Universidade do Minho
Escola de Engenharia

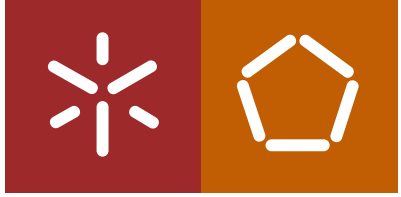
Ana Isabel Gonçalves Sequeira

Mineração de Dados em Sistemas OLAP

Ana Isabel Gonçalves Sequeira Mineração de Dados em Sistemas OLAP

UMinho | 2012

Dezembro de 2012



Universidade do Minho
Escola de Engenharia

Ana Isabel Gonçalves Sequeira

Mineração de Dados em Sistemas OLAP

Dissertação de Mestrado
Mestrado em Engenharia Informática

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

Dezembro de 2012

AUTORIZADA A REPRODUÇÃO PARCIAL DESTA TESE APENAS PARA EFEITOS DE INVESTIGAÇÃO, MEDIANTE DECLARAÇÃO ESCRITA DO INTERESSADO, QUE A TAL SE COMPROMETE;

Universidade do Minho, ___/___/_____

Assinatura: _____

Aos meus pais.

Agradecimentos

No decorrer desta dissertação foram muitas as pessoas que me apoiaram, incentivaram e colaboraram na sua realização, às quais não posso deixar de manifestar o meu mais sincero agradecimento.

Em primeiro lugar, gostaria de agradecer ao meu orientador, Professor Doutor Orlando Belo, antes de mais nada pela sua compreensão, apoio e orientação ao longo de todo este percurso. Por todos os seus ensinamentos e por ter contribuído para o meu crescimento a nível científico, pessoal e profissional.

À minha grande colega e amiga Mariana Carvalho, pela paciência, força e companheirismo que demonstrou em todos os momentos. Por ter estado a meu lado nos momentos menos fáceis, sempre pronta a colocar-me um sorriso na cara.

Quero também agradecer a três pessoas bastante importantes por terem ajudado a definir a pessoa que sou hoje: Sílvia Machado, Maria de Fátima Gonçalves e Ana Cláudia Palma. Por todos os conselhos que me deram, pelo seu apoio, amizade e pelos momentos de descontração.

Ao Nelson Correia, pelo seu incentivo durante todo este ano. Por ter estado a meu lado, demonstrando uma paciência incansável. Obrigada pelo amor, alegria e atenção sem reservas...

Por fim, manifesto um agradecimento especial aos meus pais, a quem dedico esta dissertação, pois sem eles nada disto teria sido possível. Agradeço-lhes pelo apoio recebido ao longo de todos estes anos, por sempre me incentivarem a alcançar caminhos cada vez mais distantes. Muitíssimo obrigada pelo investimento que fizeram em mim, pela confiança, pela oportunidade. A eles, a quem tudo devo, o meu eterno agradecimento pelo amor incondicional, carinho, apoio, preocupação e insistência.

O meu profundo e sentido agradecimento a todas as pessoas, amigos e colegas que, mesmo não citados nesta lista de agradecimentos, contribuíram de forma direta ou indireta não só para a concretização desta dissertação, mas também para o meu crescimento pessoal e profissional.

A todos um muito obrigada!

Resumo

Mineração de Dados em Sistemas OLAP

As diversas vantagens que os *data warehouses* têm proporcionado no que toca ao armazenamento e processamento de informação levaram a uma subida substancial na aquisição deste tipo de estruturas por parte das organizações. De facto, os *data warehouses* são caracterizados por um modelo de dados que permite, entre várias opções, realizar pesquisas complexas, selecionar conjuntos de dados de maior interesse, executar operações de sintetização, fazer comparações de dados e proporcionar diferentes visualizações dos dados. No entanto, a sua complexidade acarreta diversos custos, nomeadamente custos de computação e de materialização. Por um lado, a pré-computação de um cubo a partir de um data warehouse proporciona tempos de resposta reduzidos às pesquisas realizadas, mas, por outro lado, isso causa problemas no que toca à quantidade de espaço de armazenamento necessário. As técnicas de mineração de dados, nomeadamente aquelas que consideram os algoritmos de mineração de regras de associação, permitem encontrar conjuntos de itens frequentes entre os dados, permitindo, consequentemente, definir um conjunto de preferências de exploração ou de utilização. O estudo de preferências OLAP apresentado nesta dissertação visa identificar os dados mais acedidos por parte dos utilizadores, de forma a ser possível chegar a um consenso sobre quais as partes de um cubo que não são necessárias materializar, uma vez que não são utilizadas em processos de análise, mantendo tempos de resposta das pesquisas aceitáveis e reduzindo significativamente a quantidade de memória utilizada.

Palavras-Chave: Descoberta de Padrões; Preferências OLAP; *Queries* MDX; Algoritmos de Mineração, e Regras de Associação.

Abstract

OLAP Mining

The many benefits provided by data warehouses, in particular regarding to storage and data processing, have led to a substantial growth of the data warehousing market and in the number of organizations who adopted these systems.

In fact, the data model of this type of structures allows the user to perform a large number of different operations: complex queries, find the most interesting information, aggregate and compare different values, and to provide an interactive data visualization.

However, its complexity brings some computation and materialization costs. The pre-computation of the all data cube can provide a precise and fast response to analytical queries, but it requires an enormous quantity of space to storage all materialized views.

The application of data mining techniques, such as algorithms for mining association rules, allows the discovery of frequent items among data and, consequently, the definition of OLAP preferences.

The study of OLAP preferences presented in this dissertation aims to identify the most accessed parts in a data cube and to define which parts should be materialized. With the identification and materialization only of the important parts for the analysis, it is possible to preserve a satisfactory query response time, achieving a significant reduction of memory costs.

Keywords: Pattern Discovery; OLAP Preferences; MDX *Queries*; Data Mining, and Association Rules.

Índice

Índice de Figuras

Índice de Tabelas

Lista de Siglas e Acrónimos

AIS	Adaptive Importance Sampling
DB	Database
FP-tree	Frequent Pattern Tree
KDD	Knowledge Discovery in Databases
LDA	Linear Discriminant Analysis
MDX	Multidimensional Expressions
OLAM	On-Line Analytical Mining
OLAM SE	Self Explaining On-Line Analytical Mining
OLAP	Online Analytical Processing
SQL	Structured Query Language
TID	Transaction Identification Code
UML	Unified Modeling Language
XML	Extensible Markup Language

Capítulo 1

Introdução

1.1 Contextualização

Nos últimos tempos tem vindo a registar-se um enorme crescimento na quantidade de dados armazenados e na percepção da informação útil que daí pode ser retirada e aplicada em diversas áreas. Por isso, é de extrema importância os desenvolvimentos que têm sido feitos nas áreas de *Data Warehousing*, processamento analítico de informação (OLAP) e mineração de dados.

A mineração de dados, também conhecida por *Knowledge Discovery in Databases* (KDD), consiste na análise exploratória dos dados em grandes bases de dados, de forma a descobrir conhecimento relevante, padrões e relações entre eles. Hoje em dia, as técnicas de mineração de dados são aplicadas em várias áreas, desde vendas, seguros, medicina, telecomunicações, entre outras, com os mais diversos propósitos, tal como a detecção de fraudes, aumento de lucros, ou assistência em pesquisas médicas (Seifert, 2004). A descoberta de conhecimento em bases de dados envolve várias fases antes da aplicação de técnicas de mineração, que passam, nomeadamente, pela limpeza dos dados, a sua integração, seleção, e transformação e, só depois, a sua mineração, a avaliação de padrões e a sua apresentação do conhecimento adquirido. As diversas técnicas de mineração têm vindo a ser exploradas e aplicadas em diferentes tipos de bases de dados, desde

bases de dados relacionais, espaciais, multimédia, ou *data warehouses*, entre outras (Zaiane, 1999).

Um *data warehouse* é um dos elementos essenciais nas tecnologias de informação. Estas estruturas multidimensionais permitem uma melhor análise e visualização dos dados, uma vez que armazenam valores quantitativos, denominados medidas, ou seja, os objetos de análise, que são usualmente definidos através de várias dimensões de análise. As dimensões indicam o contexto em que essa medida se insere. Em termos práticos, podem ser, por exemplo, produtos, cidades, fornecedores, entre outros. A estrutura multidimensional vai-se formando através da agregação dessas dimensões, obtendo diversas medidas para as diferentes combinações de dimensões (Chaudhuri and Dayal, 1997). Este modelo de dados permite, entre várias opções, realizar pesquisas mais complexas, selecionar conjuntos de dados de maior interesse, executar operações de sintetização, fazer comparações de dados sobre várias dimensões e ter diferentes visualizações dos dados, através de ferramentas de processamento analítico de informações (Sarawagi et al., 1998).

Um facto pertinente na implementação deste tipo de estruturas é a sua computação. A computação de cubos de dados envolve o cálculo das várias agregações. A pré-computação afeta diretamente o tempo de resposta nas pesquisas realizadas. Quanto mais partes do cubo tivermos materializadas, menor será o tempo de resposta. No entanto isto traz alguns problemas, nomeadamente no que se refere à quantidade de espaço de armazenamento necessário:

- uma pré-computação de todo o cubo torna os tempos de resposta mais rápidos, mas vai requerer bastante memória;
- a não computação de qualquer parte do cubo minimiza a quantidade de memória requerida, mas os tempos de resposta vão ser mais lentos.

