este capítulo apresenta o módulo de seleção de visões e também o módulo de controle das hierarquias, ambos incorporados à ferramenta *WAVE*.

A finalidade do módulo SVM é apresentar uma solução para a seleção de visões a materializar. A ideia da elaboração deste módulo é proporcionar novos recursos ao projetista, em tempo de projeto, de modo que o conjunto de visões selecionadas, que possa contribuir no desempenho das consultas, seja definido logo na implantação do *DW*, antes mesmo que este seja colocado em uso. Para a definição desse conjunto de visões a materializar, o módulo SVM considera todas as consultas definidas no projeto do modelo dimensional.

O algoritmo SVM, uma contribuição deste trabalho, foi incorporado ao módulo SVM. O módulo proporciona um ambiente automatizado para que o projetista do *DW* possa ter facilidade na obtenção do conjunto das visões para a materialização, por meio do conjunto de visões obtidas dos metadados do *DW*, ou seja, do modelo dimensional.

5.1. ARQUITETURA DA FERRAMENTA *WAVE*

A arquitetura da nova versão da ferramenta foi estendida com a finalidade de cumprir os objetivos deste trabalho. As extensões compreendem o Módulo de Seleção de Visões para Materialização (SVM) e o Módulo para Controle das Hierarquias.

O módulo SVM atua como complemento às funcionalidades já existentes na ferramenta. O módulo SVM busca selecionar um conjunto de visões que possa trazer benefícios ao processamento das consultas do *DW,* logo após sua implantação, antes de se ter as estatísticas de frequência de uso das consultas.

O módulo para controle das hierarquias é utilizado para criá-las e disponibilizar recursos para que elas possam ser utilizadas no processo de geração do modelo dimensional. Mais detalhes sobre o módulo de controle de hierarquias será tratado na seção 5.3 deste capítulo. Na Figura 5.1 é apresentada a nova arquitetura da ferramenta, de acordo com os dois novos módulos.

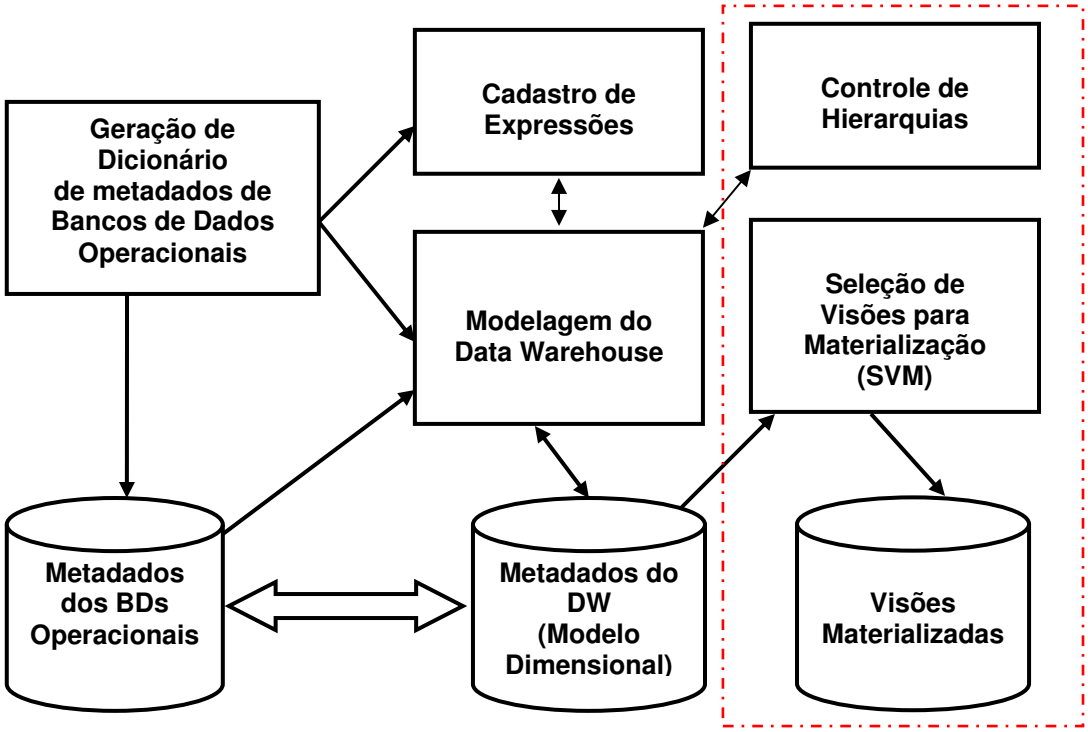


Figura 5.1 – Nova Arquitetura para a Ferramenta – *WAVE*

5.2. DESENVOLVIMENTO DO MÓDULO DE SELEÇÃO DE VISÕES (SVM)

O módulo de seleção de visões disponibiliza, para o projetista, de maneira interativa, as visões geradas por meio do módulo de modelagem do *DW*, que por sua vez, estão armazenadas como metadados do *DW*.

O projetista é conduzido pelas etapas do módulo SVM, iniciando pela criação de um projeto de seleção de visões, até obter o conjunto das visões selecionadas. Além do conjunto de visões selecionadas para a materialização, o módulo SVM gera um arquivo texto contendo as instruções *SQL*, de acordo com o padrão *SQL3* ou *SQL-99* (SQL,1999), previsto como finalização da última etapa do módulo. Fornece também um relatório detalhado com os resultados de cada projeto que o projetista possa criar.

A política adotada, no módulo SVM, é baseada na restrição de espaço, no tamanho das visões, na frequência de uso das visões, no custo de materialização e no cálculo dos benefícios por freqüência de uso e por custo de materialização.

As visões submetidas à seleção são entradas no módulo SVM e referem-se a um determinado assunto. Cada assunto representa um *DW* diferente, estratégia utilizada pela *WAVE*, para que seja possível armazenar na ferramenta vários projetos de *Data Warehouse.*

A ferramenta *WAVE*, por meio do módulo SVM, possibilita recuperar os dados de cada projeto criado, para possíveis alterações e/ou comparações, proporcionando a busca por um conjunto de visões mais adequado e benéfico de acordo com resultados anteriores.

5.2.1. DEFINIÇÃO DOS BENEFÍCIOS

Benefícios são fatores que proporcionam alguma vantagem, diante da possibilidade da seleção de visões, em razão da obtenção de um conjunto ótimo ou quase ótimo de visões. Estes fatores contribuem na definição dos critérios para a seleção. Os benefícios considerados para este trabalho são: benefício por

frequência de uso e o benefício por custo de materialização (custo para atender à consulta).

Levando em consideração os cálculos dos benefícios, tanto por frequência de uso, quanto por custo de materialização, são previstos dois conjuntos de visões: o conjunto **VC** de visões candidatas à seleção e o conjunto **VS** de visões selecionadas.

O conjunto de visões candidatas **VC** é formado por visões que, a princípio, não atenderam o primeiro critério de seleção do algoritmo SVM, mas que podem ser escolhidas devido a outros critérios em que as mesmas são submetidas para seleção, considerados pelo algoritmo.

O conjunto de visões selecionadas **VS** é o conjunto que, a princípio, atende o primeiro critério de seleção de acordo com o algoritmo, porém o conjunto **VS** não é o conjunto final de visões, e sim um conjunto inicialmente selecionado. Para ser definido o conjunto final de visões selecionadas, o algoritmo submete as visões de **VS** a outros critérios, os quais são descritos posteriormente.

O cálculo dos benefícios requer a análise dos conjuntos **VC** e **VS** em que, o benefício de cada elemento $\mathbf{VC_i}$ de **VC** seja submetido ao benefício do elemento $\mathbf{VS_j}$ de **VS**, buscando atender o critério para troca da visão candidata por uma visão já selecionada.

Os benefícios por frequência de uso e por custo de materialização buscam atender, nos dois casos, o benefício mínimo definido pelo usuário para que a troca da visão candidata pela visão selecionada seja possível.

Os benefícios mínimos são definidos como constantes $\boldsymbol{\beta_f}$ e $\boldsymbol{\beta_c}$ e são parametrizados pelo projetista do *DW*, durante o processo de seleção de visões, por meio do módulo SVM. As constantes $\boldsymbol{\beta_f}$ e $\boldsymbol{\beta_c}$ são definidas na faixa de 0.1 a 1. A faixa é convertida em percentagem e equivale, na mesma proporção, de 10 a 100% do índice de satisfação desejado para cada caso.

As equações dos benefícios, por frequência de uso e por custo de materialização são definidas, respectivamente, nos Quadros 5.2 e 5.3.

$$B_{fu} = \frac{\text{Freq.Uso de VC}_i}{\text{Freq.Uso de VS}_j} \geq \beta f \qquad (28)$$

Onde:

B_{fu} **:** benefício por frequência de uso;

βf **:** benefício mínimo de frequência de uso. Constante definida pelo usuário;

VC_i : i-ésima visão candidata: Elemento **$_i$** do conjunto de Visões Candidatas à seleção, ordenadas por frequência de uso;

VS_j : elemento **$_j$** do conjunto de visões ordenadas por frequência de uso.

Equação 28 – Benefício por Frequência de Uso

$$B_{cm} = \frac{\text{Num.acessos de VC}_i}{\text{Num.acessos de VS}_j} \geq \beta c \qquad (29)$$

Onde:

B_{cm} **:** benefício por Custo de Materialização de Visões;

βc **:** benefício mínimo de custo a considerar, ou seja, constante definida pelo usuário;

VC_i : i-ésima visão candidata: Elemento **$_i$** do conjunto de visões candidatas à seleção;

VS_j : elemento **$_j$** do conjunto de visões selecionadas.

Equação 29 – Benefício por Custo de Materialização

5.2.2. DEFINIÇÃO DO CUSTO

A definição do custo é baseada na análise da quantidade de acessos a disco, levando em consideração as junções realizadas para atender a uma determinada consulta. São estimados os acessos a disco realizados, considerando a ordem: primeiro a junção da tabela fato com a dimensão tempo e, em seguida, a junção dos resultados intermediários com cada uma das demais dimensões. A forma como as tuplas da tabela fato são registradas obedece a ordem cronológica de tempo, portanto um certo número de tuplas da fato faz referência a uma mesma unidade de tempo (aquela de granularidade mais fina, que serviu de base para a identificação da tabela tempo). Logo, para o cálculo do número de acessos à dimensão tempo, divide-se a tabela de fatos em grupos de blocos que

contêm o mesmo valor para o identificador da dimensão tempo; o número de acessos à dimensão tempo para realizar sua junção com a tabela fato será a quantidade desses grupos de blocos. Para o cálculo do custo para as demais junções, entre as tabelas intermediárias e as outras dimensões envolvidas, considera-se que serão necessários tantos acessos quantos forem o número de tuplas da tabela fato, uma vez que o acesso a essas dimensões é realizado via índice, usando o identificador da dimensão presente nas tuplas da tabela fato. Essa estimativa de custo leva em consideração o pior caso. Desta forma, o custo total de cada visão será a somatória dos acessos a disco obtidos pelas várias junções acima citadas. Observa-se que a ordem das tabelas na junção para o cálculo do custo é indiferente.

Como a abordagem do módulo SVM considera a seleção das visões em tempo de projeto, o custo é uma estimativa e não um cálculo exato. Para estimar o cálculo real do número de acessos é necessário considerar a política de otimização e dos planos de consulta que o SGBD envolvido utiliza. O otimizador do SGDB lança mão do uso de *buffers* para diminuir a quantidade de acessos a disco e, por consequência, diminuir o tempo de processamento (SILBERSCHATZ; KORTH; SUDARSHAN, 1999). Para estimar o cálculo do custo de uma consulta, é necessário estimar como as junções são implementadas. A definição dessa estimativa em tempo de projeto torna-se muito complicada, pois são muitas possibilidades a considerar.

Nesta linha de estimativa foram considerados os seguintes dados para o cálculo do custo: número de tuplas da tabela fato, o número de bytes para 1 tupla de cada dimensão envolvida e da tabela fato, o fator de bloco (página) e o número de blocos e o número de bytes por página (bloco). O número de bytes por página é importante, visto que, pode haver variação desta informação de um banco de dados para outro.

Para a dimensão tempo, em especial, foi considerada, como entrada manual no algoritmo, o valor correspondente ao tamanho de 1 tupla em bytes. Para as outras dimensões e para a fato é possível calcular os valores em bytes das tuplas, devido à ferramenta já ter feito o levantamento dos metadados do banco operacional, durante o processo de geração do modelo dimensional.

A estratégia adotada para o cálculo do custo é definida, genericamente, da forma a seguir:

Fato ⋈ Tempo ⋈ Dimensão_1 ⋈ Dimensão_n, onde o número total de acessos a disco é calculado levando em consideração a somatória do número de acessos realizado, considerando cada junção, da seguinte forma: Fato ⋈ Tempo, Temp1 ⋈ Dimensão_1, Temp2 ⋈ Dimensão_2 e assim por diante. Para o cálculo do custo da operação *group by* estão envolvidos: o particionamento da tabela em subconjuntos de tuplas, a utilização de ordenação ou *hashing* e o calculo da função de agregação para cada tupla em cada grupo. Se existir um índice *clustering* para o(s) atributo(s), então os registros já estarão particionados (agrupados) em subconjuntos apropriados (ELMASRI; NAVATHE, 2005).

Para maior esclarecimento da estratégia de custo toma-se, por base, o exemplo a seguir:

Visão1 = Tempo(Mês) Vendas(Valor_Total,Quantidade) Produtos(Descricao) Lojas(Nome_loja, CNPJ).

O primeiro passo é fazer a junção da dimensão Tempo com a tabela fato Vendas (Tempo ⋈ Vendas). Após essa junção, tem-se o resultado em uma tabela temporária (Temp1). Então, faz-se a junção de Temp1 com a próxima dimensão Produtos (Temp1 ⋈ Produtos) que, como resultado, obtém-se a tabela temporária Temp2. O último passo é executar a junção de Temp2 com a última dimensão Lojas (Temp2 ⋈ Lojas) que, por sua vez, tem seu resultado na tabela temporária Temp3.

Segue nas Figuras 5.2, 5.3, 5.4 e 5.5 a demonstração dos cálculos, de acordo com a **Visão 1**, onde:

fb: fator de bloco; nb: número de blocos; qo: quantidade de ocorrências realizadas, em média, por grão de tempo e $Temp1, Temp2, Temp3$ são tabelas temporárias 1,2 e 3.

