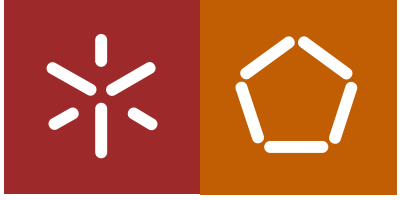
Universidade do Minho
Escola de Engenharia

Ana Sofia da Silva Duarte

Materialização “à Medida” de Vistas Multidimensionais de Dados

Outubro de 2011

UMinho | 2011 Ana Sofia da Silva Duarte Materialização “à Medida” de Vistas Multidimensionais de Dados



Universidade do Minho
Escola de Engenharia

Ana Sofia da Silva Duarte

Materialização “à Medida” de Vistas Multidimensionais de Dados

Dissertação de Mestrado
Mestrado em Informática

Trabalho efectuado sob a orientação do
Professor Doutor Orlando Manuel de Oliveira Belo

Outubro de 2011

Universidade do Minho, ___/___/_____

Assinatura: _____

À minha mãe, às minhas irmãs e em especial ao meu pai.

Agradecimentos

No percurso deste trabalho muitas foram as conquistas mas também muitas foram as dificuldades. Para que fosse possível, com sucesso, concluir este percurso de aprendizagem, académica e até mesmo pessoal, muitas foram as pessoas que estiveram ao meu lado a apoiar-me e a elas quero expressar os meus sinceros agradecimentos. Sem querer desfazer de qualquer pessoa que me apoiou neste percurso, e desde já muito obrigada a todos os que estiveram ao meu lado, gostava sem qualquer ordem de importância de salientar em particular:

- O meu orientador Professor Doutor Orlando Manuel Belo, pelas críticas, sugestões e correcções ao longo deste trabalho além da confiança transmitida.

- A Alice Monteiro Marques, pela disponibilização do seu trabalho de mestrado que serviu de apoio nesta tese e pelas horas passadas a discutir resoluções de problemas que muito ajudaram nas avaliações das soluções possíveis.

- A Helena Correia, a Maria Manuel Castro e o Pedro Carvalho pelo tempo despendido, o apoio prestado e pela oportunidade cedida.

- A Sophia Pereira Martins, o André Rafael Fernandes e o João Pedro Ferreira pelo apoio incondicional em todas as horas e por sempre acreditarem em mim.

- A Maria Clarisse Duarte pelo apoio e pela ajuda na revisão do documento.

- À minha família pelas oportunidades que me propuseram, pelo apoio e compreensão.

- A todos os colegas e amigos que sempre me apoiaram, incentivaram e até mesmo descontraíram nas horas boas e nas mais difíceis.

Resumo

Materialização "à medida" de vistas multidimensionais de dados

Com o emergir da era da informação foram muitas as empresas que recorreram a *data warehouses* para armazenar a crescente quantidade de dados que dispõem sobre os seus negócios. Com essa evolução dos volumes de dados surge também a necessidade da sua melhor exploração para que sejam úteis de alguma forma nas avaliações e decisões sobre o negócio. Os sistemas de processamento analítico (ou OLAP – *On-Line Analytical Processing*) vêm dar resposta a essas necessidades de auxiliar o analista de negócio na exploração e avaliação dos dados, dotando-o de autonomia de exploração, disponibilizando-lhe uma estrutura multiperspectiva e de rápida resposta. Contudo para que o acesso a essa informação seja rápido existe a necessidade de fazer a materialização de estruturas multidimensionais com esses dados já pré-calculados, reduzindo o tempo de interrogação ao tempo de leitura da resposta e evitando o tempo de processamento de cada query. A materialização completa dos dados necessários torna-se na prática impraticável dada a volumetria de dados a que os sistemas estão sujeitos e ao tempo de processamento necessário para calcular todas as combinações possíveis. Dado que o analista do negócio é o elemento diferenciador na utilização efectiva das estruturas, ou pelo menos aquele que selecciona os dados que são consultados nessas estruturas, este trabalho propõe um conjunto de técnicas que estudam o comportamento do utilizador, de forma a perceber o seu comportamento sazonal e as vistas alvo das suas explorações, para que seja possível fazer a definição de novas estruturas contendo as vistas mais apropriadas à materialização e assim melhor satisfaçam as necessidades de exploração dos seus utilizadores. Nesta dissertação são definidas estruturas que acolhem os registos de consultas dos utilizadores e com esses dados são aplicadas

técnicas de identificação de perfis de utilização e padrões de utilização, nomeadamente a definição de sessões OLAP, a aplicação de cadeias de Markov e a determinação de classes de equivalência de atributos consultados. No final deste estudo propomos a definição de uma assinatura OLAP capaz de definir o comportamento OLAP do utilizador com os elementos identificados nas técnicas estudadas e, assim, possibilitar ao administrador de sistema uma definição de reestruturação das estruturas multidimensionais "à medida" da utilização feita pelos analistas.

Palavras-chave: Sistemas de *Data Warehousing*, Processamento Analítico de Dados, OLAP, Sessões OLAP, Cadeias de Markov, Classes de Equivalência, Assinaturas OLAP.

Abstract

Multi-dimensional Views Materialization Based on User Queries

With the emergence of the information era many companies resorted to data warehouses to store an increasing amount of their business data. With this evolution of data volume the need to better explore this data arises in order to be somewhat useful in evaluating and making business decisions. OLAP (On-Line Analytical Processing) systems respond to the need of helping the business analyst in exploring the data by giving him the autonomy of exploration, providing him with a multi-perspective and quick answer structure. However, in order to provide quick access to this information the materialization of multi-dimensional structures with this data already calculated is required, reducing the query time to the answer reading time and avoiding the processing time of each query. The complete materialization of the required data is practically impossible due to the volume of data that the systems are subjected to and due to the processing time needed to calculate all combinations possible. Since the business analyst is the differentiating element in the effective use of these structures, this work proposes a set of techniques that study the user's behaviour in order to understand his seasonal behaviour and the target views of his explorations, so that it becomes possible to define new structures containing the most appropriate views for materialization and in this way better satisfying the exploration needs of its users. In this dissertation, structures that collect the query records of the users will be defined and with this data techniques of identification of user profiles and utilization patterns are applied, namely the definition of OLAP sessions, the application of Markov chains and the determination of equivalence classes of queried attributes. In the end of this study, the definition of an OLAP signature capable of defining the OLAP behaviour of the user with the elements identified in the studied techniques

will be proposed and this way allowing the system administrator a definition for restructuring of the multi-dimensional structures in "size" with the use done by the analysts.

Key Words: *Data Warehousing Systems*, On-Line Analytical Processing, OLAP Sessions, Markov Chains, Equivalence Classes, OLAP Signatures.

Índice

Índice de Figuras

Índice de Tabelas

Índice de Algoritmos

Lista de Siglas e Acrónimos

ACM	*Association for Computing Machinery*
API	*Application Programming Interface*
BUC	*Bottom-Up Computation*
Ci	Custo de Interrogação
Cm	Custo de Manutenção
CM	Cadeias de Markov
DOLAP	*Desktop On-Line Analytical Processing*
DW	*Data Warehouse*
DWeb	*Data Webhouse*
ER	Entidade Relacionamento
FK	*Foreign Key* (Chave estrangeira)
FASMI	*Fast Analysis o Shared Multidimensional Information*
HOLAP	*Hibrid On-Line Analytical Processing*
IDE	*Integrated Development Environment*
LQ	Log *Queries*
MDX	*Multi-Dimensional eXpressions*
MOLAP	*Multidimensional On-Line Analytical Processing*
OLAP	*On-Line Analytical Processing*
PROMISE	*Predicting User Behavior in Multidimentional Information Systems Enviroments*
PK	*Primary Key* (Chave Primária)
ROLAP	*Relational On-Line Analytical Processing*
S	Sessões OLAP
SIGMOD	*Special Interest Group on Management of Data*
SQL	*Structured Query Language*
TDC	*Top-Down Computing*
VRGA	*View Recommendation Greedy Algorithm*
XML	*eXtensible Markup Language*

Capítulo 1

Introdução

1.1 Contextualização

Com o crescente desenvolvimento dos mercados e a evolução imposta pelo ritmo que se vive numa era de informação e competitividade, as empresas são sujeitas a pressões de dinamismo perante o seu negócio, tendo necessidade de resoluções assertivas e atempadas para as novas oportunidades que surgem no dia-a-dia. Os decisores, com necessidades de análise e previsão sobre o seu negócio anseiam por sistemas capazes de os ajudar em análises rápidas, intuitivas e ajustadas. Necessitam de disponibilidade de informação do seu negócio não só a nível geral, mas com capacidade de descer ao detalhe se assim surgir a necessidade. Tudo isto ajustado a um tempo de resposta que lhes permita manter a competitividade e rigor nas decisões. Para uma grande parte das empresas, a resposta à necessidade de guardar os dados num formato orientado à consulta foi a criação de sistemas de *data warehousing (DW)*, definidos como "uma colecção de tecnologias de suporte à decisão, destinada a permitir o conhecimento ao trabalhador (executivo, gerente, analista) para tomar decisões melhores e mais rápidas" (Chaudhuri & Dayal, 1997).

Estes novos modelos de dados, geralmente alimentados por sistemas operacionais e com fluxos de inserção que garantam a consistência e pureza dos dados, vêm permitir definir uma separação entre sistemas operacionais de sistemas de processamento analítico. Esta mudança de paradigma trouxe aos analistas um ganho na fiabilidade nas suas análises e decisões como na própria

compreensão dos dados. No entanto as suas necessidades de análise, tal como os seus negócios, têm exigências de avaliações sobre vários pontos de vista. Assim, as várias medidas de análise podem ser consultadas sobre várias perspectivas – as designadas dimensões de análise, de forma a permitirem ao utilizador uma manipulação directa dos dados de forma que vá adaptando a sua consulta à necessidade de análise que lhe for surgindo. Esta nova necessidade de exploração dos dados segundo diferentes perspectivas de negócio e de adaptabilidade segundo a necessidade do analista após as consultas prévias veio ser colmatada pelos sistemas de processamento analítico, ou OLAP (*On-Line Analytical Processing*).

Os sistemas OLAP surgiram em 1993 (Codd, Codd & Salley, 1993) como resposta a essa necessidade de interacção do analista com os sistemas de armazenamento da informação, dotando-os de capacidade de manipulação dos dados de uma forma ágil e sempre rigorosa que os auxiliasse na sua avaliação e até previsão do negócio. Muitas já são as empresas com produtos líderes nesta área (Microsoft, Oracle, MicroStrategy, IBM) (Gartner, 2011). Para suportar estes novos sistemas, surgiram as estruturas multidimensionais de dados, os cubos de dados (conceito adoptado pela simplificação da sua visualização a três dimensões). Estas estruturas permitem organizar os dados segundo várias perspectivas de análise e tendo em conta as respectivas hierarquias, níveis de agregação lógicos definidos sobre as dimensões (ex: para uma dimensão tempo, uma hierarquia possível poderia ser: dia -> mês -> ano). Esta organização dos factos de uma forma multidimensional e hierárquica permite fazer manipulações dos dados como *drilldown*, *rollup*, *pivoting* e *slice and dice*, típicas dos sistemas de processamento analítico (Chaudhuri & Dayal, 1997).

Esta interactividade com os sistemas OLAP deve ter como requisito uma resposta rápida, tanto às interrogações quanto à manipulação da estrutura, permitindo a fluidez da análise, a produtividade dos analistas e a qualidade das suas decisões. Uma análise que seja constantemente quebrada por processos demorados, além de dificultar o trabalho do utilizador, pode propiciar tarefas alternativas e induzir quebras de raciocínio (Business Application Research Center, 2011). Este tempo de resposta está associado não só às condições de *hardware* da solução interveniente mas sobretudo às condições de disponibilização dos dados consultados. Imaginemos que, por exemplo, um analista pretende comparar os resultados obtidos nas lojas do concelho de Braga com as do concelho do Faro para o ano corrente e o ano anterior. Esta informação necessita dos valores de vendas (medida em análise) para níveis de agregação correspondentes às dimensões com os dados das lojas (loja), do concelho (localização) e do ano (tempo). Se este pedido for lançado ao

servidor e tiver que ser processado no momento da consulta, com a agravante de serem grandes volumes de dados, o tempo de resposta será demorado pois implica o cálculo da métrica para todos os registos existentes. Por outro lado, se este valor já estiver previamente agregado e armazenado, o tempo de resposta reduz-se ao tempo de consulta dos valores.

Este tipo de análise leva-nos ao pré cálculo e materialização das agregações, uma resposta evidente ao problema na demora das respostas às interrogações, mas que por outro lado levanta problemas de espaço de armazenamento, tempo de processamento e materialização das agregações, além da manutenção dessas mesmas agregações, dada a necessidade de assegurar a consistência da sua informação. Esta solução de pré-materialização dos dados é ainda agravada pela diversidade do número de agregações possíveis, por exemplo se considerarmos cinco dimensões e que cada uma delas tenha também cinco níveis de agregação possíveis, as hierarquias, que é um cenário bastante modesto quando comparado com algumas situações reais, o número de agregações possíveis seria de 1024. Este número, quando aplicado a casos reais, cresce descontroladamente, tornando-se muitas vezes incomportável a pré-materialização de todas as agregações possíveis e obrigando os administradores dos sistemas a tomar decisões de escolha das agregações que devem ser materializadas.

Além do crescente número de agregações possíveis, face ao número de dimensões e níveis de análise, existe ainda a percepção de que nem todas as agregações possíveis são alvos de interrogações, chegando a haver uma boa parte delas que nem sequer chega a ser questionada. Outras são direccionadas a grupos de utilizadores específicos que têm necessidades distintas, levando à necessidade de selecção de vistas, agregações, do cubo a materializar que satisfaçam as necessidades de cada utilizador e não prejudicando o fluxo de análise com necessidade de processamento imediato. Este é considerado o problema de selecções de vistas ou subcubos que Harinarayan, Rajaraman & Ulman (1996) caracterizou de NP-Hard. Esta selecção quando adequada às necessidades de interrogação, não só minimiza a quantidade de agregações necessárias a materializar como diminui o tempo de processamento e o espaço necessário para armazenar essas mesmas agregações. Contudo deve existir precaução nesta selecção, pois uma má selecção das vistas pode levar os sistemas OLAP às dificuldades iniciais, ou seja, se a selecção não contemplar uma agregação requerida pelo utilizador, implica a necessidade do seu cálculo após o seu pedido e consequente demora no tempo de resposta, se esta situação se repetir para um grande número de pedidos, então a selecção não foi correctamente feita e não vai de encontro às necessidades de utilização do sistema.

1.2 Motivação e Objectivos

Rapidamente se percebe que os sistemas de processamento analítico são uma mais-valia para a tomada de decisões dos analistas de negócio. Propiciam consultas orientadas e de resposta atempada. No entanto essa resposta atempada pode ser influenciada por inúmeros factores, nomeadamente a disponibilidade dos dados. O facto de uma dada consulta ter os dados pré-agregados ou não faz toda a diferença no momento da interrogação, que implica uma simples consulta ou um processamento que pode ser mais ou menos complexo segundo a própria complexidade da *query*, do volume de dados e mesmo do desenho da estrutura de suporte aos dados. Esta necessidade de processamento momentâneo aquando a consulta, tal como já vimos, pode ser minimizada com a pré-materialização, no entanto implica que haja uma decisão sobre quais vistas devem ser materializadas, tornando-se este um problema alvo de vários estudos no meio científico.

Esta tarefa de decisão de selecção é bastante complexa e dependendo de alguns factores pode ser certa ou errada, por exemplo, um dado conjunto de agregações pode ser alvo de análise de um utilizador mas nunca ser consultado por outro, dependendo do foco do seu trabalho. Se, a selecção de vistas de cada um destes analistas não for a apropriada para as suas consultas, podemo-nos deparar com dois problemas, por um lado, se as vistas materializadas não responderem às suas necessidades, vamos de encontro ao problema de demora na resposta que já identificámos, por outro, se materializámos agregações que não chegam a ser consultadas, ou que raramente são alvo de pesquisas, então estamos a desperdiçar recursos para o processamento e para a materialização que seriam bastante úteis para outras estruturas devidamente identificadas.

Com tudo isto chegámos ao objectivo macro do nosso trabalho, propor estruturas multidimensionais que respondam às necessidades de utilização, optimizando o processamento das agregações e reduzindo o espaço necessário para a sua materialização. Contudo temos vindo a analisar a dificuldade de decisão destas estruturas, principalmente porque elas devem responder às necessidades de consulta dos utilizadores. Necessidades essas que vão variando conforme o negócio ou até mesmo conforme a sazonalidade, por exemplo as análises de valores do final do mês, ou ano, devem ser diferentes àquelas que são feitas todos os dias. Todos estes parâmetros

fazem parte da utilização, que está constantemente a mudar, quanto mais não seja pela experiência adquirida pelo analista que vai aprimorando as suas análises e avaliações do negócio.

Ao longo desta nossa análise, percebemos que o factor comum é a utilização, assim como o analista que interroga o sistema. Se o sistema deve ser capaz de responder às necessidades do utilizador, então nada melhor que conhecer o percurso do utilizador a fim de perceber o seu perfil e tentar identificar as suas necessidades constantes. Se por um lado algumas das pesquisas vão mudando, por outro, uma grande parte do trabalho feito é repetido com alguma frequência, se formos conhecedores desses padrões, mais facilmente podemos decidir as pré-materializações para irem de encontro às necessidades identificadas.

Várias técnicas de *profiling* de utilização já foram implementadas em outras áreas, nomeadamente na utilização da Web. Estas permitem traçar perfis de utilização e prever o comportamento do utilizador, com intuito de propor os passos seguintes ou simplesmente disponibilizar a informação de uma forma mais atempada. Esse é o nosso objectivo, conseguir até certo ponto prever a utilização, identificando a sazonalidade e focos de interrogações e tirar partido desse conhecimento para propor estruturas multidimensionais mais ajustadas à utilização de cada analista, ou grupo de analistas.

Para conseguir este estudo e propor estruturas ajustadas à medida dos requisitos de negócio e dos analistas, com base em perfis de utilização corrente e considerando tendências sazonais de processos de tomada de decisão definimos alguns objectivos para este projecto:

- Capturar registos de *queries* multidimensionais lançadas sobre a plataforma analítica.
- Identificar, caracterizar e definir os hipercubos alvos de consultas, de acordo com as sessões de exploração multidimensional.
- Gerar padrões de utilização de hipercubos – sessões OLAP – e enriquecê-los, tanto quanto possível, com informações de sazonalidade das consultas e do negócio.
- Prever as próximas consultas segundo probabilidades obtidas com técnicas de mineração de dados.
- Identificar elementos de queries que ocorram frequentemente conjuntas.

- Propor estruturas que considerem dimensões, assim como as suas hierarquias, e medidas para materialização e ainda a periodicidade e tipo de refrescamento das mesmas.

O desenvolvimento deste estudo ocorreu no âmbito de um projecto de investigação e desenvolvimento intitulado UPonICE, Modelos e Estratégias para a Optimização de Estruturas Multidimensionais de Dados (Belo & Duarte, 2010). O projecto, com financiamento da empresa Portugal Telecom Inovação, visava disponibilizar um sistema de análise de *queries* multidimensionais capaz de estabelecer perfis de utilização (exploração) de estruturas multidimensionais de dados com vista à optimização, quer a nível do processamento, bem como ao nível do armazenamento.

1.3 Estrutura do Documento

As propostas de sistemas de processamento analítico ajustados a satisfazer tempo de respostas o mais reduzido possível e disponibilidade de informação adequada ao negócio e sua análise, são recomendações difíceis de conseguir e geralmente são readaptadas e aperfeiçoadas consoante o crescimento dos dados, o aumento do número de utilizadores e até com a complexidade do negócio. Como temos vindo a ver a problemática da selecção de vistas é um dos pontos fulcrais de optimização de um sistema desta natureza.

Neste trabalho, após um estudo do estado da arte, propomos técnicas de reestruturação de hipercubos com base na utilização que foram aqui documentadas e estruturadas nos seguintes capítulos:

- **Capítulo 2** – Sistemas de Processamento Analíticos. Neste capítulo, realiza-se uma breve introdução de conceitos relacionados com os Sistemas de Processamento Analítico, nomeadamente como e porquê surgiu, quais as estruturas de dados associadas, a exploração de dados, como é que as estruturas multidimensionais são processadas e ainda as arquitecturas OLAP.

- **Capítulo 3** – Métodos de Reestruturação de Hipercubos. Nesta parte, começamos com uma apresentação da evolução do estudo da problemática da selecção de vistas com a

identificação de alguns dos trabalhos existentes e de seguida o estudo sobre técnicas relacionadas com as propostas de reestruturação feitas no âmbito deste trabalho, entre estas a paralelização do estudo feito sobre consultas *web* com as consultas multidimensionais comparando os *clicks* em páginas *web* com as *queries* lançadas pelo utilizador OLAP. Finaliza-se este capítulo com a apresentação de técnicas de identificação de classes de equivalência.

- **Capítulo 4** – Reestruturação de Hipercubos à Medida. Aqui podemos encontrar toda a descrição da nossa proposta de reestruturação de hipercubos. Desde uma apresentação inicial do ciclo de reestruturação até uma proposta final de assinaturas OLAP que caracterizam a utilização de um analista segundo as técnicas apresentadas ao longo do trabalho. No decorrer da segunda secção é também descrito a fase de recolha e armazenamento de dados que se torna essencial para o desenvolvimento do projecto, contemplando as ferramentas de suporte, a base de dados de trabalho, o acesso e exploração dos dados e ainda a recolha e armazenamento das consultas. Na apresentação das técnicas, mais propriamente nas secções 4.3 e 4.4, são descritas as formas de aplicação ao trabalho em causa, Sessões OLAP, Cadeias de *Markov* e Classes de Equivalência, assim como o proveito que se pode tirar de cada uma delas para a proposta de reestruturação.

- **Capítulo 5** – Conclusões e Trabalho Futuro. O capítulo final apresenta um breve comentário final ao trabalho realizado, bem como uma apreciação à abordagem desenvolvida. Por fim apresentámos algumas linhas de trabalho futuro de forma a complementar e continuar o estudo que aqui está apresentado.

Capítulo 2

Sistemas de Processamento Analítico

2.1 Sistemas de Processamento Analítico

Cada vez mais a rapidez de evolução, o dinamismo e a adaptabilidade das empresas são factores diferenciadores na concorrência dos mercados. Os analistas de negócio são todos os dias confrontados com novos desafios, o que exige decisões atempadas e ajustadas ao problema em causa. Para conseguirem responder a estes estímulos, os analistas necessitam, cada vez mais, de terem acesso a toda a informação relativa às áreas de negócio com que estão envolvidos. Desde o acesso a dados mais detalhados até à consulta de métricas globais, a disponibilidade de informação correcta, de acesso rápido e intuitivo pode fazer a diferença num processo de tomada de decisão. Para responder a estas necessidades, muitas empresas criaram sistemas de *data warehousing* (SDW), definidos em torno de "uma colecção de dados empresariais, orientados a assuntos específicos, integrados, revelando a sua variação ao longo do tempo e, obviamente, não voláteis" (Inmon, 1996).

Estes sistemas, alimentados por processos que os ligam a sistemas operacionais, vieram trazer aos decisores o ganho da informação, permitindo-lhes perceber realmente o significado dos dados provenientes do negócio. No entanto cada vez mais os analistas sentem a necessidade de poderem manusear essa informação e obterem resultados segundo vários eixos de análise (dimensões). Para responder a esta necessidade, surgiram alguns sistemas capazes de proporcionar a

investigação dos dados segundo as mais variadas perspectivas e navegar ao longo das suas estruturas de forma a decidir a próxima análise baseada no resultado de observações anteriores.

Para responder a estas necessidades, Codd propôs em 1993 uma nova tecnologia de processamento analítico chamada *OnLine Analytical Processing* (OLAP) (Codd, Codd & Salley, 1993) (Kirkgöze et al., 1997), uma tecnologia de processamento analítico que proporciona a descoberta de informação de negócio a partir dos dados existentes. Estes sistemas são suportados por estruturas multidimensionais conhecidas como cubos OLAP e optimizadas para a consulta de dados. Esta multidimensionalidade deriva da multiperspectiva que o analista tem do seu negócio, sendo que uma das características dos sistemas deve ser a permissão e facilidade do utilizador poder interagir com o esquema lógico multidimensional de forma a construir as suas consultas.

Para consolidar as características fundamentais dos sistemas de processamento analítico (Codd, Codd & Salley, 1993) Codd apresentou um conjunto de 12 regras de avaliação das ferramentas OLAP candidatas a serem comercializadas. No entanto, Nigel Pendse e Richard Creeth consideram o termo com um significado pouco claro, assim como as regras definidas serem inadequadas à caracterização das ferramentas. Para colmatar estas falhas, em 1995 surge o conceito FASMI (***F**ast **A**nalysis of **S**hared **M**ultidimensional **I**nformation*) (Pendse, 2008), um termo simples, de fácil memorização e independente de produtos já existentes. Esta proposta ditou 5 regras que caracterizam as aplicações OLAP de uma forma específica, mas sem ditar a sua implementação. Essas regras são:

- **F (*Fast*) – Rápida**. O sistema deve oferecer respostas rápidas. Neste caso, isto quer dizer que tendencialmente o sistema é voltado para respostas dentro dos cinco segundos, ou de um a vinte segundos dependendo da complexidade, devendo ser poucas as que demorem o tempo máximo recomendado. Estudos revelam que os utilizadores após 30 segundos, se não forem avisados que a resposta vai ser mais demorada, consideram que o sistema entrou em erro e cancelam a operação. Mesmo quando advertidos da demora, os utilizadores acabam por distrair-se e quebrar a linha de pensamento do estudo e consecutivamente prejudicar a qualidade da análise. O *BI Survey*[1] (Business Application

[1] *BI Survey* é um relatório anual de utilização de Business Intelligence e Performance Management com respostas de fornecedores independentes de todo o mundo. Este relatório compara ainda os principais produtos no mercado através de critérios como desempenho, escalabilidade e suporte. O BI Survey 9 atraiu 3.093 respostas.

Research Center, 2011) conclui que as respostas lentas continuam a ser citadas como problema técnico frequente em produtos OLAP.

- **A (*Analysis*) – Analítica**. Significa que o sistema lida com qualquer lógica de negócio e análise estatística relevante para a aplicação e para o utilizador, assim como é de fácil uso para o utilizador final. Deve proporcionar a possibilidade de definir novos cálculos *ad_hoc* assim como relatórios que satisfaçam as necessidades de flexibilidade e cálculo orientado dos utilizadores sem necessidade de programação. O sistema deve ser funcional e intuitivo para o utilizador final.

- **S (*Shared*) – Partilhada.** O sistema deve implementar todos os requisitos de segurança e confidencialidade (de preferência até à célula) e ter vários tipos de permissões de escrita, contudo com cuidados adaptados a actualizações simultâneas. Mesmo quando não são necessárias as permissões de escrita, elas devem ser acauteladas para futuras evoluções. Esta é uma das principais áreas de fraqueza nos produtos OLAP, que tendem a assumir que todas as aplicações OLAP apenas necessitam de permissões de leitura e adoptam controlos de segurança simplista.

- **M (*Multidimentional*) – Multidimensional**. É a principal característica de um sistema OLAP. O sistema deve fornecer uma visão multidimensional, incluindo o suporte a hierarquias e a sua exploração, permitindo ao utilizador uma análise mais abrangente.

- **I (*Information*) - Informação**. Os sistemas devem ser capazes de lidar com grandes quantidades de dados, independentemente do modelo de servidor. Isto salvaguarda não só a necessidade de replicação de dados, exigindo características relacionadas com o desempenho e espaço, mas também a integração com outros sistemas que sejam necessários à análise, como por exemplo regras de cálculo de valores.

Em suma, um sistema de processamento analítico é um sistema capaz de responder às necessidades analíticas de um negócio, baseando-se num esquema multidimensional permite que o utilizador seja capaz de conduzir as suas consultas conforme as necessidades que for encontrando.

2.2 Estruturas Multidimensionais de Dados

As estruturas multidimensionais são o componente central dos sistemas OLAP. São elas que dotam os utilizadores de liberdade de exploração dos dados permitindo-lhes reorganizar a disposição da informação assim como responder às necessidades de uma visão multiperspectiva dos factos. Estes modelos de dados são construídos com objectivo de permitir acções como: navegar nos dados a vários níveis de agregação, permitir mostrar e omitir perspectivas, usar funções de agregação, rodar as perspectivas, etc..

Se considerarmos fazer uma avaliação de uma análise de factos sobre três perspectivas, ou dimensões, podemos visualizar esses dados como um cubo. Esta analogia, que permite uma materialização visual do conceito, é uma imagem recorrente quando falámos das estruturas multidimensionais. No entanto, nos sistemas de processamento analítico reais o número de dimensões é, geralmente, superior a três. Para acautelar esta generalização para *n* dimensões, as estruturas dever-se-ão chamar hipercubos ou multicubos. Contudo a designação simplista de cubo é mais usual, mesmo quando a referência é a uma estrutura $n > 3$, devido, simplesmente, à sua empatia visual.