Por isso, quando se trata de estruturas multidimensionais é útil analisar que tipo de dados é mais importante, os preferenciais, e chegar a um compromisso entre memória e tempo de processamento/resposta para definir quais as células do cubo que se devem pré-computar (Mahar, 2009).

1.2 Motivações e Objetivos

Usualmente, quando estamos a trabalhar sobre estruturas como os cubos de dados, existem várias células materializadas que são pouco ou nada utilizadas, o que leva a uma utilização ineficiente de espaço de armazenamento. Desta forma, a descoberta de padrões de utilização destas estruturas, o estudo e análise das pesquisas efetuadas, a identificação de células mais e menos acedidas, assim como uma série de outros dados, podem ser bastante úteis para contornar esse problema, e consequentemente reduzir custos de execução/computação.

Visando o estudo de padrões e a identificação de relacionamentos entre os dados optou-se pela aplicação de técnicas de mineração sobre uma estrutura de dados multidimensional. A estrutura multidimensional em estudo consistiu num conjunto de dados relativos aos acessos realizados por utilizadores a um outro cubo de dados, isto é, um conjunto de *queries* MDX (*Multidimensional Expressions*) que foram realizadas sobre o cubo de dados ao longo do tempo. Este tipo de *queries*, utilizadas para realizar pesquisas em fontes de dados OLAP, retorna um conjunto de células desse mesmo cubo. Assim, julga-se ser de grande utilidade analisar em pormenor as pesquisas efetuadas, identificar as dimensões e células mais acedidas, as menos acedidas e ainda outras informações que possam ser relevantes. A escolha de um algoritmo de mineração de dados, como, por exemplo, algoritmos de regras de associação, podem ser bastante úteis para a realização deste tipo de estudos.

A informação que poderá ser obtida através deste estudo poderá ter alguma relevância, não só a nível académico mas também a nível empresarial, dado que se nota um crescente investimento por parte das empresas e organizações em *data warehouses*. É de salientar que investir num Sistema de *Data Warehousing* pode ser custoso, não só a nível da implementação mas também a nível monetário. Muitas vezes as empresas optam por ter toda a estrutura de dados materializada de modo a obter uma melhor performance nas consultas realizadas, mas isso traz custos acrescidos, nomeadamente na quantidade de memória necessária. Identificando os dados mais utilizados é possível chegar a um consenso das partes de um cubo que podemos não materializar (as que considerarmos, com base em algum critério, serem irrelevantes para os utilizadores), mantendo tempos de resposta das pesquisas aceitáveis e reduzindo significativamente a quantidade de memória utilizada.

1.3 Organização da dissertação

Além do presente capítulo, esta dissertação está organizada da seguinte forma:

- Capítulo 2 – Preferências de Utilização OLAP. Neste capítulo, será feita uma introdução ao conceito de estrutura multidimensional e *queries* MDX. Será ainda apresentada a definição de preferências OLAP, em que consistem, e como surgiram a partir da necessidade de contornar alguns problemas de utilização de cubos OLAP, nomeadamente pesquisas ineficientes, cujos resultados não satisfazem as necessidades de análise.

- Capítulo 3 – Extração de Conhecimento em Estruturas Multidimensionais. Aqui serão apresentadas diversas abordagens de mineração de estruturas multidimensionais, dando ênfase especial a técnicas de mineração de regras de associação. Também serão revelados alguns casos em que a mineração de estruturas multidimensionais é vantajosa.

- Capítulo 4 – O Caso de Estudo. Todo o processo de trabalho realizado com base neste caso de estudo será demonstrado neste capítulo, desde a recolha e preparação dos dados, até à aplicação de técnicas de mineração e extração dos resultados. Os resultados foram focalizados em duas vertentes de análise, uma mais geral e outra temporal. Este capítulo termina com a apresentação do *CubesDefiner*, uma ferramenta desenvolvida com o objetivo de permitir apresentar os resultados das preferências obtidas de forma simples ao utilizador e como adaptá-los às necessidades requeridas.

- Capítulo 5 – Conclusões e Trabalho Futuro. Finalmente, neste capítulo, serão apontados alguns comentários e conclusões sobre o trabalho realizado, assim como algumas ideias e sugestões para estudos futuros.

Capítulo 2

Preferências de Utilização OLAP

2.1 Cubos OLAP e Queries MDX

Ao se falar em estruturas multidimensionais, ou simplesmente em cubos de dados, é inevitável falar-se em *queries* MDX. A linguagem de expressões multidimensionais (MDX) (Microsoft, 2012) foi criada pela Microsoft de forma a permitir a exploração de bases de dados multidimensionais. Estes dois conceitos estão diretamente relacionados, uma vez que as *queries* MDX são executadas sobre cubos de dados e o resultado da sua execução é o próprio cubo ou parte dele (Niemi et al., 2001).

Desta forma, antes de procedermos à definição formal de uma *query* MDX é necessário definir a própria estrutura multidimensional sobre a qual esta será aplicada, assim como alguns conceitos relacionados (Golfarelli and Rizzi, 2008).

Definição 1 (Esquema multidimensional). Um esquema multidimensional é definido por M=<A, H, M> onde:

- A= $\{a_1,\ldots,a_p\}$ é um conjunto finito de atributos, cada um definido sobre um domínio categórico $Dom(a_k)$.

- H={h_1,…,h_n} é um conjunto finito de hierarquias, cada uma caracterizada por (1) um subconjunto de atributos Attr(h_i) $\subseteq$ A, (2) uma ordem total de roll-up $\geq_{hi}$ sobre Attr(h_i), e (3) e uma família de funções de roll-up, incluindo a função $RollUp_{aj}^{ak}$: Dom(ak) $\rightarrow$ Dom(aj) para cada par de atributos a_k e a_j tal que $a_k \geq_{hi} a_j$.
- Um conjunto finito de medidas M={m_1,…, m_l}, cada uma definida por um domínio numérico Dom(m_i).

Para cada hierarquia h_i, o atributo superior da ordem é definido por DIM_i, e determina um nível de agregação mais fino. Por outro lado, o atributo inferior é definido por ALL_i, e determina o maior nível de agregação, ou seja, o que apresenta menos nível de detalhe. Através do conceito de hierarquia apresentado, assim como a definição de esquema multidimensional, já é possível definir o que é o conjunto de agregações.

Definição 2 (Conjunto Group-by). Dado um esquema multidimensional M=<A,H,M>, de domínio Dom(H)= Attr(h1)×…×Attr(hn), temos um conjunto de agregações (*group-by*) G∈Dom(H), onde G=<ak1,…akn> e Dom(G)=Dom(ak1)×…×Dom(akn). Cada g∈Dom(G) é uma coordenada de G na *lattice* de agregações.

A *lattice* de um cubo representa os diferentes níveis de agregação da estrutura e podem ser observados na figura seguinte.

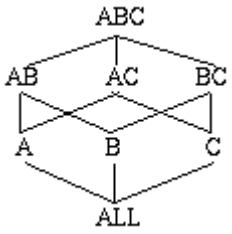


Figura 1 - *Lattice* de um cubo multidimensional (Stonebraker and Hellerstein, 1998)

Nesta figura 1 está representada a *lattice* de um esquema multidimensional com três atributos ou dimensões. O nó designado por ABC define o nível de agregação mais fino, enquanto que o nó ALL define o nível menos detalhado. Dentro da *lattice* do cubo podem existir diversos níveis hierárquicos para cada atributo. Consequentemente, para cada coordenada da *lattice* podem existir mais que um conjunto de agregação.

Consideremos o seguinte exemplo no qual temos a *lattice* de um cubo composto por três dimensões: Customer, Product e Date, cada uma delas com as seguintes hierarquias, representadas em ordem *roll-up*:

Customer → City → Nation → AllNats
Product → Type → Category → AllCats
Date → Month → Year → AllYears

Para este esquema, podemos representar o conjunto de elementos do topo da *lattice* como $G^{\top}$=<Customer, Product, Date> e da parte inferior da lattice como $G^{\perp}$=<AllNats, AllCats, AllYears>. Ao realizarmos uma operação de *roll-up* sobre a dimensão Customer vamos ter, por exemplo, um conjunto de agregação definido pelos atributos <Nation, Product, Date>.

Após definir o esquema e a sua constituição já é possível fazer uma definição formal dos seus dados. Os dados de um esquema multidimensional representam factos e que são caracterizados por uma determinada granularidade ou conjunto de agregações. Os factos podem ser dados elementares, definidos pelo nível de granularidade mais fino ou dados mais agregados, representados pelos restantes níveis de granularidade.

Definição 3 (Facto). Tendo em conta um esquema multidimensional M=<A,H,M> e um conjunto de agregação G∈Dom(H), um facto sobre G é dado por f = <G, g, v>, onde g ∈ Dom(G) e v ∈ Dom(M) = Dom(m1) × . . . × Dom(ml). Um exemplo de um facto pode ser:

f1 =<G1, <'Italy', 'Crunchy', 'Oct. 19 2007'>, <100, 30>>

Definição 4 (Cubo de dados). Dado um esquema multidimensional M, uma instância de M, denominado cubo de dados, é um conjunto de factos $C \in F_M$, onde F_M é o conjunto de todos os factos para um dado esquema M.

Por outras palavras, um cubo de dados é constituído por um conjunto de factos primários e por um conjunto de factos obtidos pela aplicação dos diversos níveis de agregação apresentados acima.

Após definidos os conceitos base de uma estrutura multidimensional, chegou a altura de definir formalmente o que é uma *query* MDX. As expressões multidimensionais, denominadas *queries* MDX, foram introduzidas pela Microsoft como uma linguagem declarativa que permite realizar pesquisas sobre bases de dados multidimensionais (Niemi et al., 2001).