Para a junção: (**Tempo** ⋈ **Vendas**)

$$\left\lceil fb_{fato} = \frac{Num.bytes\ por\ página(bloco)}{Num.bytes\,1 tupla\ fato} \right\rceil \quad \left\lfloor nb_{fato} = \frac{Num.tuplas\ da\ fato}{fb_{fato}} \right\rfloor$$

$$\left\lceil fb_{Temp1} = \frac{Num.bytes\ por\ página(bloco)}{Num.bytes\,1 tupla\ fato + Num.bytes\,1 tupla\ Tempo} \right\rceil$$

$$\left\lfloor nb_{Temp1} = \frac{Num.tuplas\ da\ fato}{fb_{Temp1}} \right\rfloor$$

O número de acessos a disco resultante dessa primeira junção é:

$$Result1 = nb_{fato} + num.acessos\,Tempo + num.acesso\,índice_{Tempo} + nb_{Temp1} + num.acessos\,Group\,by$$

Figura 5.2 – Junção de Tempo com Vendas

Para a junção: (**Temp1** ⋈ **Produtos**)

$$\left\lceil fb_{Temp2} = \frac{Num.bytes\ por\ página(bloco)}{Num.bytes\,1 tupla\,Temp1 + Num.bytes\,1 tupla\,\text{Produtos}} \right\rceil$$

$$\left\lfloor nb_{Temp2} = \frac{Num.tuplas\ da\ fato}{fb_{Temp2}} \right\rfloor$$

O número de acessos resultante dessa junção é:

$$Result2 = nb_{Temp1} + num.tuplas\ Fato + num.acesso\,índice_{\text{Produto}} + nb_{Temp2} + num.acessos\,Group\,by$$

Figura 5.3 – Junção de Temp1 com Produtos

Para a junção: (**Temp2** ⋈ **Lojas**)

$$\left\lceil fb_{Temp3} = \frac{Num.bytes\ por\ página(bloco)}{Num.bytes\,1 tupla\,Temp2 + Num.bytes\,1 tupla\ Lojas} \right\rceil$$

$$\left\lfloor nb_{Temp3} = \frac{Num.tuplas\ da\ fato}{fb_{Temp3}} \right\rfloor$$

O número de acessos resultante dessa junção é:

$$Result3 = nb_{Temp2} + num.tuplas\ Fato + num.acesso\,índice_{Lojas} + nb_{Temp3} + num.acessos\,Group\,by$$

Figura 5.4 – Junção de Temp2 com Lojas

$$Custo\ da\ Visão = \text{Result1} + \text{Result2} + \text{Result3}$$

Figura 5.5 – Custo para Materialização da Visão

5.2.3. ALGORITMO SVM

Diante das pesquisas realizadas, na literatura, foi observado que os autores apresentam suas abordagens de forma muito específica e, os algoritmos aplicados acompanham essa especificidade no contexto estudado. Adotar um algoritmo da literatura para este trabalho seria parcialmente válido, pois o mesmo teria que ser adaptado à realidade da ferramenta *WAVE.* Portanto, foi decidido que o ideal seria o desenvolvimento de um algoritmo próprio que pudesse contar com as entradas que a ferramenta *WAVE* atende de imediato. Fazer modificações na ferramenta, para adequá-la fielmente a algum algoritmo da literatura, seria quebrar os objetivos deste trabalho, pois a maioria das funcionalidades da ferramenta tem que ser adequadas. Sob tais condições de mudança, a ferramenta perde sua originalidade. O algoritmo proposto leva em consideração que as visões são as consultas obtidas durante a etapa de geração do modelo dimensional, de acordo com o Módulo de Modelagem do Modelo Dimensional da *WAVE*.

ALGORITMO SVM

Entrada: Conjunto de visões inseridas pelo projetista, Nº Tuplas da tabela fato, Tamanho de cada visão, Nº Tuplas das Dimensões, Espaço de Projeto, Frequência de Uso das Visões, Granularidade de cada Visão, Número de bytes para 1 tupla de cada dimensão, Número em bytes por página(bloco), β_f (benefício mínimo de frequência de uso) e β_c (benefício mínimo para o custo de materialização da visão).

Saída: Conjunto de visões selecionadas.
Início

1. Ordenar o conjunto **V** das visões, por frequência de uso;
2. Percorrer o conjunto **V** em busca de visões iguais, ou seja, formadas pelas mesmas tabelas e mesmos atributos. Caso encontre visões iguais, formadas pela mesma sequência de tabelas e atributos, eliminar a visão que contém a menor frequência de uso. Se encontrar visões com sequências diferentes, mas com a mesma frequência de uso, qualquer visão pode ser eliminada.
3. Ordenar as visões com mesma frequência de uso, em ordem decrescente com relação ao custo de cada visão;

4. Percorrer o conjunto **V** do elemento mais significativo para o menos significativo e obter o conjunto das visões que caibam no espaço de projeto. Guardar em **VS_inicial**.
5. Gerar o conjunto das visões candidatas **VC_inicial** com as visões que não foram selecionadas para o conjunto **VS_inicial** (colocar em ordem da menos significativa para a mais significativa).
6. **VC ← VC_Inicial**
7. **VS ← VS_Inicial**
8. Percorrer o conjunto **VC** em busca de visões que podem ser geradas (descendente direta) das visões do conjunto **VS**. Se $\mathbf{VC_i}$ for descendente direta de $\mathbf{VS_j}$ elimina $\mathbf{VC_i}$ do conjunto **VC** e guarda $\mathbf{VS_j}$ no conjunto **VS_final** ($\mathbf{VS_j}$ tem que pertencer obrigatoriamente ao conjunto das visões selecionadas **VS_final**).

REPITA

CONDIÇÃO DE PARADA: Se não existir opção de troca em função dos benefícios de frequência de uso, custo de materialização e espaço disponível para no mínimo 2 candidatas ou por não existir elementos no conjunto **VC.**

9. Calcular os benefícios para cada elemento $\mathbf{VC_i}$ de **VC** com relação a $\mathbf{VS_j}$ de **VS**; iniciar pelo elemento mais significativo de cada conjunto.
10. Buscar nos conjuntos **VC** e **VS** as opções de troca se a visão do conjunto selecionado puder ser substituída (em termos de existência de espaço) por pelo menos duas visões candidatas, considerando o cálculo dos benefícios **(frequência de uso + custo de materialização da visão)**.
11. Colocar as visões selecionadas no conjunto **VS_final**;
12. Retirar, do conjunto **VS,** cada visão $\mathbf{VS_j}$ substituída e colocar no conjunto **VT** (Visões de Troca).

FIM REPITA

13. Colocar em **VS_final** as visões que permanecerem em **VS**, em ordem decrescente de frequência de uso e para as visões com mesma frequência de uso, em ordem decrescente por custo de materialização.

Fim do Algoritmo.

5.2.4. APRESENTAÇÃO DO MÓDULO SVM

A ideia da construção do módulo de seleção de visões é a de conduzir o projetista no projeto de visões, de forma natural, minimizando seu envolvimento em questões técnicas a respeito de políticas de seleção de visões. A estratégia de implementação do algoritmo, por meio do módulo SVM, caracteriza-se por um assistente. O assistente do módulo de seleção de visões é dividido em quatro passos.

O primeiro passo do módulo SVM é caracterizado por apresentar, ao projetista, estatísticas das tabelas de dimensões de um *DW*. As estatísticas incluem a quantidade de tuplas e o tamanho em *megabytes* de cada dimensão. O tamanho das dimensões é calculado usando, como base, as tabelas do banco operacional, ou seja, o tamanho em bytes de uma tupla da(s) tabela(s) que a(s) compõem, multiplicado pela quantidade de tuplas existente na(s) tabela(s). É fornecido também, nessa primeira etapa, pelo projetista, o valor do espaço em disco previsto para o projeto. O espaço destinado ao projeto considera o tamanho do *DW* (fato e dimensões) e o espaço para a materialização das visões.

O projetista pode fornecer uma previsão de crescimento do *DW* ao longo de um período de tempo em que o conjunto de visões escolhidas ainda seja adequado. O valor do aumento do crescimento é levado em consideração para os cálculos das visões a materializar. A tela que representa este passo do módulo pode é mostrada na Figura 5.6.

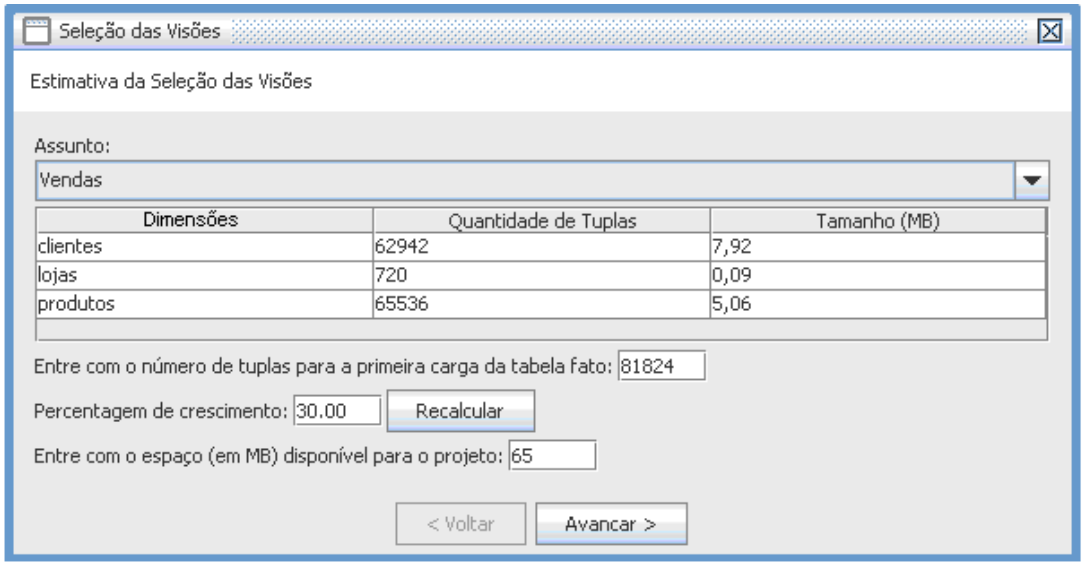


Figura 5.6 – Estimativa de Tamanho das Dimensões

O segundo passo consiste em exibir as visões, ao projetista, de acordo com o assunto selecionado. A cada visão exibida, a ferramenta estima o seu tamanho. O projetista insere o valor que representa a frequência de uso, para cada visão, e também a periodicidade, podendo ser: diária, semanal, mensal, bimestral, semestral ou anual. Para efeito de cálculo, no algoritmo SVM, todas as periodicidades serão convertidas para o menor grão, ou seja, para diário.

A descrição de cada visão é exibida, ao projetista, de acordo com a sequência de escolha das colunas feitas por ele. A descrição é montada da seguinte forma: Primeiro aparece a dimensão tempo, seguida do atributo escolhido "Tempo(atributo)". Na sequência aparecem as tabelas de fato e dimensões, e as colunas envolvidas de acordo com as escolhas "Fato/Dimensão1 (coluna1, coluna2...)". Quando é escolhida mais de uma coluna de uma tabela fato ou dimensão, as mesmas são separadas por vírgula. Na Figura 5.7 é mostrado o segundo passo do módulo SVM.

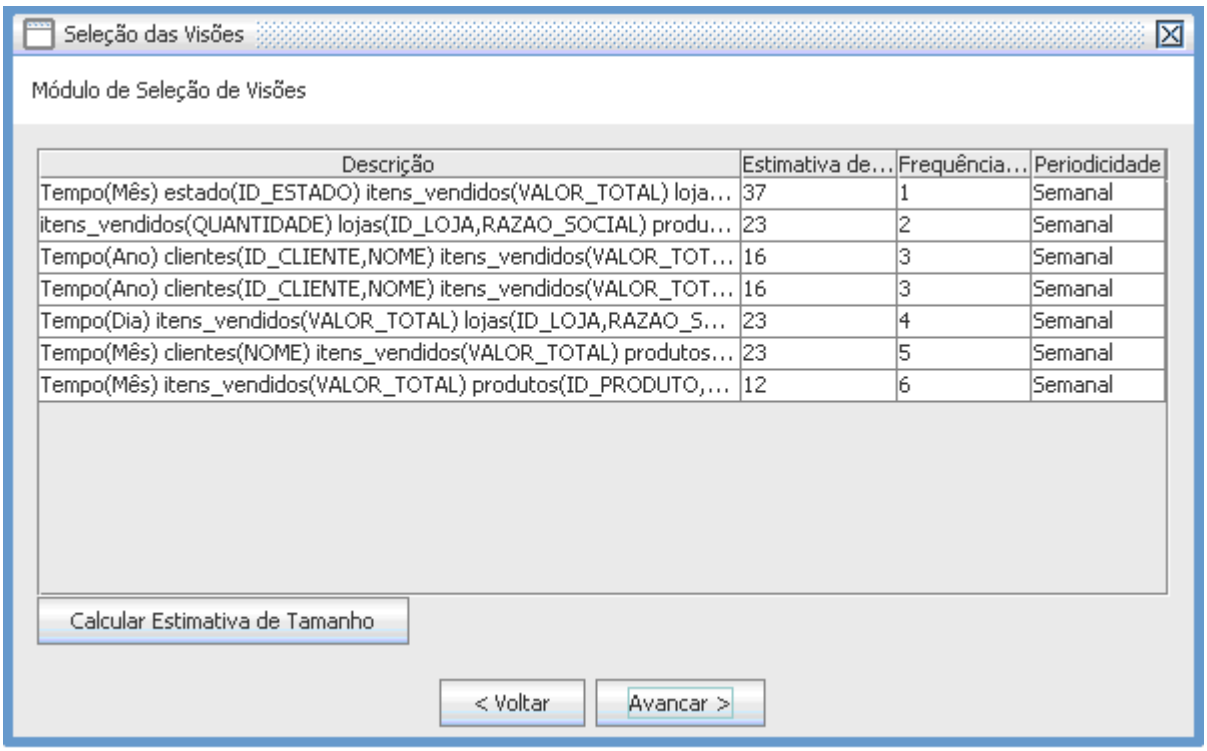


Figura 5.7 – Definição do Tamanho e Frequência de Uso das Visões

O terceiro passo apresenta os parâmetros para uso do algoritmo SVM. Os parâmetros podem ser alterados de acordo com a necessidade do projetista. As constantes dos benefícios e os parâmetros: número de bytes por

página (bloco) e tamanho em bytes para 1 tupla da dimensão tempo são definidos como entradas manuais pelo projetista, visto que tais parâmetros podem variar de um projeto para outro. A alteração dos mesmos provoca mudanças nos resultados, pois se referem diretamente aos cálculos, para obtenção do conjunto de visões a materializar.

As constantes de benefício por freqüência de uso (β_f) e por custo de materialização (β_c) são responsáveis por comparar o resultado do cálculo em nível de satisfação. O algoritmo adota o índice de 50% de satisfação (correspondente à 0.50), sendo que o projetista pode alterar esse valor. A seguir, na Figura 5.8, é exibida a tela de parametrização do algoritmo. Além dos parâmetros, a tela de parametrização possui uma opção para eliminar as visões descendentes diretas[6]. Se esta opção estiver marcada, o algoritmo irá percorrer o conjunto de visões candidatas **VC**, em busca das visões que podem ser geradas, ou seja, descendentes diretas do conjunto **VS**.