Um cubo de dado é uma projecção multidimensional resultante da combinação das coordenadas de cada eixo de análise, concretamente cada dimensão. Cada célula contém os valores correspondentes às métricas de negócio caracterizadas por essa combinação de coordenadas. Se considerarmos um exemplo tridimensional <Cliente; Produto; Tempo> e a métrica vendas, obteríamos um cubo como aquele que está apresentado na Figura 2.1. Estas estruturas foram propostas por Gray em 1997, como uma generalização dos operadores de *SQL Group By; Cross-Tab* e *Sub-Totals* (Gray et al., 1997).

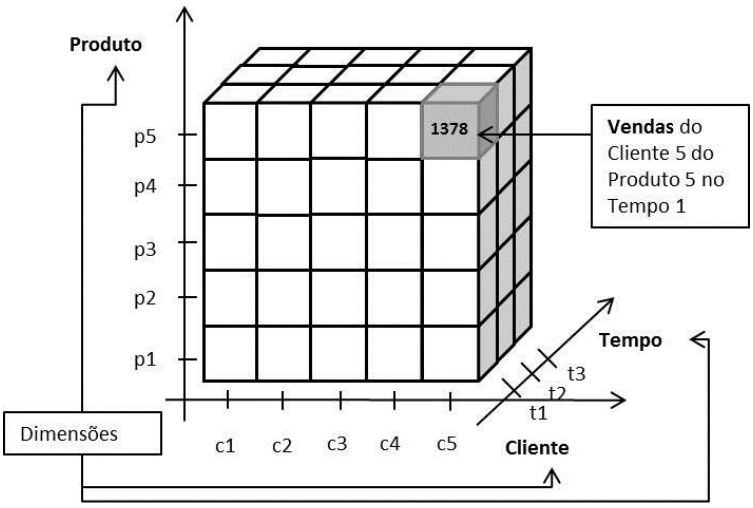


Figura 2.1 – Exemplo de um cubo com três dimensões <Cliente; Produto; Tempo>

O conceito de dimensão é essencial para a compreensão dos dados multidimensionais. Com elas podemos seleccionar e agrupar os dados. Cada dimensão pode ter um ou mais atributos, sendo que estes estão organizados em uma ou mais hierarquias. As hierarquias são níveis de detalhe que o analista poderá escolher para a sua análise. Por exemplo, a dimensão `'tempo'` pode ter como hierarquias `Dia -> Mês -> Semestre -> Ano` e `Dia -> Estação do Ano -> Ano` (Figura 2.2).

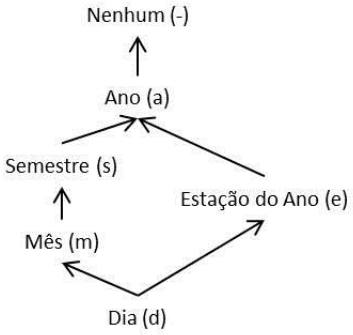


Figura 2.2 - Exemplo de duas hierarquias para a dimensão `'tempo'`

Para cada um destes níveis a dimensão tem um conjunto de instâncias - por exemplo, para `Mês` teremos *{Janeiro; Fevereiro;…},* que são os valores das dimensões. A combinação de níveis, de cada dimensão, dá-nos a granularidade dos factos consultados, isto quer dizer, o grão da consulta será dado pelo valor do facto caracterizado pelas dimensões nos níveis de agregação escolhidos. Por exemplo, para o cubo que temos na Figura 2.1 se considerarmos a hierarquia tempo `Dia -> Mês -> Semestre -> Ano` e considerarmos que o tempo da ilustração está ao nível de `Mês`, então a granularidade do cubo de exemplo seria *"Produto por Cliente por Mês"*, ou numa versão mais simplista, apenas com as iniciais (cpm). Além de detalharmos as dimensões e conseguirmos consultas com granularidades mais finas, também é possível obter uma granularidade mais grossa colapsando uma dimensão, ou seja, não escolhermos nenhuma instância (Nenhum (-)) para uma dada dimensão e, assim, serem consideradas todas as instâncias (em algumas situações em vez de aparecer nenhum para representar esta agregação, aparece "todos"). Por exemplo, no cubo que temos vindo a observar, se quisermos saber *"As Vendas de todos os Produtos em todos os Meses"*, a nossa consulta teria que considerar a agregação de todos os clientes, ou seja, (-pm), onde '–' significa que não escolhemos nenhum nível da hierarquia de cliente e assim considerámos todas instâncias (Figura 2.3).

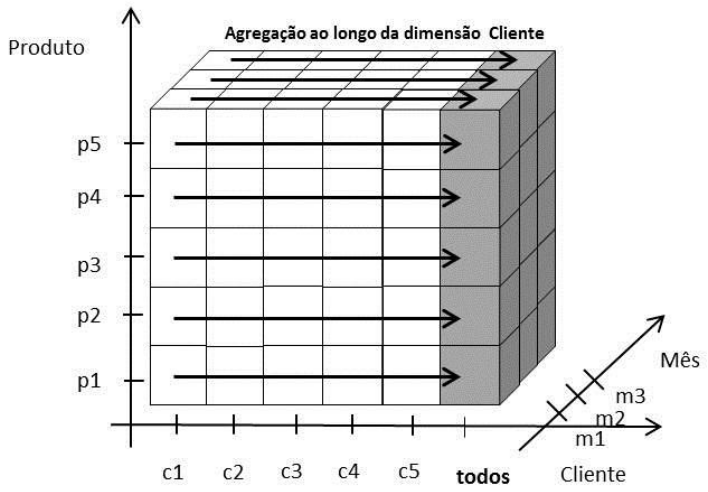


Figura 2.3 - Exemplo de uma agregação num cubo

Este tido de granularidade mais agregadora, geralmente, pode ser obtida a partir dos dados de granularidades mais finas. A sequência de níveis das hierarquias é a sequência de níveis de agregação numa dada dimensão. No seguimento deste conceito, de reutilização de dados gerados para o cálculo de dados mais agregados, surge a *lattice* de dependências proposta por Harinarayan, Rajaraman & Ullman (1996), que nos permite definir todas as agregações possíveis dentro do nosso conjunto de dimensões, assim como as respectivas dependências. Considerando o cubo que temos usado como exemplo `<Cliente; Produto; Tempo>` a *lattice* de dependências seria a apresentada na Figura 2.4.

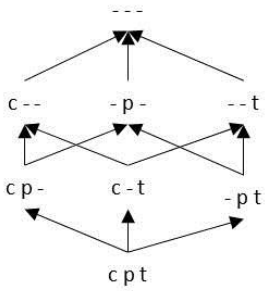


Figura 2.4 - Lattice de dependências do cubo `<Cliente; Produto; Tempo>`

Esta estrutura permite-nos perceber que para três dimensões podemos considerar 2^3 cubos, sem, no entanto, considerámos qualquer nível respeitante a uma ou mais hierarquias. De uma forma genérica, para calcularmos o número de cubos possíveis para d dimensões com h_i níveis sobre a respectiva hierarquia, poderemos usar a seguinte expressão:

$$\prod_{i=1}^{d}(h_i + 1)$$

Através desta fórmula, podemos perceber que o crescimento do número de cubos possíveis é bastante acentuado conforme vamos aumentando o número de dimensões e respectivas

hierarquias. Por exemplo, se considerámos cinco dimensões com quatro hierarquias cada uma, seriam 4*4*4*4*4=1024 cubos possíveis. Uma estrutura gigantesca.

Numa perspectiva genérica vimos que o Hipercubo não passa de um conjunto de dados agregados de acordo com uma dada granularidade. Quando efectuamos uma consulta a um sistema de processamento analítico, estamos a definir um conjunto de vistas de dados que queremos visualizar. Estes dados são métricas agregadas segundo dimensões de análise – perspectivas de análise – que podem ser exploradas em diferentes níveis de agregação. Podemos assim considerar que as queries são vistas ou subcubos da estrutura disponível. Observando a *lattice* apresentada na Figura 2.4 conseguimos perceber que o grão da agregação *(cpt)* é menor do que, por exemplo, o da agregação *(-p-)* isto porque as dimensões `cliente` e `tempo` seriam agregadas e o detalhe seria apenas ao nível da dimensão `produto`. Este tipo de respostas a consultas mais agregadas pode não precisar de pré-calculo ou armazenamento dos dados, isto porque os dados de um maior nível de detalhe podem ser utilizados para o cálculo dos níveis superiores, embora este tipo de técnica implique a agregação e cálculo das métricas aquando da consulta. Por exemplo, se tivermos materializado um cubo com o `tempo` ao nível do `mês` e a consulta for com agregação ao `ano`, o cálculo desta vista é bastante mais rápido se a agregação for feita a partir dos registos do `mês` do que se tiver que consultar, por exemplo, a tabela de factos ao `dia`. Com este tipo de técnicas, mesmo que o subcubo mais adequado não esteja disponível, no caso do exemplo seria com a agregação ao `ano`, as dependências entre os cubos materializados podem ser úteis para encontrar a alternativa com o menor número de linhas e, assim, menos custosa para o cálculo e mais rápida na resposta à *query* em questão.

2.3 Exploração de Hipercubos

Os sistemas de processamento analítico foram criados para auxiliar análises sobre dados e ajudar os analistas a perceber os seus conteúdos que até então não eram evidentes. Estes novos sistemas vieram permitir, acima de tudo, que o utilizador possa agora manusear os dados de forma a chegar a respostas concretas, passo a passo, *query* a *query*. Na maioria das situações as consultas não são casos isolados, mas sim sequências interactivas que partem, por exemplo, de um relatório e vão sendo manipuladas e ajustadas segundo o percurso de análise que o utilizador

pretende. Para que este tipo de processo seja possível é necessário poder fazer algumas operações sobre os dados de forma a poder vê-los de várias perspectivas ou níveis de agregação diferentes e que levem a um entendimento claro dos dados.

Retomemos o nosso exemplo de cubo <Cliente, Produto, Tempo>, agora para melhor perceber que tipo de manuseamento dos dados é possível realizar no ambiente deste tipo de sistemas. Para tal, vamos considerar as seguintes hierarquias: Cliente(c) -> AnoNascimento(n); Produto(p) -> TipoProduto(t); Dia(d) -> Mês(m) -> Ano(a) e a métrica Vendas. Este primeiro exemplo corresponderia a (cpd) no detalhe mais fino e a (nta) para o caso da agregação maior.

O facto de as dimensões estarem organizadas em hierarquias, permite ao utilizador ver os dados em vários níveis de agregação. Assim, se o utilizador começar com uma consulta "*as vendas de todos os clientes (c1, c2, c3, c4 e c5), de todos os produtos (p1, p2, p3, p4 e p5) para os dias d1, d2 e d3*", *(cpd)*, a consulta estaria a se dirigir a um detalhe de granularidade máximo. Se de seguida, o analista perceber que prefere ver as vendas para um nível de agregação ao nível do mês e assim "*consultar as vendas dos clientes e produtos ao nível do mês*", *(-pm), o* utilizador estaria a executar uma operação de *rollup*. No entanto se o utilizador decidir fazer uma operação do tipo inverso, por exemplo se quisesse voltar a ver o detalhe ao dia, realizaria uma operação que é reconhecida por *driilldown* (desagregar) ao longo da dimensão tempo, do nível do mês para o nível do dia (Figura 2.5) (Kirkgöze et al., 1997).

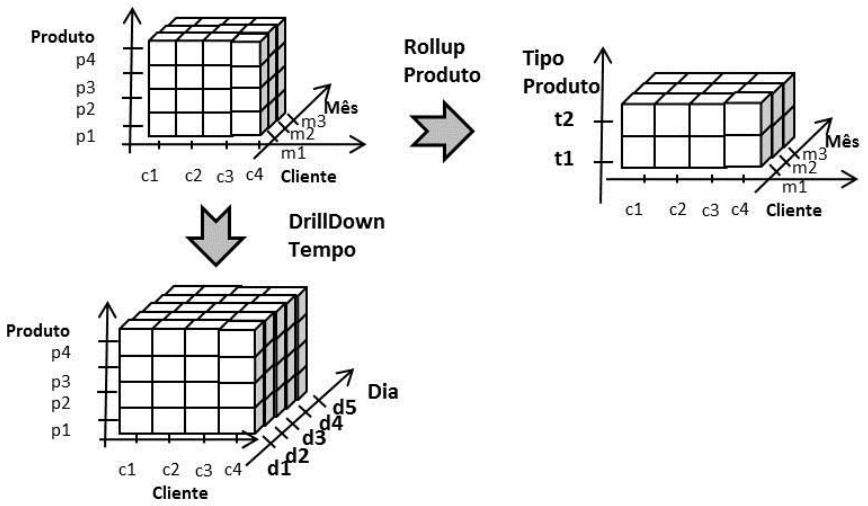


Figura 2.5 – Representação das operações de *drilldown* e *rollup*

Outro conjunto de operações bastante usual é constituído pelas operações de *slicing*, *dicing* e *pivoting*. Estas operações permitem ao utilizador restringir as dimensões a valores exactos e também girar as estruturas de forma que as linhas de resposta sejam menores e com disposições mais lógicas à análise (Figura 2.6).

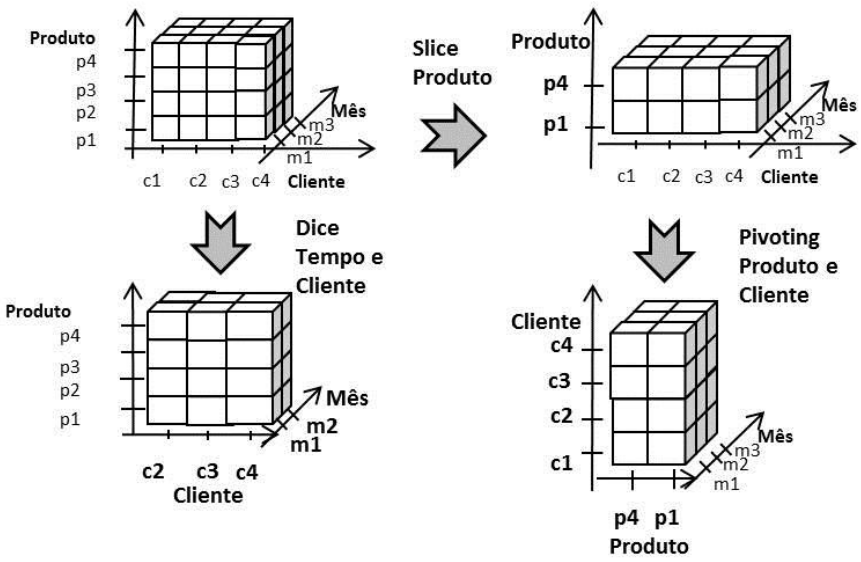


Figura 2.6 - Ilustração das operações de *slicing*, *dicing* e *pivoting*

Estas operações podem ser feitas a partir de uma linguagem de interrogação de estruturas multidimensionais. A mais usual entre as ferramentas é a linguagem *Multidimensional Expressions* (MDX) introduzida pela *Microsoft* em 1998 (Microsoft MDX, 2011) e posteriormente adoptada por muitas outras ferramentas analíticas como a *MicroStrategy* (MicroStrategy, 2011) e *Mondrian* (Hyde, 2009), por exemplo. A MDX é uma linguagem semelhante à SQL (*Structured Query Language*), a linguagem padrão para consultas em sistemas relacionais. No entanto não é uma sua extensão, embora possa em algumas situações ser traduzida para SQL.

A linguagem MDX permite não só consultas a cubos multidimensionais mas também a formatação desses mesmos resultados, assim como executar tarefas de *design* de cubos ou mesmo determinar indicadores de desempenho. Esta linguagem permite ainda executar tarefas administrativas, incluindo fazer a definição da dimensão e dos critérios de segurança da célula de um cubo. Esta linguagem contém os seguintes tipos de dados:

- escalar: número ou string que pode ser especificado literalmente ou por retorno de uma função MDX, como por exemplo string 'mdx'; Sum(número);

- boleano: funções ou expressões lógicas podem devolver boleanos com valores de verdadeiro ou falso;

- dimensão / Hierarquia: Dimensão corresponde a uma dimensão de um cubo contendo membros; Uma hierarquia por sua vez pode conter níveis; ambos os tipos podem ser usados como os nomes directos das estruturas, ou por função, exemplos: [Tempo], mdx.Dimension = [Tempo], mdx.HyerarchyName = [Tempo]. Sendo que mdx corresponde a uma expressão MDX.

- nível: Correspondente a um nível de uma hierarquia que pode ser especificado directamente ou por uma função, exemplo: [Tempo].[Mês] ou mdx.Level = [Mês];

- membro, uma instância num nível de uma dimensão ou hierarquia que pode ser definido directamente ou por invocação de uma função, exemplo: [Tempo].[Mês].[Outubro] ou mdx.PrevMember = [Outubro];

- tuplo: Colecção ordenada de um ou mais membros que podem ser de hierarquias ou dimensões diferentes, exemplo: ([Tempo].[Mês].[Outubro], [Produto].[Guarda Chuva]);

- conjunto: representa uma colecção ordenada de tuplos com a mesma hierarquia ou dimensão, exemplo: {([Measure].[Vendas] , [Tempo].[Mês].[Setembro]), ([Measure].[Vendas] , [Tempo].[Mês].[Outubro])} (Microsoft MDX, 2011);

Além dos tipos de dados a linguagem ainda dispõe de um vasto conjunto de funções para cálculos e para operações sobre os cubos, como por exemplo a função *DrilldownLevel* que aplica uma operação de *drilldown* em um nível a todos os elementos de um conjunto. Existem ainda um conjunto de funções direccionadas a cada um dos tipos de dados. Esta linguagem permite ainda que os utilizadores definam as suas próprias funções, possibilitando definições mais apropriadas às necessidades de cada pesquisa. Em suma, a MDX permite conjugar um conjunto de expressões e recorrer a variadas funções de forma a construir queries que satisfaçam as consultas necessárias (Microsoft MDX, 2011).

```
SELECT {[Tempo].[Mês].[Outubro]} ON COLUMNS,
 {[Produto].Chidren} ON ROWS
FROM Vendas
WHERE ([Measures].[Vendas])
```

Figura 2.7 - Exemplo de consulta MDX

Na Figura 2.7 podemos observar um exemplo de uma consulta MDX que pode ser dividida em três partes, nomeadamente *SELECT*, *FROM* e *WHERE*. Na primeira parte da *query*, SELECT, é identificado o que se pretende ver nas colunas, ON COLUMNS, e nas linhas, ON ROWS, neste exemplo selecciona-se todas as ocorrências de Produto, nas linhas, e o mês de Outubro, na coluna. A segunda parte da *query*, FROM, identifica o cubo que será consultado, no exemplo apresentado o cubo Vendas. Na última parte da *query* MDX, WHERE, é especificada a métrica a ser calculada para cada uma das células, neste caso a métrica vendas (Figura 2.8).

[Measures].[Vendas]	*[Tempo].[Mês].[Outubro]*
[Produto].[Aquecedor]	2.976
[Produto].[Frigorífico]	72.456
[Produto].[Fogão]	89.474
[Produto].[Microondas]	56.934
[Produto].[Máquina de Lavar Roupa]	15.954

Figura 2.8 - Exemplo de um resultado de uma consulta MDX

2.4 Processamento de Hipercubos

A base dos sistemas de processamento analítico são as estruturas multidimensionais que armazenam os dados pré-calculados, os hipercubos, ou adoptando o termo mais simplista os cubos. A implementação destas estruturas levanta à partida dois problemas, o processamento dos dados e o seu armazenamento. A necessidade de pré-calculo das agregações, consoante as funções necessárias, são actividades demoradas e com bastante exigência de processamento que são escaláveis com a quantidade de dados (Morfonios et al., 2007).

A melhor forma de processar os hipercubos é alvo de estudos desde que os sistemas de processamento analítico surgiram. Alguns foram os algoritmos que foram propostos para abordar o processamento das estruturas multidimensionais para suporte analítico (Morfonios et al., 2007). O algoritmo 2^D (Gray et al., 1997) foi um dos primeiros, proposto também como introdução à estrutura dos cubos de dados. É um algoritmo simples mas pouco eficiente devido à sua complexidade exponencial que computa todos os *group-by*[2] directamente da tabela de factos e de seguida faz a união de todos os resultados. Já o algoritmo GBLP (Gray et al., 1997) melhora a eficiência do anterior tirando partido da *lattice* e calculando a agregação a partir da menor já calculada. Este algoritmo mesmo sendo mais eficiente, ainda apresenta algumas desvantagens de acesso a grandes quantidades de dados, não sendo possível manter um cubo total em memória, que rapidamente se torna suficientemente grande para não ser passível de o manter. O Algoritmo PipeSort Agarwal et al. (1996) é a primeira tentativa de melhorar o GBLP minimizando o custo de computação total do cubo. Este algoritmo associa pesos estimados a cada possibilidade de calcular os vários nós da *lattice*, escolhendo no final qual o nó "pai" mais vantajoso ao cálculo da agregação. A desvantagem deste algoritmo dá-se pela má escalabilidade, tornando-se ineficiente para grandes quantidades de dimensões, dado que pesa cada caminho possível para cálculo de cada agregação. Ainda em (Agarwal et al., 1996) é paralelamente proposto o algoritmo Overlap. Este algoritmo tira partido da ordenação dos tuplos de agregação, de maneira a reaproveitar *subsets* já disponíveis. Além disso este algoritmo recorre a métodos de *prunning*[3] da *lattice* e a

[2] *group-by* – instrução SQL que em conjunto com funções de agregação agrupa os resultados das linhas intervenientes na consulta segundo uma coluna.
[3] *prunning* – redução de nós de forma a evitar cálculos desnecessários

computação conjunta de nós para reduzir os custos de acesso a disco. Em comparação ao algoritmo anterior, o Overlap além de ter uma forma diferente de travessia da árvore ele tira partido da ordenação dos tuplos a partir de subsequências anteriores com a mesma ordenação. Em Agarwal et al. (1996) é ainda proposto o PipeHash, uma alternativa ao PipeSort que tira partido de *hashing*[4]. Este algoritmo tem a particularidade de seleccionar o nó "pai", para cálculo da agregação, levando em conta o tamanho estimado do ascendente e escolhendo o menor. Em 1997 Ross & Srivastava (1997) propõe um algoritmo que considera bases de dados com grandes volumes de dados dispersos que levam a cálculos de agregações desnecessárias. Este novo factor de análise torna o algoritmo Partitioned-Cube (Ross & Srivastava, 1997) mais eficiente que os anteriores. Este algoritmo combina as vantagens das propostas anteriores de forma a particionar melhor os dados e a executar um processamento mais rápido e adequado a grandes volumes de dados dispersos. Mais tarde surgem algoritmos como BUC (Beyer & Ramakrishnan, 1999) que apenas computam agregações que satisfaçam condições mínimas, Iceberg-Cube.

2.5 Arquitectura OLAP

Os sistemas de processamento analítico são uma mais-valia para analistas que pretendem analisar os dados referentes ao seu negócio. São sistemas capazes de dotar o utilizador de capacidade de exploração dos dados sobre multiperspectivas. São por isso sistemas que consomem dados armazenados e os disponibilizam para o utilizador, ou em certos casos para técnicas de mineração de dados (Figura 2.9).

[4] *hashing* – funções de transformação de sequências de caracteres num valor ou sequência, geralmente, mais curta, ou numa chave da sequência original. São utilizadas, por exemplo, para indexar e recuperar dados em bases de dados de forma rápida e também em algoritmos de criptografia.

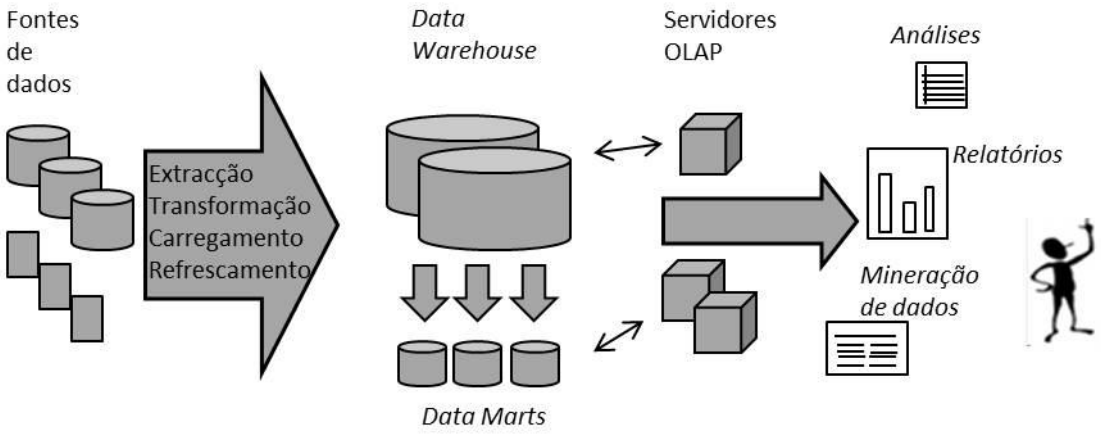


Figura 2.9 - Arquitetura *ent-to-end* (Chaudhuri & Dayal, 1997)

Os sistemas de processamento analítico têm várias implementações possíveis, entre elas as mais comuns são a Relacional OLAP (ROLAP), Multidimensional OLAP (MOLAP), Híbrido OLAP (HOLAP) e *Desktop*[5] OLAP (DOLAP), escolhidas segundo as necessidades de exploração ou a quantidade de dados envolvida.

Um sistema de processamento analítico Relacional, ROLAP, é baseado no acesso directo a fontes de dados relacionais. Esta arquitectura não exige uma pré-computação das agregações e baseia-se numa consulta directa à fonte de dados no momento da consulta. Isto permite a manipulação de grandes quantidades de dados, visto que não exige memória para pré-armazenamento dos dados, no entanto pode implicar consultas com tempos de resposta longos implicados pelo processamento dos dados no momento da consulta. Este tempo de resposta pode ser minimizado se a fonte estiver preparada para responder a grandes volumes de dados. Contudo as limitações deste tipo de arquitectura estendem-se às limitações impostas pelo uso de SQL nas consultas, dificultando por exemplo, cálculos complexos de métricas. Para minimizar o custo da consulta com um sistema ROLAP é frequente recorrer-se a tabelas agregadas disponíveis no sistema fonte, no entanto é implícito o custo de processamento e armazenamento destas tabelas.

[5] *Desktop*, termo inglês para área de trabalho.

Na arquitectura MOLAP os dados são pré-calculados e armazenados em estruturas multidimensionais no servidor OLAP. Esta técnica é recomendada para respostas rápidas pois os dados estão já pré computados sendo apenas necessário o tempo de consulta. Contudo esta solução tem limitações contrárias ao ROLAP. Nesta arquitectura as limitações são relativas à quantidade de dados disponíveis para a consulta. Esta limitação advém da necessidade de pré-calcular os dados do cubo, impondo limitações no processamento e armazenamento das agregações. Visto que todos os cálculos são pré-gerados quando o cubo é criado, o conjunto de métricas disponíveis será aquele que foi definido e calculado à partida na construção do cubo, limitando os dados disponíveis para consulta.

A arquitectura Híbrida OLAP é uma combinação do melhor dos dois "mundos", permitindo ao utilizador fazer consultas sobre dados pré-calculados (MOLAP) e permitindo também a possibilidade de fazer consultas sobre dados que não estejam disponíveis nos cubos através de consultas directas ao sistema fonte (ROLAP). Este tipo de solução continua a limitar os dados de consulta rápida disponíveis para consulta de grandes volumes e a poder ter tempos de resposta longos em queries que sejam consultadas na fonte, contudo o ganho está na possibilidade de disponibilizar um conjunto de agregações para respostas rápidas, não limitando as consultas ao universo de agregações previamente calculadas.

Desktop OLAP é a arquitectura escolhida para disponibilizar um cubo multidimensional par a um dado utilizador na sua própria máquina. Os dados são processados previamente independentemente da estrutura da fonte e ajustados às necessidades desse mesmo utilizador. Uma vez os dados entregues, o utilizador pode fazer as suas pesquisas normalmente embora com as limitações de apenas poder consultar os dados disponíveis na estrutura pré-calculada (M. Anandarajan, A. Anandarajan & Srinivasan, 2003).

Capítulo 3

Métodos para a Reestruturação de Hipercubos

3.1 Identificação da Problemática

Os sistemas de processamento analítico são cada vez mais um factor diferenciador no trabalho dos analistas de qualquer negócio, permitindo-lhes um acesso rápido e orientado à consulta. No entanto, para que sejam capazes de fornecer respostas com um bom desempenho, torna-se necessário recorrer muitas das vezes à pré-computação e à materialização apriori de cubos. Contudo este tipo de soluções, devido às grandes quantidades de dados que usualmente envolvem, acarreta grandes custos em termos de espaço de armazenamento, tempo de cálculo, tempo de materialização e tempo de actualização. Dadas estas dificuldades, verificámos que os recursos chave são o espaço e o tempo, e que a materialização total de um cubo se torna, na prática, incomportável.