Definição 5 (Query MDX). Seja C = <D1, …, Dn, F> um cubo de dados multidimensional e Ri um conjunto de membros da dimensão Di, para todo o $i \in [1,n]$, uma *query* MDX realizada sobre C é dada por um conjunto de referências R1×…×Rn (Giacometti et al., 2008).

A estrutura de uma *query* MDX pode parecer à primeira vista semelhante à de uma *query* SQL, pois também é representada por uma operação SELECT-FROM-WHERE. No entanto, uma *query* MDX utiliza uma estrutura específica que permite efetuar pesquisas sobre hierarquias e agregações de dados. De forma geral, uma *query* MDX tem a seguinte representação (Niemi et al., 2001):

```
SELECT <axis specification> [, <axis specification>, …]
FROM <cube specification>
WHERE <slicer specification>
```

em que 'Axis specification' permite definir os eixos de análise sobre os quais queremos efetuar a pesquisa. O resultado de uma *query* MDX retorna um subconjunto de dados multidimensionais, que pode ter até 128 eixos de análise. Por isso, é importante definir os eixos de análise assim como o nível de agregação sobre o qual queremos o resultado.

No 'cube specification' é necessário especificar qual o cubo de dados sobre o qual pretendemos fazer a pesquisa. Já a cláusula WHERE, tal como numa *query* SQL, é utilizada para restringir o

conjunto de resultados. No entanto, neste caso esta cláusula pode ser utilizada para especificar a parte do cubo que será vista no resultado final (Nolan, 1999, Microsoft, 2012).

O exemplo apresentado a seguir representa uma *query* efetuada sobre um cubo de dados, que permite analisar as vendas de produtos alimentares e bebidas em todas as lojas de São Francisco.

Exemplo. *Query* MDX sobre um cubo de dados de vendas:
SELECT {[Store].[All Stores].[USA].[CA].[San Francisco]} ON COLUMNS, {[Product].[All Products].[Food], [Product].[All Products].[Drink]} ON ROWS
FROM [Sales]
WHERE {[Measures].[Unit Sales]}

Formalmente, tendo em conta a definição apresentada acima, a *query* apresentada é dada pelo seguinte conjunto de referências: {<Drink, alltime, allcustomer, San Francisco>, <Food, alltime, allcustomer, San Francisco>} (Giacometti et al., 2009).

2.2 Preferências OLAP

Os *data warehouses* são desenvolvidos de forma a conseguirem suportar enormes quantidades de dados que possam ser relevantes para o negócio das organizações. Devido ao seu tamanho e complexidade pode tornar-se bastante complicado para os utilizadores realizarem a procura e exploração de dados. Por vezes, a falta de familiarização com a estrutura existente e dos dados que poderão ser úteis para a análise pretendida podem levar o utilizador à realização de pesquisas ineficientes, ou seja, à execução de *queries* MDX cujos resultados não satisfazem as necessidades de análise. Aquando a exploração de cubos, existem dois cenários possíveis de pesquisa ineficiente: por um lado a execução da *query* MDX pode retornar pouca ou nenhuma informação, e por outro lado pode devolver ao utilizador demasiados dados, o que torna difícil identificar quais os mais importantes e, daí, retirar alguma informação útil (Golfarelli and Rizzi, 2009).

Foi na sequência das tentativas de superar este problema, e na necessidade de restringir e especificar melhor a informação relevante para os utilizadores, que foi definido o conceito de preferências OLAP. O conceito de preferência em si já tinha aparecido anteriormente nas bases de

dados transacionais. No entanto, quando se trata de preferências OLAP existe um conjunto de fatores a ter em conta para além dos tuplos, ou melhor dizendo neste caso dos factos, nomeadamente as agregações e hierarquias existentes. A hierarquia define o nível de agregação dos dados, e este, consequentemente, acaba por ter um forte impacto no tamanho do resultado que é retornado. Enquanto que numa base de dados transacional uma preferência é definida como um conjunto de possíveis combinações de valores de atributos, num cubo de dados uma preferência pode envolver combinações de factos de diferentes grupos de agregação e com diferentes níveis de hierarquia (Golfarelli and Rizzi, 2008).

Desta forma, uma preferência OLAP pode ser definida formalmente da seguinte maneira (Golfarelli and Rizzi, 2009):

Definição (Preferência OLAP). Dado um esquema multidimensional M, uma preferência P sobre M é definida como um par ($<_p$, $\cong_p$) onde $<_p \subseteq F_M \times F_M$ é uma s.p.o. (*strict partial order*, ou, ordem parcial estrita) e $\cong_p \subseteq F_M \times F_M$ é uma SV-relation (*substitutable values relation* ou relação de valores substituíveis) sobre $<_p$.

Seguindo esta definição, e supondo que estamos a analisar dois factos, f1 e f2, ao ter f1 $<_p$ f2, podemos afirmar que f2 é preferido a f1. Se tivermos f1 $\cong_p$ f2, podemos afirmar que, em termos de preferência, os dois factos são equivalentes.

Para melhor compreensão desta definição é necessário recordar alguns conceitos já referidos, como é o caso de s.p.o e SV-relation. Uma ordem parcial estrita sobre um conjunto S representa uma relação irreflexiva e intransitiva entre elementos desse conjunto. Considerando dois elementos x, y e z pertencentes a S, temos uma ordem parcial estrita entre x e y se forem respeitadas as seguintes propriedades:

$$[\text{propriedade irreflexiva}]\ \neg(x <_p x)$$
$$[\text{propriedade transitiva}]\ x <_p y \wedge y <_p z \Rightarrow x <_p z$$

Por outro lado, a relação diz-se de valores substituíveis (SV-relation) sobre $<_p$ se, para esses mesmos elementos do conjunto (Kieβling, 2002):

$$x \cong_p y \Rightarrow \neg(x <_p y) \wedge \neg(y <_p x)$$
$$x \cong_p y \wedge \exists z : z <_p x \Rightarrow z <_p y$$

$$x \cong_p y \wedge \exists z : x <_p z \Rightarrow y <_p z$$

Tendo em conta as condições acima referidas, necessárias a ter em conta na definição deste tipo de preferências, Golfarelli e Rizzi conseguiram mostrar em (Golfarelli and Rizzi, 2008) que era possível definir preferências de exploração analítica com base em todo domínio OLAP. Desta forma, especificaram três tipos base de preferências, nomeadamente preferências sobre atributos, medidas e hierarquias, assim como uma gramática, baseada na gramática proposta por Werner Kießling em (Kießling, 2002), para a sua representação.

Preferências sobre atributos

Quando lidamos com bases de dados relacionais os atributos são utilizados para caracterizar tuplos. O mesmo acontece com os factos nas bases de dados multidimensionais. No entanto, neste caso, os atributos que caracterizam um facto estão relacionados a valores agregados, assim como a hierarquias. Por exemplo, para um valor específico de um atributo, os outros atributos podem ter mais que um valor, e, consequentemente, existirão diversos valores agregados.

Desta forma, as preferências sobre atributos, definidas por Golfarelli e Rizzi, permitiram determinar quais o factos preferenciais com base nos valores dos seus atributos, isto é, dada uma determinada hierarquia h, um determinado atributo a e um valor categórico c assumido por a, podemos definir um facto como preferencial ou não se para qualquer nível de agregação de h, o valor do atributo assume o valor de c (Golfarelli and Rizzi, 2008).

Para melhor compreender este conceito consideremos os construtores POS e NEG da gramática definida por Golfarelli (Golfarelli and Rizzi, 2009):

- POS (Month, 'October-2008') – com base no conceito de preferência sobre atributos é possível afirmar que factos cujo mês é Outubro de 2008 e factos obtidos através da aplicação de operações de *roll-up* ou *drill-down* sobre esta hierarquia, ou seja, factos sobre qualquer dia do mês de Outubro de 2008, ou factos de 2008, são preferenciais.
- NEG (Month, 'October-2008') – com este construtor indicamos que são preferenciais quaisquer factos cujo mês **não** é Outubro de 2008 ou factos que **não** são obtidos através da aplicação de operações de *roll-up* ou *drill-down* sobre esta hierarquia.

Preferências sobre medidas

Tal como os atributos, também as medidas são utilizadas para caracterizar factos. Desta forma, faz sentido que preferências OLAP também considerem este elemento de análise. Golfarelli definiu assim preferências com base em medidas que permitem definir se um facto é preferencial ou não tendo em conta o domínio numérico das medidas que o caracterizam.

Para representar este tipo de preferências Golfarelli e Rizzi definiram três novos construtores para a sua gramática:

- BETWEEN (m, vlow, vhigh): que permite definir um facto como preferencial, se o valor da sua medida m estiver compreendido entre um valor mínimo vlow e um valor máximo vhigh. Para os restantes factos foram do intervalo, podemos compará-los com base na distância do seu valor para o intervalo.

Por exemplo: um facto f2 diz-se preferencial a um facto f1, se a distância do seu valor de m para o intervalo for menor que a distância do valor de m do facto f1. Ou seja:

$f1 <_P f2,\ iff\ \Delta(f1.m, [vlow, vhigh]) > \Delta(f2.m, [vlow, vhigh])$, onde a distância Δ é dada por:

$$\Delta(v, [vlow, vhigh]) = \begin{cases} 0, & if\ v \in [vlow, vhigh] \\ vlow - v, & if\ v < vlow \\ v - vhigh, & if\ v > vhigh \end{cases}$$

- LOWEST(m), HIGHEST(m): permite definir um facto como preferencial se o seu valor de m for o mais baixo (lowest) ou mais alto (highest) possível.