Uma visão **A** pode ser descendente direta de uma visão **B** se a visão **A** possuir todas as colunas e atributos de uma visão **B**, conforme o exemplo a seguir:

Visão A: Tempo(Dia) itens_vendidos (QUANTIDADE) lojas (ID_LOJA, RAZAO_SOCIAL).

Visão B: Tempo(Dia) itens_vendidos (VALOR_TOTAL,QUANTIDADE) lojas (ID_LOJA, RAZAO_SOCIAL, ID_CIDADE) produtos(ID_PRODUTO, DESCRICAO).

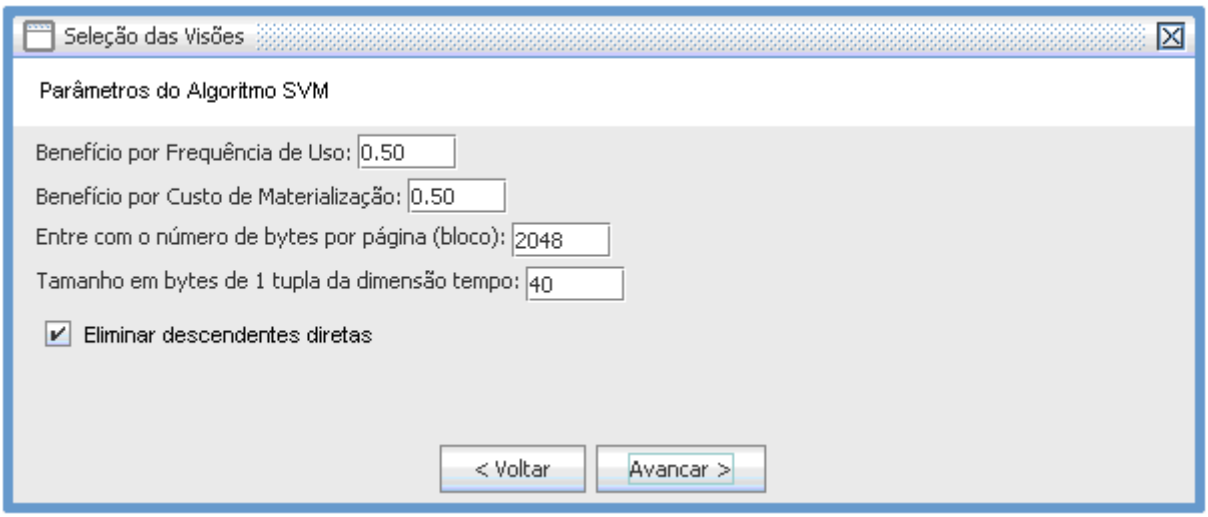


Figura 5.8 – Parâmetros do Algoritmo SVM

[6] Uma visão V1 é descendente direta de uma visão V2 se, no grafo de derivação, V1 estiver diretamente ligada à V2 e estiver localizada em um nível inferior a V2.

O quarto passo do módulo realiza os cálculos para escolha das visões a materializar, de acordo com as entradas que o projetista forneceu durante os passos anteriores para o algoritmo. A tela de resultados pode ser visualizada na Figura 5.9.

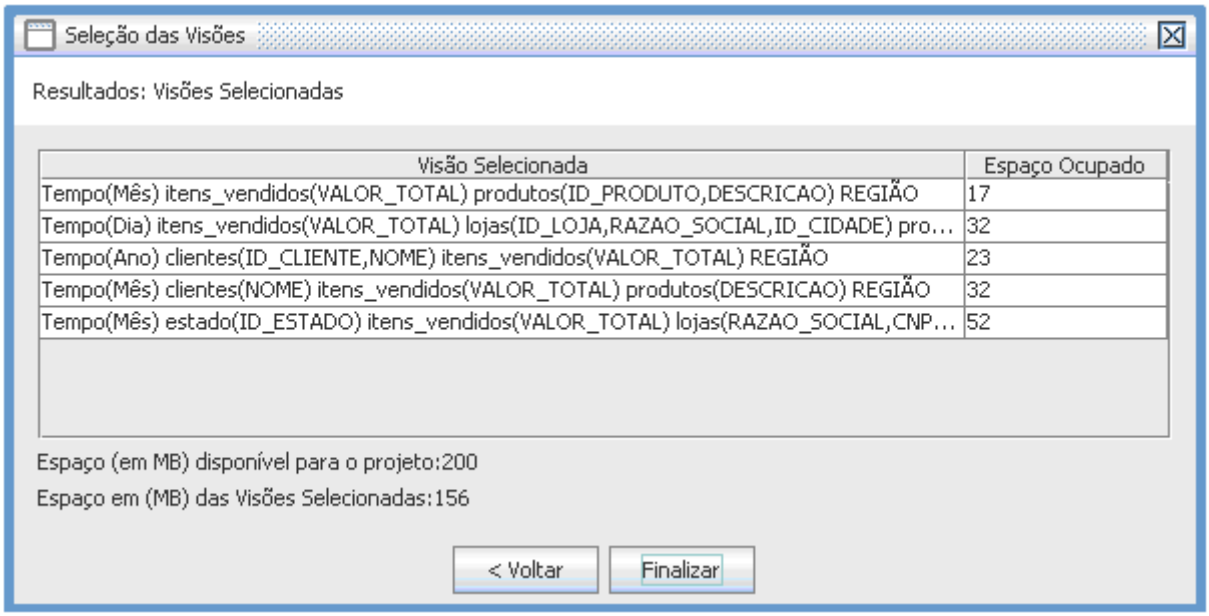


Figura 5.9 – Conjunto de Visões Selecionadas

Há possibilidade, nesse quarto passo, de retornar aos passos anteriores para alterar parâmetros do algoritmo, tais como: a frequência de uso, periodicidade, espaço disponível para o projeto e as estimativas das tabelas dimensões e fato, por meio do fator de crescimento. A partir de possíveis mudanças, ao retornar ao quarto passo novamente, um novo conjunto de visões será gerado. Os procedimentos de avanço e recuo garantem flexibilidade ao projetista, permitindo a obtenção de resultados que possam atender melhor às necessidades.

Ao finalizar os procedimentos realizados pelo módulo de seleção de visões, o projeto é salvo e seus parâmetros podem ser recuperados posteriormente, para análise e possíveis alterações. Após o projeto ser salvo, são geradas as instruções *SQL*, em forma de arquivo texto, para importação pelos SGBDs. Para a geração das instruções *SQL,* foram consideradas as agregações definidas pelo módulo de controle de expressões, de acordo com os operadores de agregação: *SUM*(soma), *AVG*(média), *COUNT*(contagem), *MIN*(mínimo), *MAX*(máximo). A

Figura 5.10 ilustra um exemplo de instruções geradas que atendem ao padrão *SQL3 (SQL,1999).*

```
CREATE OR REPLACE MATERIALIZED VIEW "Vendas_VISAO5" AS
SELECT tempo.Dia,
    itens_vendidos.VALOR_TOTAL ,
    lojas.ID_LOJA,
    lojas.RAZAO_SOCIAL,
    lojas.ID_CIDADE,
    produtos.ID_PRODUTO,
    produtos.DESCRICAO
  FROM  tempo, itens_vendidos,
       lojas,
       produtos
  WHERE itens_vendidos.pk_tempo = tempo.pk_tempo AND
        lojas.ID_LOJA = itens_vendidos.ID_LOJA AND
        produtos.ID_PRODUTO = itens_vendidos.ID_PRODUTO
  GROUP BY tempo.Dia, itens_vendidos.VALOR_TOTAL, lojas.ID_LOJA, lojas.RAZAO_SOCIAL,
lojas.ID_CIDADE, produtos.ID_PRODUTO, produtos.DESCRICAO
;
```

Figura 5.10 – Instruções *SQL* em Arquivo Texto

O módulo SVM permite também visualizar o log do processo ocorrido, durante o processamento do algoritmo SVM. É disponibilizado, por projeto criado, por meio de um relatório, o log do processamento das visões submetidas à seleção, ou seja, quais visões participaram de VS_Inicial (primeiro conjunto de visões selecionadas), VC_Inicial (primeiro conjunto de visões candidatas), VC (quais visões permaneceram como candidatas, no final do processamento), VT (as visões que participaram de troca, entre visões selecionadas e visões candidatas), as visões que foram excluídas, seja porque são iguais ou descendentes diretas, o custo de cada visão, a frequência de uso e a periodicidade, e por fim, o conjunto VS de visões selecionadas.

Os resultados dos cálculos são exibidos, tais como: o espaço disponível para o projeto, o espaço ocupado pelas visões, o tamanho do *DW*, os parâmetros do algoritmo (benefícios, número de bytes por página, status do uso de eliminação das visões descendentes), o número de tuplas da tabela fato e seu tamanho em bytes, o número de tuplas de cada dimensão e seus respectivos tamanhos em bytes. Na Figura 5.11, é exibida parte do relatório de um projeto. Esse relatório é importante para que o projetista tenha condição de analisar os resultados obtidos. A análise do relatório permite que o projetista faça modificações nos parâmetros, no sentido de conseguir um novo conjunto de visões selecionadas. É permitido também, ao projetista, gerar um novo projeto com novos parâmetros, a fim de comparar a solução desenvolvida com resultados de projetos anteriores.

Para cada visão exibida no relatório de LOG, a interpretação das informações relativas aos conjuntos VS_Inicial, VC_Inicial, VS, VC, VT é feita da seguinte forma: considerando o exemplo em destaque na Figura 5.11, o número “1” embaixo de VC_Inicial e de VC representa que esta visão participou dos dois conjuntos durante o processamento do algoritmo e, por sua vez, ocupou a posição “1” do vetor de cada conjunto. Se essa visão também tivesse participado de outros conjuntos, além dos representados no exemplo, estariam também representados, no relatório, os outros conjuntos participantes e suas respectivas posições no vetor. No caso de eliminação da visão pelo algoritmo, é mostrado embaixo de “Mot.Elimin.” qual o motivo da eliminação (Igual: se foi eliminada por ser uma visão igual ou “Descendente direta”), caso o algoritmo esteja habilitado para verificar esta condição.

Projeto: PROJETO_02

Espaço disponível ao projeto: 200MB
Espaço total ocupado: 177MB
Tamanho do DW: 21MB
Benefício por Frequência de Uso: 0.50
Benefício por Espaço Ocupado: 0.50
Número de Bytes por Página (Bloco): 2048
Eliminar descendentes diretas: S
Número de Tuplas da Tabela Fato: 113295
Tamanho da Tabela Fato: 9.08MB

Dimensões

Nome	Tuplas	Tamanho
clientes	62942	7.92
lojas	720	0.09
produtos	65536	5.06

Visões

Descrição da Visão

itens_vendidos(QUANTIDADE) lojas(ID_LOJA,RAZAO_SOCIAL) produtos(DESCRICAO)

Tam.	Freq. Uso	Period.	Acessos Disco	vs_inicial	vc_inicial	vs	vc	vt	Mot. Elimin.
32	1	Diário	405137		1		1		

Tempo(Ano) clientes(ID_CLIENTE,NOME) itens_vendidos(VALOR_TOTAL) REGIÃO

Tam.	Freq. Uso	Period.	Acessos Disco	vs_inicial	vc_inicial	vs	vc	vt	Mot. Elimin.
23	2	Mensal	258799						Igual

.....
.....

Figura 5.11 – LOG ou Relatório de Configurações do Projeto de Visões

De acordo com o relatório, ilustrado na Figura 5.11, o tamanho da tabela fato exibido é calculado sob o mesmo critério em que são calculadas as dimensões (i.e., tamanho de uma tupla da tabela envolvida, multiplicada pela

quantidade de tuplas estimada pelo projetista). E, o tamanho do *DW*, é calculado somando os tamanhos das dimensões e da tabela fato.

5.3. MÓDULO DE CONTROLE DE HIERARQUIAS

O módulo de controle de hierarquias foi desenvolvido para auxiliar a geração do modelo dimensional. O objetivo é definir as hierarquias para uso no modelo dimensional e exercer o controle sobre os atributos pertencentes a essas hierarquias, de modo que elas possam ser associadas ao modelo dimensional de forma correta. Na nova versão da ferramenta, apresentada em detalhes no capítulo 6, o módulo de controle de hierarquias define quais as colunas que fazem parte de uma hierarquia. Por exemplo: Se o projetista quer definir para um *DW* de vendas uma hierarquia “Região”, por meio do cadastro de hierarquias da ferramenta, então é definido primeiro o nome da hierarquia, “Região”. Depois são escolhidas as colunas que participam desta hierarquia. Não é permitido definir uma hierarquia com o mesmo nome, a partir de uma já cadastrada. A próxima tela, na Figura 5.12, mostra como é feito o cadastro das hierarquias.

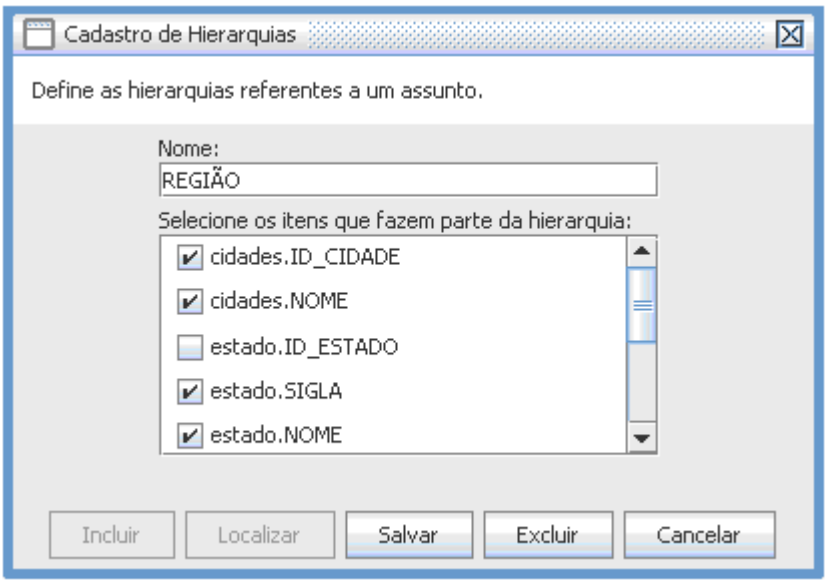


Figura 5.12 – Cadastro de Hierarquias

Após a definição de uma hierarquia, a mesma será considerada na geração no modelo dimensional. Se, porventura, em uma nova consulta, o projetista escolher colunas pertencentes às tabelas já definidas em uma hierarquia, tais atributos serão inseridos na hierarquia e não será criada uma nova dimensão.

O projetista pode associar uma hierarquia às consultas, de acordo com o contexto abordado pelo *DW*, veja o exemplo para a consulta 1: “Obter os valores mensais das vendas por produto e por Região”. Neste caso, é disponibilizado ao projetista, na etapa 2 do assistente do modelo dimensional, as hierarquias definidas previamente que, por sua vez, podem ser associadas à consulta 1. Se a consulta 1 determina que quer, como resultado, as vendas mensais por produto e por região, o projetista deve escolher “Região” entre as hierarquias existentes.