É neste contexto que surge a problemática de selecção de vistas a materializar (Harinarayan, Rajaraman & Ullman, 1996) (Gupta, 1997) que é definida por Chirkova, Halevy & Suciu (2002) como:

> "… seja $\mathcal{R}$ o esquema de base de dados, $\mathcal{B}$ o espaço de armazenamento disponível, $\mathcal{Q}$ a carga de processamento sobre o esquema $\mathcal{R}$, C uma função de

> estimativa de custo para processar a consulta e $\mathcal{E}$ uma função para estimar o tamanho das consultas sobre $\mathcal{R}$. O problema de selecção de vistas é encontrar um conjunto de vistas V sobre $\mathcal{R}$ cujo tamanho total é no máximo $\mathcal{B}$ minimizando $C(\mathcal{R},V,Q)$".

Esta mesma problemática tem vindo a ser explorada por vários autores desde 1996 e desde essa altura que é classificada como um problema *NP-hard,* tal como Harinaryan, Rajaraman & Ullman provaram (Harinarayan, Rajaraman & Ullman, 1996). Assim, importa sobretudo reduzir o custo da consulta, tendo em consideração o que isso implica. Por outro lado, esta diminuição do custo de interrogação (Ci) será mais efectiva quanto mais disponível estiver o resultado questionado, o que implica que o resultado esteja materializado. levando-nos a considerar o custo de materialização e de manutenção (Cm) dos dados disponíveis.

Muitos foram os autores que propuseram várias soluções para minimizar os custos desta problemática. A primeira técnica a ser abordada, em 1996 (Harinarayan, Rajaraman & Ullman, 1996) utilizou algoritmos *Greedy*, uma das técnicas mais recorrentes na literatura para solucionar o problema de selecção de vistas. Este trabalho propõe uma definição de *lattice* para modelos multidimensionais e aplica um algoritmo que tem em conta o benefício de cada vista. Para tal, calcula o custo pelo número de linhas da mesma. Este algoritmo mesmo apresentando resultados bastante satisfatórios, para o tempo de resposta à interrogação, tem como desvantagem o tempo de decisão de quais as vistas mais apropriadas para materializar, o que o torna bastante custoso.

Este mesmo algoritmo foi alvo de várias optimizações e comparações. Em 1998 Shukla, Deshpande & Naughton (1998) propõe o PBS (*Pick By Size*), uma optimização do algoritmo de Harinarayan, Rajaraman & Ullman, com recurso a *chunks* e que tem como *input* o espaço disponível para a pré computação, o que o leva a considerar as limitações de espaço que não tinham sido levadas em conta pelo primeiro algoritmo. Mais recentemente Kumar, Haider & Kumar (2011) propõem o VRGA ("*View Recommendation Greedy Algorithm*") como um algoritmo de duas fases que determina as top-T vistas de uma *lattice* multidimensional, superando o problema de escalabilidade do algoritmo proposto por Harinarayan, Rajaraman & Ullman.

Por sua vez, Gupta em 1997 (Gupta, 1997) recorre a índices e a algoritmos *Greedy* com grafos *AND/OR* para a representação das várias possíveis materializações. Esta mesma abordagem é a

opção de Theodoratos & Bouzeghoub em 2000 para uma abordagem ainda mais abrangente, tendo em consideração as preocupações de processamento e manutenção. Mais tarde, em 2001, Liang, Wang & Orlowska propõe um algoritmo heurístico em duas fases que considera o espaço ocupado pelas vistas que propõe materializar assim como o custo de manutenção das mesmas, para tal, este autor, recorre a grafos OR. Ainda em 2000 Shukla, Deshpande & Naughton considera o problema de selecção de vistas para modelos multi-cubo. Neste trabalho propõe três algoritmos segundo dois parâmetros. Um dos algoritmos corre localmente a cada um dos cubos mas não considera métricas derivadas e cubos virtuais. Os outros dois algoritmos escolhem as agregações entre todos os cubos disponíveis, dos dois algoritmos, um considera métricas derivadas e outro não. Por sua vez, Bauer & Lehner em 2003 apresenta o algoritmo "*distributed node set greegy*", a primeira proposta de selecção de vistas para uma solução com cubos distribuídos. Esta proposta considera uma *lattice* de agregações distribuídas, assim como, cálculo de custos de interrogação e benefícios para cubos distribuídos. Já em 2004, Lin & Kuo propõe uma solução com recurso a algoritmos genéticos.

Todas estas propostas incidem na selecção a partir de algoritmos que partem da estrutura do cubo. Além disso, implicam também altos custos na execução dos algoritmos, levando a que a reestruturação não possa ser considerada como resposta imediata às frequentes mudanças de tendências de consultas.

Iceberg Cubing

Para responder aos custos de espaço que são requeridos pelas estruturas que suportam os sistemas de processamento analítico, surgiram em paralelo um conjunto de trabalhos que propuseram materializações baseadas não na estrutura mas sim na frequência dos dados. Este estudo é conhecido como o problema de cubos *Icerberg* que tem por objectivo propor para materialização um conjunto de tuplos mínimos que suportem o maior número de agregações que podem ser questionadas num cubo. Por outras palavras Beyer & Ramakrishnan (1999) e Findlater & Hamilton (2003) definem o problema *Iceberg-Cube* com o objectivo de identificar e computar os valores que serão mais prováveis de consulta, para tal usam as agregações para identificar as células no cubo mais significativas. Assim sendo, os autores propõem uma *query* que simplifica a definição do problema, considerando apenas as dimensões A, B e C:

"SELECT A,B,C, COUNT(*),SUM(X)
FROM TableName
CUBE BY A,B,C
HAVING COUNT(*)>=minsup "

Por suporte mínimo (minsup) entende-se o número mínimo de tuplos onde cada agregação proposta aparece. Por exemplo, para um cubo com três dimensões, se definirmos um suporte mínimo de 25% (2 tuplos de 2^3 combinações), serão seleccionados todas as agregações que responderem a pelo menos 2 ocorrências entre as agregações restantes. Para esta abordagem, que recorre à frequência das agregações nos dados, apareceram vários algoritmos que propõe determinar os tuplos emergentes, e por este ponto de vista, mais apropriados à materialização.

O primeiro algoritmo proposto em 1994 por Agrawal é o *APRIORI* (Agrawal & Srikant, 1994), (Agarwal et al., 1996). Este método começa por percorrer a base de dados para encontrar os valores individuais frequentes. De seguida pesquisa quais os pares mais frequentes das combinações possíveis entre os valores da primeira passagem. Na terceira passagem o processo repete-se e assim sucessivamente até que fiquem seleccionados os tuplos mais frequentes que satisfaçam as iterações anteriores e que obedeçam ao suporte mínimo. Assim, este algoritmo irá iterar o número de vezes equivalente ao tamanho do tuplo proposto no final. No entanto, a exigência de recursos de memória, devido às várias iterações, torna o ARIORI mais desvantajoso.

Em 1999 Beyer propõe o algoritmo *Bottom-Up Computation* (BUC) (Beyer & Ramakrishnan, 1999), um método menos exigente no consumo de memória que considera uma *lattice* em árvore. Primeiro o algoritmo determina qual o valor mais frequente do primeiro atributo, de seguida quais as combinações deste valor com o segundo atributo que aparecem mais vezes, e assim sucessivamente. Quando não consegue determinar uma combinação mais frequente, o algoritmo desce um nó na árvore e prossegue com o atributo imediatamente a seguir. Este comportamento garante que todos os atributos e combinações são tidos em consideração.

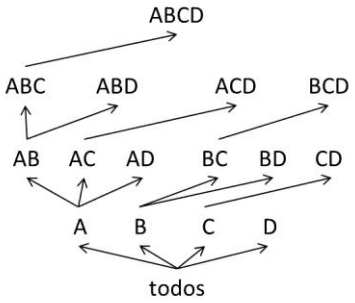


Figura 3.1 - Árvore de processamento do algoritmo BUC para quatro dimensões

Por sua vez o algoritmo *MultiWay* (Zhao, Deshpande & Naughton, 1997) opera baseado no algoritmo *Top-Down Computing* (TDC) partindo da combinação de todos os atributos para as ocorrências mais individuais. No entanto a eficácia deste método é tanto maior quanto mais ordenada, por cardinalidade mais alta, estiver a tabela. Entende-se por cardinalidade o número de valores distintos para cada atributo.

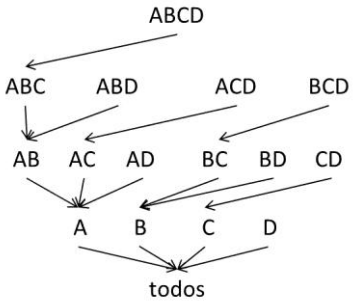


Figura 3.2 - Árvore de processamento do algoritmo TDC para quatro dimensões

Muitos outros algoritmos surgiram para abordar o problema de cubos *Iceberg*, *H-Mine* (Pei et al., 2001), *MM-Cubing* (Shao, Han & Xin, 2004) (Feng et al., 2004) são alguns exemplos, assim como outros que partiram de alguns dos métodos aqui apresentados, como por exemplo *H-Cubbing* (Han et al., 2001) e *Star-Cubing* (Xin et al., 2003).

Tanto esta problemática de cubos *Iceberg* como a anterior estudada, selecção de vistas, são convergentes no intuito da optimização dos sistemas de processamento analítico, no entanto estes estudos mesmo tendo o propósito de satisfazer a consulta e auxiliar os analistas, não têm em consideração a utilização feita pelos mesmos. As consultas feitas são o espelho do que realmente é relevante para o utilizador naquele momento. E se essa consulta se repetir e se tornar um padrão? Será que não é relevante tê-la em consideração na hora da decisão da escolha do cubo que deve ser materializado? Alguns foram os autores que consideraram esta mudança de paradigma. Para tal recorreram desde técnicas de cache a técnicas de *data mining*.

Kotidis & Roussopoulos em 1999 propõe o *Dynamat*, um gestor de caches OLAP, que armazena a granularidade mais fina das consultas efectuadas, mesmo das agregadas. Este trabalho gere a cache à custa de comparações, dada uma *query*, é validado se a resposta a essa *query* já existe em cache, mesmo que numa agregação diferente, e à custa do cálculo de benefícios do fragmento guardado, para poder gerir o espaço disponível. Este cálculo de benefício tem em conta várias variáveis como por exemplo a frequência de acessos ao fragmento ou o tempo passado desde a última consulta. Ainda neste ano Shim, Scheuermann & Vingralek (1999) propõem um gestor de caches dinâmicas que tenta satisfazer a pesquisa através da comparação das consultas e recurso a um grafo dinâmico que correlaciona as mesmas. Em 2002 Kalnis (Kalnis et al., 2002) propõe uma arquitectura PeerOLAP que tira partido dos terminais de cada utilizador, em conjunto com a rede, para disponibilizar as respostas às consultas. Dada uma pesquisa, esta propaga-se pela rede de forma a ser satisfeita por uma cache ou por fragmentos provenientes de várias caches.

Mas o conhecimento do utilizador pode ainda ser mais proveitoso. O recurso a cache, com o armazenamento de resultados de pesquisas, por si só, não demonstra as tendências de cada utilizador, ou de um grupo de utilizadores. Para tirar partido deste tipo de conhecimento surgem trabalhos com recurso a técnicas de *profiling* e aplicação de métodos preditivos. Em 2000 Belo propõe um assistente pessoal que actualiza dinamicamente as vistas mais requeridas por um utilizador, ou grupo de utilizadores, num ambiente distribuído. Esta solução tira partido do conhecimento dos perfis dos utilizadores, não só para reestruturar e recalcular as vistas disponíveis, como também propõe as próximas consultas ao utilizador. Também em 2000 é proposto por Sapia (2000) o PROMISE (*Predicting User Behavior in Multidimentional Information Systems Enviroments*), uma solução com recurso a modelos de Markov para modelar o processo de sucessão de interrogações e um conjunto de variáveis associadas. Esta solução também antecipa a consulta do utilizador com recurso a métodos preditivos.

Definição formal do problema de Selecção de vistas

Considerada a necessidade da materialização de vistas e dada a impossibilidade de materializar e manter todas as vistas possíveis de um *Data Warehouse,* devido aos requisitos de espaço e ao tempo de actualização de todas as vistas, que seria sempre maior do que a janela temporal de actualização disponibilizada, Kalnis, Mamoulis & Papadias em 2002 define o problema de selecção de vistas.

Seja *L* um conjunto de vistas de um cubo e *M* o conjunto de vistas a materializar ($M \subseteq L$). Cada vista $v \in L$ tem três valores associados: a frequência de interrogação f_v, a frequência de actualização g_v e o custo de leitura r_v. Sendo que o custo de responder a uma *query q*, correspondente a uma vista *v*, é o tamanho de *v*, ou seja, o número de tuplos (r_v). Considerando a existência de L, o custo de armazenamento de M é dado por:

$$S(M) = \sum_{v \in M} r_v$$

Seja *u(v,M)* o custo de actualização da vista *v* para o conjunto de vistas a materializar *M*. Então o custo de actualização de M é dado por:

$$U(M) = \sum_{v \in M} g_v \times u(v, M)$$

Seja *q(v,M)* o custo de resposta a uma interrogação *v* quando *M* é o conjunto de vistas materializados. O custo total de responder a interrogações é dado por:

$$Q(M) = \sum_{v \in L} f_v \times q(v, M)$$

Assim sendo, o problema de selecção de vistas é dado pela escolha do conjunto *M*, $(M \subseteq L)$, tal que $Q(M)$ é o mínimo, desde que $S(M) \leq S_{max}$ e $U(M) \leq U_{max}$. Sendo que S_{max} é o espaço disponível para a materialização e U_{max} a janela temporal disponível pata a manutenção das vistas materializadas.

3.2 Caracterização de Utilizadores OLAP

As técnicas de *profiling* sob comportamentos de utilizadores têm sido abordadas em várias finalidades, como por exemplo detecção de fraude, *marketing* personalizado e promoções comerciais (Chen, Dayal & Hsu, 1999). No caso do problema que vimos a abordar, o essencial é a escolha de um conjunto de dados a materializar, que satisfaça as *queries* do utilizador com um tempo de resposta o mais curto possível. Assim percebemos que o elemento de ligação entre os sistemas de processamento analítico e o utilizador são as consultas. Já em 2000, Sapia estudou as características do utilizador OLAP e identificou algumas que considera serem consequentes da filosofia e ambientes de análise associados a estas aplicações:

- Sessão orientada à exploração de dados. Um utilizador de um sistema analítico inicia a sua pesquisa com o intuito de responder a uma estudo que tem em mente, por exemplo, um analista que quer perceber a evolução das vendas de chocolates em várias lojas espalhadas pelo país. Para chegar a esse resultado o utilizador pode começar com consultas parciais, por exemplo "Quais as lojas mais rentáveis do último trimestre?", depois "Quais os chocolates mais vendidos nessas lojas?" e vai analisando até chegar às conclusões que queria obter inicialmente. Todas estas *queries,* feitas por um utilizador e com um propósito de análise, são consideradas no âmbito de uma sessão.

- Interrupções por processos cognitivos. Toda esta actividade entre o sistema analítico e o utilizador é definido por um processo de duas fases que se vão alternando sucessivamente. Na primeira, de interacção directa, o utilizador lança uma consulta e aguarda a resposta. De seguida a segunda fase é a correspondente à análise da resposta. Esta segunda fase é um processo cognitivo do utilizador que o autor chama de caixa preta, um processo não determinístico mas preditivo. Ou seja, não conseguimos determinar que uma dada resposta a uma consulta vai implicar uma decisão ou uma próxima consulta, mas é possível prever uma possível acção do utilizador.

- A navegação iterativa na formulação de *queries*. Os utilizadores OLAP têm duas formas para interagir com as ferramentas. A primeira por uma *query* já feita antecipadamente, por

exemplo, com um relatório, e a segunda por operações iterativas sobre uma *query* de forma a obter novos resultados, por exemplo, operações de *slice* ou *drilldown.*

- Tarefas e padrões específicos de utilizadores. As *queries* que um analista utiliza para chegar ao resultado do seu estudo são o espelho do procedimento adoptado por ele. Se esse utilizador tiver que repetir essa análise, o mais provável é repetir o processo, ou até por um processo bastante idêntico. Isto leva-nos a sequências de consultas que poderão ser padrões de utilização para uma dada tarefa. Assim, dada uma tarefa para um determinado utilizador ou até mesmo para um grupo de utilizadores, pode-se mapear numa probabilidade de sequências de consultas.

Estas características permitem-nos perceber que a utilização destas ferramentas remete-nos à determinação de sessões que podem conter padrões relacionados com a tarefa e com o utilizador, ou grupo de utilizadores.

Para se poder fazer uma observação e estudo do comportamento do utilizador devemos recorrer ao registo de comportamento, que neste caso será através da captação das consultas e dos respectivos dados associados, tudo isto recorrendo ao registo de *logs*. No registo da actividade alguns dos dados que são relevantes registar para os estudos posteriores são: o utilizador, a *query* de pesquisa, o tempo de resposta, a data e a hora do pedido. Estes últimos, podem ser utilizados para estudos de sazonalidade do acesso.

Sessões

Após isso necessitámos de determinar as sessões propriamente ditas. Para tal, é necessário delimitar as sessões. Isto quer dizer, no conjunto de registos de um dado utilizador, perceber quando ele iniciou e terminou uma tarefa de análise. Assim conseguiremos isolar os acessos e agrupa-los de forma a percebermos qual os comportamentos dos utilizadores aquando as suas pesquisas.

Em mineração de dados *web*, foram desenvolvidas técnicas para melhorar a qualidade dos sites e a satisfação dos utilizadores. Estas recorrem, usualmente, a métodos baseados no estudo comportamental dos seus utilizadores. Estes estudos têm fins comerciais e também de

aconselhamento de sites identificados como da preferência do utilizador, com base no seu perfil. Estas recomendações são traçadas segundo o estudo feito sobre os percursos escolhidos por utilizadores anteriores. Para este objectivo, este tipo de técnicas passa por processos de captação e limpeza de dados, identificação de utilizadores, definição de sessões, entre outras.

Ao longo desta dissertação vamos fazer em paralelo o estudo de sessões dos utilizadores dos sistemas de processamento analítico com trabalhos existentes sobre técnicas aplicadas a mineração de dados web, como é o caso de sessões web, aproveitando o conhecimento e trabalho já existente nesta área. Esta opção deve-se à semelhança das acções dos utilizadores, em ambas as situações os utilizadores fazem pedidos a servidores, analíticos ou web, para fazer consultas de informação e após a sua avaliação decidem a próxima acção, de abandonar a sessão ou fazer o próximo consulta / clique respectivamente.

A definição das sessões em trabalhos de sessões *web* pode ser caracterizada como pró-activa, identificando as sessões ao mesmo tempo que os pedidos estão a ser registados, ou reactiva, quando as sessões são definidas durante a leitura dos ficheiros de *logs* (Spiliopoulou et al., 2003). No caso dos sistemas de processamento analítico, a definição das sessões pode surgir após a consulta, avaliando das *logs,* mas como a identificação do utilizador é feita à partida por autenticação, o sistema pode ser preparado para identificar a sessão no seu decorrer. No entanto podemo-nos deparar com situações em que os utilizadores não efectuem o *logout* após uma tarefa e passado algum tempo continuem nessa sessão para fazer uma nova pesquisa referente a uma análise diferente.

Algumas heurísticas são apresentadas na literatura para resolver o caso de definição das sessões. Algumas consideram um tempo máximo de sessão, e particionam as sessões por esse tempo. No entanto, há heurísticas que definem um tempo máximo de inactividade, que vem a resolver o problema de sessões onde o utilizador não efectua *logout*. Assim estas heurísticas definem que a diferença temporal entre duas as páginas / consultas $t_{c+1} - t_c$ não deve exceder o tempo de inactividade definido.

tempo	consulta	t_{c+1}-t_c
Sessão 1		
10:00	c1	-
10:05	c2	5min
10:07	c3	2min
Sessão 2		
10:40	c4	33min
10:48	c5	8min
Sessão 3		
11:30	c6	42min

Figura 3.3 - Exemplo de definição de sessões para um tempo de inactividade de 30 minutos

Outras fontes de dados

Tal como em mineração de dados web, todo este processo pode ser enriquecido com outras fontes de dados. No caso OLAP uma mais-valia, para conseguirmos identificar a estrutura emergente que melhor se adequa a cada utilizador ou grupo de utilizadores, será a estrutura do cubo ou cubos que estão a ser consultados, a disponibilização da estrutura de suporte analítico permitirá mapear as consultas. Esta estrutura pode ser disponibilizada em vários formatos, como por exemplo *xml*, (Pentaho Corporation , 2011).

Integração de dados

Após a recolha das consultas, registo de *logs*, definição das sessões, composta pela sequência das consultas, e a recolha dos dados referentes às estruturas que suportam essas mesmas consultas analíticas, é importante integrar estes dados em estruturas que facilitem a pesquisa e exploração dos mesmos. No caso dos estudos de sessões *web*, a opção recai por um *Data Webhouse* (Dweb), uma instanciação de um *Data Warehouse* (DW) para fontes relacionadas com dados *web*.

No caso OLAP a situação é idêntica, nesse sentido, o DW deve suportar duas estruturas multidimensionais, uma para os dados provenientes dos registos log, que registará todas as

queries feitas pelo utilizador, e a outra para guardar as sessões identificadas. Estas estruturas são ainda enriquecidas pelos dados provenientes da estrutura do cubo e por métricas calculadas, como por exemplo o tempo de sessão e número total de *queries*. Assim sendo a primeira permite uma análise sobre as *queries* lançadas pelos utilizadores e a segunda o estudo do comportamento dos utilizadores ao longo do seu trabalho (Kimball et al., 1998) (Kimball & Caserta, 2004).

3.3 Padrões Sequenciais

Com os dados todos guardados em estruturas adequadas, seguem-se os procedimentos de análise com intuito da avaliação levar a uma nova decisão de cubo mais adequada à utilização. A mineração de dados é um processo não trivial de descoberta de padrões de conhecimento em bases de dados, textos, imagens, etc. Estes padrões, que à partida não são perceptíveis, devem ser válidos, inovadores, úteis e compreensíveis (Fayyad, Piatetsky-Shapiro & Smyth, 1996). A mineração de dados recorre ao uso de várias técnicas para alcançar os diversos objectivos. Por sua vez, estes mesmos objectivos podem variar, conforme a área de trabalho alvo, estatística, bases de dados, inteligência artificial, *data warehouses*, etc. Entre as muitas técnicas disponíveis, as mais usadas são: a classificação, a previsão, o *clustering*, as regras de associação e os padrões sequenciais. O processo de utilização de técnicas de mineração de dados divide-se em três fases, a preparação dos dados, a descoberta de padrões/conhecimento e por fim uma avaliação dos resultados. Este processo deve ser iterativo e melhorado a cada ciclo, de forma a afinar o processo e obter dados mais vantajosos e próximos da realidade. No entanto antes de se iniciar a preparação dos dados, deve haver uma avaliação do domínio de análise, das fontes de dados e identificar que tipo de resultados se pretende obter. Na preparação dos dados é feita a recolha, transformação e integração dos dados. É nesta fase que os dados serão limpos de ruídos, define-se algoritmos de tratamentos de nulos, excluem-se dimensões desnecessárias à análise e transformam-se os dados para serem consumidos pelos algoritmos de mineração de dados. Na fase seguinte, os dados são submetidos aos algoritmos seleccionados e desses extraem-se padrões e conhecimentos que serão avaliados na fase final do processo e que não eram perceptíveis à partida. Nesta fase final, cabe ao analista perceber se as técnicas aplicadas foram apropriadas e se trouxeram uma mais-valia de conhecimento, identificar se e quais os padrões que trazem ganho

em relação ao que era conhecido antes do processo (Fayyad, Piatetsky-Shapiro & Smyth, 1996) (Chen, Han & Yu, 1996).

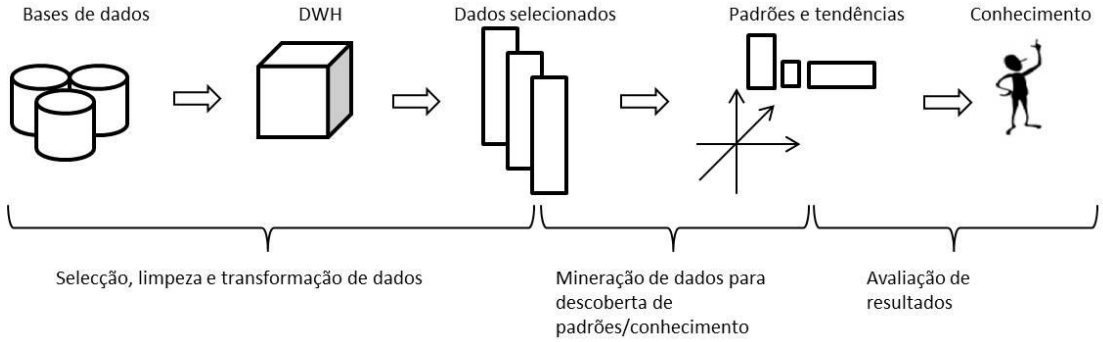


Figura 3.4 - Sequência de acções para descoberta de padrões e conhecimento

A seguir será feita uma breve apresentação de passos e técnicas relacionadas com a mineração de dados e úteis para o estudo que vimos a realizar.

Modelação de dados

No processo que temos vindo a paralelizar com a mineração de dados *web*, conseguimos determinar um conjunto de *queries* $Q = \{q_1, q_2, \ldots, q_n\}$, assim como um conjunto de sessões $S = \{S_1, S_2, \ldots, S_j\}$, em que cada sessão é dada por uma sequência de *queries* pertencentes ao conjunto Q, $S_t = \{q_1^t, q_2^t, \ldots, q_l^t\}$. Neste conjunto podem ainda ser associados pesos a cada ocorrência de *queries* numa dada sessão. Este peso pode ser binário, associando há ocorrência ou não da dada *querie* nessa sessão, ou um valor que determine um valor relevante dessa mesma ocorrência da *querie* na sessão, por exemplo, o tempo de execução da *querie* ou o tempo de inactividade entre essa *querie* e a *query* seguinte. Neste último caso, para a última *querie* da sessão, pode-se optar pelo valor médio na sessão.

Esta modelação de dados, permite-nos ainda criar matrizes de execução de *queries* por utilizadores, descartando a sequencialidade de ocorrência. Por exemplo, considerando o conjunto de *queries* $Q = \{q_1, q_2, q_3, q_4, q_5, q_6\}$ e uma matriz de ocorrências Figura 3.5:

query / utilizador	q_1	q_2	q_3	q_4	q_5	q_6
U_1	0	1	1	1	0	0
U_2	1	1	1	1	1	0
U_3	1	0	1	1	0	1

Figura 3.5 - Exemplo de uma matriz de ocorrências

Neste exemplo (Figura 3.5), quer-se-ia dizer que o utilizador 1 lançou as *queries* q_2, q_3 e q_4 durante a sua sessão. Este tipo de modelação pode ser útil para técnicas que tomem em consideração a sequência das consultas, como por exemplo *clusters* de sessões, para descoberta de grupos de utilizadores, ou regras de associação, para descoberta de ocorrências relacionadas entre *queries* (Liu, 2007).

Análise estatística

A análise estatística, por sua vez, remete-nos à geração de relatórios e análises sobre as diversas informações que podem ser armazenadas e extraídas através de um *data warehouse*. No caso da problemática em análise, podem ser feitas pesquisas sobre as consultas pelo registo de *logs*, como por exemplo, as *queries* mais e menos consultadas ou as médias de tempo de execução, ou então métricas sobre as sessões, como tempos médios de sessões, sessões mais longas ou mesmo o número de *queries* consultadas por sessão para um dado utilizador, assim como outras análises quando os dados estão enriquecidos, por exemplo, determinar qual o período do dia mais recorrente entre os registos para um conjunto de utilizadores.

Padrões Sequenciais

No caso do estudo que temos vindo a realizar, temos observado as vantagens que podemos tirar da sequência das *queries*. Por exemplo, se soubermos que a seguir a uma *query* $q_{i,}$ que consulta as dimensões <`Tempo, Cliente`>, existe uma grande probabilidade de o utilizador executar a *query* $q_{i,}$ onde consulta <`Tempo, Cliente, Produto`>, podemos tirar partido desse conhecimento para prever a próxima consulta. As técnicas de padrões sequenciais tentam

encontrar sequências que se repitam num dado conjunto de sessões. Este tipo de técnicas leva-nos a descobrir sequências de consultas que o utilizador, ou grupo de utilizadores, repete e, em conjunto com as observações que temos vindo a fazer, decidir quais devem estar materializadas.