Preferências sobre hierarquias

Por último, Golfarelli e Rizzi, definiram preferências sobre hierarquias, ou seja, permitiram definir preferências sobre os diversos níveis de agregação dos factos.

Para tal, foi definido o conceito de distância entre atributos pertencentes a uma determinada hierarquia e quatro novos operadores. O conceito de distância entre atributos contempla a distância entre o nível hierárquico em que cada atributo se encontra.

Por exemplo: considerando a hierarquia h={Product→Type→Category→AllCats}, a distância Dist(Product, Category)=2.

Para representar este tipo de preferências os autores definiram os seguintes operadores:

- CONTAIN(h, a) – em que são preferenciais os factos cujo conjunto de agregação contenha o atributo a da hierarquia h.
- NEAR(h, afine, acoarse) – em que são preferenciais os factos cujo conjunto de agregação contenha um atributo *a* compreendido entre dois níveis hierárquicos definidos pelos atributos *afine* e *acoarse* da hierarquia h. Este operador é em certa parte semelhante operador BETWEEN definido para as preferências sobre medidas, pois também este contempla a distância, neste caso entre atributos, para comparar dois factos cujo conjunto de agregação não se encontra compreendido entre os níveis hierárquicos definidos. Ou seja, um facto f1 é preferido a f2 se o seu nível de agregação for mais próximo ao definido pelo operador.
- COARSEST(h) – que define como preferenciais os factos mais agregados sobre uma hierarquia h.
- FINEST(h) – ao contrário do construtor COARSEST, este operador define como preferenciais os factos cujo grão seja mais fino, ou seja, os factos mais detalhados sobre uma determinada hierarquia h.

Para melhor compreender estes construtores seguem-se os seguintes exemplos:

Considere as hierarquias Time = {Date→ Month→ Year→ AllYears}, Product = {Product→ Type→ Category→ AllCats} e Customer = {Customer→ City→ Nation→ AllNats}.

Exemplo 1. CONTAIN(PRODUCT,Type) – São preferenciais todos os factos cujo conjunto de agregação contenha Type. Por exemplo, factos definido pelo conjunto de agregação G^T=<Customer, Type, Month> são preferidos.

Exemplo 2. NEAR(TIME,Month,Year) – São preferenciais os factos cujos dados estejam agregados por Month ou Year. Factos definidos, por exemplo, pelo conjunto de agregação G^T=<Customer,

Type, Date> **não** são preferidos. Se, por outro lado, fossem preferenciais os factos definidos por NEAR(TIME,Date,Year), tanto os dados agregados por Date, Month ou Year seriam preferidos.

Exemplo 3. FINEST(TIME) – São preferenciais os factos mais detalhados sobre a hierarquia TIME. Por exemplo, factos definidos pelo conjunto de agregação G^T=<Customer, Type, Date> são antepostos a factos definidos por G^T=<Customer, Type, Month>

2.3 Vantagens das preferências

Com o que já vimos sobre as preferências, podemos afirmar que estas refletem os dados que são mais interessantes para os agentes de decisão que utilizam cubos no seu dia-a-dia. Assim, conhecer as preferências de utilização de cubos pode ter bastante impacto nos resultados a serem retornados ao utilizador. Conseguir fornecer ao agente de decisão exatamente a informação relevante e necessária para a análise pretendida é uma das maiores vantagens quando se conhecem as preferências OLAP. No entanto, diretamente relacionados com esta vantagem, encontram-se outros pontos em que o facto de conhecermos as preferências de um utilizador tem bastante impacto. Ao se conhecer o que o utilizador pretende e ao devolver-lhe apenas partes do cubo com a informação necessária, uma série de aspetos relacionados com a sua computação e materialização podem ser melhorados e, consequentemente, haver uma redução significativa nos custos de computação e de memória utilizada.

Para além da melhoria da qualidade dos resultados das *queries*, todo o processo de análise pode ser melhorado. Como já foi referido, a própria estrutura multidimensional é bastante complexa e pode ser por vezes complicado para o analista procurar a informação que necessita. Ter uma interface simples que mostre ao utilizador recomendações de *queries*, baseadas em utilizações anteriores, ou que permita a introdução das suas próprias preferências torna bastante mais simples e eficaz a extração de informação. Também a visualização dos próprios resultados podem ser personalizados e adaptados consoante o utilizador.

A análise prolongada de preferências de utilização de cubos OLAP pode também ser útil para indicar se este satisfaz todas as necessidades da organização, e, se necessário, proceder a uma reestruturação da própria estrutura multidimensional (Kozmina and Niedrite, 2010).

O estudo de preferências OLAP acaba por ser bastante semelhante ao que já era feito para serviços de *e-commerce*, por exemplo, ou de análise de *clickstreams*, em que são traçados perfis de utilização de forma a analisar as preferências dos utilizadores e adaptar os serviços para uma melhor satisfação (Golfarelli and Rizzi, 2009).

Capítulo 3

Extração de Conhecimento em Estruturas Multidimensionais de Dados

3.1 OLAM – Mineração de sistemas OLAP

Ao longo dos anos têm sido realizados diversos estudos nas áreas de OLAP e *data mining* com o objetivo de otimizar a exploração e análise de dados, e com isso a descoberta de padrões e relacionamentos entre os mesmos. Apesar de ambas fornecerem boas ferramentas e soluções neste campo, estas ainda apresentam algumas limitações. O *Online Analytical Processing* tem poderosas ferramentas para organização e visualização de dados dentro do *data warehouse*, desde que esses dados sejam simples, pois não consegue lidar com dados complexos como imagens, texto, som ou vídeo. Por outro lado, as ferramentas de *data mining* não são adequadas para a organização e visualização, uma vez que são direcionadas para conjuntos de dados unidimensionais. No entanto, estas técnicas apresentam bons resultados na análise e descoberta de conhecimento, tanto em dados simples como complexos (Messaoud et al., 2004, Messaoud et al., 2006b).

De forma a tirar partido das melhores características de cada uma dessas áreas, Jiawei Han, cientista de renome na área de base de dados, dedicou grande parte do seu trabalho de investigação à mineração de dados e ao desenvolvimento de sistemas, como por exemplo o DBMiner (Han et al., 1997a), que permitem integrar OLAP com técnicas de mineração de dados. O principal objetivo desse trabalho de investigação foi aplicar um conjunto de métodos de mineração de dados, como por exemplo caracterização, classificação, associação, previsão ou segmentação, interactivamente a diversas partes de uma base de dados multidimensional e em diversos níveis de abstração (Han et al., 1998).

Segundo Han (1997), a mineração de estruturas multidimensionais é uma área ainda muito pouco explorada, mas que se poderá revelar bastante promissor devido principalmente às seguintes razões:

1) num *data warehouse* os dados são integrados, consistentes e limpos. Características essenciais tanto para o processamento analítico como para a mineração de dados;

2) a integração permite aplicar técnicas de mineração de dados em diversos níveis de abstração e ainda a diferentes subconjuntos de dados, devido à sua utilização conjunta com técnicas de OLAP como o *slice, dice* e *pivoting*;

3) a interação entre utilizadores e sistema torna-se mais fácil e flexível, permitindo que estes possam selecionar e alterar funções de mineração dinamicamente.

Nesta abordagem, a integração destas duas tecnologias resulta da adaptação de técnicas tradicionais de mineração às estruturas multidimensionais de dados, obtendo-se métodos de mineração de dados baseados em cubos, também denominados de métodos de OLAP *Mining* ou OLAM (Han et al., 1998).

3.1.1 Abordagens estudadas

Nos últimos anos diversos autores viram em OLAP e *Data Mining* duas áreas complementares e debruçaram as suas pesquisas e trabalhos científicos sobre esta área de OLAM. Ao analisar esses trabalhos foi possível identificar soluções e diversas propostas. No entanto, penso que podemos

agrupar cada uma dessas soluções em três principais abordagens de integração e de descoberta de informação em estruturas multidimensionais.

Sistemas OLAM

Como já foi mencionado, Jiawei Han foi o grande pioneiro no que toca à integração destas duas áreas. Ele introduziu o conceito de On-Line Analytical Mining (OLAM) como o processo de extração de conhecimento de bases de dados multidimensionais através da aplicação de métodos de mineração de dados da mesma maneira que as técnicas de processamento analítico são aplicadas a essas estruturas (Han, 1998).

Han e outros investigadores desenvolveram um sistema OLAM, o DBMiner (Han et al., 1997a), que permite a análise interativa sobre diversas partes dos cubos de dados e sobre diferentes níveis de abstração. Tendo por base o conceito de OLAM, este sistema integra técnicas de OLAP e mineração de dados, para além de outras tecnologias da área de base de dados, que permitem tanto a descoberta de conhecimento em *data warehouses* como em bases de dados relacionais. O DBMiner integra um sistema de base de dados com conceitos de hierarquia, que permitem explorar e analisar cubos de dados e, ainda, módulos com tecnologias de descoberta de conhecimento. Estes módulos contêm métodos de mineração de dados, baseados nos métodos tradicionais como a classificação, caracterização, associação, entre outros, que foram adaptados de forma a ser possível a sua aplicação em estruturas multidimensionais.