Na versão atual da ferramenta só pode ser associada uma hierarquia para cada consulta. A Figura 5.13 exibe a tela deste exemplo.

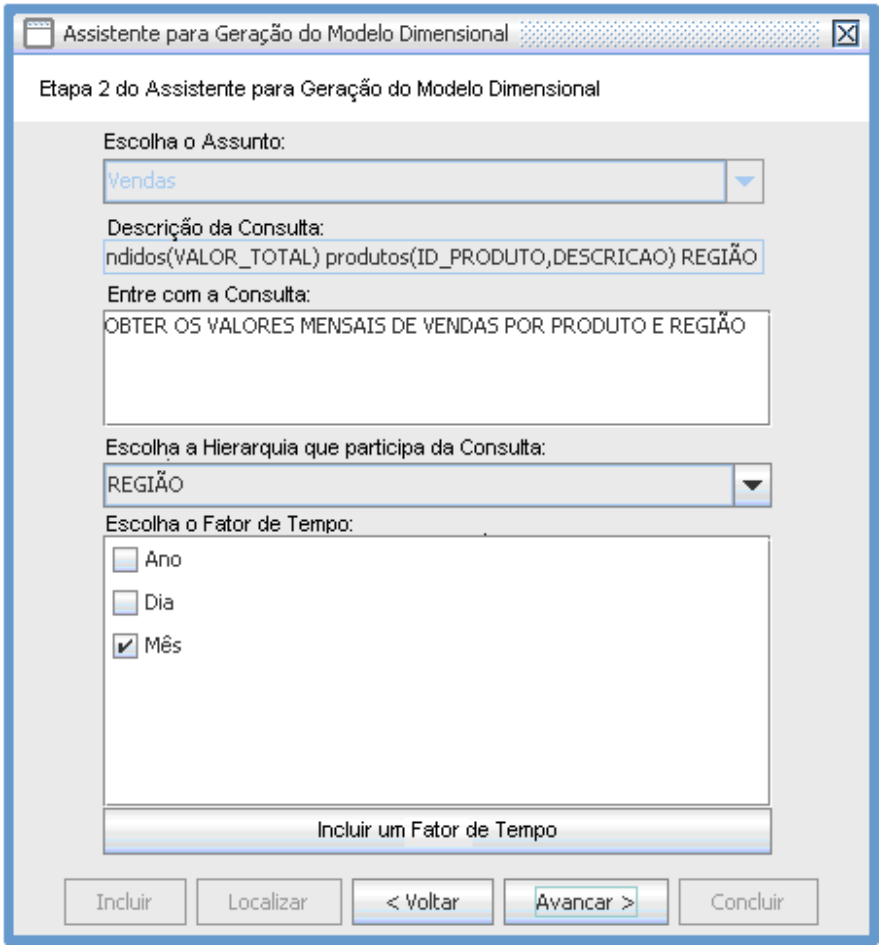


Figura 5.13 – Exemplo do Uso da Hierarquia “Região”

5.4. CONSIDERAÇÕES FINAIS

Este capítulo apresentou uma visão geral do módulo de seleção de visões a materializar (SVM) e do módulo de controle de hierarquias. Foram discutidos os conceitos sobre os dois módulos, a arquitetura incorporada à *WAVE,* a definição dos benefícios, a definição do custo e o algoritmo SVM.

A abordagem apresentada no capítulo de trabalhos relacionados tem semelhança com a abordagem apresentada neste capítulo sob os seguintes aspectos: Golfarelli, Maniezzo e Rizzi (2004) fazem uso da fragmentação vertical, fundamentada por sua metodologia, e esta é aplicada ao protótipo da ferramenta WanD, desenvolvida por Golfarelli, Rizzi e Saltarelli (2002). A metodologia trata o uso da fragmentação vertical, sobre fragmentos de visões. No módulo SVM as visões são analisadas sob o aspecto vertical, ou seja, por colunas nas tabelas de interesse. Utiliza a metodologia desenvolvida por Lima (2006) para definir as consultas e o modelo dimensional. Por consequência, o módulo SVM, assim como os outros módulos também estão aplicados a uma ferramenta, a *WAND*.

Golfarelli, Maniezzo e Rizzi (2004) utilizam fragmentos provenientes de um cubo de dados de um esquema multidimensional. As *workloads* são formadas por um conjunto de consultas, cujos dados são obtidos por meio de uma conexão ao banco operacional, utilizando o driver *ODBC*. No módulo SVM não se utiliza fragmentos e sim as visões obtidas de cada consulta, através do módulo de geração do modelo dimensional da *WAVE*. As visões são consultas definidas na etapa de modelagem e armazenadas na *WAVE* como metadados do *DW.*

É utilizado por Golfarelli, Maniezzo e Rizzi (2004), na ferramenta *WanD*, algoritmos da literatura para seleção de um conjunto ótimo de fragmentos. Para a melhor fragmentação o custo de cada fragmento é um fator primordial, assim como os fragmentos candidatos. O módulo SVM possui seu próprio algoritmo para seleção das visões, e este se utiliza do custo das visões, entre outras restrições, para definir quais visões serão selecionadas. Visões candidatas também fazem parte da estratégia do algoritmo do módulo SVM, havendo a possibilidade de troca desde que os benefícios sejam atendidos.

Ao final do processo, proposto por Golfarelli, Maniezzo e Rizzi (2004), é obtido o conjunto de fragmentos que minimiza o tempo de processamento. A estratégia apresentada, de acordo com o módulo SVM, não trabalha com a minimização do tempo de processamento, e sim com um conjunto satisfatório de visões para atuar na implantação do *DW*.

E, por fim, a abordagem de Golfarelli, Maniezzo e Rizzi (2004) define o conjunto de fragmentos a partir do *DW* já populado, uma característica diferente da proposta deste trabalho. O módulo SVM faz a seleção das visões a materializar com base em dados estatísticos do banco operacional e a partir dos metadados do *DW*, armazenados na ferramenta durante o processo de modelagem dimensional da *WAVE*, ou seja, durante a fase de projeto.

No próximo capítulo é apresentada uma visão geral da nova versão da ferramenta e os resultados do experimento realizado.

6. EXPERIMENTO REALIZADO

Este capítulo tem por objetivo apresentar um experimento para a avaliação do funcionamento do módulo SVM, utilizando o contexto tradicional de vendas.

6.1. EXPERIMENTO: UM EXEMPLO DE VENDAS

Na realização deste experimento, foi levado em consideração o modelo lógico do contexto de vendas, apresentado na Figura 6.1 e um conjunto de consultas, definidas pelo projetista, apresentado na Tabela 4. As visões definidas pela ferramenta a partir das consultas definidas podem ser vistas na Tabela 5.

As etapas para geração do modelo dimensional, a exemplo da primeira consulta apresentada na Tabela 4, de acordo com o assistente de modelagem dimensional da *WAVE*, são exibidas nas Figuras 6.2, 6.3 e 6.4.

Os parâmetros adotados para uso deste experimento de vendas são: espaço disponível para projeto (100 MB), percentagem de crescimento (0%), benefício por frequência de uso (0.5), benefício por custo de materialização (0.5), número de bytes por página (4096), *status*=SIM para eliminar visões descendentes diretas. O resultado da seleção das visões a materializar, de acordo com o algoritmo do módulo SVM, assim como os passos que antecedem o resultado, são apresentados nas Figuras 6.5, 6.6, 6.7 e 6.8.

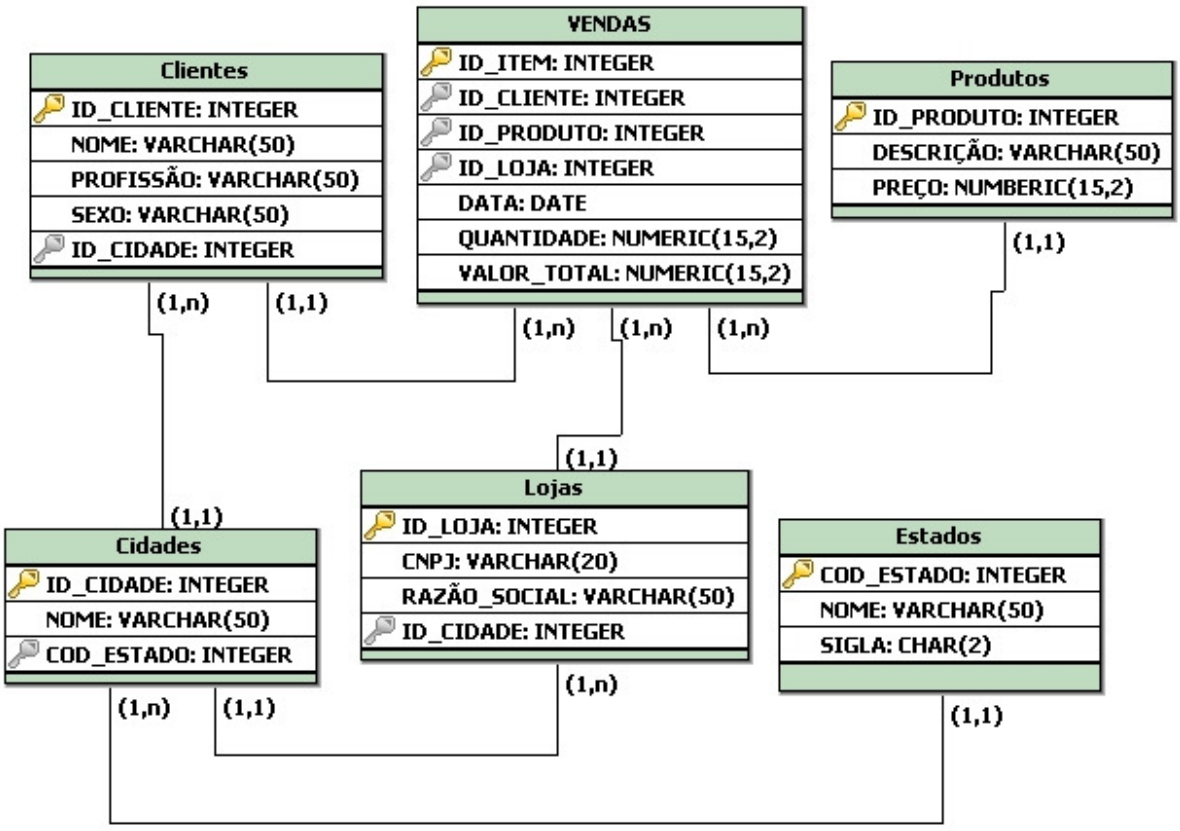


Figura 6.1 – Modelo Lógico de Vendas

Tabela 4 – Conjunto de Consultas Submetidas

1	Obter a média bimestral dos valores de vendas por loja e por produto
2	Obter a média da quantidade mensal de vendas trimestrais por loja e por cliente
3	Obter a média dos valores anuais de vendas separadas por sexo do cliente
4	Obter os valores mensais das vendas por loja e por produto
5	Obter a soma das quantidades diárias de vendas por loja e por produto
6	Obter a soma das vendas mensais por região
7	Obter a soma das vendas mensais por cliente e região
8	Obter a soma dos valores das vendas mensais por profissão do cliente e por região
9	Obter a quantidade trimestral das vendas por região
10	Obter a quantidade semanal das vendas por região
11	Obter o valor máximo semanal das vendas por região
12	Obter o valor mínimo semestral das vendas por loja
13	Obter os valores anuais de vendas por cliente e por região
14	Obter os valores anuais de vendas por cliente e por região (repetida)
15	Obter os valores bimestrais de vendas por produto e por região
16	Obter os valores diários de vendas por loja e por produto
17	Obter os valores mensais das vendas por cliente por produto e por região
18	Obter os clientes que em determinado mês realizaram a maior quantidade em compras

Tabela 5 – Conjunto de Visões Geradas

V 1	Tempo(Bimestre) itens_vendidos(VALOR_TOTAL) lojas(ID_LOJA,RAZAO_SOCIAL) produtos(ID_PRODUTO,DESCRICAO)
V 2	Tempo(Trimestre) itens_vendidos(ID_PRODUTO,QUANTIDADE) lojas(ID_LOJA,RAZAO_SOCIAL,CNPJ)
V 3	Tempo(Ano) itens_vendidos(VALOR_TOTAL) clientes(SEXO)
V 4	Tempo(Mês) itens_vendidos(VALOR_TOTAL) lojas(RAZAO_SOCIAL,CNPJ) produtos(DESCRICAO)
V 5	Tempo(Dia) itens_vendidos(QUANTIDADE) lojas(ID_LOJA,RAZAO_SOCIAL) produtos(DESCRICAO)
V 6	Tempo(Mês) itens_vendidos(ID_LOJA,VALOR_TOTAL) REGIÃO
V 7	Tempo(Mês) itens_vendidos(VALOR_TOTAL) clientes(ID_CLIENTE,NOME) REGIÃO
V 8	Tempo(Mês) itens_vendidos(VALOR_TOTAL) clientes(PROFISSAO) REGIÃO
V 9	Tempo(Trimestre) itens_vendidos(QUANTIDADE) REGIÃO
V 10	Tempo(Semana) itens_vendidos(QUANTIDADE) REGIÃO
V 11	Tempo(Semana) itens_vendidos(VALOR_TOTAL) REGIÃO
V 12	Tempo(Semestral) itens_vendidos(VALOR_TOTAL) lojas(ID_LOJA,CNPJ)
V 13	Tempo(Ano) itens_vendidos(VALOR_TOTAL) clientes(ID_CLIENTE,NOME) REGIÃO
V 14	Tempo(Ano) itens_vendidos(VALOR_TOTAL) clientes(ID_CLIENTE,NOME) REGIÃO
V 15	Tempo(Bimestre) itens_vendidos(VALOR_TOTAL) produtos(ID_PRODUTO,DESCRICAO) REGIÃO
V 16	Tempo(Dia) itens_vendidos(VALOR_TOTAL) lojas(ID_LOJA,RAZAO_SOCIAL,ID_CIDADE) produtos(ID_PRODUTO,DESCRICAO)
V 17	Tempo(Mês) itens_vendidos(VALOR_TOTAL) clientes(NOME) produtos(DESCRICAO) REGIÃO
V 18	Tempo(Mês) itens_vendidos(VALOR_TOTAL) clientes(ID_CLIENTE,NOME)