À semelhança das transacções Web, as interrogações analíticas e respectivas sessões são fonte de dados para a aplicação de cadeias de *Markov*. Esta técnica permite-nos fazer a previsão de acessos a uma dada vista (ou *query*) e pode ainda ser visualizada e analisada como uma grafo (Deshpande & Karypis, 2004) (Sarukkai, 2000) (Borges & Levene, 1999). Para a definição da cadeia de *Markov* é então necessário o conjunto de estados, que serão as *queries* lançadas pelo utilizador $Q = \{q_1, q_2, \ldots, q_n\}$, assim como uma matriz de transacção T, de ordem n, com a probabilidade de passar de um estado para o estado seguinte, ou seja, a probabilidade de um utilizador lançar a *query x* e de seguida a *query y*. Esta técnica tem ainda associado o conceito de ordem da cadeia, que é o número de estados que considerámos para a definição da probabilidade de chegar a um dado estado, ou seja, uma cadeia de ordem *n*, considera *n-1* estados anteriores para prever o estado seguinte. As cadeias de *Markov* são ainda vantajosas pela sua capacidade de serem incrementais, não havendo assim necessidade de reconstrução total da cadeia cada vez que for necessário acrescentar novos dados.

Consideremos o exemplo a seguir, para uma cadeia de Markov de ordem 1 (Figura 3.6).

Id de sessão	Sequência de interrogações
S_1	q_1 -> q_2 -> q_3 -> q_4 -> q_5
S_2	q_1 -> q_2 -> q_3
S_3	q_1 -> q_2 -> q_3 -> q_5
S_4	q_2 -> q_4 -> q_5
S_5	q_2 -> q_4 -> q_5 -> q_3
S_6	q_2 -> q_4 -> q_1 -> q_5
S_7	q_4 -> q_1 -> q_2 -> q_5

Figura 3.6 - Exemplo de algumas sequências de interrogações

Para gerarmos a cadeia de *Markov* correspondente a este exemplo de sequência de interrogações primeiro devemos considerar o estado inicial -2. De seguida determinar a probabilidade de passar do estado inicial para cada uma das *queries* do conjunto Q, dada pelo número de vezes que essa *query* foi consultada como a primeira de uma sessão $/q_1/$, a dividir pelo número total de consultas efectuadas.

$$PI(q1) = |q1| / consultas$$

De seguida calcula-se a probabilidade das passagens para o estado final, -1, desta vez, apenas das *queries* que ocorram como últimas de uma sessão. De seguida calculam-se as restantes probabilidades entre os estados. Esta é dada pela contagem do número de vezes em que a *query y* foi precedida pela *query x*, $/ q_x \rightarrow q_y /$, a dividir pelo número de consultas feitas com a *query x*.

$$P(qx \rightarrow qy) = | qx \rightarrow qy | / | qx |$$

O exemplo que estamos a seguir daria a cadeia de Markov representada na Figura 3.7.

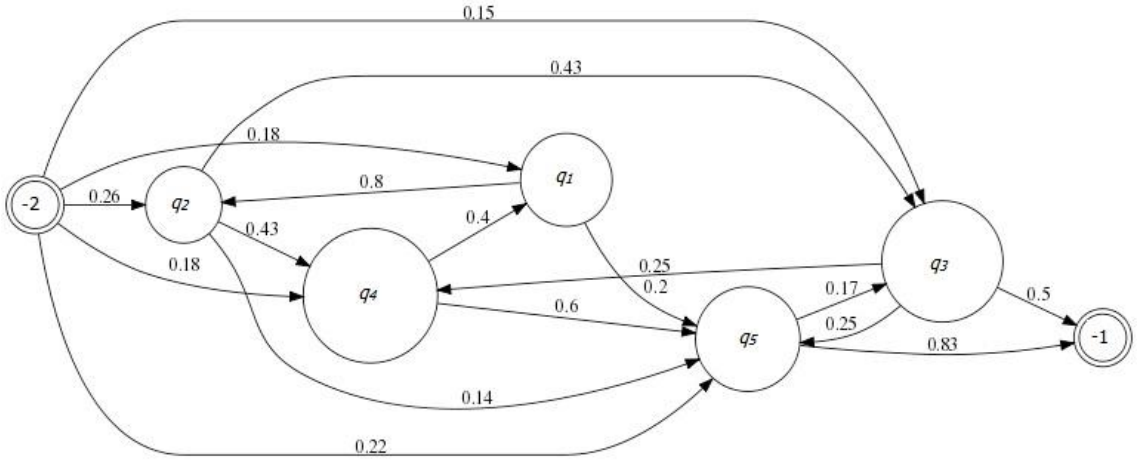


Figura 3.7 - Visualização de exemplo de Cadeia de Markov

Após a construção das cadeias de *Markov*, o passo seguinte é o cálculo do percurso de consultas mais frequente, ou caminhos mais frequentes. Visto que estamos na presença de grafos, os caminhos mais frequentes podem ser calculados com recurso a algoritmos de pesquisa em grafos

que nos darão todos os caminhos passíveis de ser percorridos no grafo. Este tipo de cálculo pode ser feito através de uma pesquisa em largura, *Breast-First Search* (Weiss, 1993) ou em profundidade *Depth-First Search* (Tarjan, 1972). No entanto para a determinação dos caminhos mais frequentes são necessários os conceitos de suporte e confiança. O primeiro é dado pela probabilidade inicial do caminho. Já a confiança é definida como a probabilidade de um utilizador percorrer um dado caminho, que pode ser calculada pelo produto das probabilidades das ligações da cadeia. Este tipo de parâmetros é determinado à partida e passados como *input* ao cálculo dos caminhos (Marques & Belo, 2010) (Marques, 2009).

3.4 Classes de equivalência

Todas as técnicas que temos vindo a estudar têm permitido perceber várias formas para recolhermos dados que nos permitam caracterizar o uso de ferramentas analíticas, assim como definirmos áreas de um ou mais cubos que são mais visitadas e que serão mais propícias à materialização. As classes de equivalência são mais uma técnica estudada que tem intuito de identificar a sobreposição de domínios de consulta em hipercubos. Esta técnica não é um conceito recente na área da matemática e tem vindo a ser explorado pelos domínios das Ciências da Computação. Conceptualmente uma classe de equivalência é dada por (Scheinerman, 2003):

Seja uma relação de equivalência R num conjunto C e seja a ϵ C. A classe de equivalência de a, representada por [a], é o conjunto de todos os elementos de C relacionados com a pela relação R, ou seja

$$[a] = \{ x \epsilon C : x R a \}.$$

Em processamento analítico, este conceito já foi aplicado por outros autores, em 2001 Niemi, Nummenmaa & Thanisch apresentaram um trabalho que recorre a classes de equivalência e às *queries MDX* lançadas pelo utilizador para propor uma reestruturação de um hipercubo que reflectisse a exploração dos cubos. Este tipo de técnicas permite uma aproximação entre a estrutura do cubo e a exploração efectivamente feita pelo utilizador, ou seja, o espaço de soluções é determinado pelas *queries* multidimensionais lançadas. Para uma melhor percepção desta ideia,

consideremos um espaço de soluções a duas dimensões no qual representámos um conjunto de vistas (também a duas dimensões), de forma a identificar sobreposições (Figura 3.8).

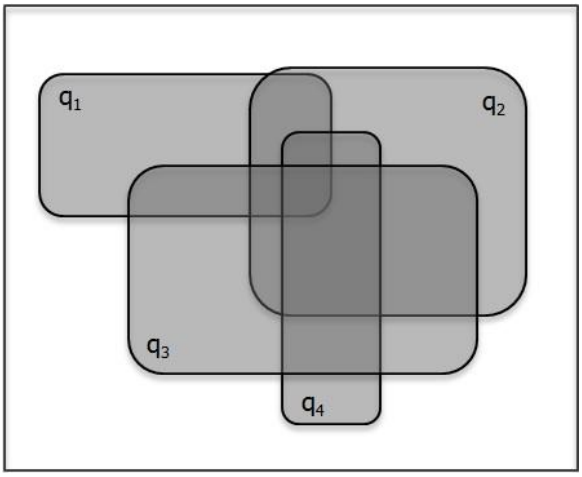


Figura 3.8 - Sobreposição de queries a duas dimensões

Colorindo cada uma das vistas geradas, a sua eventual sobreposição será representada por uma cor mais escura quanto maior for o número de sobreposições das vistas. No limite, a zona mais escura representará o bloco de dados mais solicitado, possivelmente identificando uma classe de equivalência. No entanto, esta observação de carácter empírico, não é o suficiente para a reestruturação efectiva do cubo mas sim um princípio de avaliação.

No seguimento desta ideia em (Niemi, Nummenmaa & Thanisch, 2001) é proposto um algoritmo que divide *queries* MDX em classes de equivalência (Algoritmo 3.1).

Input: Esquema H de um cubo OLAP, um conjunto Q de *queries* MDX lançadas a H e um conjunto de dependências funcionais sobre os atributos em H.
Output: Um conjunto de classes de equivalência de *queries* em Q.

1. Para cada *query* Q_i em Q, adicionar os atributos da cláusula SELECT ao conjunto X_i. Ignorar dimensões demasiados gerais, por exemplo, tempo e localidade.
2. Para cada conjunto X_i, com base nas informações de dependências funcionais, construir o conjunto Y_i de chaves de dimensões do conjunto H, tal que Y_i contenha a chave da dimensão K se existir o atributo $A \in X_i$, para $A \in K^{+}$.
3. Construção de classes de equivalência para as *queries* da seguinte forma:
   Duas *queries* Q e Q' pertence à mesma classe de equivalência E se podemos formar uma sequência de *queries* existentes $<Q_0 = Q, Q_1, \ldots Q_n, Q' = Q_{n+1}>$ tal que $Y_i \cap Y_{i+1} \neq 0$, $0 \leq i \leq n$, onde Y_i denota o conjunto de chaves de dimensões da *query* Q_i.
4. (Uma fase adicional para melhorar a eficiência para a análise menos detalhada dos cubos):
   Para cada classe de equivalência E_i:
   Para cada *query* Q em E_i:
   Se há um $Q' \in E_i$ com as mesmas dimensões que Q mas algumas dimensões de Q' estão em níveis mais detalhados do que em Q, então, construir uma nova classe de equivalência E 'como se segue: $E' = E_i - \{Q'\}$
5. Output do conjunto de classes de equivalência obtido.

Algoritmo 3.1 – Definição de classes de equivalência (Niemi, Nummenmaa & Thanisch, 2001)

Este algoritmo coloca duas *queries* na mesma classe de equivalência, se directa ou transitivamente as duas tiverem dimensões comuns. Consideremos o exemplo de três *queries* MDX sobre o mesmo cubo.

1. Q_1: *SELECT* dia, grupo_produto *WHERE* lucro, ano.[2010]
2. Q_2: *SELECT* id_empregado, id_cliente *WHERE* lucro
3. Q_3: *SELECT* ano, id_store *WHERE* quantidade
4. Q_4: *SELECT* ano, id_produto, loja *WHERE* quantidade
5. Q_5: *SELECT* dia, id_produto, id_cliente *WHERE* quantidade
6. Q_6: *SELECT* loja, id_produto *WHERE* quantidade

Na primeira fase do algoritmo, seleccionam-se os atributos da cláusula *SELECT*, X_1 = {dia, grupo_produto}, X_2 = {id_empregado, id_cliente}, X_3 = {ano, loja}, X_4 = {ano, id_produto, loja}, X_5 = {dia, id_produto, id_cliente} e X_6 = {loja, id_produto}. De seguida os respectivos conjuntos com as chaves Y_1 = {dia, id_produto}, Y_2 = {id_empregado, id_cliente} Y_3 = {dia, id_loja}, Y_4 = {dia, id_produto, id_loja}, Y_5 = {dia, id_produto, id_cliente} e Y_6 = {id_loja, id_produto}. Destes seis conjuntos podemos formar seis classes de equivalência, E_1 (tempo, produto) = {Q_1, Q_4, Q_5}, E_2 (empregado, cliente) = {Q_2}, E_3 (loja, produto) = {Q_6, Q_4}, E_4 (tempo, produto, loja) = {Q_4}, E_5 (produto cliente, tempo) = {Q_5} e E_6 (loja, tempo) = {Q_3, Q_4}. Além disso, de acordo com o passo 4, a classe de equivalência E1 pode ainda ser dividido em duas, E_1' (dia, produto) = {Q_1, Q_5} e E_1'' (ano, produto) = {Q_4}, visto que as duas *queries* operam em níveis de detalhe diferentes.

Ainda neste trabalho, (Niemi, Nummenmaa & Thanisch, 2001), é proposto um algoritmo para a construção de estruturas de cubos OLAP com base num algoritmo proposto em (Niemi, Nummenmaa & Thanisch, 2000).

Capítulo 4

Reestruturação de Hipercubos à Medida

4.1 O Processo de Reestruturação

Com o processo de reestruturação de hipercubos que aqui apresentámos pretende-se criar um sistema capaz de propor a reestruturação das estruturas multidimensionais com base na observação do comportamento dos utilizadores. Para tal inicialmente devemos observar, em tempo útil, todas as *queries* lançadas sobre hipercubos contidos numa plataforma OLAP e armazená-las em formatos adequados. Posteriormente, com base nessa informação, o processo deve ser capaz de estabelecer novos perfis de utilização OLAP e consequentemente propor as novas estruturas dos hipercubos adaptados à medida desses perfis. Estes novos perfis de exploração multidimensional indicarão assim as estruturas dos hipercubos e assim materializar apenas o que realmente se estimou que vai ser usado.

Para cumprir este objectivo, foi definido um conjunto de fases necessárias à definição de todo o processo e consequentemente à geração das novas estruturas (Figura 4.1).

1. Recolha e armazenamento de informação. Captação das *queries* multidimensionais (MDX) e integração em estruturas de dados apropriadas.

2. Identificação de perfis de utilização. Processamento dos dados, definição de sessões OLAP e reflexos em cadeias de Markov para análise comportamental dos utilizadores.

3. Descoberta de padrões de comportamento. Determinação de classes de equivalência a partir das *queries* realizadas pelos utilizadores.

4. Definição de Assinaturas OLAP e novas estruturas. Definição de uma assinatura de utilizadores OLAP a partir dos padrões observados e consequentemente proposta de estruturas para os hipercubos.

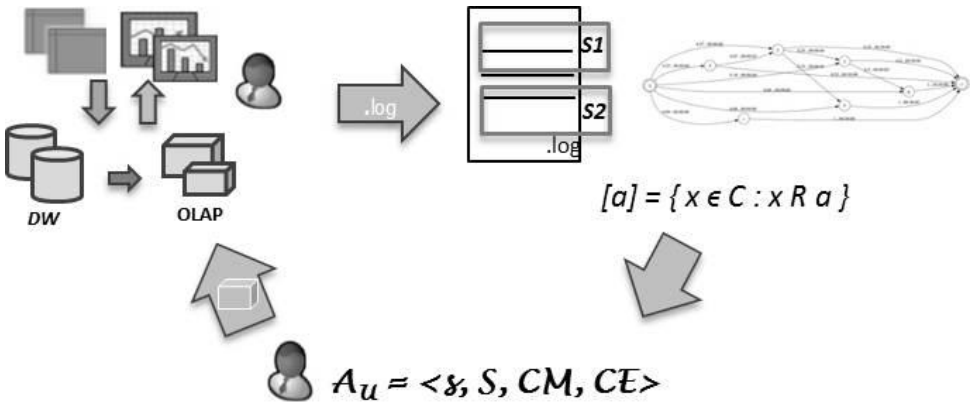


Figura 4.1 - Processo de reestruturação

Refira-se aqui que, todos os trabalhos de estudo e desenvolvimento apresentados e descritos ao longo deste capítulo, para concepção de um método expedito para a reestruturação de hipercubos à medida, foram realizados no âmbito do projecto UPonICE[6] (Belo & Duarte, 2010).

[6] O projecto UPonICE - Optimização de HiperCubos com Base em Perfis de Utilização e Tendências Sazonais – foi um projecto financiado pela Portugal Telecom Inovação, tendo sido realizado ente Março de 2009 e Fevereiro de 2010.

4.2 Recolha e Armazenamento de Dados

Desde o início assumimos que o processo que aqui propomos de optimização da estrutura do hipercubo seria desencadeado através da determinação de perfis de exploração de dados a partir da análise das consultas lançadas pelos utilizadores. Assim teríamos de recriar um ambiente que nos permitisse recolher as *queries* que fossem sendo lançadas e armazená-las em estruturas apropriadas para que posteriormente fossem fonte para as fases seguintes do processo. Para tal recorremos a um conjunto de ferramentas de suporte que nos permitisse simular algumas consultas e implementar um mecanismo de captação da informação dessas mesmas consultas. Além das ferramentas tivemos também que dispor de uma base de dados exemplo que se aproximasse muito de um caso real.

4.2.1 Ferramentas de suporte

Logo no início dos trabalhos foi necessário escolher uma ferramenta OLAP. Como características interessávamo-nos um sistema de processamento analítico versátil, com mecanismos de processamento analítico eficientes, de fácil instalação, integrável com ambientes de desenvolvimento Java[7] e sem custos de licenciamento. Dadas estas características escolhemos trabalhar com o Mondrian, um servidor OLAP de domínio público e código aberto que foi desenvolvido pela Pentaho Corporation[8] (Pentaho Corporation , 2011). Este servidor processa consultas MDX, armazena os dados numa base de dados relacional e apresenta os resultados em formato multidimensional. É desenvolvido em Java, independente da plataforma podendo ser utilizado em diferentes sistemas operacionais. Como é uma ferramenta de código aberto, permite incorporar novas funcionalidades, assim como disponibilizar recursos para estender a linguagem MDX, por exemplo para a criação de novos operadores para as consultas. A instalação do Mondrian é um processo facilitado não só pela simplicidade mas também pela documentação disponível.

[7] Java, linguagem de programação orientada a objectos. Surge em 1995 na empresa *Sun Microsystems* com a liderança de James Gosling. Esta linguagem é compilada para *bytecode* e executado por uma máquina virtual.
[8] Pentaho Corporation, empresa de origem norte americana que desenvolve soluções de Business Intelligence de código aberto. Foi fundada em 2004 por um grupo de executivos vindos de outras empresas com experiência na área.

Para assegurar a comunicação entre o sistema Web de interacção e o sistema analítico, necessitámos de um servidor de aplicações, neste caso o Apache[9] Tomcat (Tomcat, 2011). Este irá permitir o acesso aos hipercubos disponíveis na base de dados multidimensional de trabalho assim como a sua exploração. As consultas realizadas sob os dados nos hipercubos serão enviadas pelo interface *Web* ao servidor analítico, este irá tratar a *query* MDX assim como disponibiliza-la num ficheiro de log que mais tarde o processo irá consumir.

Quanto ao sistema de gestão de base de dados, para suportar a base de dados exemplo, o sistema operacional de monitorização e o sistema de *data warehousing,* escolhemos o MySQL (Oracle Corporation, 2011). O MySQL é compatível com várias plataformas e de licenciamento livre, tem vindo a ser referido como um sistema de gestão de base de dados com bom desempenho e escalabilidade pouco exigente quanto a recursos. Além destas características o MySQL é de fácil integração com o servidor analítico escolhido. Para a parte de administração, desenho e criação de bases de dados recorreu-se ao MySQL Workbrench. Quanto ao desenvolvimento dos componentes necessários a todo o processo descrito neste trabalho, foram desenvolvidos com recurso à linguagem Java.

4.2.2 A base de dados de trabalho

Para ser possível simular as *queries* MDX e consequentemente os perfis de utilização, torna-se necessária a base de dados de trabalho. Mesmo que as exigências dimensionais não fossem o foco, pois o interesse era na interrogação, a dimensão e diversidade dos dados iria ajudar na criação de perfil de utilizadores e definir o modo de actuação desses perfis. Neste caso a escolha é facilitada pela disponibilização de uma base de dados na demonstração do Mondrian. A `FoodMart` é uma base de dados sobre vendas que preenche estes requisitos além de que dispõe de uma estrutura multidimensional bastante rica, alargada e diversificada além de que é sobre uma área de negócio bastante comum nas demonstrações de sistemas de suporte à decisão, a área de vendas, tornando as consultas mais intuitivas.

[9] The Apache Software Foundation, fundação que fornece suporte organizacional, legal e financeiro a projectos de software de código livre. Foi constituída em 1999 como uma sociedade sem fins lucrativos para garantir os projectos Apache, (The Apache Software Foundation, 2011).

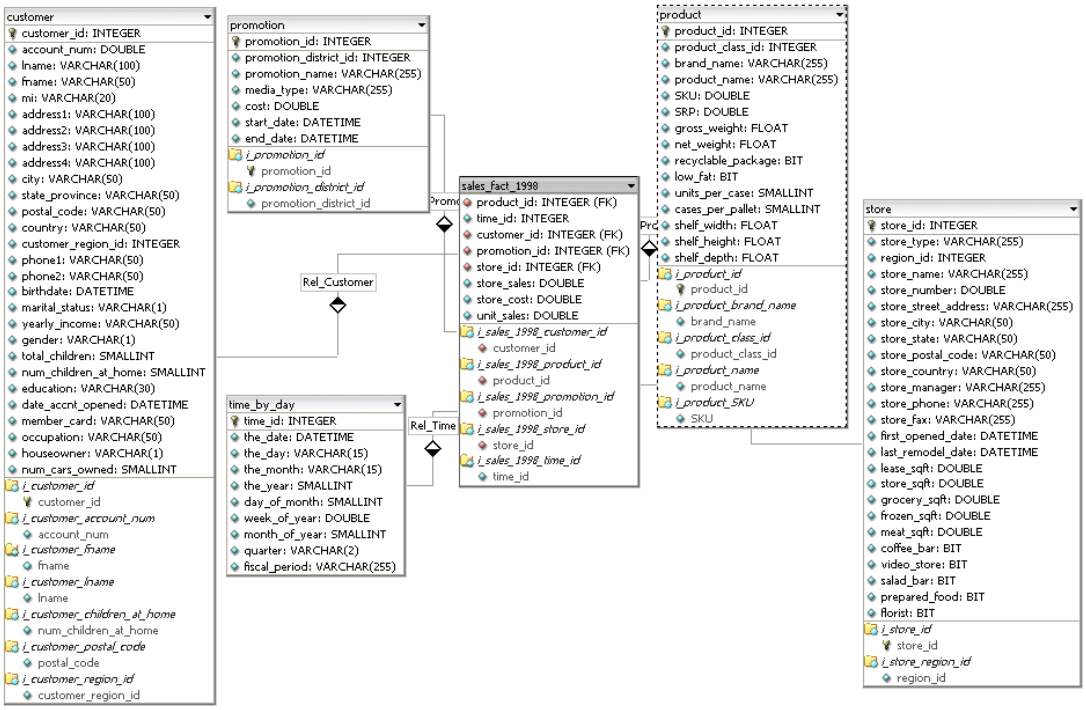


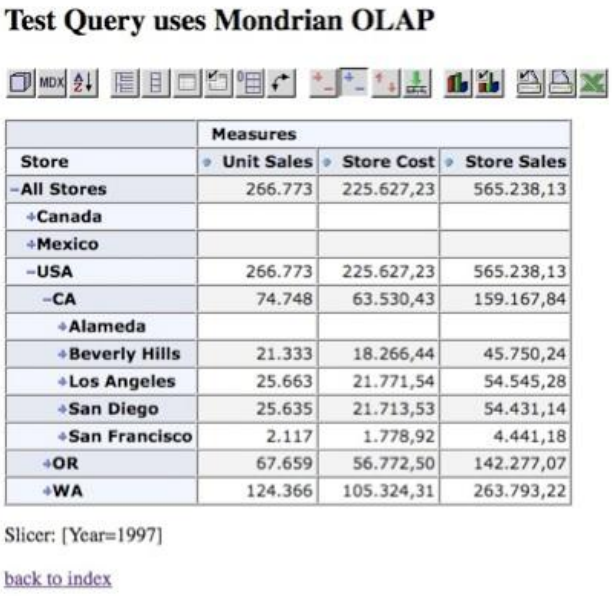
Figura 4.2 - Esquema da `FoodMart`

Na Figura 4.2 podemos observar o esquema em estrela de um *data mart*, que por exemplo, pode ser de apoio às actividades de um departamento de vendas de uma empresa. Este esquema inclui uma tabela de factos (`sales_fact_1998`), com três métricas de análise (`store_sales`, `store_cost` e `unit_sales`) e cinco dimensões (`customer`, `promotion`, `time_by_day`, `product` e `store`). Este conjunto de métricas, dimensões e mesmo as hierarquias presentes nas dimensões permite-nos combinar os dados segundo várias perspectivas e assim criar uma vasta variedade de consultas. Por exemplo se considerarmos a dimensão `store` e a respectiva hierarquia, `store_id -> store_city -> store_state` podemos fazer consultas como "o cálculo das vendas realizadas em cada loja" (`store_id` ou `store_name`) e de seguida agregarmos os dados seguindo a hierarquia, aplicando a função soma (*sum*()) sobre a métrica `store_sales`, passando de nível para nível, de grão mais fino para grão mais grosso (sentido de *rollup*), obteremos as vendas por cidade (`store_city`) e depois por estado (`store_state`) no qual as lojas estão situadas. Apenas devemos ter cuidado com os conceitos

para não construir consultas com agregações que não se materializem em nenhuma realidade chegando a ser inconsistentes.

4.2.3 Exploração das Estruturas Multidimensionais

Nesta fase iremos perceber a potencialidade de navegação do ambiente criado, para posteriormente procedermos à captação das *queries* MDX lançadas. Na Figura 4.3 pode-se observar o interface da aplicação Web que nos permitiu explorar os dados contidos num hipercubo. Apesar da interface de exploração estar longe da sofisticação das actuais ferramentas OLAP que concorrem nos mercados, esta responde às nossas necessidades de navegação e captação das consultas MDX.



Test Query uses Mondrian OLAP

	Measures		
Store	Unit Sales	Store Cost	Store Sales
−All Stores	266.773	225.627,23	565.238,13
+Canada			
+Mexico			
−USA	266.773	225.627,23	565.238,13
−CA	74.748	63.530,43	159.167,84
+Alameda			
+Beverly Hills	21.333	18.266,44	45.750,24
+Los Angeles	25.663	21.771,54	54.545,28
+San Diego	25.635	21.713,53	54.431,14
+San Francisco	2.117	1.778,92	4.441,18
+OR	67.659	56.772,50	142.277,07
+WA	124.366	105.324,31	263.793,22

Slicer: [Year=1997]

back to index

Figura 4.3 - Ambiente de exploração de dados

Conseguimos com facilidade executar as operações mais frequentes em sistemas de processamento analítico, *rollup*, *drilldown*, *slicing*, etc. As operações muitas vezes foram executadas com um simples *click* nas células ou nos botões correspondentes à operação

pretendida e tinham reflexos imediatos. Além desta simplicidade de utilização, também é possível a alteração das *queries* MDX geradas pelas selecções no ambiente gráfico ou mesmo a edição integral da *query* que seja respondida pelo servidor (Figura 4.4).

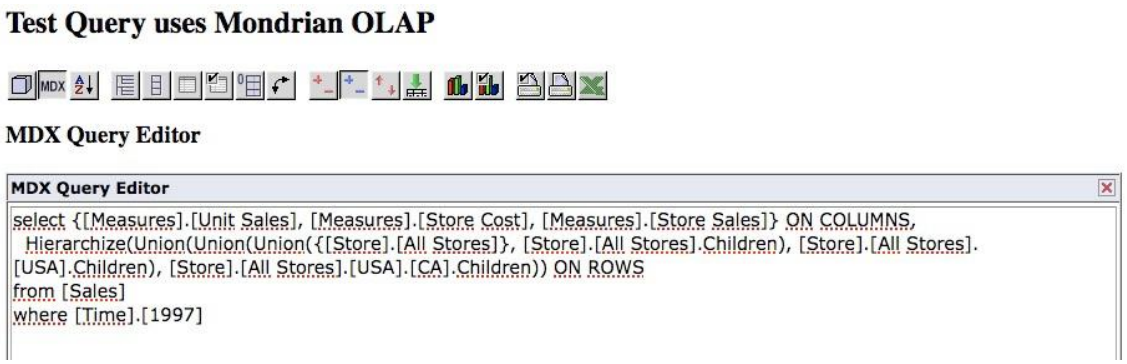


Figura 4.4 - Edição de uma *query* MDX

4.2.4 Recolha e Inventariação das *Queries* MDX

Para que todo este processo fosse possível, o primeiro passo a ser conseguido seria a captação das *queries* MDX lançadas pelos utilizadores. Para tal foi necessário saber como é que o Mondrian recolhia e processava as consultas, no entanto não conseguimos detectar esse processo então recorremos a uma ferramenta de *logging* disponibilizada pela Apache Software Foundation (The Apache Software Foundation, 2011), que foi especialmente concebida para programadores que desejam incorporar no seu código fonte instruções específicas de *logging*, a log4j (Apache Logging Services, 2011). Esta ferramenta disponibiliza uma API[10] que depois de parametrizada e em cooperação com o Mondrian permite o registo das *queries* MDX e alguns dados associados à consulta num ficheiro de *logs*. Para cada instrução MDX lançada ao servidor analítico a log4j gera duas linhas de texto nesse ficheiro com o formato apresentado na Figura 4.5.