Também Goil and Choudhary (1998) desenvolveram uma infraestrutura que tira partido das características destas duas áreas. A ideia principal destes dois autores foi adaptar abordagens já existentes de mineração de dados orientada a atributos, utilizadas para descoberta de regras de associação entre esses mesmos atributos, a cubos de dados. A adaptação e integração desta abordagem com OLAP permite encontrar relacionamentos entre objetos do cubo de dados, nomeadamente dimensões, e com isso identificar as agregações que devem ser materializadas e reduzir o espaço necessário para o seu armazenamento. Este trabalho traz assim um enorme contributo para o estudo destas estruturas e sua otimização.

Por sua vez, Chmelar e Stryka (2008) propuseram um sistema OLAM, o OLAM SE (Self Explaining On-Line Analytical Mining), que foi baseado no sistema OLAM de Han, mas que apresenta algumas novas funcionalidades. De forma a desenvolver um sistema mais simples e fácil de interagir, em comparação com o DBMiner, os autores implementaram as seguintes melhorias:

1) Introduziram duas novas métricas, denominadas *cover* e *obviosity*. O primeiro parâmetro, *cover*, determina quais as classes de dados que são essenciais para a análise, ou seja, este parâmetro é utilizado para identificar quais os valores que são importantes para a análise, evitando que dados insignificantes sejam analisados. O segundo parâmetro, *obviosity*, é utilizado como suporte máximo, prevenindo que sejam aplicados métodos de mineração sobre classes com ganho de informação muito alto. Por outras palavras, dados que nos proporcionem informação direta, sem ser necessário aplicar métodos de extração de conhecimento, são excluídos do processo de mineração.

2) A camada de apresentação do sistema foi melhorada e simplificada, permitindo a sua fácil interação tanto com utilizadores experientes como inexperientes. A estrutura desta camada foi baseada nos diagramas UML (*Unified Modeling Language*), de forma a permitir a representação de relacionamentos, associações, agregações e hierarquias entre dimensões. A interface permite ainda esconder ou mostrar atributos, simulando as operações de *drill-down* e *roll-up* do OLAP, e uma opção para realizar operações de *slice and dice* sobre hierarquias.

3) Foram criados dois modos de mineração: *online* e *offline*. No modo *online* a mineração de dados é aplicada interactivamente de forma a extrair conhecimento rapidamente. O modo *offline* foi desenvolvido para permitir a execução de algoritmos de mineração de dados com elevada complexidade computacional. Neste modo esses algoritmos não executados iterativamente em background e é-lhes atribuído um valor de prioridade baixo. O objetivo principal deste modo é permitir executar operações de mineração adicionais - mas não essenciais -, que são um complemento às operações de mineração realmente necessárias, sem que tenham impacto muito relevante sobre a performance do sistema.

Transformação de dados multidimensionais

Nesta abordagem os dados multidimensionais são transformados de forma a permitir a aplicação de métodos de mineração de dados tradicionais. Helen Pinto (2001) apresenta um método que permite extrair e analisar informação de estruturas multidimensionais através de mineração de padrões sequenciais. Nesta abordagem os dados multidimensionais são extraídos e colocados em sequências de dados, sendo posteriormente aplicados os métodos de mineração sobre o conjunto

de sequências. Cada valor de uma dimensão é tratado como um item de dados sequencial. Sobre a nova estrutura com a informação multidimensional é aplicado um algoritmo baseado no Apriori, denominado PrefixSpan, de forma a encontrar as sequências frequentes. Ou seja, o algoritmo é utilizado para encontrar os conjuntos de itens frequentes que satisfaçam um valor de suporte mínimo estabelecido.

Seguindo também esta abordagem, Tjioe and Taniar (2004) desenvolveram dois algoritmos, Vavg e Havg, que permitem extrair dados de *data warehouses* e tratá-los de forma a ser possível aplicar sobre eles métodos tradicionais de mineração de dados, nomeadamente algoritmos de mineração de regras de associação. Tradicionalmente, estes algoritmos de regras de associação são aplicados sobre bases de dados transacionais, pois o facto de as bases de dados multidimensionais conterem dados sumarizados torna-se um obstáculo no processo de descoberta das regras. Desta forma, os autores desenvolveram estes dois algoritmos e optaram por utilizarem as medidas da tabela de factos do *data warehouse* como filtro dos dados que seriam extraídos. O algoritmo Vavg calcula verticalmente a quantidade média para cada chave da tabela de factos, procurando e calculando os valores desde a primeira até à última linha. Se a média calculada é maior ou igual a um valor mínimo definido, então esta é guardada numa nova tabela, bem como as dimensões que estão diretamente relacionadas. Posteriormente um algoritmo de mineração, como por exemplo o Apriori, é aplicado sobre a tabela resultante. O algoritmo Havg é semelhante ao Vavg, no entanto o cálculo da quantidade média é executado horizontalmente para cada linha na tabela de factos.

Adaptação de algoritmos de mineração

Nesta abordagem os métodos de mineração de dados tradicionais são transformados e adaptados de forma a ser possível aplicá-los diretamente sobre os dados multidimensionais. Os métodos que têm sido mais utilizados e adaptados são os de associação. Os algoritmos de mineração de regras de associação tradicionais permitem identificar relacionamentos entre valores dentro da mesma dimensão. No entanto, quando os valores se encontram em dimensões diferentes a descoberta regras de associação com os métodos tradicionais já não é possível. Desta forma, alguns autores em (Chen, 1999), (Nestorov and Jukic, 2003), (Vijayalakshmi and Raja, 2005) e (Messaoud et al., 2006a), apresentam novas abordagens nas quais algoritmos tradicionais, como o Apriori ou o FP-Growth, são adaptados de forma a possibilitar a descoberta de regras de associação relevantes em dados multidimensionais.

3.1.2 Técnicas de mineração

As técnicas de mineração OLAP são baseadas nos métodos de mineração de dados tradicionais, usualmente aplicados sobre bases de dados relacionais. Estas novas técnicas incorporam novas funcionalidades, derivadas do processamento analítico, aos métodos de mineração, de forma a ser possível a sua aplicação em estruturas de dados multidimensionais (Han et al., 1998). Han (1998) definiu um conjunto de técnicas de OLAM, baseadas em técnicas tradicionais, tais como a caracterização, associação, classificação e segmentação.

Caracterização baseada em cubos

Os métodos de caracterização baseados em cubos são utilizados para descrever conjuntos de dados relevantes com base em técnicas de generalização de dados. A aplicação do método resulta num conjunto de regras de caracterização que representam as principais características dos dados. Por exemplo, esta técnica pode ser útil para traçar perfis de clientes, uma vez que permite definir as principais características dos clientes mais frequentes. Os algoritmos de caracterização são integrados com técnicas de OLAP, como por exemplo o *drill-down* (*progressive deepening*) e o *roll-up* (*progressive generalization*), de forma a ser possível descobrir essas características em diferentes níveis de abstração.

Comparação/Discriminação baseada em cubos

Este método é utilizado para encontrar características que distinguem uma classe específica das restantes. O resultado da sua aplicação é um conjunto de regras discriminantes que descrevem as diferenças entre a classe em análise e as outras classes. As técnicas de comparação são bastante semelhantes às técnicas utilizadas para derivar regras de caracterização, no entanto estas utilizam medidas comparativas para conseguir fazer a distinção entre classes.

Associação baseada em cubos

Quando se trata de métodos de associação, Han (1998) salienta que as estruturas multidimensionais oferecem maior flexibilidade e eficiência na mineração de regras de associação, em comparação com as bases de dados relacionais. Nas bases de dados relacionais, os métodos de mineração de regras de associação são utilizados para encontrar um conjunto de relacionamentos de dependência entre diversos atributos. Ao lidar com estruturas multidimensionais é importante ter em consideração tanto as dependências entre atributos dentro da mesma dimensão, como entre diferentes dimensões. Desta forma, para este tipo de estruturas foram definidos dois tipos de associação: a inter-dimensão e a intra-dimensão. A associação inter-dimensão representa um conjunto de regras de associação entre duas ou mais dimensões do cubo. Por outro lado, a associação intra-dimensão representa um conjunto de regras de associação entre uma ou mais dimensões de referência, com um determinado nível de referência, sendo as outras dimensões agrupadas como um conjunto transacional de dados.

Todas as outras técnicas de OLAM são bastante semelhantes às técnicas tradicionais. No entanto, a mineração de regras de associação multidimensional pode ser um pouco mais confusa. Para uma melhor compreensão de como é feita a mineração em cubos de dados, segue-se o seguinte exemplo apresentado em (Han, 2000):

Exemplo: Suponhamos que temos uma estrutura multidimensional com várias dimensões, em que três delas são: Customer, Country e Product.

Regras de Associação Inter-dimensão:

- Assumindo que necessitamos saber quais os produtos que usualmente são comprados por consumidores do Canadá.
- Para responder a essa questão, pode ser definida uma regra de associação inter-dimensão, tal como a seguinte:

  Customer-Country("Canada")) Product("Coffee")

- Para esta regra foi utilizada a agregação entre as dimensões "Customer" e "Country", e analisados os relacionamentos entre esta agregação e a dimensão "Product".

Regras de Associação Intra-dimensão:

- Assumindo que necessitamos saber, para um determinado país, como por exemplo "Canada", quais os produtos que são comprados em conjunto.