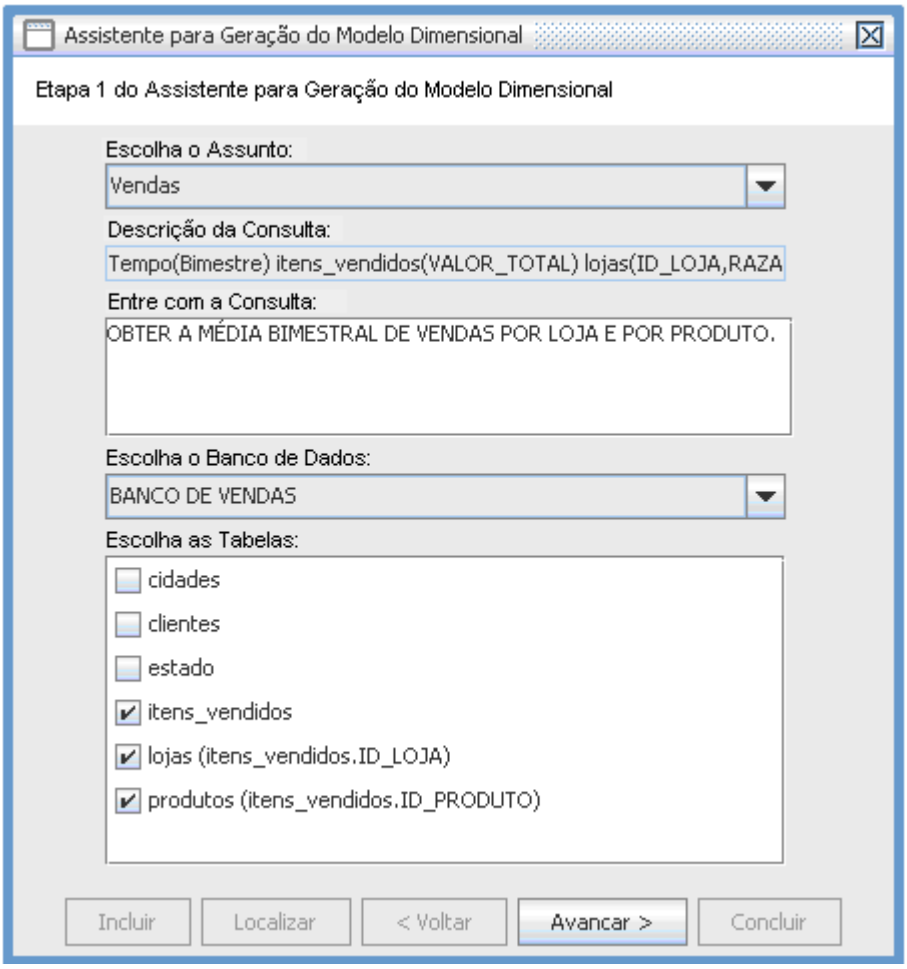


Figura 6.2 – Definição da Consulta e Tabelas

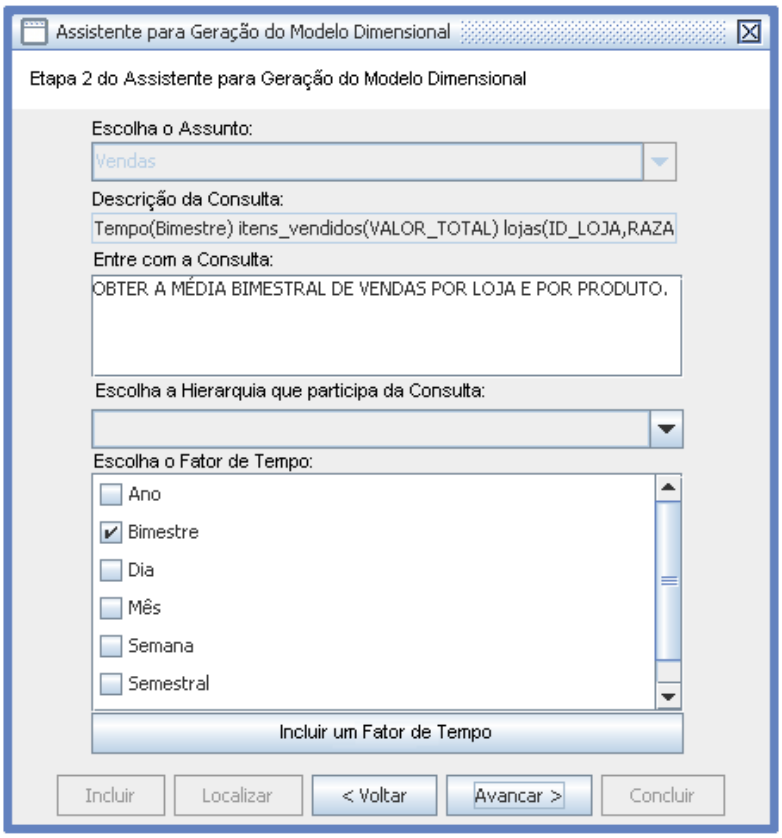


Figura 6.3 – Escolha da Hierarquia e Fator de Tempo

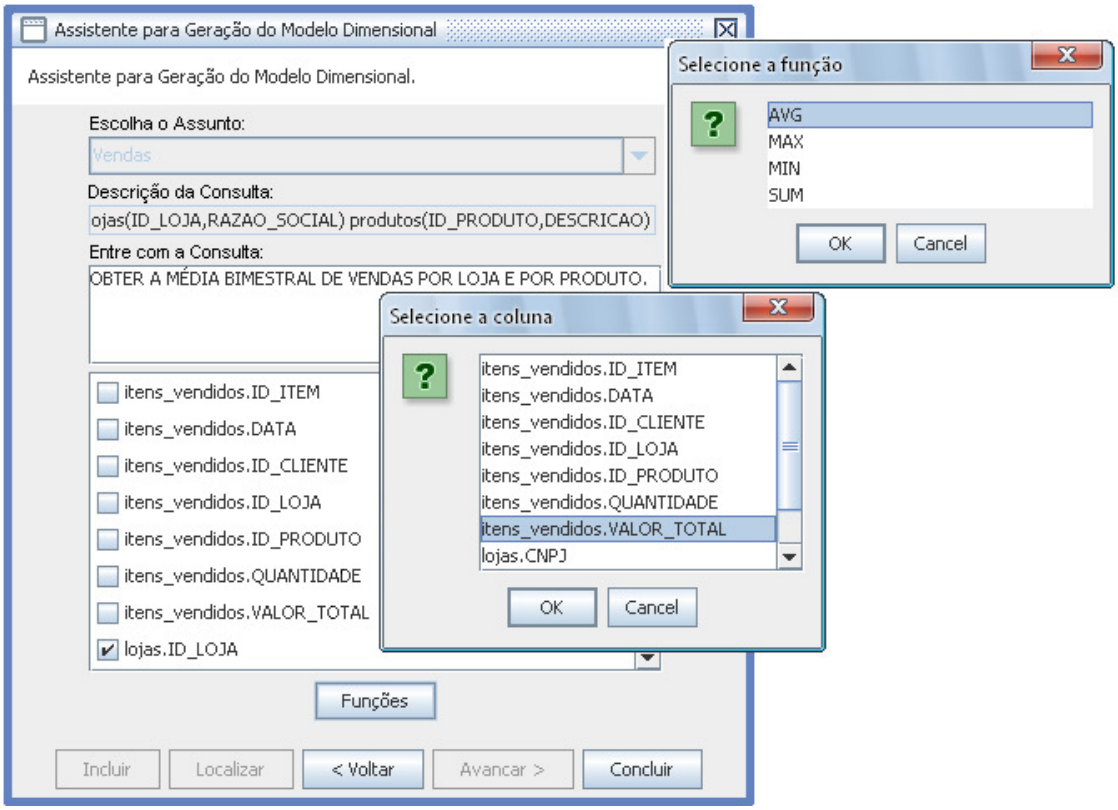


Figura 6.4 – Escolha das Colunas e Definição das Agregações

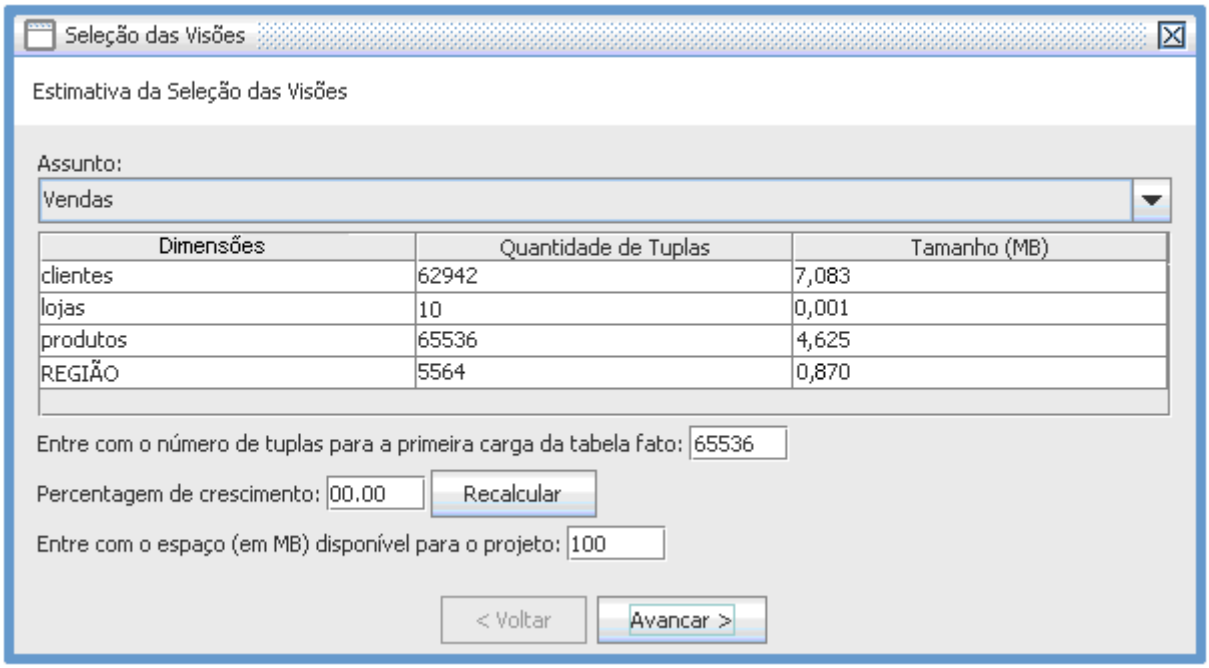


Figura 6.5 – Estimativas para a Seleção de Visões

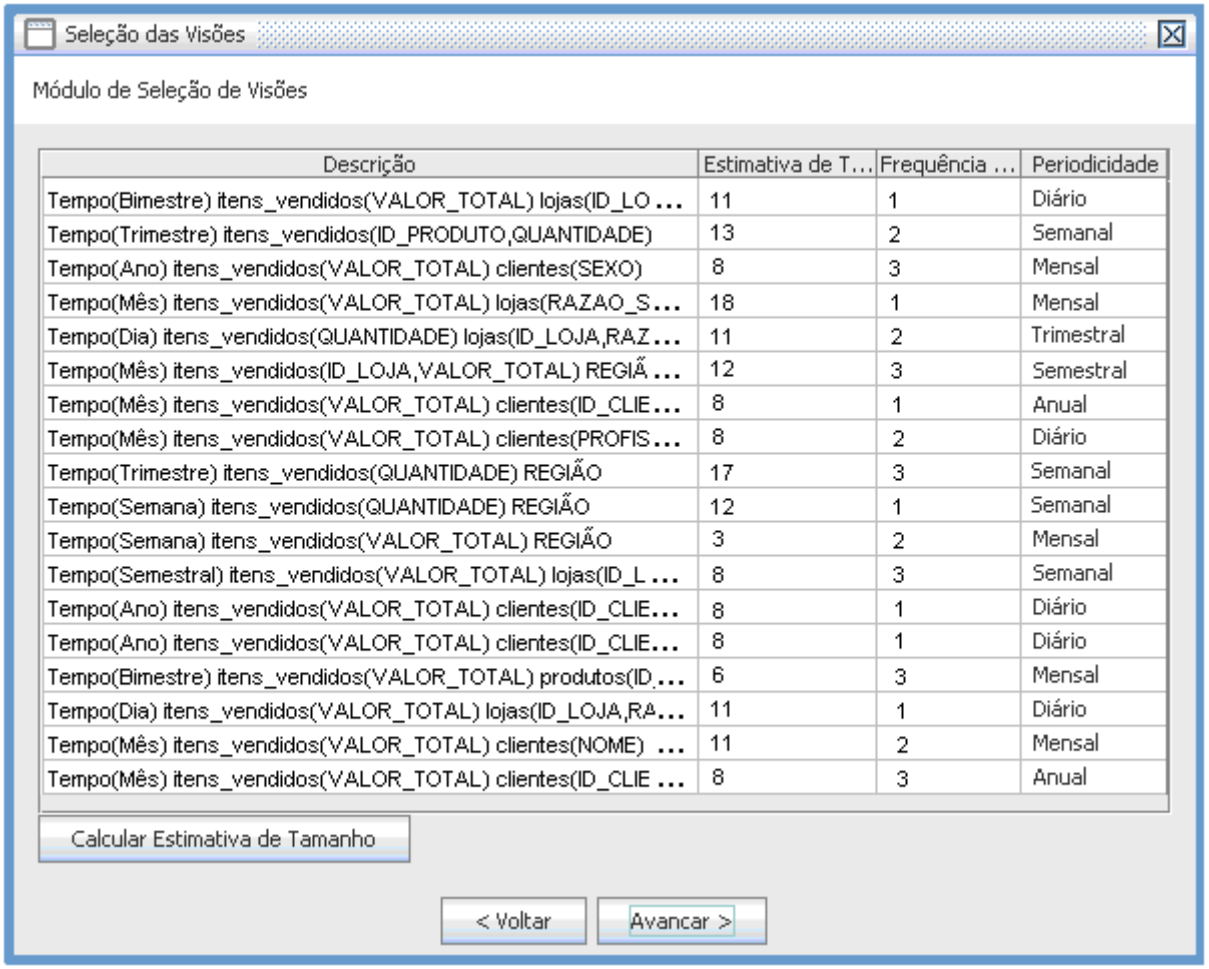


Figura 6.6 - Definição do Tamanho e Frequência de Uso das Visões

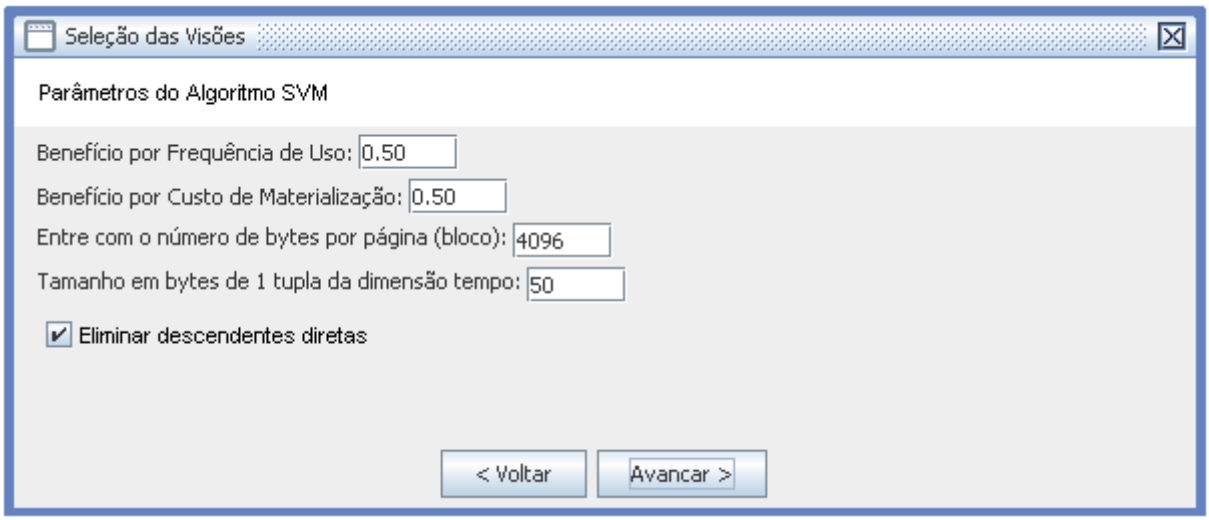


Figura 6.7 – Parâmetros do Algoritmo SVM

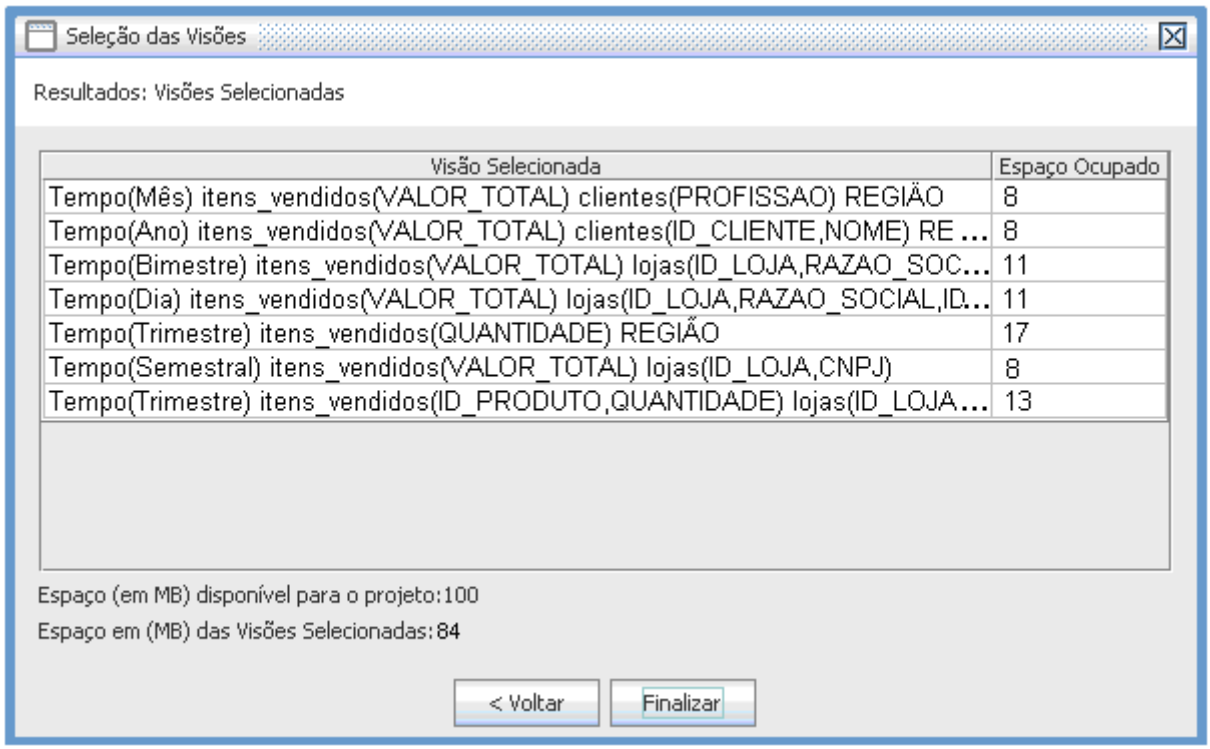


Figura 6.8 – Conjunto de Visões Selecionadas

O *SQL* das visões geradas pela ferramenta está disponível no Anexo 1, e o arquivo de log (Relatório de Configurações) disponível no Anexo 2.

Para finalizar o teste de vendas, foi implementado o *DW*, gerado de acordo com o relatório do modelo dimensional, vide Anexo 3, utilizando o SGBD Oracle 10g Express. A carga no *DW* foi realizada com dados fictícios, de acordo com

o contexto, cujos números de tuplas para cada tabela de dimensão e fato estão relacionadas na Tabela 6, totalizando aproximadamente 18 MB de tamanho.

Tabela 6 – Dados de Carga do *DW de* Vendas

Dimensão Clientes	62.942 tuplas	114 bytes por tupla
Dimensão Lojas	10 tuplas	118 bytes por tupla
Dimensão Produtos	65.536 tuplas	53 bytes por tupla
Dimensão Tempo	734 tuplas	50 bytes por tupla
Hierarquia Região	5.564 tuplas	94 bytes por tupla
Fato(Itens_Vendidos)	65.536 tuplas	43 bytes por tupla

6.2.1. RESULTADOS DO EXPERIMENTO