[10] http://logging.apache.org/log4j/1.2/apidocs/org/apache/log4j/PatternLayout.html

```
data hora,xxx utilizador DEBUG mondrian.mdx -  número da query: query MDX
data hora,xxx utilizador DEBUG mondrian.mdx -  número da query: tempo ms
```

Figura 4.5 - Informação relativa a cada *query* MDX executada

Na primeira linha é guardada informação relativa à *query* MDX executada, enquanto que na segunda, essencialmente, é acrescido o tempo que o servidor analítico gastou na sua execução. Esta informação é suficiente para suportar o processo de análise das estruturas utilizadas, assim como para identificar os elementos envolvidos (dimensões, medidas, filtros, etc.). Com base na informação registada poderemos criar estruturas próprias para acolher a caracterização das *queries* MDX interrogadas bem como as partes das estruturas multidimensionais que foram acedidas.

4.2.5 A Informação das *Queries* MDX

Com o acesso às *queries* MDX lançadas sobre o servidor e conhecendo os vários elementos multidimensionais envolvidos, armazenámos esses dados num sistema capaz de os acolher. Para conseguir alimentar esse sistema foi necessário desenvolver um pequeno *parser* capaz de analisar cada uma das *queries* MDX e separar os diversos elementos que tínhamos interesse em guardar. Assim sendo, começámos por desenhar e implementar uma base de dados que acolhesse os dados provenientes das *logs* (Figura 4.6).

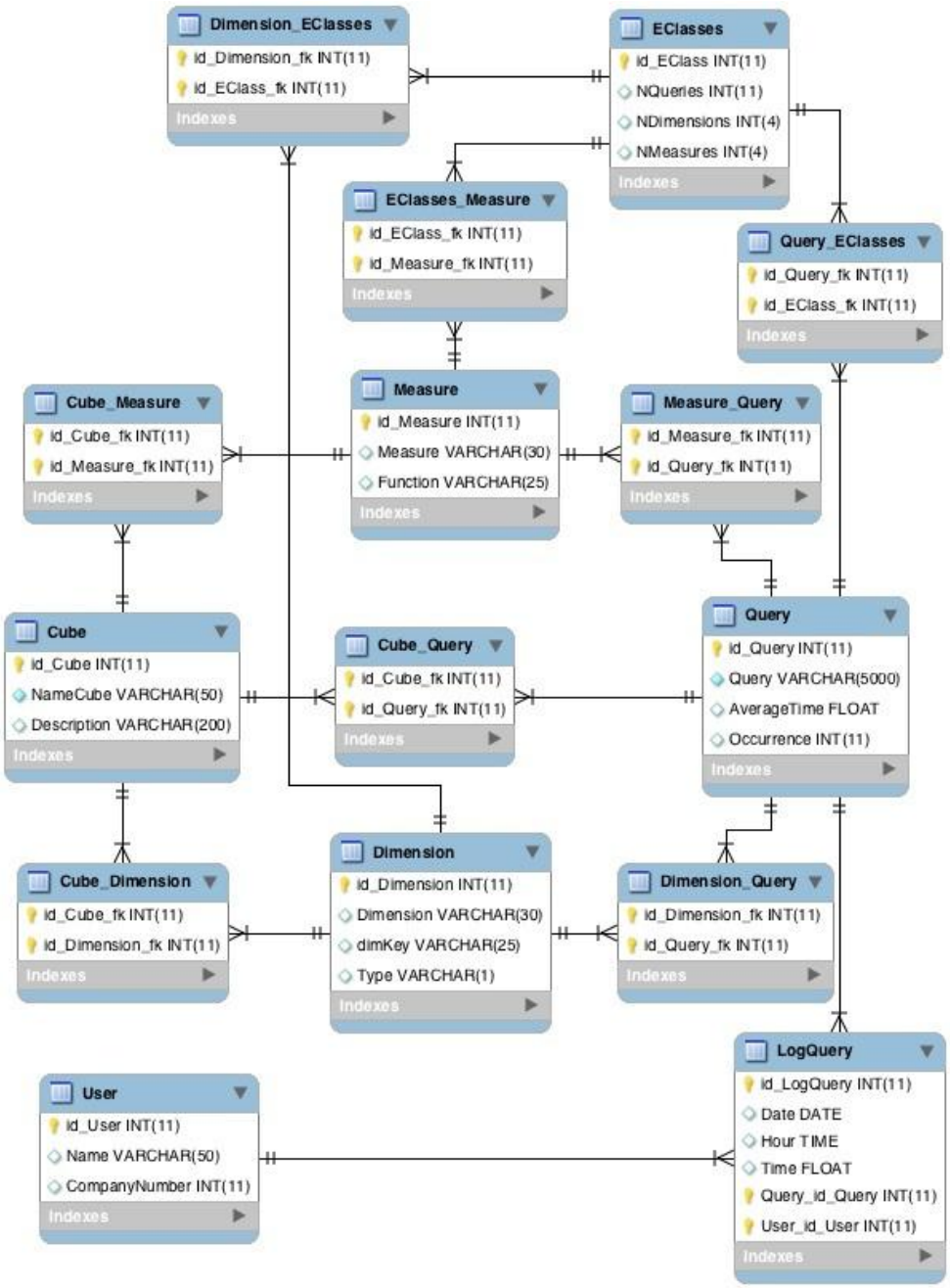


Figura 4.6 - Esquema Lógico da base de dados de acolhimento das *logs*

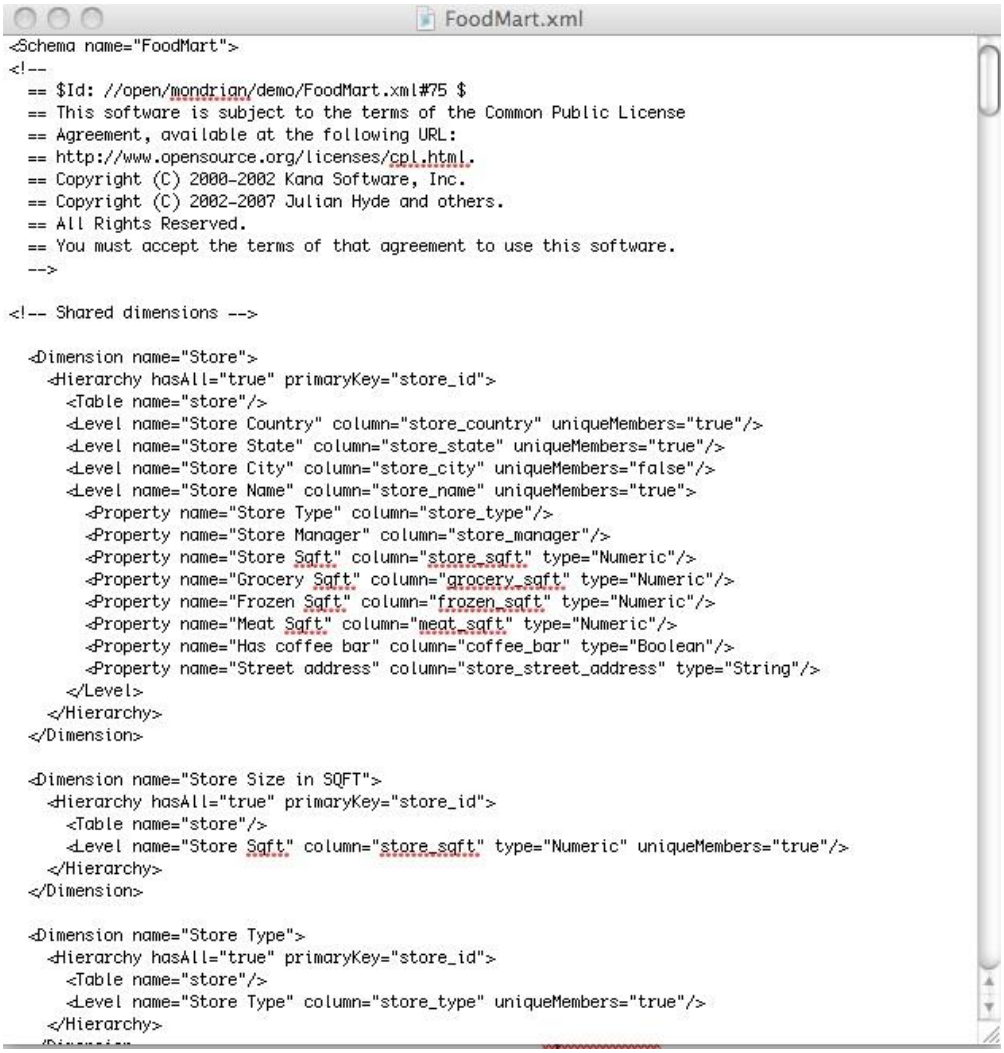
No esquema da base de dados podemos observar a existência de quinze tabelas, onde sete delas correspondem às entidades base envolvidas no processo, `User`, `Cube`, `LogQuery`, `Query`,

`Dimension`, `Measure` e `EClasses`, e as restantes oito são resultantes de relacionamentos N:M identificados entre as entidades referidas, `Dimension_Query`, *`Measure_Query`*, *`Cube_Query`*, *`Cube_Dimension`*, *`Cube_Measure`*, *`Query_EClasses`*, *`EClasses_Measure`* e *`Dimesnion_EClasses`*.

Esta base de dados de suporte ao rastreio foi desenhada para acolher não só as *queries* multidimensionais mas também os elementos que nela se pudessem identificar. Isto levou ao desenho de uma estrutura que acompanha de perto a organização dos componentes da *query*. No entanto para que fosse possível manter a integridade referencial entre os elementos identificados e obtivéssemos coerência entre o que estávamos a registar e a estrutura que estava a ser consultada, recorremos também à fonte da estrutura multidimensional que estava a ser interrogada. Uma vez que o registo nos ficheiros de log depende funcionalmente da estrutura multidimensional, começamos o povoamento da base de dados por extrair os elementos intervenientes nas estruturas multidimensionais, e consequentemente, quando inserirmos os registos log conseguimos identificar os elementos correspondentes.

A definição destas estruturas multidimensionais está guardada em ficheiros XML[11] (Figura 4.7). Para conseguirmos extrair os metadados dos cubos consultados necessitávamos de entender a estrutura do ficheiro assim como perceber o código XML. Visto que já tínhamos definido trabalhar apenas com um cubo de dados, já que essa única estrutura era suficiente para suportar o processo de rastreio de dados que pretendíamos fazer, identificámos esse cubo no ficheiro e a estrutura correspondente ao mesmo. De seguida definimos o processo de captura da informação relativa a esses metadados e estabelecemos a correlação do cubo referido com as respectivas dimensões e medidas (Algoritmo 4.1). Com estes dados aparecem os primeiros registos na base de dados, com a caracterização do cubo alvo do nosso processo de análise, abrangendo as tabelas `Cube`, `Dimension`, `Measure`, `Cube_Dimension` e `Cube_Measures`.

[11] XML, um dos subtipos da SGML (*Standard Generalized Markup Language*, ou Linguagem Padronizada de Marcação Genérica) capaz de descrever diversos tipos de dados. Com o propósito de facilitar a partilha de informações pela internet.


FoodMart.xml

```
<Schema name="FoodMart">
<!--
  == $Id: //open/mondrian/demo/FoodMart.xml#75 $
  == This software is subject to the terms of the Common Public License
  == Agreement, available at the following URL:
  == http://www.opensource.org/licenses/cpl.html.
  == Copyright (C) 2000-2002 Kana Software, Inc.
  == Copyright (C) 2002-2007 Julian Hyde and others.
  == All Rights Reserved.
  == You must accept the terms of that agreement to use this software.
  -->

<!-- Shared dimensions -->

  <Dimension name="Store">
    <Hierarchy hasAll="true" primaryKey="store_id">
      <Table name="store"/>
      <Level name="Store Country" column="store_country" uniqueMembers="true"/>
      <Level name="Store State" column="store_state" uniqueMembers="true"/>
      <Level name="Store City" column="store_city" uniqueMembers="false"/>
      <Level name="Store Name" column="store_name" uniqueMembers="true">
        <Property name="Store Type" column="store_type"/>
        <Property name="Store Manager" column="store_manager"/>
        <Property name="Store Sqft" column="store_sqft" type="Numeric"/>
        <Property name="Grocery Sqft" column="grocery_sqft" type="Numeric"/>
        <Property name="Frozen Sqft" column="frozen_sqft" type="Numeric"/>
        <Property name="Meat Sqft" column="meat_sqft" type="Numeric"/>
        <Property name="Has coffee bar" column="coffee_bar" type="Boolean"/>
        <Property name="Street address" column="store_street_address" type="String"/>
      </Level>
    </Hierarchy>
  </Dimension>

  <Dimension name="Store Size in SQFT">
    <Hierarchy hasAll="true" primaryKey="store_id">
      <Table name="store"/>
      <Level name="Store Sqft" column="store_sqft" type="Numeric" uniqueMembers="true"/>
    </Hierarchy>
  </Dimension>

  <Dimension name="Store Type">
    <Hierarchy hasAll="true" primaryKey="store_id">
      <Table name="store"/>
      <Level name="Store Type" column="store_type" uniqueMembers="true"/>
    </Hierarchy>
```

Figura 4.7 - Extracto do ficheiro XML da definição da `FoodMart`

Input: Ligação à estrutura de dados e ao ficheiro XML
1. Identificar os cubos e inserir os respectivos registos na entidade `Cubo`.
2. Identificar as `Dimensões` e `Medidas` e inseri-las na entidade respectiva `Dimensão/Medida` com os respectivos dados e características.

Algoritmo 4.1 - Processamento do ficheiro XML com os metadados do Cubo

Estando agora a base de dados com os registos dos elementos intervenientes no cubo em análise, podemos proceder à fase de recolha de informação relativa ao registo das *queries* MDX, geradas pelo servidor OLAP e registadas nos ficheiros de *log* pela ferramenta *log4j*. Como período de análise para as *queries* definimos o dia, 24 horas de trabalho (0:00-23:59). A cada período, o ficheiro é substituído por um vazio e o anterior serve de fonte ao processo definido para a leitura do ficheiro e povoamento da base de dados (Algoritmo 4.2).

Input: Ligação à estrutura de dados e ao ficheiro *log*
1. Para cada registo de consulta:
    a) Identificar os dados:
        i. `data`
        ii. `hora`
        iii. `utilizador`
        iv. `cubo`
        v. `queryMDX`
        vi. `tempoSatisfação`.
    b) Para o caso do `utilizador`, `cubo` e da `queryMDX` quando:
        i. Não existe já um registo nas respectivas tabelas deve ser inserido
        ii. Existe, então identificar os códigos correspondentes.
    c) Inserir um registo na tabela de consultas, `LogQuery`, com os respectivos dados.

Algoritmo 4.2 - Processamento do ficheiro de *log*

Neste povoamento podemos perceber que a tabela mais afectada é a `LogQuery`. Esta acolhe os registos relativos à execução das *queries*: data, hora da execução, qual o utilizador que lançou a *query* e a própria *query*. São também assegurados os principais relacionamentos da tabela `LogQuery`, garantindo a integridade referencial dos atributos do utilizador e da *query*, estão as tabelas `User` e `Query`, que mantêm a informação relativa aos utilizadores do sistema de processamento analítico e da caracterização das *queries*, respectivamente.

Neste momento já temos a nossa base de dados povoada pela caracterização dos cubos e pelo registo das *queries* no entanto existem outras tabelas que não foram referenciadas como já tendo sido preenchidas, isto porque em algumas das situações os dados são enriquecidos. Por exemplo a tabela `User` guarda uma série de elementos que tiveram que ser preparados por nós, de forma a poder apresentar uma caracterização mais real de um utilizador. Isto também se justifica pelo facto de que as *queries* lançadas ao servidor são simulações criadas por nós, consequentemente o perfil de utilização também o é. Além desta tabela também aparecem as tabelas

`Query_EClasses`, `EClasses` e `Dimension_EClasses` que não foram referidas até agora mas surgem no esquema lógico ilustrado. Estas servirão de suporte às classes de equivalência, um processo de descoberta de padrões que surge no contexto deste trabalho e serão descritas com maior pormenor numa secção mais a frente.

Tabela	**Descrição**
User	Caracterização dos utilizadores que fazem as consultas
Cube	Descrição dos cubos alvo para exploração
LogQuery	Entidade que regista a ocorrência da consulta de uma *query* MDX
Query	Entidade que regista a *query* MDX
Dimension	Entidade que regista as dimensões consultadas numa *query*
Measure	Entidade que regista as medidas consultadas numa *query*
EClasses	Entidade que regista as classes de equivalência geradas
Dimesnion_Query	Relacionamento N:M entre as entidades dimensão e *query*
Measure_Query	Relacionamento N:M entre as entidades medida e *query*
Cube_Query	Relacionamento N:M entre as entidades cubo e *query*
Cube_Dimension	Relacionamento N:M entre as entidades cubo e dimensão
Cube_Measure	Relacionamento N:M entre as entidades cubo e medida
Query_EClasses	Relacionamento N:M entre as entidades *query* e classe de equivalência
EClasses_Measure	Relacionamento N:M entre as entidades classe de equivalência e medida
Dimesnion_EClasses	Relacionamento N:M entre as entidades dimensão e classe de equivalência

Tabela 4.1 - Descrição geral das tabelas da base do sistema operacional

Apresentámos agora três tabelas que auxiliarão na descrição e compreensão dos objectos que foram identificados neste sistema operacional (esta designação surge uma vez que numa fase posterior desenhámos e implementámos um pequeno *data warehouse*). Estas tabelas seguem as formas tradicionais dos dicionários de dados destas estruturas. Na Tabela 4.1 encontrámos uma descrição geral das tabelas da base de dados, na Tabela 4.2 e na Tabela 4.3, a título de exemplo, uma descrição mais detalhada de uma das entidades e um relacionamento respectivamente.

Entidades					
Atributo	**Descrição**	**Tipo**	**Exemplo**	**Nulo**	**RI**
LogQuery Registo sequencial, por ordem de execução, de todas as *queries* MDX lançadas ao sistema.					
Id_LogQuery	Número atribuído a cada nova consulta	*Integer*	1	N	PK[12]
Date	Data em que ocorreu a consulta	*Date*	2008-01-01	N	
Hour	Hora em que ocorreu a consulta	*Time*	09:53:39	N	
Time	Tempo de satisfação da consulta	*Real*	1042	N	
Query_Id_Query	Identificação da *query* perante o sistema. Uma *query* pode aparecer várias vezes na mesma consulta.	*Integer*	2	S	FK[13]
User_Id_User	Identificação do utilizador que lançou a consulta. Um utilizador pode lançar várias vezes a mesma *query*.	*Integer*	3	S	FK

Tabela 4.2 - Descrição detalhada das entidades do sistema operacional

As tabelas aqui descritas foram desenhadas a fim de acolher a informação das *queries* MDX e os dados relacionados: cubos, dimensões e medidas. Os processos de povoamento são desencadeados pelos registos das *logs*. No entanto a tabela de *logs* é a última a ser povoada para que sejam mantidas todas as regras de integridade referencial, para tal, como descrito nos processos de povoamento apresentados, inicialmente reconhece-se a *query*, bem como os seus diversos componentes (utilizador, cubo, dimensões e medidas), levando ao preenchimento das tabelas correspondentes e só depois ao registo da ocorrência de consulta da *query* MDX. Quanto às tabelas relacionadas com as classes de equivalência, o seu povoamento será detalhado quando abordarmos a sua determinação num subcapítulo à frente.

[12] PK, *Primary Key*, Chave primária da tabela.
[13] FK, *F*oreign Key, chave estrangeira da tabela.

Relacionamentos N:M				
Atributo	**Descrição**	**Tipo**	**Nulo**	**RI**
Cube_Query Identificação das *queries* que foram lançadas sobre um ou mais cubos.				
Id_Cube	Identificação do cubo envolvido no relacionamento com a query	*Integer*	N	PK, FK
Id_Query	Identificação da query envolvida no relacionamento com o cubo	*Integer*	N	PK, FK

Tabela 4.3 - Descrição detalhada das relacionamento do sistema operacional

4.3 Caracterização de Sessões OLAP

No processo de proposta de reestruturação dos hipercubos que vimos a analisar, torna-se indispensável o conhecimento do comportamento do utilizador. Nessa linha de pensamento para melhor identificarmos e caracterizarmos o comportamento de consulta do analista, percebemos que podemos facilmente adoptar o conceito de sessão de utilização e a partir daí caracterizar o comportamento e tentar identificar padrões de utilização. O conceito de sessão OLAP não é frequente na terminologia dos sistemas de processamento analítico. Do que conseguimos encontrar com a pesquisa que foi feita sobre o conceito de sessão OLAP, este apenas aparece nesta área referido por Sapia em 2000 num trabalho de caracterização do utilizador OLAP (Sapia, 2000). No entanto este tipo de conceito de sessão é bastante usual nos trabalhos de utilização e exploração da Web (W3C, 1999) - as sessões Web. Na realidade no nosso ponto de vista se considerarmos que um "click" em OLAP é uma *query* MDX, podemos aplicar os processos que se fazem sobre *clickstreams* às *queries*, idealizando que uma sequência de *queries* MDX poderá ser vista como uma *querystream* ou *MDXstream*. Partindo desta analogia entre as sessões Web e as sessões OLAP considerámos a caracterização do utilizador com aplicação de métodos idênticos, como por exemplo o reflexo das sessões em cadeias de Markov para análise comportamental.

4.3.1 Preparação do Cálculo das Sessões

O processamento de sessões OLAP, tal como acontece para uma sessão Web, requer alguma preparação dos dados armazenados nas *logs*, assim como algum processamento intensivo sobre os registos de utilização. Neste sentido, com o intuito de uma análise temporal sobre as *queries* lançadas sobre o servidor analítico e para suporte ao cálculo das sessões OLAP, decidimos desenhar e implementar um pequeno *data warehouse* com duas estruturas multidimensionais que acolhesse os registos das log *queries* e as sessões identificadas. Na primeira a análise que será permitida será essencialmente sobre as *queries* e a respectiva ocorrência, já na segunda a análise recai sobre o comportamento dos utilizadores ao longo das sessões de trabalho bom como estabelecer padrões de comportamento OLAP.

Dadas as estruturas que foram previamente criadas no sistema operacional para acolhimento dos registos, não foi necessário preparar processos adicionais de limpeza de dados ou de readaptação de metadados. O processo de povoamento foi simplificado e na maioria das vezes foi satisfeito por scripts SQL, correspondentes a vistas do sistema operacional, que respondiam às necessidades da estrutura multidimensional correspondente ao registo de *logs*. Analisemos agora com mais detalhe as duas estruturas (Figura 4.8 e Figura 4.9). Verifica-se que estamos perante uma constelação de duas estrelas, que integram duas tabelas de factos e cinco dimensões de análise, onde quatro delas são partilhadas pelas duas estruturas. Além destas dimensões identificadas surgem ainda algumas dimensões degeneradas, dimensões sem tabela, que serão detalhadas mais a frente. O conjunto de dimensão garantem a caracterização das diversas perspectivas de análise definidas e as tabelas de factos os registos da informação base considerada para análise. As duas estruturas identificadas no *data warehouse* desenhado pretendem auxiliar a encontrar estruturas mais adequadas à exploração efectivamente feita, assim como ambas caracterizam as interrogações dos analistas, no entanto têm perspectivas diferentes:

- As ***Queries* MDX** executadas permitem-nos analisar os processos de exploração OLAP que um utilizador (ou grupo de utilizadores) realizou ou longo do tempo, sabendo-se também as estruturas afectadas – os cubos – e, consequentemente, quais as vistas multidimensionais de dados obtiveram – cada *query* MDX executada disponibiliza uma vista do cubo alvo.

- As **Sessões** OLAP permitem saber com exactidão os períodos de trabalho que os utilizadores realizam ao longo do tempo. Podemos analisar a duração e a frequência desses períodos de trabalho, bem como, mais uma vez, as estruturas multidimensionais com que costumam trabalhar. A análise de uma sessão OLAP, em particular, corresponderá a uma sequência de execução de várias *queries* MDX.

Podemos assim perceber que estas duas estruturas são complementares em termos de análise. Na primeira podemos traçar um perfil com base na consulta das *queries* e com a segunda perceber comportamentos a nível de períodos de temporais de trabalho. Isto leva-nos a identificar o grão de análise das duas estruturas, a primeira ao nível da *query* e a segunda da sessão. Além da complementaridade percebemos que as duas estruturas são dependentes visto que as sessões OLAP são calculadas a partir do registo das *queries* MDX. Por este motivo e para que possam ser mais claros as especificações e os processos correspondentes a cada estrutura, os processos serão apresentados separadamente.

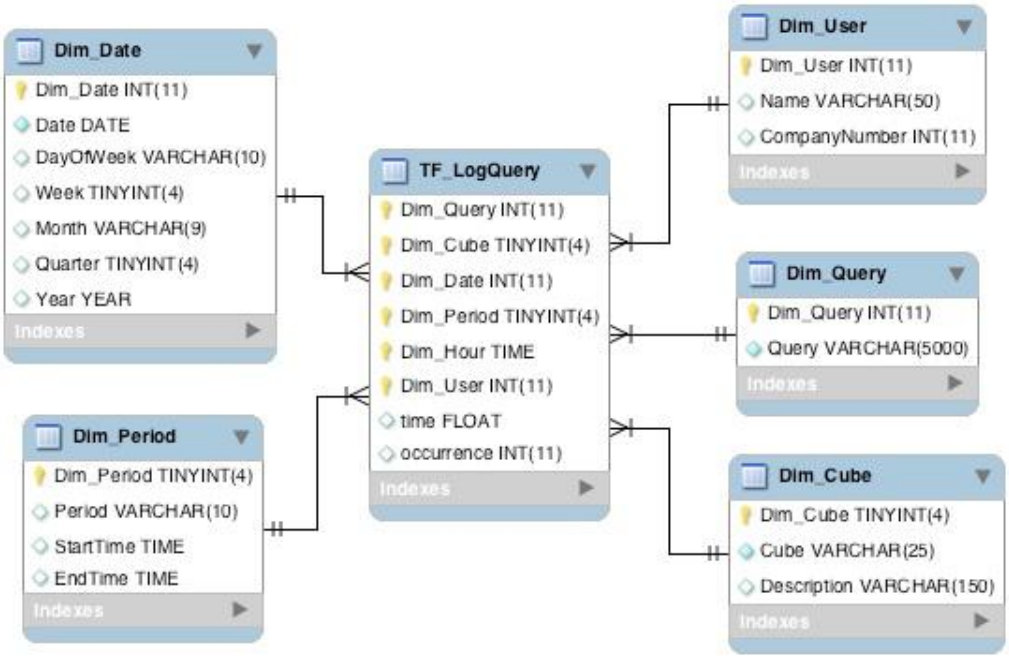


Figura 4.8 - Esquema estrela para guardar as *queries* MDX executadas

Dimensões			
Atributo	**Descrição**	**Tipo**	**Exemplo**
Dim_Date (Partilhada) Caracterização do calendário pretendido para análise.			
Dim_Date (PK)	Chave de identificação da data	*Integer*	2
Date	Data em que ocorre a consulta	*Date*	2010-01- 01
DayOfWeek	Nome do dia da semana correspondente à data	*Varchar* (10)	Friday
Week	Número da semana correspondente à data	*Integer*	1
Month	Nome do mês correspondente à data	*Varchar* (09)	January
Quarter	Número do trimestre correspondente à data	*Integer*	1
Year	Ano correspondente à data	*Integer*	2010
Dim_*Query* Caracterização de todas as *queries* MDX distintas lançadas até ao momento ao servidor analítico.			
Dim_Query (PK)	Chave de identificação da *query*	*Integer*	47
Query	Instrução completa da *Query* MDX	*Varchar* (5000)	SELECT {[Measures] ... WHERE [Time].[1997]
Dim_Cube (Partilhada) Identificação e caracterização de todos os cubos que foram consultados até ao momento.			
Dim_Cube (PK)	Chave de identificação do cubo	*Integer*	1
Cube	Designação do cubo	*Varchar* (25)	Sales
Description	Descrição do contexto de negócio e informação disponível para consulta no cubo	*Varchar* (150)	Cubo que guarda informaçã o sobre as vendas ...

Tabela 4.4 - Descrição detalhada das dimensões do esquema de registo das queries MDX

Dimensões			
Atributo	**Descrição**	**Tipo**	**Exemplo**
Dim_User Identificação e caracterização dos utilizadores com credenciais para realizarem consultas OLAP.			
Dim_User (PK)	Chave de identificação do utilizador	*Integer*	23
Name	Nome do Utilizador que executou pelo menos uma consulta até ao momento	*Varchar* (50)	Francisco Duarte
CompanyName	Número de identificação do utilizador dentro da empresa	*Integer*	45
Dim_Period (Partilhada) Identificação e caracterização dos períodos do dia – manhã, tarde, etc.			
Dim_Period (PK)	Chave de identificação do período do dia	*Integer*	1
Period	Período do dia (Manhã, Tarde ou Noite)	*Varchar*(10)	Morning
StartTime	Hora a que começa o período correspondente	*Time*	08:00:00
EndTime	Hora a que termina o período correspondente	*Time*	12:59:59

Tabela 4.4 - Descrição detalhada das dimensões do esquema de registo das *queries* MDX (continuação)

Comecemos então pela descrição da estrutura correspondente ao registo das *queries* MDX executadas pelo sistema de processamento analítico. Na Figura 4.8 podemos observar o esquema estrela correspondente a esta estrutura nas Tabela 4.4 e Tabela 4.5 a descrição mais detalhada tanto das dimensões intervenientes neste esquema quanto da tabela de factos, respectivamente. A tabela de factos incorpora apenas duas métricas, `time` e `occurrence` que correspondem ao tempo de execução da *query* e ao número de ocorrências da *query* respectivamente. Ambas as métricas são aditivas (entenda-se agregáveis).