- Para responder a essa questão, pode ser definida a seguinte regra de associação intra-dimensão:

  Within Customer-Country("Canada"):
  Product ("CarryBags")) Product ("Tents")

- Para esta regra a dimensão de referência utilizada foi o conjunto de consumidores que respeitassem um determinado nível de referência., aqui definido pelo nome do país "Canada". Para essa dimensão de referência foi analisada a dimensão "Product", ou seja, foram analisados quais os produtos adquiridos na mesma compra e estabelecidas as relações entre os mesmos

A derivação deste tipo de regras pode ser feita através da adaptação de algoritmos de mineração de regras de associação, como o Apriori ou o FP-Growth, de forma a ser possível a análise inter-dimensão e intra-dimensão em cubos de dados.

Classificação baseada em cubos

A aplicação deste tipo de métodos de classificação resulta num conjunto de modelos que descrevem classes de objetos. Estes modelos são obtidos através da análise de um conjunto de dados de treino. Ao analisar esses dados são extraídas as características de cada classe de dados identificada. Para cada uma das classes é criado um modelo com as suas características. Após serem definidos todos os modelos são geradas um conjunto de regras de classificação, que permitirão classificar novos objetos e indicar a classe à qual pertence. Os métodos de classificação baseados em cubos integram técnicas de mineração de dados tradicionais, como as árvores de decisão, classificador Naive Bayes ou a Análise de Discriminantes Lineares (LDA), com tecnologias de exploração de cubos de dados. Desta forma, antes de ser aplicado o algoritmo de mineração, é realizada uma operação de generalização sobre o conjunto de dados de treino inicial, permitindo desta forma a classificação de objetos nos diferentes níveis de abstração da estrutura multidimensional.

Segmentação baseada em cubos

Os métodos de segmentação permitem dividir os dados, agrupando os itens que têm características comuns em classes, denominadas por *clusters*. O principal objetivo deste tipo de métodos é manter um grau de similaridade elevado dentro de cada classe e um grau de similaridade baixo entre as diversas classes. O processo de segmentação é baseado nos métodos de mineração conhecidos, como é o caso da segmentação particional, que engloba, por exemplo, o algoritmo K-means, e da segmentação hierárquica. Da mesma forma que outros métodos já referenciados, a segmentação baseada em cubos também pode ser aplicada nos diversos níveis de abstração.

Previsão baseada em cubos

Com a integração das duas áreas tornou-se possível a aplicação de modelos de previsão sobre estruturas multidimensionais. Estes modelos são utilizados para prever valores de atributos específicos que são relevantes para a análise. A previsão é feita através da análise de fatores que possam influenciar esses mesmos atributos. Para a realização da análise desses fatores podem ser utilizados métodos estatísticos ou métodos de classificação, como os apresentados anteriormente. A previsão baseada em cubos integra ferramentas OLAP com métodos de previsão, permitindo que operações como o *drill-down* possam ser realizadas sobre o resultado da previsão, ou métodos de previsão possam ser aplicados a diferentes partes do espaço de abstração do cubo OLAP.

3.1.3 Áreas de aplicação

Tal como já foi mencionado, as tecnologias de *data warehousing* têm vindo a ser extremamente usadas em diferentes áreas de negócio, tal como retalho, serviços financeiros, telecomunicações e até mesmo na medicina. Nestas áreas são utilizadas técnicas de mineração de dados nos seus sistemas OLAP com os mais diversos propósitos, como por exemplo traçar perfis de clientes, análise de risco, detecção de fraude na indústria financeira, análise de chamadas telefónicas na

área de telecomunicações e ainda ajudar em diversos tipos de análise e investigação nas áreas de medicina e biomédica (Blazewicz, 2003). Para além da aplicação de técnicas de mineração OLAP nos tradicionais *data warehouses* orientados ao negócio, estas técnicas também têm sido usadas em *data warehouses* espaciais e de multimédia, a fim de permitir superar algumas lacunas existentes na análise de dados complexos, assim como na Web.

Aplicação na Web

Devido à enorme quantidade de informação que se encontra disponível *online*, a mineração de dados tem sido neste campo extremamente utilizada. Diversos métodos de mineração Web têm sido explorados e utilizados para extrair informação dos *Web Services* e de documentos disponíveis na *Web*. No entanto, têm vindo a aparecer novos interesses de análise nesta área, nomeadamente no que se refere ao e-commerce e ao estudo dos acessos por parte dos utilizadores às páginas Web. É neste tipo de análise que a mineração OLAP tem-se demonstrado bastante útil. Normalmente, as páginas *Web* guardam dados relativos ao seu histórico de acessos. A análise deste tipo de dados e a extração de informação nestes casos pode revelar-se bastante útil para efetuar por exemplo reestruturação nas próprias páginas *Web*, aplicar campanhas de marketing orientadas aos seus utilizadores, e, entre outras coisas, ajustar o desempenho dos servidores com base na sua utilização. Usualmente estes dados encontram-se armazenados em ficheiros de log nos servidores (Fung and Wong, 2002). Segundo Han (1998), uma boa abordagem para analisar esses dados é extraí-los, pré processá-los e armazená-los em cubos de dados e posteriormente aplicar técnicas de mineração sobre esse cubo.

Zaïane, Xin e Han (1998b) desenvolveram um sistema baseado no sistema DBMiner desenvolvido por Han, o WebLogMiner, que utiliza técnicas de OLAM para extrair padrões de ficheiros de log com os registos dos acessos às páginas Web. O processo de descoberta de informação executado por este sistema passa por quatro passos essenciais:

(1) A informação relevante contida no ficheiro de log é extraída e colocada numa base de dados relacional. A ideia é tornar a posterior construção do cubo muito mais simples.
(2) O cubo de dados é construído.
(3) São aplicadas operações OLAP sobre o cubo (*slice, dice, roll-up, drill-down*, etc).
(4) Os métodos de mineração são aplicados dentro do cubo de dados.

Vários autores em (Hu and Cercone, 2002), (Fung and Wong, 2002) e (Alves and Belo, 2003) seguiram esta abordagem para analisar *clickstreams*, ou, por outras palavras, a sequência de operações ou "cliques" que um utilizador realizou numa página Web, ou para extrair informação relativa às compras realizadas por consumidores na área de e-commerce. Um dos objetivos comuns em todos estes trabalhos é o estudo do comportamento dos consumidores/utilizadores de forma a ser possível melhorar a sua satisfação.

Dados espaciais

As bases de dados ou os *data warehouses* espaciais são utilizados para armazenar tanto dados espaciais, como pontos, linhas, regiões, etc., como dados não espaciais, que representam propriedades e características referentes aos objetos espaciais (Han, 1998). Tal como referi acima, analisar dados complexos, como dados espaciais, pode ser uma tarefa complicada e com custos de computação bastante elevados. Uma abordagem que pode ser utilizada para a otimização da análise deste tipo de dados é colocá-los em estruturas multidimensionais e, seguidamente, aplicar os algoritmos de mineração sobre essa estrutura (Pedersen and Tryfona, 2001). Também para este tipo de análise Han desenvolveu um sistema, o GeoMiner, que contem um módulo que permite construir cubos de dados espaciais a partir das bases de dados com os dados de origem e módulos de mineração que contêm algoritmos de mineração de dados que poderão ser aplicados sobre os cubos resultantes (Han et al., 1997b). Os algoritmos de mineração foram adaptados de forma a ser possível a sua aplicação tanto a bases de dados como a *data warehouses* espaciais.

Uma das abordagens que tem tido grande impacto na melhoria dos custos de computação deste tipo de cubos é a identificação de dados relevantes para a análise, através de algoritmos de mineração de dados, antes da fase de pré-computação do cubo, e a exclusão dos dados irrelevantes desse mesmo processo (Han, 1998).

Texto e dados multimédia

A abordagem apresentada na secção anterior pode ser aplicada a diversos tipos de dados complexos, nomeadamente a texto e a dados multimédia. Também nestes casos os dados são extraídos para estruturas multidimensionais e sobre eles são aplicados algoritmos de mineração de dados. Zaïane et al. (1998a) também desenvolveram um sistema baseado no DBMiner, o

MultiMediaMiner. Este sistema, para além de conter os módulos de OLAM existentes do DBMiner, incorpora outros módulos de extração e manipulação de dados multimédia.

3.2 Regras de associação

Os algoritmos de mineração de regras de associação são utilizados para encontrar associações ou relacionamentos entre os dados, ou seja, são úteis para determinar quais os conjuntos de itens que são frequentes e que ocorrem em conjunto. Este tipo de análise é utilizado complementarmente a métodos como o de classificação ou segmentação, permitindo ajudar no processo de tomada de decisão em diversas áreas, como telecomunicações, serviços comerciais, análise comportamental de clientes na *Web*, entre outras. Antes de definir uma regra é necessário conhecer alguns conceitos relacionados.

Definição 1 (Transação). Seja I={a1, a2,…, an} um conjunto de itens, podemos definir uma transação T como um conjunto de itens tal que $T \subseteq I$.

Definição 2 (Base de dados de transações). Seja I um conjunto de itens e T uma transação, podemos definir uma base de dados de transações DB=<T1, T2,…Tn> como um conjunto de transações Ti ($i \in [1..n]$). Desta forma uma base de dados de transações D sobre I, é um conjunto de transações sobre I.

Definição 3 (Padrão frequente). Seja A um padrão, que é um conjunto de itens, A diz-se frequente se o seu suporte, definido como a frequência absoluta de ocorrências, satisfaz um valor de suporte mínimo estabelecido.