A partir da implementação do modelo dimensional (Anexo 3) e da carga de dados, de acordo com a Tabela 6, as consultas foram executadas no SGBD Oracle e medido o tempo de processamento de cada visão. A máquina utilizada para o experimento possui um processador Intel Core 2 Duo de 2GHz (Giga Hertz) com 2 GB (Giga Bytes) de memória *RAM*.

Em seguida são apresentados os resultados obtidos com relação ao custo, na sequência selecionada pelo algoritmo, para materialização das visões. O gráfico apresentado na Figura 6.9 é proveniente do conjunto de visões selecionadas para materialização e, o gráfico ilustrado na Figura 6.10, representa o resultado das visões não selecionadas para materialização.

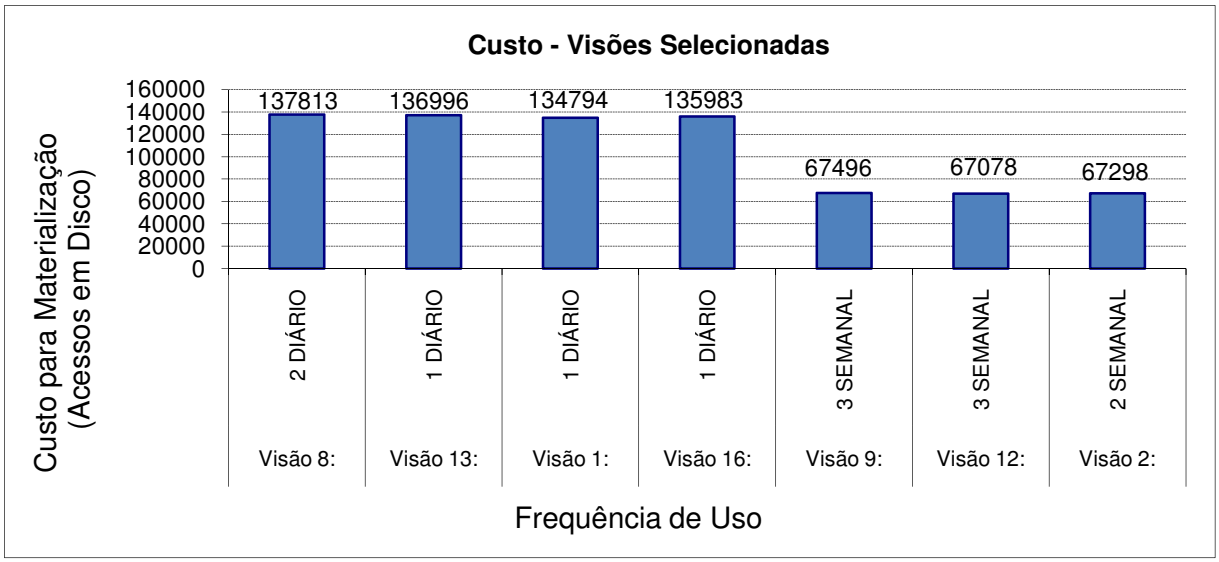


Figura 6.9 – Gráfico de Custo: Visões Selecionadas para Materialização

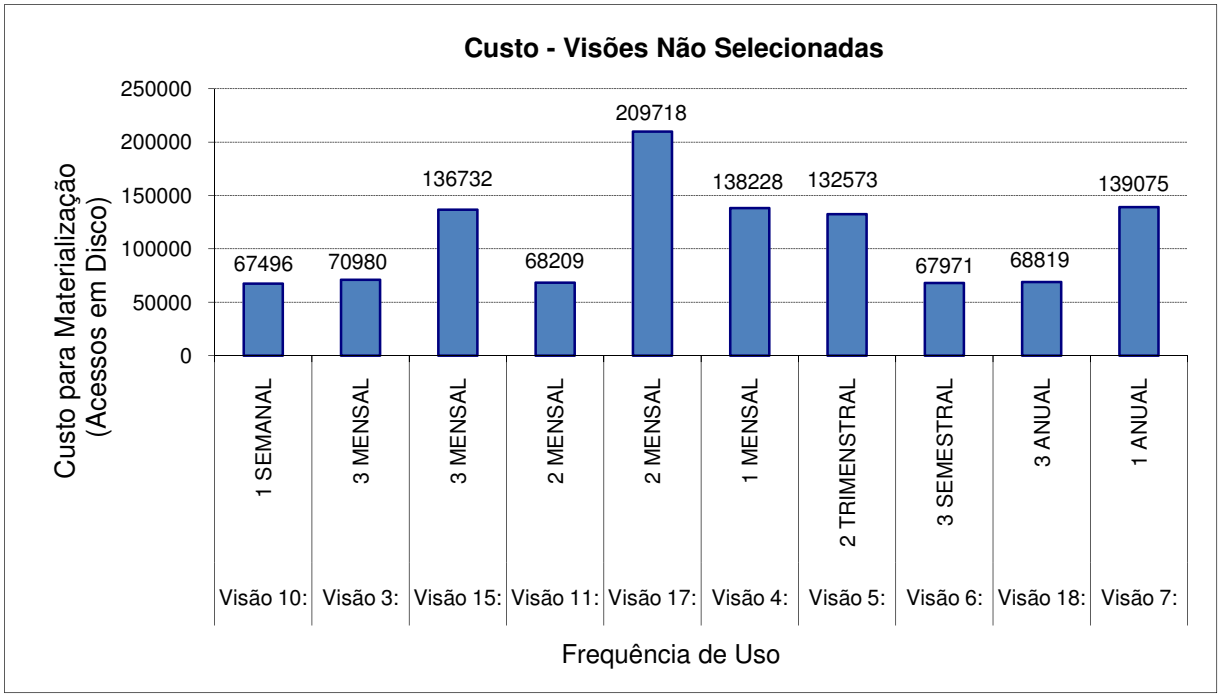


Figura 6.10 – Gráfico de Custo: Visões não Selecionadas

O espaço ocupado pelas visões, selecionadas e não selecionadas, pode ser visto nos gráficos das Figuras 6.11 e 6.12, respectivamente.

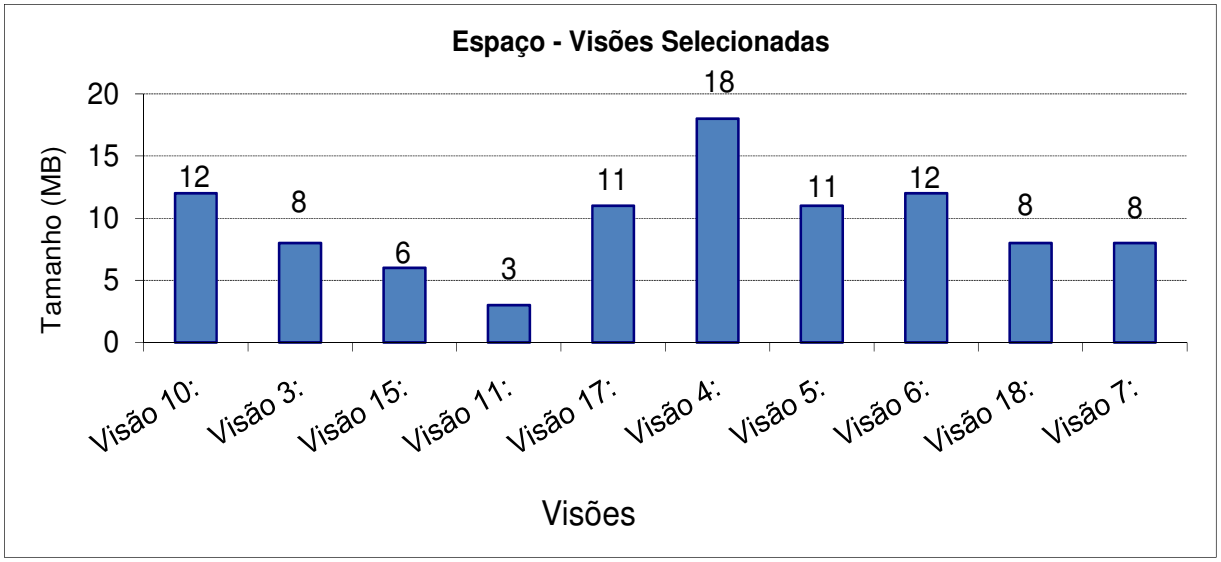


Figura 6.11 – Espaço Ocupado: Visões Selecionadas

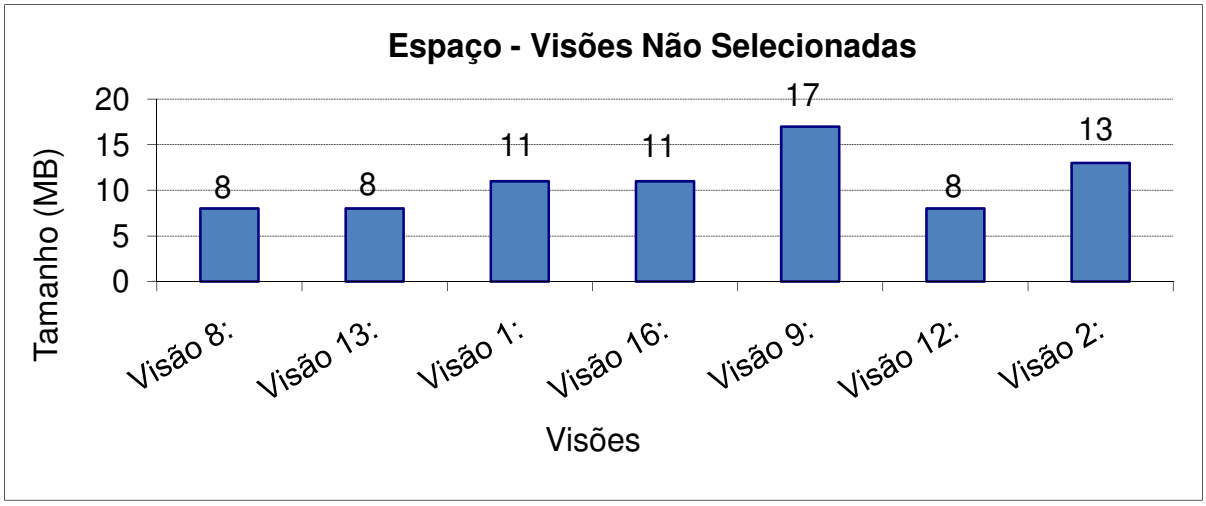


Figura 6.12 – Espaço Ocupado: Visões Não Selecionadas

6.3. DISCUSSÕES DOS RESULTADOS

De acordo com os resultados apresentados nos gráficos das Figuras 6.10 e 6.9, na seção anterior, tendo em vista as condições estabelecidas para a seleção das visões, o conjunto de visões selecionadas atende a este quesito, pois foram selecionadas, em função do espaço disponível, as visões com maior frequência de uso.