Tabela de Factos			
TF_LogQuery (6 Dimensões e 2 Medidas) Tabela que guarda cada uma das *queries* MDX que foi executada por um utilizador, sobre um dado cubo, numa determinada data a uma certa hora. Equivale, em certa medida, à tabela `LogQuery` do sistema operacional, mas organizada de acordo com um espaço multidimensional.			
Atributo	**Descrição**	**Tipo**	**Exemplo**
Dim_Query (FK/Dim_query)	Identificação da *query* MDX executada na consulta	*Integer*	1
Dim_Cube (FK/Dim_Cube)	Identificação do cubo consultado na *query* executada	*Integer*	1
Dim_Date (FK/Dim_Date)	Data em que a *query* foi Executada	*Integer*	1
Dim_Period (FK/Dim_Period)	Período do dia em que a *query* foi executada	*Integer*	3
Dim_User (FK/Dim_User)	Utilizador que executou a *query*	*Integer*	1
Dim_Hour	Hora a que ocorreu a consulta	*Time*	17:00
Time (SUM(),AVG())	Tempo de satisfação da *query* (em milisegundos)	*Integer*	1423
Occurrence (SUM())	Contagem do número de ocorrências da *query*	*Integer*	1

Tabela 4.5 - Descrição detalhada da tabela de factos de registo das *queries* MDX

Quanto à estrutura desenhada para acolher as sessões OLAP (Figura 4.9), incluímos as dimensões e medidas que nos pareciam mais convenientes na análise de uma sessão. Para melhor identificarmos as medidas recorremos à análise do estudo que geralmente é feito sobre as sessões Web. Ao observarmos os dois esquemas das estruturas definidas, rapidamente percebemos que o espaço multidimensional é bastante idêntico quanto às dimensões escolhidas, no entanto a análise torna-se claramente diferente devido ao conjunto de medidas bastante mais alargado de oito medidas todas aditivas. O conjunto é constituído por: `SessionTime`, o tempo que a sessão OLAP demorou; `NumberQueries`, o número de queries executados durante a sessão; `AvgProcessingQueries`, o tempo médio de processamento das queries da sessão; `AvgBetweenQueries`, o tempo médio entre a execução de duas queries;

`MaxBetweenQuery`, o tempo máximo entre a execução de duas queries; `MinBetweenQuery`, o tempo mínimo entre a execução de duas queries; `MaxProcessingQuery`; o maior tempo de processamento de uma *query*; e, por fim, o `MinProcessingQuery`, o menor tempo de processamento de uma *query*. Este conjunto de medidas permite-nos perceber algumas características do trabalho do utilizador, conseguimos perceber desde a periodicidade de execução até à forma como ele recebe e analisa os dados através do tempo entre queries ou o número de queries que lança. Conseguimos ainda identificar algumas características de utilização OLAP, como o tempo de resposta ou a exigência de processamento. Este conjunto de medidas quando combinadas com a informação proveniente da primeira estrutura apresentada, pode indicar-nos factores de exploração como quais os "verdadeiros" cubos (partes do cubo inicial) é que são realmente explorados e quais as células que nunca foram alvo de uma *query*, além do tempo de acesso a esses alvos que pode ser um factor indicativo de pré-materialização. Quanto mais hábil for a análise destes indicadores, mais elucidativas serão estas medidas.

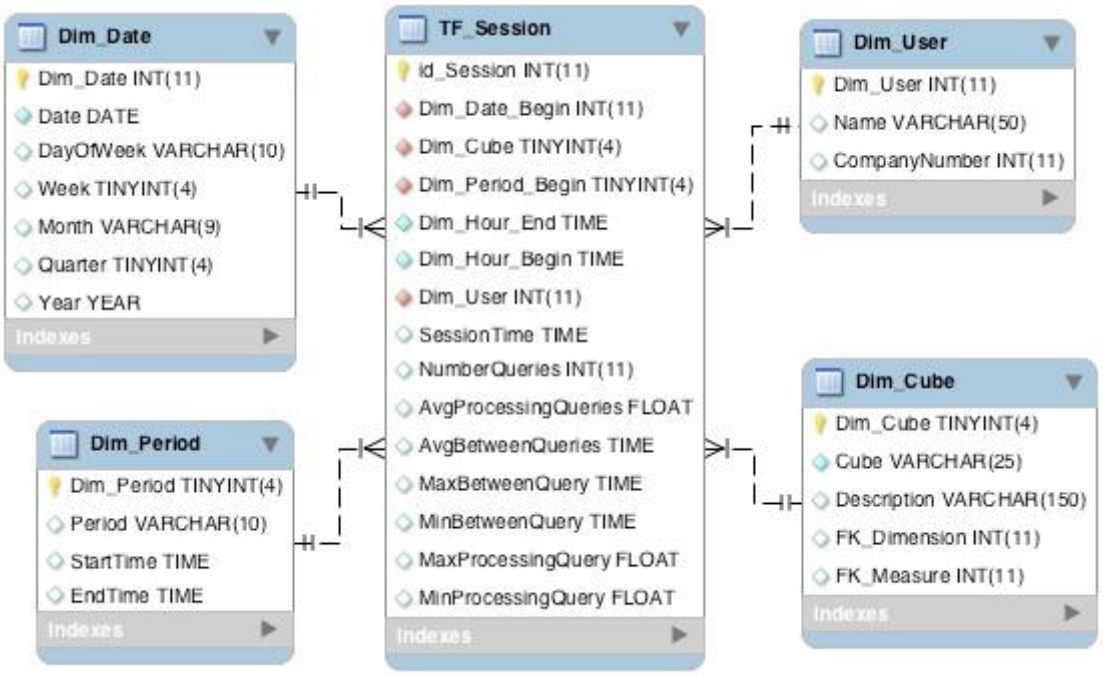


Figura 4.9 - Esquema estrela para guardar as sessões OLAP

Tabela de Factos			
TF_Session (6 Dimensões e 2 Medidas) Informação detalhada sobre todas as sessões OLAP que foram calculadas a partir dos registos das queries MDX lançadas pelos utilizadores OLAP e executadas pelo servidor analítico.			
Atributo	**Descrição**	**Tipo**	**Exemplo**
Id_Session (PK)	Número da sessão OLAP	*Integer*	1
Dim_Date (FK/Dim_Date)	Data em que a primeira *query* da sessão OLAP executada	*Integer*	1
Dim_Cube (FK/Dim_Cube)	Identificação do cubo consultado nas queries que foram executadas na sessão	*Integer*	1
Dim_Period (FK/Dim_Period)	Período do dia em que a sessão foi iniciada	*Integer*	3
Dim_Hour_Begin (FK/Dim_Period)	Hora em que a sessão OLAP foi iniciada	*Time*	3
Dim_Hour_End	Hora em que a sessão OLAP foi considerada terminada – hora da execução da última *query* MDX da sessão.	*Time*	1
Dim_User	Identificação do utilizador da sessão	*Integer*	17:00
SessionTime	Tempo de duração da sessão OLAP (minutos)	*Time*	10
NumberQueries	Número de queries executadas na sessão	*Integer*	10
AvgProcessingQueries	Tempo médio de satisfação das queries da sessão (milisegundos)	*Time*	1543
AvgBetweenQueries	Tempo médio entre as queries executadas na sessão (minutos)	*Time*	3
MaxBetweenQuery (AVG())	Tempo máximo entre queries da sessão (minutos)	*Time*	10
MinBetweenQuery (AVG())	Tempo mínimo entre queries da sessão (minutos)	*Time*	1
MaxProcessingQuery (AVG())	Tempo máximo de satisfação das queries da sessão (milisegundos)	*Time*	1923
MinProcessingQuery (AVG())	Tempo mínimo de satisfação das queries da sessão (milisegundos)	*Time*	1121

Tabela 4.6 - Descrição detalhada da tabela de factos de registo das sessões OLAP

Na Tabela 4.6 apresentámos a descrição mais ao pormenor da tabela de factos interveniente neste esquema multidimensional desenhado para acolher as sessões OLAP, quanto à descrição das dimensões é a mesma que na Tabela 4.4 visto que as dimensões são partilhadas.

Devemos apenas salvaguardar que ao longo deste processo foi considerado que, durante uma sessão OLAP, o utilizador apenas consulta um cubo. Daí a existência de um relacionamento de um-para-muitos com a tabela `Dim_Cube`. Para que seja considerado possível a consulta de mais do que um cubo na mesma sessão, a estrutura deverá ser adaptada para o acolhimento de uma tabela ponte entre a tabela de factos e a dimensão `Dim_Cube` de forma a acolher um relacionamento de muitos para muitos entre estas duas tabelas.

Após percebermos o desenho e implementação das estruturas multidimensionais que acolhem as *queries* MDX e as sessões, vamos agora perceber o processo de povoamento das mesmas. Este povoamento segue o procedimento típico de alimentação de um *data warehouse*, passando pelas fases de extracção, transformação e carregamento de dados. Numa primeira fase, os dados são extraídos do sistema operacional para uma área de retenção, onde as chaves de substituição são atribuídas a quase todas as dimensões. De seguida integram-se os dados recolhidos na tabela correspondente ao armazenamento das *queries* MDX, a `TF_LogQuery`. Por fim, são calculados os dados a integrar na tabela de factos das sessões OLAP, a `TF_Session`.

O processo de povoamento é realizado através de *scripts* SQL desenvolvidas especificamente e de algumas classes Java sobre o motor de base dados que seleccionámos para suporte aos sistemas de dados deste projecto, o My-SQL 5.1. Numa primeira fase do povoamento são criadas todas as estruturas da área de retenção: tabelas de auditoria, tabelas de chaves de substituição e uma réplica das estruturas do *data warehouse*. De seguida, procede-se ao primeiro povoamento no qual as tabelas `Cube`, `Query`, `User` e `LogQuery` são replicadas para as respectivas tabelas de auditoria. Paralelamente a este processo é criada uma chave de substituição por cada registo em cada uma das tabelas de dimensão referidas. Após a extracção estar concluída, tal como a criação das chaves de substituição, passamos à fase de integração dos dados no *data warehouse*. Aqui, e porque é o primeiro povoamento, começamos por alimentar a dimensão `Dim_Date` para um período de dez anos de vida do *data warehouse* (período de tempo que nos pareceu mais do que suficiente para os testes que pretendíamos realizar) a partir da data 2008-01-01 até à data 2018-01-01. Tal como para a dimensão `Dim_Date`, a `Dim_Period` é também alimentada apenas no

primeiro povoamento, salvaguardando actualizações, com apenas três registos correspondentes a três períodos definidos para um dia de trabalho. Com base nisso, é inserido o período da manhã - (`Morning`) das 08h00 às 12h59, o período da tarde - (`Afternoon`) das 13h00 às 19h59 e o período nocturno - (`Night`) das 20h00 às 07h59. Por fim, as dimensões `Dim_User`, `Dim_Query` e `Dim_Cube` são povoadas pelos registos nas respectivas tabelas de auditoria, embora contendo já a respectiva chave de substituição. Quanto à tabela de factos `TF_LogQuery`, por cada registo na tabela de auditoria da `LogQuery` são seleccionadas as chaves correspondentes à estrutura do *data warehouse* correspondente, relativas ao utilizador, à *query* e ao cubo, sendo também determinada a chave corresponde ao período do dia. No final do processo todas tabelas de auditoria são limpas sendo registado a data do último registo de consulta inserido no *data warehouse* na tabela `RA_LastRegister`, da área de retenção, de modo a se poder saber a partir de que data começará o povoamento seguinte. Como já referido, o povoamento da primeira estrutura é bastante linear e não acarreta grandes dificuldades, uma vez que é feito directamente através da tabela `LogQuery` do sistema operacional. No Algoritmo 4.3 podemos ver em esquema geral do código relativamente ao povoamento da sua tabela de factos `TF_LogQuery`. De salientar que, o povoamento das dimensões terá que ser feito sempre antes de qualquer uma das tabelas de factos que contenha na sua estrutura uma chave estrangeira para uma qualquer dimensão.

Input: Todos os registos da tabela de auditoria da *LogQurey*

1. Para cada registo (*query*, *utilizador*, *data*, *hora*, *tempo*)
   a) *Squery* = seleccionar a respectiva chave de substituição;
   b) *Sutilizador* = seleccionar a respectiva chave de substituição;
   c) *Scube* = determinar qual o cubo correspondente e seleccionar a respectiva chave de substituição;
   d) *Período* = determinar a chave do período do dia correspondente à *hora;*
   e) *Cdata* = seleccionar a chave correspondente à *data;*
   f) `Ocorrência` = 1;
   g) Inserir registo na `TF_LogQuery` com os valores (`Squery`, `Scubo`, `Cdata`, `Período`, `hora`, `Sutilizador`, `tempo`, `Ocorrência`).

Algoritmo 4.3 - Povoamento da tabela de factos `TF_LogQuery`

Quando observámos o povoamento que foi feito na tabela de factos `TF_LogQuery` poderia ser praticamente uma cópia dos registos contidos no sistema operacional. No entanto dada a

necessidade de preparar o registo da tabela de factos de acordo com a sua estrutura, e utilizando para cada uma das suas dimensões os seus respectivos atributos de relacionamento, obrigou à implementação de alguns processos de atribuição e validação de chaves, de processos de povoamentos de cada uma das dimensões integradas no sistema e também à verificação de todos os relacionamentos estabelecidos de forma a não colocar em causa qualquer questão de integridade, quer ela fosse de integridade referencial ou de entidade.

Quanto ao povoamento da estrutura correspondente às sessões OLAP o procedimento é mais exigente no que toca ao cálculo das métricas assim como na determinação das próprias sessões, grão da tabela de factos `TF_Session`. Este povoamento é detalhado na secção seguinte (0).

4.3.2 O Processamento de Sessões OLAP

Com a identificação das sessões OLAP para os diversos utilizadores da plataforma analítica, pretendemos conhecer o conjunto de queries MDX que são lançadas em conjunto. Não nos é relevante, pelo menos no âmbito do nosso trabalho, a ordem pela qual foram realizadas mas sim o conjunto de queries MDX, correspondente a vistas do cubo, que o utilizador executou durante a sua sessão de trabalho de análise.

Comparando com o estudo que nos tem vindo a servir de exemplo e referência, as sessões Web, a determinação das sessões OLAP torna-se um processo bastante mais simples de realizar dado o número de utilizadores ser consideravelmente menor, assim como o volume de pedidos registados num servidor analítico comparando com um servidor Web. Além disso, à excepção do perfil de convidado (*guest*), pouco usual numa plataforma analítica (a não ser para demonstração, provas de conceito, etc.), os utilizadores são todos identificados com as credenciais necessárias que são atribuídas pelo administrador para o acesso às estruturas. Mesmo quando as estruturas multidimensionais de dados estão localizadas na máquina do agente de decisão, os acessos estão por inerência identificados.

A identificação das sessões OLAP tem por base o processamento dos registos armazenados na tabela de factos das queries MDX, `TF_LogQuery`. Através destes registos identificámos e caracterizámos as sessões segundo várias dimensões de análise. Estes dados são integrados no

esquema multidimensional da qual faz parte a tabela de factos `TF_Session` (Figura 4.9). Todo este processo de transformação dos dados para integração nesta estrutura é feito por um programa Java desenvolvido para este fim, cujo esquema geral é apresentado no Algoritmo 4.4.

Input: Queries da `TF_LogQueries` e tempo máximo de unactividade (`tmi`)

1. Para cada dia `dt`:
   a. Seleccionar as queries relativas à data trazendo os atributos correspondentes: ao utililizador (`dutilizador`), ao cubo (`dcubo`), à hora (`dhora`), ao período (`dperiodo`) e ao tempo de satisfação (`tempo`), ordenadas pelos três primeiros;
   b. Inicializar as variáveis da sessão (atributos da `TF_Session`–vs);
   c. Percorrer as queries duas a duas, `qactual` e `qseguinte`
      i. Se as duas pertencerem ao mesmo cubo
         1. Se pertencerem ao mesmo utilizador
            a. Se a diferença de tempo é inferior a `tmi`
               i. Então a `qseguinte` está na mesma sessão que a `qactual`;
               ii. Actualizam-se as `vs`;
               iii. A `qseguinte` passa a `qactual` e repete-se o processo com a próxima query da lista como `qseguinte`;
            b. Diferença de tempo igual ao superior a `tmi`
               i. Considera-se que as *queries* pertencem a sessões diferentes;
               ii. Actualizam-se as `vs` sendo a última *query* desta sessão a `qactual`;
               iii. Inserir o registo da sessão na `TF_Session` com os valores correspondentes às `vs`;
               iv. A `qseguinte` passa a `qactual` e repete-se o processo com a próxima query da lista como `qseguinte`;
               v. Reinicializar as `vs` para a próxima sessão.;
         2. Utilizadores diferentes
            a. Considerar sessões diferentes;
            b. Repetem-se os passos de 1.c.i.1.b.i. até a 1.c.i.1.b.v.;
      ii. Cibos diferentes
         1. Considerar sessões diferentes;
         2. Repetem-se os passos de 1.c.i.1.b.i. até a 1.c.i.1.b.v.;
   d. Actualizam-se as `vs` sendo a última *query* desta sessão a `qactual`;
   e. Inserir o registo da sessão na `TF_Session` com os valores correspondentes às `vs`.

Algoritmo 4.4 – Povoamento da tabela de factos `TF_Session`

Numa fase inicial, antes de fazer o povoamento da tabela de factos `TF_Session` o programa trata de recolher a informação relativa a cada uma das dimensões envolvidas: utilizadores (`Dim_User`), cubos (`Dim_Cube`), horas (`Dim_Hour`), períodos (`Dim_Period`) e tempo de satisfação (`Time`). Na realidade, o programa executa várias operações de projecção sobre a tabela

`TF_LogQuery` recolhendo todos os identificadores relativos a cada uma das dimensões e ordenando-os, por fim, pela seguinte ordem: utilizador, cubo e hora. Posteriormente para cada grupo de queries que obedeça a um novo par {`utilizador`, `cubo`} o algoritmo vai percorrendo todas as queries, duas a duas, (`queryActual` e `querySeguinte`) calculando o tempo que passou entre as duas. Se a diferença for inferior ao tempo máximo de inactividade considerado (o período de tempo definido previamente para indicar o início de uma nova sessão), então são actualizados os dados relativos a um registo da tabela `TF_Session`. De seguida faz-se a mudança dos delimitadores de cálculo de sessão, passando-se a *query* seguinte (`qseguinte`) para *query* actual (`qactual`) e volta-se a repetir todo o processo até não termos mais queries. Tal como para a tabela `TF_LogQuery`, neste processo de cálculo de sessões também se guarda numa tabela especificamente definida para o efeito a data da última sessão introduzida na `TF_Session`, como forma de se saber a partir de que data é que se iniciará o próximo povoamento da `TF_Session`.

Nesta fase já temos as estruturas de *queries* MDX e sessões OLAP povoadas e com dados para caracterizarmos o comportamento dos utilizadores OLAP. O modelo organizacional que adoptámos, um *data warehouse,* permite-nos com facilidade fazer as análises ao longo do tempo. A sua estrutura baseada em eixos de análise, as dimensões, e medidas que caracterizam cada uma das ocorrências registadas nas estruturas permitem-nos nesta altura fazer algumas análises sobre o elemento prioritário neste processo, o utilizador. Podemos, por exemplo, observar quais as vistas multidimensionais que consultou através das queries MDX, revelando as partes do cubo acedidas, conseguimos perceber quando é que essas consultas foram feitas e determinar se existe uma periodicidade nas mesmas. Por outro lado conseguimos também perceber quais são os períodos de inactividade ou com menos carga sobre o servidor analítico, podendo assim revelar uma janela temporal mais apropriada para processos que consumam recursos do servidor. Todas estas análises, e outras que sejam possíveis sobre estes dados, possibilitam uma decisão de estrutura dos cubos a materializar mais sustentada e próxima da realidade de consulta. Claro está que, esta decisão não deve ser tomada apenas com base nestes dados até porque pode acarretar riscos como por exemplo a demora na satisfação de algumas consultas *ah hoc* que não sejam satisfeitas pelos cubos definidos.

4.3.3 Conjugação dos Dados de Sessão com Cadeias de *Markov*

Ao longo deste trabalho temos falado na definição de perfis de utilização, no entanto esta tarefa não é simples. Implica análises exaustivas de todo o comportamento do utilizador abrangendo desde as pequenas tarefas e consultas que possa fazer no âmbito do seu trabalho até à periodicidade e janela temporal escolhidas. Para além de todo o material que já vimos a analisar queríamos enriquecer este trabalho com processos que nos ajudassem a identificar também as sequências de consultas e assim perceber se existiam padrões de consultas nas sessões OLAP, assim como determinar uma probabilidade de ocorrência sequencial de duas ou mais queries MDX e assim surge no nosso trabalho a integração com as cadeias de *Markov*. Na verdade, esta ideia é mais uma vez influenciada pelos estudos realizados na área de estudo sobre utilização Web.

Com as cadeias de *Markov* pretendemos fazer previsão de acessos. Esta previsão é dada por uma probabilidade calculada através da contagem do número de utilizadores que pedem a execução dessa mesma *query* seguida de uma outra. Uma outra vantagem do recurso a esta técnica é a possibilidade de visualização através de grafos pesados e orientados onde as queries MDX estão representadas pelos nós e os arcos dão-nos a probabilidade de ocorrência consecutiva e um conjunto de outras métricas que podem ser acrescentadas como o número de utilizadores que consultou sucessivamente as queries intervenientes nesse arco.

Para acolher os dados resultantes deste processo desenhámos e implementámos uma tabela de suporte chamada `Markov_Chains` que foi incluída no sistema operacional (Tabela 4.7). Com esta nova tabela podemos ver que conseguimos determinar, por exemplo, o tempo médio entre a consulta de duas consultas, além da probabilidade associada.

Entidade					
Atributo	**Descrição**	**Tipo**	**Exemplo**	**Nulo**	**RI**
Markov_Chains Identificação e caracterização de cada uma das cadeias de Markov geradas a partir das queries de cada uma das sessões OLAP.					
id_MarcovChains	Chave de identificação da cadeia de Markov	*Integer*	3	N	PK
Begin	Chave da *query* de onde parte o arco	*Integer*	4	N	FK
End	Chave da *query* onde chega o arco	*Integer*	2	N	FK
Probability	Probabilidade de ocorrer a sequência das queries anteriores (de 'Begin' para 'End')	*Real*	0,67	N	
AVGTimeBetweenQueries	Tempo médio entre a consulta das duas queries	*Time*	00:00:3	S	
TotalQuery	Número de vezes que a *query* indicada em 'Begin' ocorre	*Integer*	5	S	
TotalNextQuery	Número de Vezes que a *query* indicada em 'End' ocorre	*Integer*	3	S	
TotalUsers	Número total de utilizadores que executa a sequência	*Integer*	2	S	

Tabela 4.7 - Descrição detalhada da tabela `Markov_Chains`

O processo de determinação das cadeias de *Markov*, detalhado no Algoritmo 4.5, inicia-se com a selecção de todas as queries consultadas até ao momento e, posteriormente, de todas as sessões identificadas. A partir destes dados constroem-se as estruturas correspondentes aos "caminhos" (*trails*) de cada sessão. De seguida, calcula-se a probabilidade inicial de cada *query* ocorrer e de seguida a probabilidade de ocorrência sucessiva entre cada duas queries. Por fim são registados na tabela todos os arcos da cadeia, ou seja os dados correspondentes à passagem de uma *query* para outra (nós da cadeia).

Input: Ligação à estrutura de dados

1. Criar estruturas queries (Q), `trails` (caminhos), `queries_count`
2. Captar todas as queries existentes ($Q = \{q_1 ... q_n\}$) .
3. Construir os caminhos – `trails`
   a. Captar todas as sessões da `TF_Session` ($S_i = \{q_k ... q_j\}$) e para cada uma:
      i. Captar todos os registos pertencentes a esta sessão (`TF_LogQuery`) e construir uma estrutura `trail` com todas as *queries* pertencentes à sessão;
      ii. Adicional `trail` à `trails;`
4. Percorrer a `trails` e preencher a `queries_count` com o número de vezes que cada registo aparece no total.
5. Construção das indicações para o grafo
   a. Percorrer todos os registos e para cada um acrescentar uma cadeia do estado inicial (-2) para esse registo com a probabilidade inicial dada por Pqi a dividir pelo número total de ocorrências de todas as queries
   b. Percorrer novamente todos os registos e para cada um:
      i. Percorrer todas as queries que aparecem a seguir a este e adicionar uma cadeia com a probabilidade de estes registos aparecerem consecutivamente.

Algoritmo 4.5 - Processo de cálculo das cadeias de *Markov*

O processo de cálculo das cadeias de *Markov* que aqui apresentámos é uma adaptação do trabalho apresentado em Marques (2009). Para melhor percebermos esta técnica e o ganho que podemos ter com ela, apresentámos de seguida um pequeno exemplo correspondente à Figura 4.10. Neste exemplo podemos observar como as *queries* (os nodos da rede contêm os identificadores das queries) foram sendo executadas ao longo das várias sessões e através dos pesos dos ramos ter uma noção do que pode acontecer a seguir à execução de uma qualquer *query*. Por exemplo, se analisarmos o que envolve o nodo 2, verificámos que os nodos -2 (nodo inicial) e 4 o precedem, o que significa que sempre que um utilizador inicia a sua sessão de exploração ou executa a *query* 5 a probabilidade de ele pedir a execução da *query* 2 a seguir é de 0.27, em ambos os casos. Por outro lado, se analisarmos os destinos dos ramos que partem desse nodo, verificámos que o mais provável que aconteça é que o utilizador depois de pedir a *query* 2 peça a execução da *query* 3, da *query* 5 ou simplesmente termine a sessão (nodo -1). A probabilidade associada com qualquer uma dessas três acções é de 0.33. Como podemos observar, estas avaliações são bastante fáceis e intuitivas, no entanto se o grafo cresce em demasia torna-se uma leitura mais confusa e devemos recorrer à geração de cadeias parciais ou a métodos de visualização de grafos mais apropriados. Em conclusão podemos dizer que este método é também bastante útil para o entendimento da forma como actuam os analistas numa determinada plataforma OLAP.

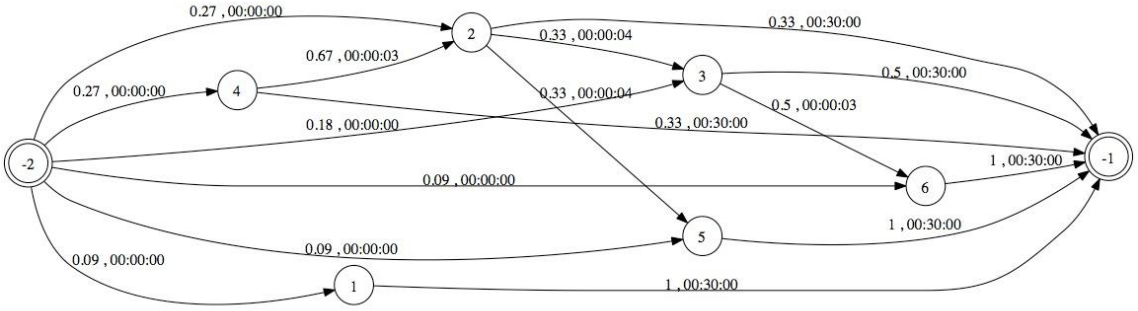


Figura 4.10 - Exemplo de uma cadeia de *Markov* gerada a partir de sessões OLAP

4.3.4 Reflexos na Proposta de Reestruturação

Para melhor percebermos o proveito das estruturas multidimensionais criadas e das cadeias de *Makov* no âmbito do estudo que temos vindo a fazer relativo a propostas de estruturas multidimensionais mais orientadas ao utilizador e à sua exploração efectiva, iremos agora abordar cinco cenários possíveis para essa reestruturação.

- `AllCube`, geração de um cubo completo que contempla todos os elementos pertencentes à estrutura inicial e considerando todos os utilizadores, esta proposta não contempla restrições consequentes da utilização.

- `AllCubeUsage`, proposta de cubo que considera todos os elementos pertencentes ao cubo que foram consultados na plataforma analítica por todos os utilizadores.