Desta forma, sejam A e B dois conjuntos de itens, conseguimos definir uma regra de associação de notação $A \Rightarrow B$, onde $A \subset I$, $B \subset I$ e $A \cap B = \varnothing$. Cada regra com a forma $A \Rightarrow B$ que é gerada tem associados dois parâmetros: o suporte e a confiança. O suporte da regra é definido como a percentagem de transações na base de dados de transações DB que contêm $A \cup B$, ou seja, pode ser visto como a probabilidade $P(A \cup B)$. Por outro lado, a confiança é a percentagem de transações em DB que contem o conjunto B, sabendo que essa transação contém o conjunto A, ou seja, a probabilidade condicionada P(B|A).

Para uma regra ser considerada satisfatória tem de respeitar um valor de suporte e outro de confiança mínimos, que podem estar contidos entre os 0% e 100% (Goethals, 2010, Győrödi et al., 2004, Han and Kamber, 2006).

3.2.1 Algoritmos de mineração de regras

Existem diversos métodos para a mineração de regras de associação, mas a abordagem mais frequente passa primeiro pela aplicação de um processo de descoberta de itens frequentes, ou padrões frequentes, antes da obtenção das regras. Esse primeiro processo costuma ser o mais exaustivo, uma vez que envolve a geração de conjuntos de itens candidatos a frequentes, através de leituras à base de dados, e, ao mesmo tempo, o mais complicado, pois um mau desempenho do algoritmo nesta fase pode levar à geração de um conjunto de padrões candidatos muito grande, o que consequentemente pode levar à criação de regras de associação pouco satisfatórias ou inconclusivas.

Ao longo dos anos, vários autores têm testado e melhorado diversos algoritmos para descoberta de associações entre os dados, nos quais os mais conhecidos e aplicados são os algoritmos Apriori e FP-Growth. Estes dois algoritmos, e algumas das suas variações, têm sido utilizados em casos já referidos com semelhanças ao que estamos a tratar, nomeadamente na análise e descoberta de padrões em logs Web (Joshi et al., 1999, Iváncsy and Vajk, 2006)**.**

i. Apriori

O algoritmo Apriori foi adaptado do algoritmo AIS por Agrawal et al., que otimizou o processo de geração de conjuntos de itens candidatos (Agrawal and Srikant, 1994). Este algoritmo segue a abordagem indicada anteriormente, determinando primeiramente o conjunto de itens frequentes. Na figura 2 podemos ver como este algoritmo atua. Inicialmente é feita a contagem das ocorrências dos itens através de uma procura *breadth-first* pelo conjunto de dados, de forma a gerar um conjunto de itens frequentes de tamanho k. Seguidamente é utilizada a função *apriori-gen* que recebe como argumento todos os (k-1)-*itemsets* e devolve um superconjunto contendo todos os k-*itemsets*, designado como conjunto de itens candidatos (Agrawal and Srikant, 1994). Seguidamente, para cada transação na base de dados, é determinado o suporte de cada conjunto de itens candidato, ou seja, o seu número de ocorrências. Posteriormente é verificado se cada um dos candidatos satisfaz um valor de suporte mínimo estabelecido (Győrödi et al., 2004).

```
L1 = {Large l-itemsets};
For (k=2 ; Lk-1≠∅ ; k++) do begin
  Ck=apriori-gen(Lk-1); //New candidates
  Forall transactions t ∈ D do begin
        Ct=subset(Ck, t);
        // Candidates contained in t
        Forall candidates c ∈ Ct do
              c.count++;
  end
  Lk = {c ∈ Ct | c.count ≥ minsup}
end
Answer = ∪ k Lk;
```

Figura 2 - Algoritmo Apriori (Győrödi et al., 2004)

Vantagens e limitações

O algoritmo Apriori tem sido um dos mais usados, não só devido aos bons resultados na geração de itens frequentes, mas também à sua simplicidade. A sua fácil implementação tem levado vários autores a criar a suas próprias versões adaptadas e otimizadas, baseadas no algoritmo Apriori.

A utilização de uma procura *breadth-first* permite a contagem eficiente dos itens frequentes, e a propriedade Apriori é útil para encontrar todos os conjuntos de itens frequentes. No entanto, esta propriedade que para alguns casos pode ser bastante útil, para outros casos pode ser problemática, uma vez que pode levar à geração de vários conjuntos de candidatos (Kumar and Rukmani, 2010).

Uma outra desvantagem deste algoritmo é o seu tempo de execução. Quando lidamos com transações muito grandes, gerar *n* conjuntos de itens para cada transação e testá-los como frequentes requer várias leituras à base de dados, assim como bastante tempo. Por outro lado, a possibilidade de paralelização do algoritmo pode contornar em parte este problema (Goethals, 2010, Kumar and Rukmani, 2010).

ii. FP-Growth

Outro algoritmo bastante utilizado para mineração de regras de associação é o FP-Growth, desenvolvido por Han et al. (2000). A principal característica deste algoritmo é o facto de utilizar uma estrutura de dados compacta, denominada *frequent pattern tree* (FP-tree), no processo de geração de itens candidatos. Desta forma o algoritmo passa por dois principais processos: a

geração da FP-tree e, posteriormente, a aplicação do FP-Growth para a obtenção das regras. O primeiro processo consiste essencialmente em armazenar todas as transações da base de dados numa única estrutura. Desta forma cada nó da árvore corresponde a um item que está associado a uma lista, que contem todas as transações na qual este ocorre, e a um contador do seu número de ocorrências (Győrödi et al., 2004).

```
Procedure FP-growth(Tree, A)
{
    If Tree contains a single path P
      then for each combination (denoted as B) of the nodes in the
      path P do
           generate pattern B ∪ A with support=minimum support of
           nodes in B
      else for each ai in the header of the Tree do
      {
           generate pattern B=ai∪A support=ai.support;
           construct B's conditional pattern base and B's conditional
           FP-tree
           TreeB;
           If TreeB≠∅
             Then call FP-growth(TreeB, B)
      }
}
```

Figura 3 - Algoritmo FP-Growth (Győrödi et al., 2004)

O segundo processo está apresentado na Figura 3. Nesta fase o algoritmo FP-Growth recebe uma FP-tree e devolve um conjunto de padrões frequentes, através da realização de uma procura *bottom-up* pela estrutura. Se a árvore recebida apenas conter um caminho, o algoritmo vai gerar padrões através da combinação dos nós nesse caminho. Se a árvore conter mais do que um caminho é gerado um padrão condicional e a sua respetiva FP-tree para cada item. Um padrão condicional consiste num conjunto de prefixos dos caminhos desse item. O algoritmo é executado recursivamente sobre as novas FP-tree geradas (Han et al., 2000).

Vantagens e limitações

A principal característica e, consequentemente, vantagem deste algoritmo é a utilização de uma estrutura de dados compacta, a FP-tree, na qual é armazenada apenas a informação considerada relevante. Esta estrutura permite não só a redução do número total de itens candidatos, mas também do número de leituras repetidas à base de dados. Desta forma, o algoritmo revela-se bastante eficiente tanto para pequenas bases de dados, como para bases de dados com elevado número de transações (Győrödi et al., 2004, Kumar and Rukmani, 2010). A única desvantagem a apontar neste algoritmo são algumas falhas apontadas no processo de geração de bons candidatos (Kumar and Rukmani, 2010).

Capítulo 4

O Caso de Estudo

4.1 Contextualização

Desde os meados dos anos 90 que os sistemas de *data warehousing* se têm tornado em ferramentas bastante valiosas nos processos de tomada de decisão das organizações. O desenvolvimento destes sistemas e a aposta na exploração e análise de informação orientada ao assunto tem tido um grande impacto na forma como as organizações delineiam os seus negócios, e definem as estratégias que melhor se adaptam a áreas específicas (Wixom and Watson, 2001). Apesar destes sistemas terem vindo a trazer um enorme conjunto de vantagens aos analistas, é necessário um conjunto de pontos a ter em consideração quando se trata do seu desenvolvimento. O sistema deve, para além fornecer a informação consistente, completa e precisa ao utilizador, ser flexível e eficiente de forma a permitir uma adaptação e resposta rápida às mudanças nas necessidades dos analisas e intervenientes nos processos de tomada de decisão (Wixom and Watson, 2001).

Como já foi referido, diversas áreas, como as telecomunicações, o retalho, a medicina, entre outros, têm optado por este tipo de sistemas de forma a melhorarem toda a estrutura de suporte aos seus negócios, e os fatores mencionados acima são tidos bastante em conta, uma vez que

envolvem custos para essas mesmas organizações. Desta forma, é necessário ponderar e chegar a uma acordo entre rapidez, memória utilizada e flexibilidade de adaptação às necessidades pontuais dos utilizadores.

A utilização de memória e custos de computação são dois pontos bastante importantes quando lidamos com este tipo de estruturas, uma vez que é essencial ter toda a informação relevante que permita responder fácil e rapidamente a todas as necessidades do utilizador, mas isso pode implicar grandes custos de memória. Proporcionalmente ligado a esses custos estão os custos de computação, pois quanto maior a quantidade de informação, maior serão os custos de computação da mesma.

Uma das questões diretamente ligadas com a utilização ineficiente de memória para armazenamento de cubos OLAP refere-se ao armazenamento de cubos esparsos. A fraca qualidade dos dados, nomeadamente itens cujos valores são nulos ou zero, ou mesmo falhas no desenho podem levar ao aumento do grau de esparsidade do cubo, provocando inevitavelmente problemas de computação, armazenamento ineficiente assim como decréscimo no próprio desempenho do sistema de suporte à decisão (Niemi et al., 2000).