Para poder avaliar as medidas de custo efetuadas, foi medido o tempo real de processamento das consultas. Observou-se que os cálculos do custo de algumas visões apresentaram valores discrepantes. Esse comportamento pode ter ocorrido devido ao fato dos custos terem sido calculados com base no pior caso.

O espaço definido pelo projetista, como espaço disponível para projeto foi de 100 MB. Desse valor, 16 MB foram utilizados para armazenar o *DW*, restando assim 84 MB de espaço livre para a materialização das visões.

Por meio do gráfico de espaço ocupado das visões selecionadas, Figura 6.11, é possível observar que a soma dos tamanhos das visões representam 76 MB dos 84 MB disponíveis. Dessa forma, houve um aproveitamento de aproximadamente 90,47% do espaço reservado para as visões.

O gráfico, na Figura 6.12, apresenta os valores em espaço ocupado por cada visão não selecionada, obedecendo a ordem em que o algoritmo processou.

A avaliação do funcionamento do algoritmo, em linhas gerais, pode ser feita, neste experimento, por meio dos dados fornecidos pelo relatório de configurações, fornecido pela ferramenta e disponibilizado no Anexo 2.

É preciso refinar o cálculo de custo para que o mesmo apresente valores mais próximos dos valores reais de processamento das visões. São necessários também, testes com bancos de dados de diferentes aplicações, para uma análise mais abrangente dos resultados.

6.4. CONSIDERAÇÕES FINAIS

Neste capítulo foi apresentada uma visão geral do funcionamento da ferramenta *WAVE*, de suas principais telas, bem como os resultados e uma discussão do experimento realizado.

O próximo capítulo trata das conclusões e discute os trabalhos futuros.

7. CONCLUSÕES

Este trabalho apresentou a automatização de uma política para seleção de visões a materializar para *Data Warehouse*, aplicada na ferramenta *WAVE*, onde o projetista, de maneira interativa e em tempo de projeto obtém um conjunto de visões satisfatório para utilização na implantação do *Data Warehouse.*

7.1. CONTRIBUIÇÕES

A principal contribuição deste trabalho foi a definição de uma estratégia para a seleção de visões a materializar em tempo de projeto do *DW*. Essa estratégia levou à definição de um algoritmo para seleção de visões, conduzido pelo módulo de seleção de visões, e aplicado na ferramenta *WAVE*. O algoritmo é de fácil entendimento e aborda várias restrições discutidas na literatura.

O controle de hierarquias foi uma contribuição bastante significativa para a ferramenta, aplicada às etapas do assistente do modelo dimensional, pois as hierarquias são muito importantes para a concepção do *Data Warehouse*.

Os refinamentos realizados no processo de modelagem, o controle das hierarquias, e a reorganização das telas que correspondem às etapas de modelagem trouxeram, ao projetista, uma maior flexibilidade para trabalhar na geração do modelo dimensional.

A documentação fornecida pela ferramenta *WAVE* se tornou mais completa. O projetista tem melhores condições de análise dos resultados fornecidos pela ferramenta, assim como: o relatório dos modelos dimensionais gerados, relatórios por projeto de seleção de visões à materializar e também tem acesso ao SQL das visões selecionadas.

7.2. TRABALHOS FUTUROS

Durante o andamento do trabalho foram observados alguns pontos importantes que são sugeridos como trabalhos futuros:

- O amadurecimento do controle de hierarquias adicionando-se recursos gráficos para visualização e escolha dos atributos pertencentes a cada hierarquia;

- A geração do script *SQL* do modelo dimensional, escolhido pelo projetista, para ser implementado em um determinado SGBD (Oracle por exemplo). O estabelecimento da conexão com o SGBD e execução do script para a consolidação da estrutura física do modelo dimensional.

- Oferecer ao projetista a opção de dar primeira carga no *DW* projetado para avaliar o desempenho das visões selecionadas por meio do módulo SVM;

- Permissão da criação de novas possibilidades de periodicidade para a frequência de uso, no módulo SVM, proporcionando mais flexibilidade ao projetista.

Referências Bibliográficas

AGRAWAL ET AL. Database tuning advisor for microsoft sql server 2005. In: 30TH VERY LARGE DATA BASES (VLDB´04), Toronto, Canada, p. 1110-1121, 2004.

BELLATRECHE, L.; KARLAPALEM, K.; SCHNEIDER, M. On efficient storage space distribution among materialized views and indices in data warehousing environments. In: NINTH INTERNATIONAL CONFERENCE ON INFORMATION AND KNOWLEDGE MANAGEMENT (CIKM 2000), Mclean, Virginia, United States, p. 397-404, 2000.

BOUKRA, A.; NACER, M. A.; BOUROUBI, S. Selection of views to materialized in data warehouse: a hybrid solution. **International Journal of Computational Intelligence Research**, v.3, n.4, p. 327-334, 2007.

CHAN, G; LI, Q; FENG, L. **Design and Selection of Materialized Views in a Data Warehousing Environment:** A Case Study. IN: 2ND ACM INTERNATIONAL WORKSHOP ON DATA WAREHOUSING AND OLAP - DOLAP 99, Missouri, United States, p.42-47, 1999.

ELMASRI, R.; NAVATHE, S.B. **Sistemas de banco de dados**. 4.ed. São Paulo: Pearson Education do Brasil, 2005.

FORLANI, D. T.; CIFERRI, C.D.A.; CIFERI, R.R. Melhorando o desempenho do processamento de consultas drill-across em ambientes de data warehousing. In: XXI SIMPÓSIO BRASILEIRO DE BANCO DE DADOS, 2006, Florianópolis – Santa Catarina. Anais do XXI Simpósio Brasileiro de Banco de Dados, 2006, p. 161-175.

GALLAS, S. Kimball Vs. Inmon. **Data Management Review.** Disponível em: <http://www.dmreview.com> Acesso em: 20 dezembro 2007.

GOLFARELLI, M.; MAIO, D.; RIZZI, S. Applying vertical fragmentation techniques

in logical design of multidimensional databases. In: 2ND INTERNATIONAL CONFERENCE ON DATA WAREHOUSING AND KNOWLEDGE DISCOVERY, DaKak 2000, Greenwich, UK, p. 11-23, 2000.

GOLFARELLI, M.; MAIO, D.; RIZZI, S. Conceptual Design of Data Warehouses from E/R Schemes. In: 31st Hawaii International Conference on System Sciences (HICSS-31), p. 334-343, 1998.

GOLFARELLI, M.; MANIEZZO, V.; RIZZI, S. Materialization of fragmented views in multidimensional databases. **Data & Knowledge Engineering**, v.49, p.325-351, Junho. 2004.

GOLFARELLI, M.; RIZZI, S.; SALTARELLI, E. WanD: a case tool for workload-based design of a data mart. IN DECIMO CONVEGNO NATIONALE SU SISTEMI EVOLUTI PER BASI DI DATI, Portoferraio, Italy, p. 442-426, 2002.

GOU, G., YU, J. and LU, H. A* Search: An Efficient and Flexible Approach to Materialized View Selection. **IEEE Transaction on System, Man, and Cybernetics -Part C: Applications and Reviews**, v. 36, n. 3, p. 411- 424, Maio.2006.

GUPTA, H. Selection of Views to Materialize in a Data Warehouse. In: 6TH INTERNATIONAL CONFERENCE ON DATABASE THEORY (ICDT´97), p.98-112, 1997.

GUPTA, H.; MUMICK, I.S. Selection of views to materialize in a data warehouse. **IEEE Transactions on Knowledge and Data Engineering**, v.17, n.1, p.24-43, Janeiro. 2005.

HARINARAYAN, V.; RAJARAMAN, A.; ULLMAN, J. D. Implementing Data cubes Efficiently. In: ACM SIGMOD INTERNATIONAL CONFERENCE OF MANAGEMENT OF DATA, Montreal, Canada, p. 205 -216, 1996.

HOBBS, L. **Oracle materialized views & query rewrite**. Disponível em: http://www.oracle.com Acesso em: 20 de novembro de 2008.

INMON, W. H. **Como construir o Data Warehouse**. Rio de Janeiro: Campus, 1997.

JUKIC, N. Modeling Strategies and Alternatives for Data Warehousing Projects. **Communications of the ACM,** v.49, n. 4, p.83-88, Abril 2006.

KIMBALL, R. **Data warehouse toolkit:** técnicas para construção de data warehouses dimensionais, São Paulo: Makron Books, 1998.

KIMBALL, R; ROSS, M. The Data Warehouse Toolkit: Guia completo para modelagem dimensional, Rio de Janeiro: Campus, 2002.

KOTIDIS, Y.; ROUSSOPOULOS, N. A Case for Dynamic View Management. **ACM Transactions on Database Systems**, v.26, n. 4, p. 388-423, Dezembro 2001.

LIMA, L. G. **Ferramenta para Geração do Modelo Dimensional para Data Warehouses**, 2006, 87f. Dissertação (Programa de Pós Graduação em Ciência da Computação) – Faculdade de Ciências Exatas e da Natureza, Universidade Metodista de Piracicaba, Piracicaba, 2006.

LIMA, L.; VIEIRA, M. Ferramenta para Geração de Modelo Dimensional para Data Warehouses. In: XXI SIMPÓSIO BRASILEIRO DE BANCO DE DADOS, Florianópolis – Santa Catarina. 2006, p.147-, 2006. Anais do XXI Simpósio Brasileiro de Banco de Dados, 2006, p. 147-175.

MACHADO, F. N. R. **Projeto de data warehouse:** Uma Visão Multidimensional. São Paulo: Érica, 2000.

MARTINS F., BELO O., NOVAIS P. Avaliação de Algoritmos para a Selecção de Vistas Materializadas em Ambientes de Data Warehousing. In: 1º CONGRESO ESPAÑOL DE INFORMÁTICA (JISBD´2005), Granada, Espanha, p. 75-88, 2005.

SILBERSCHATZ, A.; KORTH, H.; SUDARSHAN, S. **Sistema de banco de dados**. 3.ed. São Paulo: Makron Books, 1999.

SINGH, H. **Data Warehouse:** concepts, technologies, implementations and management. Saddle River: Prentice Hall PTR, 1998.

SQL (1999). International Organization for Standardization (ISO) & American National Standards Institute (ANSI) - ISO/IEC JTC1/SC32 - ANSI ISO/IEC 9075-2:1999. ISO International Standard. Database Language - SQL - Parte 2: Foundation (SQLFoundation), 1999.

THEODORATOS, D; SIMITSIS, A. **Materialized View Selection for Data Warehouse Design**. In John Wang. Encyclopedia of Data Warehousing and Mining, Montclair State University, USA: Idea Group, 2006. P.717-720, v.1.

VALLURI, S; VADAPALLI, S; KARLAPALEM, K. View Relevance Driven Materialized Selection in Data Warehousing Environment. In: Thirteenth Australasian Database Conference (ADC´2002), Melbourne, Australia, v.5, 2002.

YOUSRI, N., AHMED, K. AND MAKKY, N. Algorithms for Selecting Materialized Views in a Data Warehouse. In: 3RD ACS/IEEE INTERNATIONAL CONFERENCE ON COMPUTER SYSTEMS AND APPLICATIONS, p.27-34, 2005.

ZHANG, C.; YAO, X.; YANG, J. Evolving materialized views in data warehouse. In CONGRESS ON EVOLUTIONARY COMPUTATION (CEC´99), Canberra, ACT, p. 823-829, 1999.

ZHOU, J. ET AL. Dynamic Materialized Views, In: IEEE 23RD INTERNATIONAL

CONFERENCE ON DATA ENGINEERING (ICDE 2007), Istanbul, Turkey, p.526-535, 2007.

ZILIO, D ET AL. Recommending Materialized Views and Indexes with IBM DB2 Design Advisor. In: FIRST INTERNATIONAL CONFERENCE ON AUTONOMIC COMPUTING (ICAC´04), p.180-188, 2004.