- `AllCubeUsageTimeBetween`, proposta idêntica à anterior, `AllCubeUsage`, com uma restrição adicional de tempo, isto é, apenas são consideradas as queries que foram executadas no período de tempo que aqui é imposto.

- `AllCubeUsageUser`, geração de cubo que considera apenas a utilização de um dado utilizador, sugerindo a estrutura correspondente aos elementos intervenientes em queries já lançadas por esse utilizador.

- `AllCubeUsageUserTimeBetween`, proposta que se assemelha ao `AllCubeUsageUser`, mas apenas contemplando o espaço de tempo considerado.

Este tipo de propostas pode ser obtido a partir de selecções descritas no Algoritmo 4.6.

Input: Ligação à estrutura de dados e quando necessário as datas do período a considerar (`dtb`) e o utilizador a considerar (`user`)

1. `AllCube`, seleccionar da estrutura de dados todos os elementos, dimensões e medidas.
2. `AllCubeUsagem`, seleccionar todas as dimensões e medidas pertencentes a queries que estejam associadas a registos na tabela de factos (`TF_LogQuery`).
3. `AllCubeUsageTimeBetween`, seleccionar todas as dimensões e medidas pertencentes a queries que estejam associadas a registos na tabela de factos (`TF_LogQuery`) no espaço de tempo compreendido entre as datas `dtb`.
4. `AllCubeUsageUser`, seleccionar todas as dimensões e medidas pertencentes a queries que estejam associadas a registos na tabela de factos (`TF_LogQuery`), pelo utilizador `user`.
5. `AllCubeUsageUserTimeBetween`, seleccionar todas as dimensões e medidas pertencentes a queries que estejam associadas a registos na tabela de factos (`TF_LogQuery`), pelo utilizador `user`, no espaço de tempo compreendido entre as datas `dtb`.

Algoritmo 4.6 - Descrição de geração de propostas baseadas na `TF_LogQuery`

A título de exemplo consideremos a Figura 4.11 onde podemos observar resultados provenientes deste tipo de propostas. A primeira proposta (`AllCube@`) podemos observar uma proposta de doze elementos onde três dos quais são medidas. Esta corresponde à estrutura inicial do hipercubo que considera todas as dimensões (também considerando elementos das hierarquias) e medidas possivelmente intervenientes. As restantes (`AllCubeUsage@`, `@id_User:1`, `@id_User:2` e `@id_User:3`) consideram restrições relacionadas com a utilização (considerando os utilizadores intervenientes). Para selecções com restrições de janela temporal, o processo seria idêntico ao demonstrado. Numa avaliação mais observadora do exemplo podemos perceber que se considerássemos que o período de exploração foi suficiente e que as sessões OLAP demonstram os reflexos da comunidade de utilizadores, então o administrador ao observar uma diferençam, como

por exemplo, entre a proposta `AllCube@` e `AllCubeUsage@`, identifica que os utilizadores apenas tiraram partido de seis dos doze elementos identificados no cubo inicial, podendo levar isto em consideração na sua decisão de reestruturação. Claro está que a decisão do administrador deve ser também sustentada (e devidamente justificada) pelo conhecimento que tem sobre o negócio e os utilizadores, percebendo as reais origens destes resultados e do comportamento dos seus analistas. Tenhamos também consciência que muitas das *queries* dos analistas são *ad hoc* e se nem sempre têm padrões muito definidos do percurso do seu estudo, os resultados que querem obter e os requisitos que querem satisfazer na sua análise são na sua maioria um padrão que repetem.

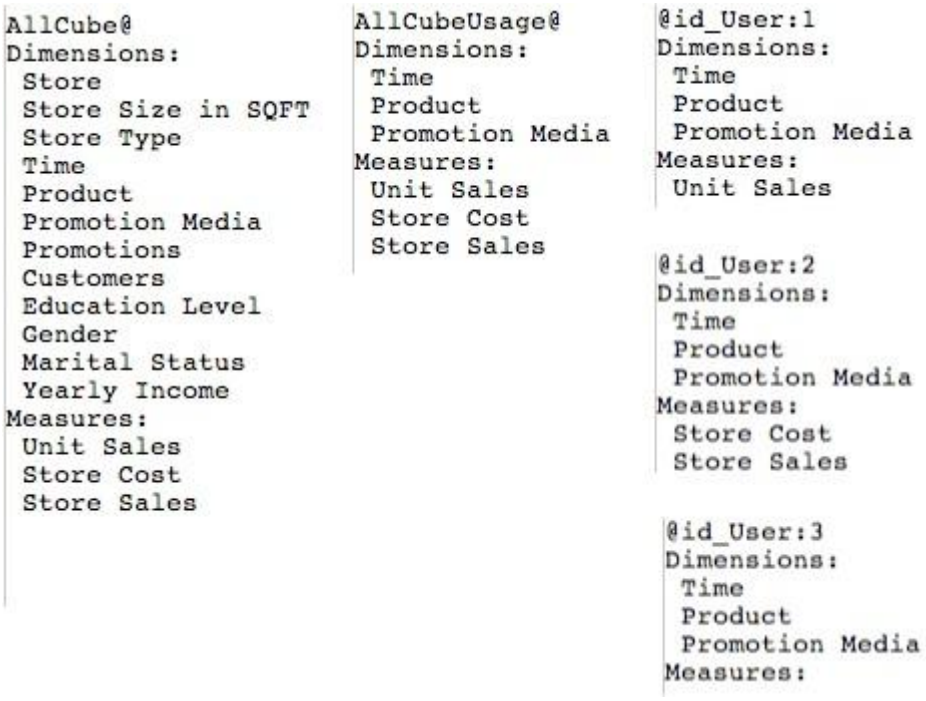


Figura 4.11 - Propostas de novas estruturas baseadas em sequências de queries

Considerando agora as cadeias de *Markov* podemos obter propostas de reestruturação baseadas no cálculo dos caminhos mais frequentes. Esta técnica que partiu do trabalho de Marques (2009) permite-nos estabelecer conjuntos de vistas multidimensionais (resultados de lançamento de queries MDX) mais frequentes. Para conseguirmos calcular este tipo de caminhos é necessário o recurso a algoritmos de pesquisa em grafos, sejam eles em profundidade (Weiss, 1993) ou em

largura (Tarjan, 1972). Dada a identificação dos caminhos que se mostrem mais recorrentes, o administrador pode toma-lo em consideração se achar que é relevante, no entanto sabendo que é um percurso de *queries* frequente entre os utilizadores.

No Algoritmo 4.7 podemos ainda perceber mais uma forma de tirar partido das cadeias de *Markov*. Como já tivemos oportunidade de perceber, as cadeias geradas contemplam uma probabilidade inicial para cada *query* considerada nessa cadeia, assim se quisermos determinar a probabilidade de cada elemento identificado surgir numa fase inicial de análise, podemos determinar a probabilidade de cada um através da probabilidade calculada para cada *query* à qual esse elemento pertença. Este método também pode ser aplicado para outra fase da análise, desde que devidamente identificada na cadeia. Mais uma vez o administrador tendo acesso à probabilidade de cada elemento surgir numa fase inicial, pode determinar por exemplo em considerar todos os elementos que tenham uma probabilidade acima da que considere relevante à pré-materialização. Consideremos um exemplo, se a dimensão `Product` (Produto) for um elemento frequente entre as *queries* com maior probabilidade (com ligação ao estado inicial) então o cálculo da sua probabilidade será também superior a outros, isto pode ser um indicativo de que o ou os analistas começam as suas análises tendo o `Product` como eixo de análise.

Input: Ligação à estrutura de dados e quando necessário as datas do período a considerar (`dtb`)

1. Criar uma estrutura com as dimensões e medidas para associar a cada uma a sua probabilidade.
2. Seleccionar da tabela `Markov_Chains` todas as cadeias que considerem o estado inicial.
   a. Para cada uma:
      i. Determinar os elementos pertencentes às *queires* presentes nesses arcos;
      ii. Calcular a probabilidade de cada elemento tendo em conta a probabilidade de cada query onde esse elemento surge.

Nota: a partir daqui pode ser devolvido o conjunto de dimensões e medidas que obedeçam a um valor mínimo de probabilidade estabelecido.

Algoritmo 4.7 - Determinar elementos mais frequentes com recurso as probabilidades iniciais

4.4 Descoberta de Padrões com Classes de Equivalência

Nesta fase do projecto recorremos ao cálculo de classes de equivalência para encontrar padrões entre os elementos das *queries* MDX.

As classes de equivalência foram abordadas no âmbito deste trabalho com intuito de melhor caracterizar a utilização de uma estrutura multidimensional de dados e chegar a uma proposta de um cubo mais adequado à consulta feita por um ou mais utilizadores identificados e consequentemente poder reduzir a estrutura a materializar assim como o seu tempo de refrescamento. As classes de equivalência são, neste trabalho, uma técnica adoptada com base num trabalho feito por Niemi, Nummenmaa & Thanisch (2001). No entanto a nossa abordagem, além de seguir o conceito do autor referido, difere ligeiramente, nomeadamente na forma como os conjuntos são determinados. No algoritmo original (Algoritmo 3.1) os conjunto são gerados a par do cálculo das diversas classes de equivalência, enquanto que nós gerámos todos os conjuntos possíveis e só depois os minimizámos de forma a obter o menor número de conjuntos possíveis que satisfaça os objectivos definidos.

Para as classes de equivalência, tal como já foi apresentado no sistema operacional, desenvolvemos estruturas que as acolhessem, além do sistema operacional sentimos a necessidade de desenhar e implementar uma estrela multidimensional (Figura 4.12), que nos permitisse armazenar estes novos dados de informação, para que posteriormente possam ser igualmente explorados e para que seja possível tirar partido deles. Na Tabela 4.8 é apresentada uma descrição detalhada da tabela de factos correspondente à estrela que acolhe as classes de equivalência, no entanto se observámos, na sua descrição, é apresentada como tendo cinco dimensões e quatro medidas, embora nas linhas que se seguem só aparece uma dimensão identificada, a dimensão correspondente ao cubo (`Dim_Cube`). Quanto às dimensões restantes (`Dim_User`, `Dim_Query`, `Dim_Dimension` e `Dim_Measure`) estão asseguradas pela chave principal da tabela de factos (`id_EClasse`), isto é, estas dimensões, devido ao seu relacionamento com a tabela de factos ser muitos-para-muitos, exigem a necessidade de tabelas ponte, isto confere uma estrutura muito particular ao esquema estrela. Estes relacionamentos de muitos-para-muitos surgem pelo facto de, por exemplo, uma medida poder fazer parte de muitas classes de equivalência, mas por outro lado cada classe de equivalência pode ter mais do que uma medida. O mesmo acontece para as dimensões, as *queries* e os utilizadores. Quanto à

caracterização destas dimensões já foi assegurada aquando da descrição das estruturas que acolhem os registos das logs (Tabela 4.4).

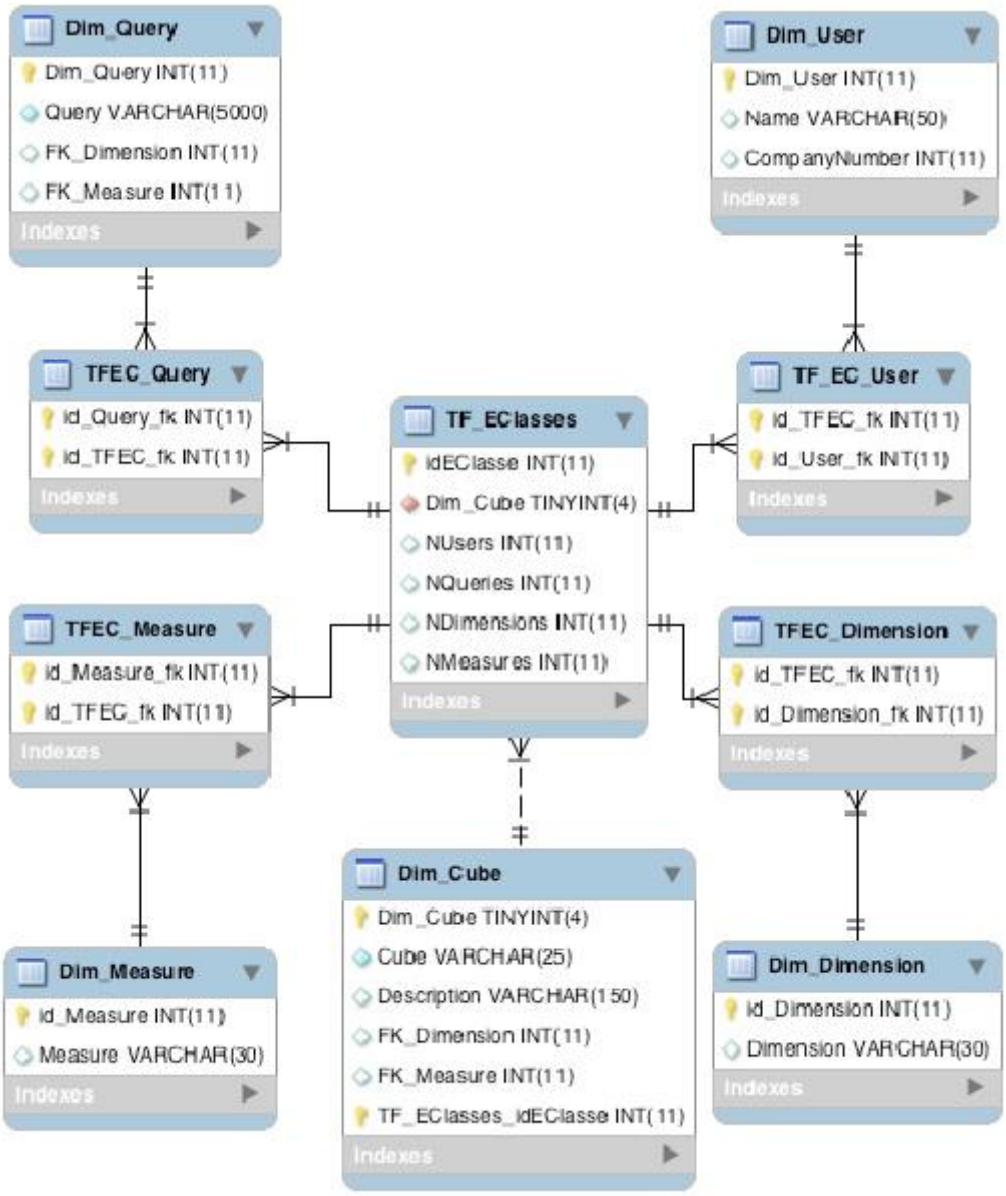


Figura 4.12 - Esquema estrela para guardar as classes de equivalência

Tabela de Factos

TF_EClasses (5 Dimensões e 4 Medidas)
Informação detalhada sobre todas as classes de equivalência geradas a partir dos registos das queries MDX lançadas pelos utilizadores OLAP e executadas pelo servidor analítico.

Atributo	Descrição	Tipo	Exemplo
id_EClasse (PK)	Número sequencial que identifica a classe de equivalência	*Integer*	1
Dim_Cube (FK/Dim_Cube)	Identificador do cubo que foi consultado nas queries executadas durante a sessão OLAP	*Integer*	1
NUsers (AVG())	Número de utilizadores que executaram as queries envolvidas no cálculo da classe de equivalência	*Integer*	3
NQueries (AVG())	Número de queries envolvidas no cálculo da classe de equivalência	*Integer*	57
NDimensions (AVG())	Número de dimensões envolvidas no cálculo da classe de equivalência	*Integer*	4
NMeasures (AVG())	Número de medidas incluídas na classe de equivalência	*Integer*	2

Tabela 4.8 - Descrição detalhada da tabela de factos `TF_EClasses`

Quanto ao processamento das classes de equivalência é apresentado no Algoritmo 4.8. Este processo inicia-se com a selecção das queries, e respectivas dimensões e medidas, que foram recolhidas no servidor analítico e armazenadas no sistema de dados operacional. Depois disso, procede-se à geração de todos os conjuntos possíveis, combinando o conjunto de dimensões e de medidas (`dimORmeas`) que temos disponível. No passo seguinte, procedemos à criação das estruturas de dados necessárias para o armazenamento de todas as possíveis classes de equivalência (`CE`). A cada uma das classes de equivalência geradas será associado o respectivo conjunto `dimORmeas`, bem como as queries que tenham sido lançadas envolvendo todos os elementos que figuram na classe definida. A partir daqui, é feita uma selecção das classes de equivalência por utilidade, reduzindo de forma significativa o conjunto das classes que foram geradas. Este passo envolveu a comparação das classes de equivalência geradas, obrigando a percorrer para cada classe cada um dos elementos que fazem parte do seu `dimORmeas` e para cada elemento deste conjunto procurarmos todas as outras classes de equivalência que tenham esse elemento. Na realidade, com este processo procurámos conjuntos de classes de equivalência

que estejam contidos noutros conjuntos de classes de equivalência e sempre que isso se verificar removemos o conjunto de menor dimensão.

Input: Acesso à base de dados com todas as queries e respectivas dimensões e medidas associadas.

1. Criar estruturas temporárias para armazenar as classes de equivalência;
2. Gerar todos os conjuntos possíveis com as dimensões e medidas exploradas;
3. Gerar uma Classe de Equivalência (`CEj`) na estrutura temporária para cada conjunto anteriormente gerado, contendo o conjunto de dimensões e medidas (`{dimORmeas}`) e outro conjunto com as queries (`{qs}`) pertencentes à respectiva `CE`;
4. Minimizar as Classes de Equivalência por um processo de comparação entre elas;
    a. Para cada `CE`:
        i. Para cada elemento do conjunto `{dimORmeas}` desta `CEi`:
            1. Se $|CEj_{\{dimORmeas\}}| > |CEi_{\{dimORmeas\}}|$
                a. Se $CEi_{\{qs\}} \supset CEj_{\{qs\}}$
                    I. Apagar a `CEi` do conjunto de todas as `CE`
5. Guardar as `CE`s restantes.

Algoritmo 4.8 - Processamento das classes de equivalência

Observando o procedimento e Figura 4.13 com exemplos de classes geradas podemos aperceber-nos que para além do cálculo das classes de equivalência ser um pouco complexo o resultado é bastante simples e intuitivo de perceber. Por exemplo, para a classe de equivalência `@idEC: 3`, determinada a partir de 108 *queries* multidimensionais, determinámos um conjunto composto por três dimensões `Time`, `Product` e `Promotion Media`[14] e três medidas `Store Cost`, `Store Sales` e `Unit Sales`. Mais uma vez, podemos tirar conclusões dos resultados embora não perdendo o contexto aplicacional assim como a utilidade do hipercubo em estudo. Se considerássemos que esta classe de equivalência levava em consideração um número de queries multidimensionais significativo no universo total, então poderíamos propor a nova estrutura multidimensional com o conjunto das três dimensões e três medidas para materializar, no entanto todas as dimensões que não fossem contempladas por esta sugestão teriam que ser consultadas por processamento executado no momento pelo servidor analítico à base de dados, isto implica

[14] Estes elementos (dimensões e medidas) são provenientes da base de dados de trabalho apresentada (`FoodMart`), no entanto alguns deles não são propriamente dimensões mas elementos das hierarquias das dimensões existentes nessa base de dados multidimensional que também estão disponíveis para navegação.

riscos de tempo de resposta e de satisfação das queries. Por outro lado se a nossa avaliação considerar também a segunda proposta apresentada de classe de equivalência, `@idEC: 4`, poderíamos concluir que a dimensão `Promotion Media` perde alguma prioridade em relação às outras duas identificadas, `Time` e `Product`, visto que esta classe considera um universo de 109 *queries* multidimensionais. Com estes dados poderíamos ainda definir pesos de cada um dos intervenientes e a partir dai considerar apenas os elementos que satisfizessem um peso mínimo definido.

```
@idEC: 3
 Dimensions:
    Time
    Product
    Promotion Media
 Measures:
    Store Cost
    Store Sales
    Unit Sales
 NQueries in this.EC = 108
```

```
@idEC: 4
 Dimensions:
    Time
    Product
 Measures:
    Store Cost
    Store Sales
    Unit Sales
 NQueries in this.EC = 109
```

Figura 4.13 - Exemplos de classes de equivalência

Tal como foi apresentado para os dados provenientes das sequências de *queries* (0), podemos também com classes de equivalência propor alguns cenários de reestruturação das estruturas multidimensionais considerando a utilização sobre o servidor de processamento analítico. Neste sentido poderíamos considerar até os mesmos cenários propostos, `AllCube`, `AllCubeUsage`, `AllCubeUsageUser` e `AllCubeUsageUserTimeBetween`, mas apresentámos nesta fase outras vertentes de exploração da `TF_EClasses`. Com recurso a simples queries SQL sobre a estrela que acolhe as classes de equivalência podemos sugerir propostas de vistas de hipercubos, que mais uma vez, não devem valer por si só para a definição da reestruturação, mas devem ser fontes para a decisão do administrador da plataforma analítica contribuindo para a sua análise e conhecimento da exploração feita sobre os sistemas de processamento analítico.

4.5 Assinaturas OLAP

Várias abordagens foram sendo apresentadas ao longo deste trabalho, desde a recolha das *queries* MDX lançadas ao servidor analítico, a determinação de sessões OLAP, o cálculo das cadeias de *Markov*, ou a descoberta de padrões com recurso a classes de equivalência, todas elas nos foram permitindo descobrir, embora com resultados diferentes, comportamentos de utilização dos analistas, exploradores do sistema de processamento analítico. No entanto ao longo da apresentação destas técnicas também fomos frisando que cada uma delas não são isoladamente suficientes para a proposta de reestruturação das estruturas multidimensionais. Embora todas nos permitam ter percepções de utilização e padrões de comportamento, podem acarretar consequências se consideradas sem contextualização. Assim, para evitar problemas na demora na resposta a uma qualquer *query*, recomendámos sempre a avaliação e interpretação destes resultados por parte do administrador do sistema, estas demoras podem ser devido à necessidade de consulta da fonte de dados aquando a interrogação ao sistema de processamento analítico consequente de uma má análise e conclusão do uso destas técnicas. Estas técnicas devem ser apoios à sua própria análise e decisão de materialização de vistas, sendo indicadores fortes de comportamentos e permitindo uma decisão mais suportada e, segundo o que apresentámos, mais acertada e indo de encontro ao que será a utilização OLAP.

Se uma assinatura pessoal é uma forma única de identificação pessoal, servindo para autenticar um documento como sendo, ou tendo participação, do assinante então podemos dizer que uma assinatura de certa forma caracteriza e identifica uma pessoa. Analogamente, podemos definir uma assinatura OLAP como sendo um conjunto de características que identificam um utilizador OLAP. Estas características podem até ser idênticas entre utilizadores, mas entre as variadas características e formas de actuação devem tornar-se únicas de alguma forma e ao mesmo tempo certificadoras do comportamento do utilizador.

Tendo todas estas percepções, propomos nesta fase a Assinatura OLAP como um conjunto de características do comportamento de exploração de uma ferramenta de processamento analítico que identifica padrões de utilização. Esta assinatura interpretada pelo administrador do sistema permite uma proposta de estrutura para um ou mais hipercubos, mais adequados à utilização futura dos analistas, baseada no seu comportamento prévio.

> Seja $\mathcal{U}$ o utilizador do sistema de processamento analítico, S um conjunto de sessões OLAP, $\mathcal{S}$ a sazonalidade do utilizador, CM uma cadeia de Markov da utilização OLAP feita por $\mathcal{U}$ e CE as classes de equivalência determinadas através das *queries* consultadas pelo utilizador, então A é a assinatura OLAP de $\mathcal{U}$, dada por $A_{\mathcal{U}} = <\mathcal{S}, S, CM, CE>$ e $f(A_{\mathcal{U}})$ a função de proposta de reestruturação do hipercubo tendo em conta o resultado da utilização de $\mathcal{U}$ sobre o sistema de processamento analítico.

Sendo que $f(A_{\mathcal{U}})$ propõe uma estrutura multidimensional ajustada à consulta de $\mathcal{U}$ tendo como *input* uma assinatura, analisemos agora a informação que pode ser extraída dessa mesma assinatura. O primeiro componente da assinatura é a sazonalidade, $\mathcal{S}$, permitindo ao administrador extrair um conjunto de informações relativas à periodicidade das consultas, se estas ocorrem em situações pontuais da semana, mês ou até mesmo ano, ou em que períodos do dia geralmente ocorrem. De seguida a informação relativa às sessões de consulta, S, definindo um comportamento associado a cada sessão ou uma visão genérica de um conjunto de sessões num dado período de tempo, por exemplo as sessões de um utilizador durante um mês. As sessões permitem ao administrador perceber quais os cubos alvo de análise, o número de queries que é lançado em cada sessão, o tempo despendido pelo utilizador nessa sessão e até a média de tempo entre as queries percebendo o tempo de análise despendido entre as consultas, não tendo em conta possíveis distracções ou tarefas paralelas, entre outras características. Com a informação das cadeias de Markov, CM, definidas em cada assinatura, é perceptível a sequencialidade das consultas assim como a probabilidade associada à consulta de vista, permitindo também determinar a probabilidade associada a cada elemento pertencente a essas mesmas queries. Por fim com as classes de equivalência, CE, permite ao administrador perceber conjuntos de dimensões com ocorrência conjunta num grupo de consultas. Percebendo assim dimensões transversais às consultas e aqueles que com maior frequência são um eixo de análise das pesquisas do utilizador. Conjugando toda esta informação podemos ainda perceber por exemplo as queries com tempos de resposta mais longos e com uma grande probabilidade de serem consultadas.

A definição deste conjunto de dados, entre outros que podem ainda ser extraídos, não só caracteriza e identifica a utilização do sistema por esse analista mas também define quais as vistas que são alvo preferencial na sua análise e até em que períodos temporais e com que frequência.

Se por exemplo considerarmos uma assinatura A_N definida pelo comportamento de utilização OLAP do utilizador N nos últimos dois meses, $A_N = <\mathcal{S}_N, S_N, CM_N, CE_N>$. Sendo que $\mathcal{S}_N$ determina que o utilizador N utiliza o sistema todas as segundas-feiras durante o período da manhã, S_N demonstra que as suas sessões incidem sobre o cubo C_{vendas} tendo uma duração média de três horas, com a consulta média de 47 queries, CM_N revela que as queries q_x, q_y e q_z são as mais prováveis de serem consultadas entre o universo de queries já consultadas pelo utilizador N, quanto às CE_N determinam um conjunto de dimensões e medidas que ocorrem nas classes de equivalência com maior número de queries associadas. Assim sendo $f(A_N)$ define que o cubo C_{vendas} deve conter as vistas associadas às consultas com uma maior probabilidade, assim como aquelas que contêm as dimensões e medidas identificadas nas classes de equivalência com maior número de queries associado. Sendo que a informação do cubo deve ser refrescada todas as madrugadas de segunda-feira. Esta poderia ser uma das avaliações do administrador tendo em conta o exemplo.

As assinaturas de cada utilizador podem ainda servir para o administrador perceber a evolução de comportamento de utilização OLAP, isto é, perceber se um dado utilizador mantém um comportamento padrão ou se a sua assinatura OLAP evolui para outro tipo de pesquisas, podendo assim o administrador adaptar as estruturas de forma a responderem às novas necessidades dos utilizadores do sistema.

Capítulo 5

Conclusões e Trabalho Futuro

5.1 Um Comentário ao Trabalho Realizado

Os sistemas de processamento analítico surgem das necessidades de exploração de dados orientadas, rápidas, capazes de auxiliarem os analistas nas descobertas e decisões necessárias no dia-a-dia dos seus negócios. Estes sistemas proporcionam o acesso a dados organizados em estruturas multidimensionais que permitem a sua exploração de forma a conhecer as métricas sobre várias perspectivas de análise. Dotam os analistas de autonomia de navegação, permitindo-lhes a cada análise ir adaptando a sua pesquisa. Contudo, para que tudo isto seja vantajoso e propicie algum valor acrescentado aos seus processos de análise, estes sistemas também devem ser rápidos e correctos nas suas respostas de forma a permitirem a fluidez e credibilidade da análise.

Usualmente, os dados dos sistemas OLAP são armazenados em estruturas multidimensionais, conhecidas por hipercubos, que necessitam de operações de processamento para cálculo das métricas com junções e funções de agregação que tornam as respostas tanto mais lentas quanto mais complexas forem as consultas. Para que estas sejam passíveis de responder em tempo útil aos analistas, sem causar interrupções no fluxo da sua análise, os sistemas de processamento

analítico recorrem a pré-processamento de agregações de dados, de forma a manterem disponíveis os dados pré-calculados e, assim, as respostas basearem-se no tempo de consulta dos seus dados pré-calculados. No entanto, para que isto seja possível deparamo-nos com a necessidade de pré-processar as agregações e de as armazenar. Isto seria mais simples se estivéssemos a referir-nos a volumes de dados reduzidos e com tempos de processamento satisfazíveis.