ANEXOS

ANEXO 1 – SQL GERADO PELA FERRAMENTA - WAVE

```
CREATE MATERIALIZED VIEW "DW"."VENDAS_VISAO1"  AS
  SELECT TEMPO.BIMESTRE,
    AVG(ITENS_VENDIDOS.VALOR_TOTAL) VALOR_TOTAL,
    LOJAS.ID_LOJA ID_LOJA,
    LOJAS.RAZAO_SOCIAL RAZAO_SOCIAL,
    PRODUTOS.ID_PRODUTO ID_PRODUTO,
    PRODUTOS.DESCRICAO DESCRICAO
  FROM  TEMPO, ITENS_VENDIDOS,
       LOJAS,
       PRODUTOS
  WHERE ITENS_VENDIDOS.PK_TEMPO = TEMPO.PK_TEMPO AND
        LOJAS.ID_LOJA = ITENS_VENDIDOS.PK_LOJAS AND
        PRODUTOS.ID_PRODUTO = ITENS_VENDIDOS.PK_PRODUTOS
  GROUP BY TEMPO.BIMESTRE, LOJAS.ID_LOJA, LOJAS.RAZAO_SOCIAL,
PRODUTOS.ID_PRODUTO, PRODUTOS.DESCRICAO;
```

```
CREATE MATERIALIZED VIEW "DW"."VENDAS_VISAO2"  AS
  SELECT TEMPO.TRIMESTRE,
    ITENS_VENDIDOS.PK_PRODUTOS PK_PRODUTO,
    AVG(ITENS_VENDIDOS.QUANTIDADE) QUANTIDADE,
    LOJAS.ID_LOJA ID_LOJA,
    LOJAS.RAZAO_SOCIAL RAZAO_SOCIAL,
    LOJAS.CNPJ CNPJ
  FROM  TEMPO, ITENS_VENDIDOS,
       LOJAS,
       CLIENTES
  WHERE ITENS_VENDIDOS.PK_TEMPO = TEMPO.PK_TEMPO AND
        LOJAS.ID_LOJA = ITENS_VENDIDOS.PK_LOJAS AND
        CLIENTES.ID_CLIENTE = ITENS_VENDIDOS.PK_CLIENTES
  GROUP BY TEMPO.TRIMESTRE, ITENS_VENDIDOS.PK_PRODUTOS, LOJAS.ID_LOJA,
LOJAS.RAZAO_SOCIAL, LOJAS.CNPJ;
```

```
CREATE MATERIALIZED VIEW "DW"."VENDAS_VISAO8"
  AS SELECT
    TEMPO.MES,
    SUM(ITENS_VENDIDOS.VALOR_TOTAL) VALOR_TOTAL,
    CLIENTES.PROFISSAO PROFISSAO,
    REGIAO.SIGLA,
    REGIAO.NOME,
    REGIAO.ID_CIDADE
FROM
    TEMPO,
    ITENS_VENDIDOS,
    CLIENTES,
    REGIAO
WHERE
    ITENS_VENDIDOS.PK_TEMPO = TEMPO.PK_TEMPO AND CLIENTES.ID_CLIENTE =
ITENS_VENDIDOS.PK_CLIENTES AND REGIAO.PK_REGIAO = ITENS_VENDIDOS.PK_REGIAO
GROUP BY
    TEMPO.MES,
    CLIENTES.PROFISSAO,
```

```
   REGIAO.SIGLA,
   REGIAO.NOME,
   REGIAO.NOME,
   REGIAO.ID_CIDADE;
```

```
CREATE MATERIALIZED VIEW "DW"."VENDAS_VISAO9"  AS
  SELECT TEMPO.TRIMESTRE,
    ITENS_VENDIDOS.QUANTIDADE QUANTIDADE,
    REGIAO.SIGLA,
    REGIAO.NOME,
    REGIAO.ID_CIDADE
  FROM TEMPO,
       ITENS_VENDIDOS,
       LOJAS,
       CLIENTES,
       REGIAO
  WHERE ITENS_VENDIDOS.PK_TEMPO = TEMPO.PK_TEMPO AND
        LOJAS.ID_LOJA = ITENS_VENDIDOS.PK_LOJAS AND
        CLIENTES.ID_CLIENTE = ITENS_VENDIDOS.PK_CLIENTES AND
        REGIAO.PK_REGIAO = ITENS_VENDIDOS.PK_REGIAO
  GROUP BY TEMPO.TRIMESTRE, ITENS_VENDIDOS.QUANTIDADE, REGIAO.SIGLA,
REGIAO.NOME, REGIAO.NOME, REGIAO.ID_CIDADE;
```

```
CREATE MATERIALIZED VIEW "DW"."VENDAS_VISAO12"  AS
 SELECT
    TEMPO.SEMESTRE,
    MIN(ITENS_VENDIDOS.VALOR_TOTAL) VALOR_TOTAL,
    LOJAS.ID_LOJA ID_LOJA,
    LOJAS.CNPJ CNPJ
FROM
    TEMPO,
    ITENS_VENDIDOS,
    LOJAS
WHERE
    ITENS_VENDIDOS.PK_TEMPO = TEMPO.PK_TEMPO AND LOJAS.ID_LOJA =
ITENS_VENDIDOS.PK_LOJAS
GROUP BY
    TEMPO.SEMESTRE,
    LOJAS.ID_LOJA,
    LOJAS.CNPJ;
```

```
 CREATE MATERIALIZED VIEW "DW"."VENDAS_VISAO13"  AS
  SELECT TEMPO.ANO,
    ITENS_VENDIDOS.VALOR_TOTAL VALOR_TOTAL,
    CLIENTES.ID_CLIENTE ID_CLIENTE,
    CLIENTES.NOME NOME_CLIENTES,
    REGIAO.SIGLA,
    REGIAO.NOME,
    REGIAO.ID_CIDADE
  FROM  TEMPO, ITENS_VENDIDOS,
       CLIENTES,
       REGIAO
  WHERE ITENS_VENDIDOS.PK_TEMPO = TEMPO.PK_TEMPO AND
        CLIENTES.ID_CLIENTE = ITENS_VENDIDOS.PK_CLIENTES AND
        REGIAO.PK_REGIAO = ITENS_VENDIDOS.PK_REGIAO
  GROUP BY TEMPO.ANO, CLIENTES.ID_CLIENTE, CLIENTES.NOME,
ITENS_VENDIDOS.VALOR_TOTAL, REGIAO.SIGLA, REGIAO.NOME, REGIAO.ID_CIDADE;
```

```
CREATE MATERIALIZED VIEW "DW"."VENDAS_VISAO16"  AS
  SELECT TEMPO.DIA,
    ITENS_VENDIDOS.VALOR_TOTAL VALOR_TOTAL,
    LOJAS.ID_LOJA ID_LOJA,
    LOJAS.RAZAO_SOCIAL RAZAO_SOCIAL,
    PRODUTOS.ID_PRODUTO ID_PRODUTO,
    PRODUTOS.DESCRICAO DESCRICAO
  FROM  TEMPO, ITENS_VENDIDOS,
      LOJAS,PRODUTOS
  WHERE ITENS_VENDIDOS.PK_TEMPO = TEMPO.PK_TEMPO AND
       LOJAS.ID_LOJA = ITENS_VENDIDOS.PK_LOJAS AND
       PRODUTOS.ID_PRODUTO = ITENS_VENDIDOS.PK_PRODUTOS
  GROUP BY TEMPO.DIA, ITENS_VENDIDOS.VALOR_TOTAL, LOJAS.ID_LOJA,
LOJAS.RAZAO_SOCIAL, PRODUTOS.ID_PRODUTO, PRODUTOS.DESCRICAO;
```

ANEXO 2 - RELATÓRIO DE CONFIGURAÇÕES(LOG)

Projeto: Projeto_Vendas

Espaço disponível ao projeto: 100MB
Espaço total ocupado: 92MB
Tamanho do DW: 16MB
Benefício por Frequência de Uso: 0.50
Benefício por Espaço Ocupado: 0.50
Número de Bytes por Página (Bloco): 4096
Eliminar descendentes diretas: S
Número de Tuplas da Tabela Fato: 65536
Tamanho da Tabela Fato: 3,5MB

Dimensões

Nome	Tuplas	Tamanho
clientes	62942	7.083
lojas	10	0.001
produtos	65536	4.625
Região	5564	0.870

Obs.: Para valores mostrados em "Acessos Disco" são considerados o valor do custo, os acessos a índices (A) e o custo das operações Group by (X).

Descrição das Visões

Visão 8: Tempo(Mês) itens_vendidos(VALOR_TOTAL) clientes(PROFISSAO) REGIÃO

Tam.	Freq. Uso	Period.	Acessos Disco	vs_inicial	vc_inicial	vs	vc	vt	Mot. Elimin.
8	2	Diário	137813 + A + X	1		1			

Visão 13: Tempo(Ano) itens_vendidos(VALOR_TOTAL) clientes(ID_CLIENTE,NOME) REGIÃO

Tam.	Freq. Uso	Period.	Acessos Disco	vs_inicial	vc_inicial	vs	vc	vt	Mot. Elimin.
8	1	Diário	136996 +A + X	2		2			Igual

Visão 1: Tempo(Bimestre) itens_vendidos(VALOR_TOTAL) lojas(ID_LOJA,RAZAO_SOCIAL) produtos(ID_PRODUTO,DESCRICAO)

Tam.	Freq. Uso	Period.	Acessos Disco	vs_inicial	vc_inicial	vs	vc	vt	Mot. Elimin.
11	1	Diário	134794 +A + X	3		3			

Visão 16: Tempo(Dia) itens_vendidos(VALOR_TOTAL) lojas(ID_LOJA,RAZAO_SOCIAL,ID_CIDADE) produtos(ID_PRODUTO,DESCRICAO)

Tam.	Freq. Uso	Period.	Acessos Disco	vs_inicial	vc_inicial	vs	vc	vt	Mot. Elimin.
11	1	Diário	135983 +A + X	4		4			

Visão 9: Tempo(Trimestre) itens_vendidos(QUANTIDADE) REGIÃO

Tam.	Freq. Uso	Period.	Acessos Disco	vs_inicial	vc_inicial	vs	vc	vt	Mot. Elimin.
17	3	Semanal	67496 + A + X	5		5			

Visão 12: Tempo(Semestral) itens_vendidos(VALOR_TOTAL) lojas(ID_LOJA,CNPJ)

Tam.	Freq. Uso	Period.	Acessos Disco	vs_inicial	vc_inicial	vs	vc	vt	Mot. Elimin.
8	3	Semanal	67078 + A+ X	6		6			

Visão 2: Tempo(Trimestre) itens_vendidos(ID_PRODUTO,QUANTIDADE) lojas(ID_LOJA,RAZAO_SOCIAL,CNPJ)

Tam.	Freq. Uso	Period.	Acessos Disco	vs_inicial	vc_inicial	vs	vc	vt	Mot. Elimin.
13	2	Semanal	67298 + A + X	7		7			

Visão 10: Tempo(Semana) itens_vendidos(QUANTIDADE) REGIÃO

Tam.	Freq. Uso	Period.	Acessos Disco	vs_inicial	vc_inicial	vs	vc	vt	Mot. Elimin.
12	1	Semanal	67496 + A +X		1		1		

Visão 3: Tempo(Ano) itens_vendidos(VALOR_TOTAL) clientes(SEXO)

Tam.	Freq.	Uso	Period.	Acessos Disco	vs_inicial	vc_inicial	vs	vc	vt	Mot. Elimin.
8	3	Mensal		70980 + A + X	2	2				

Visão 15: Tempo(Bimestre) itens_vendidos(VALOR_TOTAL) produtos(ID_PRODUTO,DESCRICAO) REGIÃO

Tam.	Freq.	Uso	Period.	Acessos Disco	vs_inicial	vc_inicial	vs	vc	vt	Mot. Elimin.
6	3	Mensal		136732 + A + X	3	3				

Visão 11: Tempo(Semana) itens_vendidos(VALOR_TOTAL) REGIÃO

Tam.	Freq.	Uso	Period.	Acessos Disco	vs_inicial	vc_inicial	vs	vc	vt	Mot. Elimin.
3	2	Mensal		68209 + A + X	4	4				

Visão 17: Tempo(Mês) itens_vendidos(VALOR_TOTAL) clientes(NOME) produtos(DESCRICAO) REGIÃO

Tam.	Freq.	Uso	Period.	Acessos Disco	vs_inicial	vc_inicial	vs	vc	vt	Mot. Elimin.
11	2	Mensal		209718 + A + X	5	5				

Visão 4: Tempo(Mês) itens_vendidos(VALOR_TOTAL) lojas(RAZAO_SOCIAL,CNPJ) produtos(DESCRICAO) regiao

Tam.	Freq.	Uso	Period.	Acessos Disco	vs_inicial	vc_inicial	vs	vc	vt	Mot. Elimin.
18	1	Mensal		138228 + A + X	6	6				

Visão 5: Tempo(Dia) itens_vendidos(QUANTIDADE) lojas(ID_LOJA,RAZAO_SOCIAL) produtos(DESCRICAO)

Tam.	Freq.	Uso	Period.	Acessos Disco	vs_inicial	vc_inicial	vs	vc	vt	Mot. Elimin.
11	2	Trimenstral		132573 + A + X	7	7				

Visão 6: Tempo(Mês) itens_vendidos(ID_LOJA,VALOR_TOTAL) REGIÃO

Tam.	Freq.	Uso	Period.	Acessos Disco	vs_inicial	vc_inicial	vs	vc	vt	Mot. Elimin.
12	3	Semestral		67971 + A + X	8	8				

Visão 18: Tempo(Mês) itens_vendidos(VALOR_TOTAL) clientes(ID_CLIENTE,NOME)

Tam.	Freq.	Uso	Period.	Acessos Disco	vs_inicial	vc_inicial	vs	vc	vt	Mot. Elimin.
8	3	Anual		68819 + A + X	9	9				

Visão 7: Tempo(Mês) itens_vendidos(VALOR_TOTAL) clientes(ID_CLIENTE,NOME) REGIÃO

Tam.	Freq.	Uso	Period.	Acessos Disco	vs_inicial	vc_inicial	vs	vc	vt	Mot. Elimin.
8	1	Anual		139075 + A + X	10	10				

Visão 14: Tempo(Ano) itens_vendidos(VALOR_TOTAL) clientes(ID_CLIENTE,NOME) REGIÃO

Tam.	Freq.	Uso	Period.	vs_inicial	vc_inicial	vs	vc	vt	Mot. Elimin.
8	1	Diário							Igual

Assunto: Vendas

Dados do Modelo

Esquema: ESTRELA Data/Hora de Geração: 10/01/2009 15:26:13

Consultas

Obter a média bimestral dos valores de vendas por loja e por produto
Obter a média da quantidade mensal de vendas trimestrais por loja e por cliente
Obter a média dos valores anuais de vendas separadas por sexo do cliente
Obter os valores mensais das vendas por loja e por produto
Obter a soma das quantidades diárias de vendas por loja e por produto
Obter a soma das vendas mensais por região
Obter a soma das vendas mensais por cliente e região
Obter a soma dos valores das vendas mensais por profissão do cliente e por região
Obter a quantidade trimestral das vendas por região
Obter a quantidade semanal das vendas por região
Obter o valor máximo semanal das vendas por região
Obter o valor mínimo semestral das vendas por loja
Obter os valores anuais de vendas por cliente e por região
Obter os valores anuais de vendas por cliente e por região (repetida)
Obter os valores bimestrais de vendas por produto e por regiao
Obter os valores diários de vendas por loja e por produto
Obter os valores mensais das vendas por cliente por produto e por região
Obter os clientes que em determinado mês realizaram a maior quantidade em compras

Modelo

Fato (itens_vendidos)

Coluna	Descrição	Tipo
PK_REGIÃO		FK
PK_TEMPO		FK
PK_lojas		FK
PK_clientes		FK
PK_produtos		FK
QUANTIDADE		
VALOR_TOTAL		

Dimensão (clientes)

Coluna	Descrição	Tipo
PK_clientes	CHAVE PRIMÁRIA	PK
NOME		
SEXO		
PROFISSAO		
ID_CLIENTE		

Dimensão (lojas)

Coluna	Descrição	Tipo
PK_lojas	CHAVE PRIMÁRIA	PK
CNPJ		
ID_LOJA		
RAZAO_SOCIAL		

Dimensão (produtos)

Coluna	Descrição	Tipo
PK_produtos	CHAVE PRIMÁRIA	PK
DESCRICAO		
ID_PRODUTO		

Hierarquia (REGIÃO)

Coluna	Descrição	Tipo
PK_REGIÃO	CHAVE PRIMÁRIA	PK
NOME		
SIGLA		
ID_CIDADE		
ID_ESTADO		

(DIMENSÃO TEMPO)

Coluna	Descrição	Tipo
PK_TEMPO	CHAVE PRIMÁRIA	PK
Semana	Semana	
Bimestre	Bimestre	
Dia	Dia	
Ano	Ano	
Mês	Mês	
Semestral	Semestral	
Trimestre	Trimestre	