Com o crescimento e dinamismo dos mercados de hoje em dia, cada vez mais existem mais dados e eixos de análise pertinentes na hora de avaliar os resultados do exercício de uma dada actividade, tornando cada vez mais os conjuntos de dados mais ricos e com volumetrias exponenciais. Estes grandes conjuntos de dados, que na sua maioria apresentam vários níveis de agregações possíveis, leva-nos à "impossibilidade" de disponibilizar a totalidade das agregações possíveis, por limitações de processamento e armazenamento. Surge então a necessidade de seleccionar apenas algumas agregações, ou vistas, para materializar. Contudo este problema torna-se bastante complexo. Já Harinarayan, Rajaraman & Ullman (1996) demonstraram que se trata de um problema *NP-Hard*. Muitos foram os estudos que abordaram esta problemática até aos dias de hoje e muitas foram as técnicas utilizadas para minimizar os custos de processamento e armazenamento das vistas que deveriam ser materializadas.

O problema de selecção de vistas infere-nos a necessidade de conhecimento do que irá ser consultado, pois uma má decisão das vistas a materializar implica que o analista não possa consultar os dados que necessita ou que seja submetido a um tempo de espera demasiado longo para uma dada *query*, dada a necessidade de processamento da informação no momento da consulta. Este conhecimento, dos dados que poderão ser consultados, não existe em muitas das situações, ou mesmo existindo pode ser deitado por terra quando um analista decide avaliar um conjunto de perspectivas resultantes de uma pesquisa e que no momento considera mais pertinente, não tendo sido considerada essa perspectiva numa avaliação anterior da decisão de vistas a materializar. Em todo este processo conseguimos perceber que o analista é um elemento diferenciador, isto porque os sistemas são criados para auxiliá-lo nas suas avaliações, devendo ser desenhados de forma a satisfazerem o mais rápido possível às suas necessidades. No entanto, sabemos que o conhecimento acerca das necessidades de um analista não é exacto, dado que a sua análise pode ser alterada a cada consulta e por consequência dos resultados da última pesquisa, tornando este processo iterativo e interactivo.

Tal como em muitos outros trabalhos que exploraram a problemática de selecção de vistas associadas a sistemas de processamento analítico, o nosso objectivo foi o de determinar um conjunto de vistas que respondam às necessidades de consulta dos utilizadores destes sistemas e paralelamente que diminuam o tempo necessário para o processamento e refrescamento das vistas escolhidas, assim como o espaço necessário para o seu armazenamento, tornando todo o processo mais ajustado às necessidades e aos recursos disponíveis. Para tal, decidimos explorar o conhecimento do comportamento do utilizador de forma a ajustar a decisão de materialização às consultas que usualmente são feitas.

Sendo o utilizador um elemento central em todo o processo e sendo que é ao analista que o sistema deve responder, então no nosso entender o conhecimento do comportamento do utilizador e das suas necessidades de análise trará um ganho para a escolha das vistas. Este novo conhecimento, relativo aos comportamentos dos utilizadores, associado ao conhecimento do administrador do sistema, que deve conhecer não só a sua utilização e finalidade mas o negócio que é analisado neste sistema, trará uma visão real da exploração usual dos sistemas assim como alguns dos factores que possam influenciar a decisão de escolhas de vistas que até então não teriam sido consideradas.

O estudo de comportamentos de utilizadores e definição do seu perfil de utilização, *profiling*, associado a consultas não é propriamente algo novo. Além de em (Sapia, 1999) já serem referidas algumas características do comportamento de um utilizador OLAP, neste trabalho sugerimos o estudo do comportamento dos analistas pelas *queries* MDX que lançam ao sistema de processamento analítico, tal como acontece em estudos de comportamentos de utilização Web, onde o estudo e caracterização dos utilizadores são feitos pelos seus "*cliks*".

Com a recolha das *queries* MDX pretendemos não só definir o perfil de utilização do analista, mas também perceber a sua sazonalidade, quais as dimensões e medidas que mais consulta e perceber assim quais as vistas alvo da sua análise. Para tal, com a aplicação de várias técnicas (sessões OLAP, cadeias de Markov e classes de equivalência), não só determinámos quais foram as consultas realizadas, mas também as probabilidades de uma dada consulta ou de uma dimensão vir a ser consultada por esse mesmo utilizador. Conseguimos, ainda, determinar um conjunto de dimensões e medidas que terão uma maior probabilidade de serem consultadas e ainda agrupar as dimensões segundo ocorrências conjuntas.

5.2 A Abordagem Desenvolvida

No trabalho que aqui apresentámos estudámos a problemática da selecção de vistas a materializar para um dado sistema de processamento analítico e propusemos e desenvolvemos um conjunto de modelos que auxiliam a optimização das estruturas em causa, baseando-nos na exploração de hipercubos. Para que todo este estudo de comportamento de utilização OLAP fosse possível, o ponto de partida foi a angariação da informação acerca das *queries* MDX lançadas pelos utilizadores ao servidor analítico. Para realizar esta tarefa necessitávamos de conhecer o funcionamento do servidor analítico escolhido e a forma que seria possível o registo e captação da utilização. Para além disso era necessária a escolha e conhecimento da base de dados de trabalho. A escolha desta foi facilitada visto que a sua disponibilização estava associada à instalação do servidor analítico, além de que o conhecimento deste negócio era facilitado por ser uma base de dados bastante usual em exemplos relacionados com sistemas de suporte à decisão e um negócio bastante intuitivo de ser percebido.

Recolhidas as *queries* MDX, e os dados associados às consultas, encontrámos a forma mais estruturada de mantermos os dados armazenados, num sistema operacional. Após a disponibilidade da informação sobre as consultas inicia-se o processo que desenvolvemos para propor novas estruturas baseadas no comportamento de utilização.

Neste trabalho não propomos apenas uma técnica de redefinição das estruturas, como também não propomos que uma das técnicas seja a escolhida para a proposta de resolução. Estudámos um conjunto de técnicas que nos permitem obter perspectivas diferentes da avaliação de utilização das estruturas existentes. Propomos com estas técnicas fundamentar e enriquecer o conhecimento do administrador dos sistemas para que a sua decisão seja mais coesa e assertiva possível.

A primeira fase do processo proposto passou pela extracção, transformação e carregamento de dados para um esquema estrela no *data warehouse,* que alojou a informação relativa às consultas mas numa perspectiva multidimensional. Com este esquema podemos já extrair informação de utilização útil para a percepção global que será interpretada pelo administrador. Nesta fase, conseguimos perceber exactamente quais as queries MDX lançadas por cada utilizador, que nos levaram às estruturas alvo da sua análise e às vistas dessas estruturas que contêm a informação

relevante às suas pesquisas. Conseguimos também perceber uma sazonalidade de utilização percebendo os dias, por exemplo de semana ou do mês, em que existem maior número de ocorrências de determinadas queries ou o período do dia em que há uma maior afluência à consulta de determinado conjunto de queries. Na exploração desta estrutura conseguimos ter uma visão ao nível da consulta e uma percepção da utilização do sistema mas com uma perspectiva mais do comportamento mais de sazonalidade de repetição de consultas dos utilizadores e estruturas alvo.

O passo que seguidamente nos pareceu natural e bastante útil para o estudo do comportamento de utilização foi o de tentar perceber e definir as sessões de utilização. Para definirmos a forma mais correcta de determinarmos as sessões, utilizámos o conhecimento adquirido do estudo de comportamento de utilização web. Percebemos uma associação entre os dois comportamentos, isto porque um analista que consulta o sistema OLAP, segundo a observação que vai tendo dos resultados anteriores ou segundo um conjunto de queries que já sabe que necessita para a sua avaliação, tem um comportamento idêntico a um utilizador web que navega de uma página para a outra pois vai seguindo uma pesquisa na web ou abre directamente um conjunto de páginas que sabe previamente que quer consultar. Ambos os comportamentos podem ser definidos por períodos/sessões de trabalho/navegação, para tal associámos o comportamento de "*click*" ao comportamento de consulta de uma *query* e a partir daqui definimos regras que definem sessões entre os registos existentes na estrutura que acolhia as consultas. Estas sessões são determinadas por utilizador e são consideradas distintas a cada mudança de hipercubo e após um dado período de inactividade, que pode ser parametrizado. Com esta estrutura que acolhe as sessões OLAP o administrador ganha novo conhecimento sobre o comportamento de exploração do analista. Consegue através desta estrutura perceber as consultas que um utilizador lança ao servidor durante os períodos de análise determinados, além de que consegue também perceber períodos de frequência e associá-los a vistas consultadas. Esta estrutura acolhe ainda um conjunto de métricas calculadas que nos permite perceber por exemplo o tempo de sessão, o tempo médio entre as consultas, o número de *queries* que um utilizador faz por sessão de trabalho e até o tempo médio que o utilizador teve de esperar por resposta às suas *queries*.

Com estas duas estruturas o administrador consegue, de uma forma fundamentada e baseada numa avaliação comportamental, caracterizar o comportamento do utilizadores OLAP, isto é, traçar o seu perfil de utilização e perceber se este se vai alterando ao longo do tempo. Por outro lado o administrador também pode inferir desta informação, quais os períodos de maior inactividade e

assim, por exemplo, definir uma janela de oportunidade para processos de refrescamento dos dados se assim for o caso.

No percurso deste trabalho, interessou-nos traçar os perfis de utilização de um analista, o que não se revelou uma tarefa fácil, pois tivemos que considerar as consultas (vistas alvo de pesquisas), e a sazonalidade (frequência das consultas e períodos em que as análises ocorriam). Com as técnicas que já apresentámos conseguimos considerar alguns destes elementos, no entanto, o passo seguinte do trabalho auxilia o administrador a perceber o percurso de análise de cada utilizador ou grupo de utilizadores, percebendo padrões de consulta e sequencialidade entre elas. As cadeias de Markov são mais uma das técnicas que incluímos no nosso estudo por percebermos o ganho que havia na avaliação de comportamentos de utilização web. Com estas cadeias conseguimos determinar a probabilidade que associada à sequencialidade das *queries*, isto é, determinar a probabilidade de uma *query* ser consultada após a outra e determinar padrões de "caminhos de consultas". Esta técnica permite perceber quais as vistas (*queries*) mais consultadas assim como prever qual a próxima *query* a ser lançada, baseada numa dada probabilidade. As cadeias de Markov permite-nos, além de uma visualização, por meio de grafos orientados, do percurso da consulta, perceber o número de utilizadores que consultou uma dada vista, o número de vezes que essa *query* foi lançada e a probabilidade associada a cada uma das queries que está a si associada como seguinte.

Com os métodos até agora apresentados, o administrador consegue, com alguma facilidade, definir os perfis de utilização dos analistas assim como perceber as vistas alvo das suas pesquisas. Consegue ainda determinar quais as queries e respectivas dimensões que têm um acesso mais frequente e a sequencialidade das consultas. Com as técnicas seguintes quisemos trazer para o nosso projecto formas de correlacionar as dimensões e medidas consultadas para que nos fosse possível propor novas estruturas baseadas na ocorrência conjunta dos elementos intervenientes nas *queries*. As classes de equivalência surgem com o estudo do trabalho de Niemi, Nummenmaa & Thanisch (2001) e permite-nos implementar um algoritmo que determina conjuntos de elementos com ocorrências conjuntas, permitindo identificar a sobreposição de domínios de consulta de hipercubos. Com as classes de equivalência o administrador consegue perceber quais zonas mais consultadas nos hipercubos dado que as classes de equivalência determinam conjuntos de elementos que repetidamente ocorrem associados num determinado número de *queries*.

Com todas estas abordagens ao estudo do comportamento de utilização OLAP passámos a ter, como já referimos anteriormente, várias técnicas que auxiliam o administrador no estudo do comportamento pela perspectiva sazonal do analista e pela perspectiva de vistas consultadas. Percebendo assim que conseguimos definir perfis de utilização dos analistas e identificar características que definem o seu comportamento. Neste sentido propomos a definição de uma assinatura OLAP como um conjunto de elementos provenientes das várias técnicas estudadas que se define por características de utilização OLAP que identificam o comportamento desse mesmo utilizador. Essa assinatura deve ser *input* para o administrador de sistema que em conjunto com o conhecimento que já possui sobre o sistema e sobre o negócio em análise deve decidir ou não reajustar a estruturas em consulta, estas decisões podem por exemplo ter em conta que os resultados de assinaturas podem estar influenciados por situações pontuais que ocorreram no negócio e não são relevantes para a reestruturação do próximo cubo. Este processo de proposta de reestruturação deve ser cíclico de forma a considerar novos comportamentos de análise que surjam.

5.3 Orientações para Trabalho Futuro

Este trabalho serve como ponto de partida para avaliação comportamental da utilização OLAP e para percebermos a importância e o ganho deste tipo de estudos na reestruturação das estruturas multidimensionais e consequentemente o ganho que isso pode reflectir na sua utilização futura. Com este ponto de partida conseguimos identificar algumas das linhas de exploração na definição de perfis de utilização OLAP, tal como na descoberta das vistas mais consultadas pelos utilizadores. Contudo, isto é apenas o ponto de partida da exploração desta visão sobre a reestruturação das estruturas multidimensionais.

Várias podem ser as melhorias e evoluções deste trabalho, nomeadamente, a incorporação das preferências definidas pelos próprios utilizadores, a melhoria de desempenho das classes de equivalência, a exploração das técnicas de mineração de dados de forma a fazer previsão, a inclusão de novas técnicas de *profiling*, definição de novas métricas de avaliação das consultas e sessões, incremento de variáveis nas cadeias de Markov e até mesmo a possibilidade de detectar

situações de descontinuidade do tipo de consultas que é feito por um dado utilizador, situações em que um utilizador saia dos seus padrões de utilização.

Outra das linhas de evolução deste trabalho é a fase final de junção de todas estas técnicas, as assinaturas OLAP, e propor uma reestruturação das estruturas multidimensionais tendo em consideração as técnicas estudadas percebendo assim a diferença entre a estrutura que está em utilização e a nova estrutura proposta. Isto de uma certa forma fecha o ciclo de trabalho, permitindo evoluir as estruturas a par da evolução das necessidades de exploração que surjam no negócio. Esta mesma evolução pode ser observada e levar o administrador a perceber a evolução das estruturas para que satisfaçam as necessidades de exploração dos analistas e assim refinando as estruturas.

Bibliografia

Agarwal, S., Agrawal, R., Deshpande, P., Gupta, A., Naughton, J.F., Ramakrishnan, R. & Sarawagi, S. 1996, 'On the Computation of Multidimensional Aggregates', *Proceedings of the 22th International Conference on Very Large Data Bases (VLDB '96)*, San Francisco, Califórnia, Estados Unidos da América, 506-521.

Agrawal, R. & Srikant, R. 1994, 'Fast Algorithms for Mining Association Rules', *Proceedings of the 20th International Conference on Very Large Data Bases (VLDB '94)*, Saõ Francisco, Califórnia, Estados Unidos da América, 487-499.

Apache Logging Services 2011, *Apache Logging Services*, 16 de Setembro de 2011, <http://logging.apache.org/log4j/>.

Apache Software Foundation 2011, *Apache Tomcat*, 16 de Setembro de 2011, <http://tomcat.apache.org/>.

Bauer, A. & Lehner, W. 2003, 'On solving the view selection problem in distributed data warehouse architectures', *Proceedings of the 15th International Conference on Scientific and Statistical Database Management (SSDBM '03)*, Cambridge, MA, 43-54.

Belo, O. 2000, 'Putting Intelligent Personal Assistants Working on Dynamic Hypercube Views Updating', *Proceedings of 2nd International Symposium on Robotics and Automation (ISRA'2000)*, Monterrey, México.

Belo, O. & Duarte, A. 2010, 'UPonICE, Modelos e Estratégias para a Optimização de Estruturas Multidimensionais de Dados', Departamento de Informática, Universidade do Minho

Berkhin, P. 2002, *Survey of clustering data mining techniques*, Springer.

Beyer, K. & Ramakrishnan, R. 1999, 'Bottom-up computation of sparse and Iceberg CUBE', *Proceedings of the 1999 ACM SIGMOD international conference on Management of data (SIGMOD'99)*, Philadelphia, Pennsylvania, United States, 359-370.

Borges, J. & Levene, M. 1999, 'Data Mining of User Navigation Patterns', in Masand, B. and Spiliopoulou, M. *Web usage analysis and user profiling*, San Diego, Califórnia, Estados Unidos da América: Springer.

Business Application Research Center 2011, *The BI Survey 9 The Customer Veredict*, 27 de Setembro de 2011, <http://www.bi-survey.com/>.

Chaudhuri, S. & Dayal, U. 1997, 'An overview of data warehousing and OLAP technology', *SIGMOD Rec.*, vol. 26, 1, Março, pp. 65-74.

Chen, Q., Dayal, U. & Hsu, M. 1999, 'OLAP-based scalable profiling of customer behavior', *1st International Conference on Data Warehousing and Knowledge Discovery (DAWAK99)*, Italy.

Chen, M.S., Han, J. & Yu, P.S. 1996, 'Data mining: An overview from a database perspective', *IEEE Transactions on Knowledge and data Engineering*, vol. 8, 6, pp. 866--883.

Chirkova, R., Halevy, A.Y. & Suciu, D. 2002, 'A formal perspective on the view selection problem', *The VLDB Journal*, vol. 11, 3, pp. 216-237.

Codd, E.F., Codd, S.B. & Salley, C.T. 1993, 'Providing OLAP (On-line Analytical Processing) to User-Analysts: An IT Mandate'.

Deshpande, M. & Karypis, G. 2004, 'Selective Markov models for predicting Web page accesses', *ACM Trans. Internet Technol.*, vol. 4, 2, Maio, pp. 163-184.

Fayyad, U.M., Piatetsky-Shapiro, G. & Smyth, P. 1996, 'From data mining toknowledge discovery: An overview', in Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P. and Uthurusamy, R. (ed.) *Advances in knowledge discovery and data mining*, Menlo Park, Califórnia, Estados Unidos da América: American Association for Artificial Intelligence.

Feng, Y., Agrawal, D., El Abbadi, A. & Metwally, A. 2004, 'Range cube: efficient cube computation by exploiting data correlation ', *Proceedings. 20th International Conference on Data Engineering (ICDE'04)*, 658--669.

Findlater, L. & Hamilton, H.J. 2003, 'Iceberg-cube algorithms: An empirical evaluation on synthetic and real data', *Intell. Data Anal.*, vol. 7, 2, Abril, pp. 77 - 97.

Gartner 2011, *Magic Quadrant for Business Intelligence Platfoms*, *Gartner*, 29 de Outubro de 2011, <http://www.gartner.com/technology/research.jsp>.

Gray, J., Chaudhuri, S., Bosworth, A., Layman, A., Reichart, D., Venkatrao, M., Pellow, F. & Pirahesh, H. 1997, 'Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals', *Data Mining and Knowledge Discovery*, vol. 1, 1, pp. 29--53.

Gupta, H. 1997, 'Selection of Views to Materialize in a Data Warehouse', *Proceedings of the 6th International Conference on Database Theory*, London, UK, 98-112.

Han, J., Pei, J., Dong, G. & Wang, K. 2001, 'Efficient computation of Iceberg cubes with complex measures', *Proceedings of the 2001 ACM SIGMOD international conference on Management of data (SIGMOD '01)*, Santa Barbara, California, United States, 1 -12.

Harinarayan, V., Rajaraman, A. & Ullman, J.D. 1996, 'Implementing Data Cubes Efficiently', *Proceedings of the 1996 ACM SIGMOD international conference on Management of data (SIGMOD '96)*, New York, NY, Estados Unidos da América, 205-216.

Hartigan, J.A. 1975, *Clustering Algorithms*, New York, NY, Estados Unidos da América.

Hyde, J. 2009, *Mondrian Documentation*, 23 de Outubro 2011, <http://mondrian.pentaho.com/documentation/mdx.php>.

Inmon, W.H. 1996, *Bilding the Data Warehouse*, Segunda edição edition, While Computer Publishing.

Kalnis, P., Mamoulis, N. & Papadias, D. 2002, 'View selection using randomized search', *Data and Knowledge Engineering*, vol. 42, 1, Julho, pp. 89-111.

Kalnis, P., Ng, W.S., Ooi, B.C., Papadias, D. & Tan, K.-L. 2002, 'An adaptive peer-to-peer network for distributed caching of OLAP results', *Proceedings of the 2002 ACM SIGMOD international conference on Management of data (SIGMOD '02)*, New York, NY, Estados Unidos da América, 25-36.

Kimball, R. & Caserta, J. 2004, *The Data Warehouse ETL Toolkit: Pratical techniques for extracting, cleaning, conforming, and delivering data*, Estados Unidos da América: Wiley Publishing, Inc.

Kimball, R., Reeves, L., Ross, M. & Thornthwaite, W. 1998, *The Data Warehouse Lifecycle Toolkit*, Estados Unidos da América: Wiley Computer Publishing.

Kirkgöze, R., Katic, N., Stolba, M. & Tjoa, A.M. 1997, 'A Security Concept for OLAP', Proceedings of the 8th International Workshop on Database and Expert Systems Applications (DEXA '97), Washington, DC, Estados Unidos da América, 0619-.

Kotidis, Y. & Roussopoulos, N. 1999, 'DynaMat: A Dynamic View Management System for Data Warehouses', ACM SIGMOD, Phiadelphia, Pensylvania, 371-382.

Kumar, V.T.V., Haider, M. & Kumar, S. 2011, 'A View Recommendation Greedy Algorithm for Materialized Views Selection', *Information Intelligence, Systems, Technology and Management: 5th International Conference (ICISTM 2011)*, Gurgaon, India, 61-70.

Liang, W., Wang, H. & Orlowska, M.E. 2001, 'Materialized view selection under the maintenance time constraint', *Data and Knowledge Engineering*, vol. 37, Maio, pp. 203-216.

Lin, W.-y. & Kuo, I.-c. 2004, 'A Genetic Selection Algorithm for OLAP Data Cubes', *Knowledge and Information Systems*, vol. 6, pp. 83--102.

Liu, B. 2007, *Web data mining: exploring hyperlinks, contents, and usage data*, Springer.

M. Anandarajan, A. Anandarajan & Srinivasan, C.A. 2003, *Business intelligence techniques: a perspective from accounting and finance*, Springer.

Marques, A. 2009, 'Identificação de Perfis de Utilização Web Baseada em Clickstreams', Mestrado em Informática, Universidade do Minho.

Marques, A. & Belo, O. 2010, 'Discovering Usage Profiles on Web Based e-Learning Platforms Using Markov Chains', In Proceedings of 9th European Conference on e-Learning (ECEL 2010), Instituto Superior de Engenharia do Porto, Porto, Portugal.

Microsoft 2011, *Consultando dados multidimensionais (Analysis Services - Dados Multidimensionais)*, 18 de Serembro de 2011, <http://technet.microsoft.com/pt-br/library/bb500184.aspx>.

Microsoft MDX 2011, *MDX Scripting Fundamentals (MDX)*, 23 de Outubro de 2011, <http://msdn.microsoft.com/en-us/library/ms145973.aspx>.

MicroStrategy 2011, *MicroStrategy SAP Services*, 23 de Outubro 2011, <http://www.microstrategy.com/Software/Products/Service_Modules/SAP_Services/>.

Morfonios, K., Konakas, S., Ioannidis, Y. & Kotsis, N. 2007, 'ROLAP implementations of the data cube', *ACM Computing Surveys*, vol. 39, 4, Outubro, p. 12.

Niemi, T., Nummenmaa, J. & Thanisch, P. 2000, 'Functional Dependencies in Controlling Sparsity of OLAP Cubes', *Proceedings of the Second International Conference on Data Warehousing and Knowledge Discovery (DaWaK 2000)*, Springer-Verlag, Londres, Reino Unido, 199-209.

Niemi, T., Nummenmaa, J. & Thanisch, P. 2001, 'Constructing OLAP cubes based on queries', *Proceedings of the 4th ACM international workshop on Data warehousing and OLAP (DOLAP '01)*, Atlanta, Georgia, Estados Unidos da América, 9-15.

Oracle Corporation 2011, *MySQL*, 27 de Setembro de 2011, <http://www.mysql.com/>.

Pei, J., Han, J., Lu, H., Nishio, S., Tang, S. & Yang, D. 2001, 'H-mine: hyper-structure mining of frequent patterns in large databases ', *Proceedings IEEE International Conference on Data Mining, 2001 (ICDM'01)*, San Jose, CA , Estados Unidos da América, 441 - 448.

Pendse, N. 2008, *The BI-Veredict*, 28 de Setembro de 2011, <http://www.bi-verdict.com/fileadmin/dl_temp/dcba78e90b0d9a178c0e87146d035fb4/fasmi.htm>.

Pentaho Corporation 2011, *Pentaho Mondrian Project*, 16 de Setembro de 2011, <http://mondrian.pentaho.com/>.

Rapid-I, R.t.F. 2011, *RapidMiner*, 17 de Setembro de 2011, <http://rapid-i.com/content/view/181/190/>.

Ross, K.A. & Srivastava, D. 1997, 'Fast Computation of Sparse Datacubes', Proceedings of the 23rd *International Conference on Very Large Data Bases (VLDB '97)*, 116--125.

Sapia, C. 1999, 'On Modeling & Predicting Query Behavior in OLAP Systems', *Proceedings of the International Workshop on Design and Management of Data Warehouses (DMDW'99)*, Heidelberg, Alemanha.

Sapia, C. 2000, 'PROMISE: Predicting Query Behavior to Enable Predictive Caching Strategies for OLAP Systems', *Proceedings of the Second International Conference on Data Warehousing and Knowledge Discovery (DaWaK 2000)*, 224-233.

Sarukkai, R.R. 2000, 'Link prediction and path analysis using Markov chains', *Proceedings of the 9th international World Wide Web conference on Computer networks : the international journal of computer and telecommunications netowrking*, Amsterdão, Holanda, 377-386.

Scheinerman, E.R. 2003, *Matemática Discreta - Uma Introdução*, 1st edition, São Paulo: Thomson.

Shao, Z., Han, J. & Xin, D. 2004, 'MM-Cubing: computing Iceberg cubes by factorizing the lattice space', *16th International Conference on Scientific and Statistical Database Management (SSDM'04)*, Ilha de Santorini. Grécia, 213 - 222.

Shim, J., Scheuermann, P. & Vingralek, R. 1999, 'Dynamic Caching of Query Results for Decision Support Systems', *Proceedings of the 11th International Conference on Scientific and Statistical Database Management (SSDBM '99)*, 254.

Shukla, A., Deshpande, P. & Naughton, J.F. 1998, 'Materialized View Selection for Multidimensional Datasets', *Proceedings of the 24rd International Conference on Very Large Data Bases*, San Francisco, CA, Estados Unidos da América, 488-499.

Shukla, A., Deshpande, P. & Naughton, J.F. 2000, 'Materialized View Selection for Multi-Cube Data Models', *Proceedings of the 7th International Conference on Extending Database Technology: Advances in Database Technology*, London, United Kingdom, 269-284.

Spiliopoulou, M., Mobasher, B., Berendt, B. & Nakagawa, M. 2003, 'A Framework for the Evaluation of Session Reconstruction Heuristics in Web-Usage Analysis', *INFORMS J. on Computing*, vol. 15, 2, Abril, pp. 171-190.

Tarjan, R. 1972, 'Depth-First Search & Linear Graph Algorithms', *SIAM J. Comput*, vol. 1, 2, pp. 146-160.

The Apache Software Foundation 2011 *The Apache Software Foundation*, 28 de Setembro de 2011, <http://www.apache.org/>.

Theodoratos, D. & Bouzeghoub, M. 2000,' A general framework for the view selection problem for data warehouse design and evolution', *Proceedings of the 3rd ACM international workshop on Data warehousing and OLAP*, McLean, Virginia, Estados Unidos da América, 1-8.

Tomcat, A. 2011, *Apache Tomcat*, 28 de Setembro de 2011, <http://tomcat.apache.org/>.

W3C, W.W.W.C.W.U.C.A. 1999, *W3C Working Draft: Web Characterization Terminology & Definitions Sheet*, 17 de Setembro de 2011, <http://www.w3.org/1999/05/WCA-terms/01>.

Weiss, M.A. 1993, *Data Structures and Algorithm Analysis in C*, Redwood City, Califórnia: Cummings Publishing Company.

Xin, D., Han, J., Li, X., Shao, Z. & Wah, B.W. 2007, 'Computing Iceberg Cubes by Top-Down and Bottom-Up Integration: The StarCubing Approach', *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, 1, Janeiro, pp. 111-126.

Xin, D., Han, J., Li, X. & Wah, B.W. 2003, 'Star-cubing: computing iceberg cubes by top-down and bottom-up integration', *Proceedings of the 29th international conference on Very large data bases (VLDB '2003)*, Berlin, Alemanha, 476-487.

Zhao, Y., Deshpande, P.M. & Naughton, J.F. 1997, 'An array-based algorithm for simultaneous multidimensional aggregates', *Proceedings of the 1997 ACM SIGMOD international conference on Management of data (SIGMOD '97)*, Tucson, Arizona, Estados Unidos da América, 159-170.

Zhou, B. & Pei, J. 2009, 'Answering aggregate keyword queries on relational databases using minimal group-bys', *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology (EDBT '09)*, Saint Petersburg, Russia, 108--119.