No entanto, nem só esse caso tem uma implicação direta na ineficiência de armazenamento. Como já foi referido, a utilização dos cubos está diretamente ligada às necessidades dos utilizadores e às diferentes atividade ou áreas da organização. Ou seja, usualmente, um determinado utilizador ou cada área da organização utiliza uma parte específica do cubo. Isto pode ter grandes implicações no desempenho do sistema, dado que acabamos por ter uma enorme quantidade de informação materializada, em que grande parte não é utilizada nos processos de tomada de decisão, o que leva a um consumo desnecessário de recursos.

A definição de *Data Marts*, ou seja, particularizações do *Data Warehouse* geral, pode ser uma solução para a redução de custos quando as utilizações são orientadas a áreas específicas do negócio. Estas estruturas permitem otimizar parte dos aspetos mencionados anteriormente, uma vez que permitem reduzir o volume de dados e proporcionar vistas mais simples aos utilizadores, tornando consequentemente a análise mais rápida e eficiente (Moody and Kortink, 2000). Contudo, até mesmo utilizando estas estruturas mais específicas, deparamo-nos com os mesmos problemas mencionados acima. Isto deve-se ao facto de que dentro das próprias áreas de negócio existirem diversos tipos de utilizadores, variadas perspetivas de análise e diferentes necessidades de negócio que se podem alterar ao longo do tempo.

É nesta perspetiva que é importante a realização de um estudo mais específico sobre a exploração de estruturas multidimensionais, ou seja, um estudo das *queries* realizadas sobre essas mesmas estruturas e do comportamento dos agentes de decisão face à sua utilização. Desta forma, o nosso estudo partiu de um conjunto de *queries* MDX, resultado da exploração de uma estrutura de dados multidimensional, um pequeno *data mart*, com informação relativa a vendas.

4.2 O modelo de dados

A exploração de estruturas de dados multidimensionais é feita usualmente através de *queries* MDX. Este problema, em concreto, consistiu na análise de *queries* MDX realizadas sobre um cubo de dados específico. É comum registar em ficheiros de log do servidor OLAP todas as *queries* MDX realizadas sobre um cubo. Tal como acontece em servidores *Web*, quando um utilizador inicia a sua utilização é aberta uma sessão, e em cada sessão um utilizador realiza um conjunto de *queries* MDX. Toda essa informação, desde a identificação do utilizador, dados relativos às *queries* realizadas e detalhes de data e hora de cada *query* ficam armazenados nesse ficheiro de log (Giacometti et al., 2008). Tipicamente um ficheiro de log de um servidor OLAP apresenta a seguinte estrutura:

```
"2010-07-09 01:24:28. 611 [http-8080-2] DEBUG mondrian.mdx -3:
 select     {[Measures].[Unit     Sales],     [Measures].[Store     Cost],
[Measures].[Store  Sales]}  ON  COLUMNS,  Hierarchize(  Union(  Crossjoin
({[Promotion    Media].[ALL    Media]},    {[Product].[ALL    Products]}),
Crossjoin({[Promotion  Media].[ALL  Media]},  [Product].[ALL  Products].
Children) ON ROWS from [Sales] where [Time].[1997]
2010-07-09 01:24:29, 876 [http-8080-2] DEBUG Mondrian.mdx -3: exec: 1265
ms"
```

A primeira linha indica os dados relativos à data e hora em que a *query* mdx foi realizada pelo servidor analítico, a *query* executada é apresentada logo de seguida e a última linha especifica a data e hora a que a *query* foi respondida, assim como o tempo te execução em milissegundos. Neste caso específico, o ficheiro log utilizado registava as *queries* executadas sobre uma estrutura multidimensional cujo contexto era relativo a Vendas (*Sales*). O cubo em questão, composto por treze dimensões e seis medidas, permite armazenar informação relativa a lojas, produtos, clientes,

armazéns, entre outros. Desta forma, o resultado do nosso estudo relativamente às *queries* executadas sobre esse cubo refletirá um conjunto de informação relacionada com este contexto.

Um ficheiro de log OLAP é tipicamente um ficheiro de texto com inúmeros registos semelhantes aos apresentados acima. Extrair qualquer tipo de informação relevante de um conjunto tão vasto de dados deste tipo torna-se numa operação extremamente complexa e extenuante. Assim, todos estes dados, utilizados posteriormente para o estudo e análise de preferências, foram previamente extraídos do ficheiro de log e armazenados numa base de dados relacional, com o objetivo de manter toda essa informação estruturada e normalizada, e facilitar a sua posterior análise. Na Figura 4 podemos ver o esquema da base de dados utilizada.

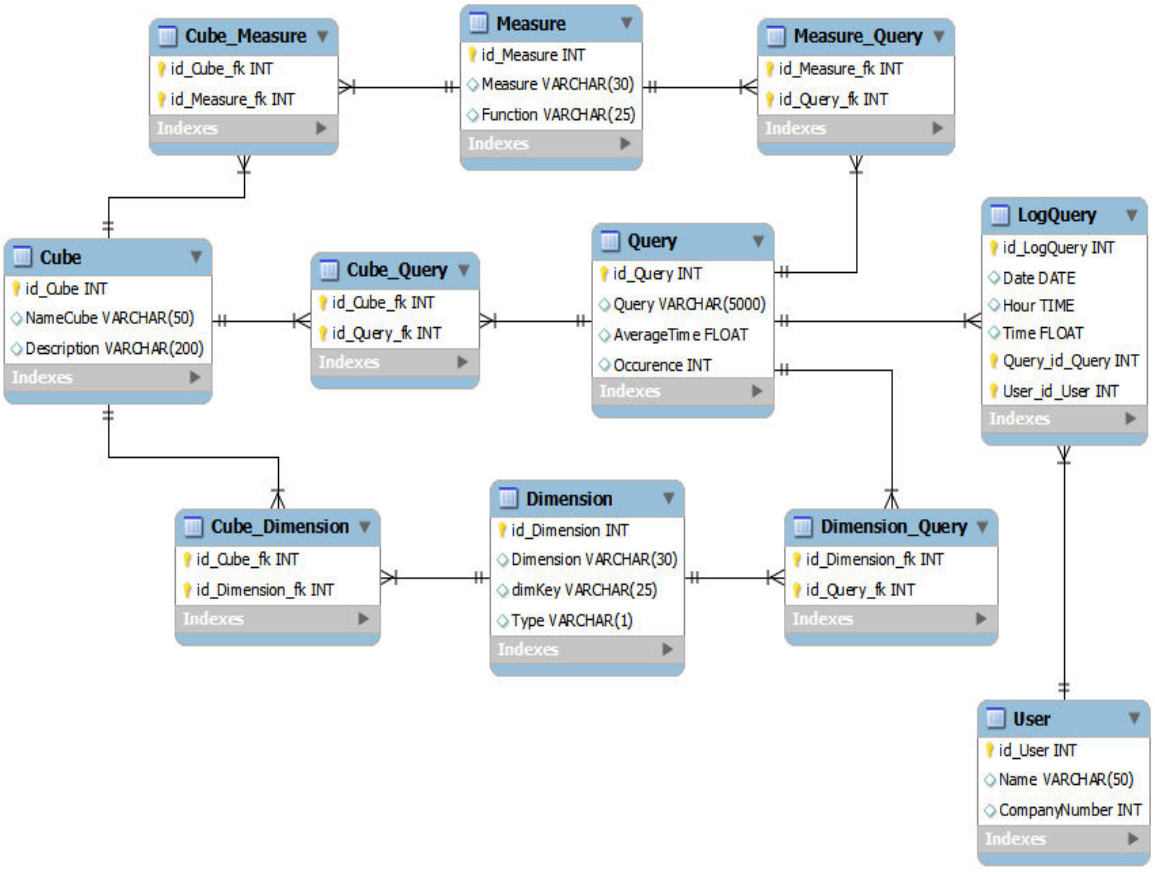


Figura 4 - Esquema da base de dados relacional

Este esquema é composto por 6 tabelas que representam as entidades envolvidas, nomeadamente: LogQuery, Query, User, Cube, Dimension e Measure; e outras 5 tabelas que representam os relacionamentos entre as entidades referidas: Cube_Measure, Cube_Dimension,

Cube_Query, Measure_Query e Dimension_Query. Cada uma das tabelas vai registar dados específicos do ficheiro de log. Na Tabela 1 está apresentado o papel que cada uma das tabelas do sistema representa.

Tabela 1 - Descrição das tabelas do modelo de dados inicial

Tabela	Papel
LogQuery	Registo da execução de uma query MDX
Query	Registo da query MDX executada pelo sistema analítico
User	Registo da informação relativa aos utilizadores que lançaram as queries
Cube	Descrição dos cubos sobre os quais as queries são executadas
Dimension	Registo das dimensões utilizadas para responder a uma determinada query
Measure	Registo das medidas utilizadas para responder a uma determinada query

Com esta base de trabalho, todo o processo de seleção, preparação e análise dos dados relevantes para este estudo tornou-se bastante mais simples.

4.3 Análise exploratória

A adesão a *data warehouses* por parte das organizações nestes últimos tempos teve grande parte a ver com a crescente necessidade destas entidades chegarem aos seus clientes e satisfazerem as suas necessidades em termos de informação para processos de tomada de decisão. Isto levou à reestruturação de departamentos de venda e à implementação de novas técnicas de análise de dados baseadas em cubos. Desta forma, penso que o nosso caso de estudo não poderia ilustrar melhor a tendência atual.

Como já referido, a análise partiu da exploração de uma estrutura de dados multidimensional, um pequeno *data mart*, com informação relativa a vendas. Relembrando, o cubo de dados "*Sales*" apresenta uma estrutura constituída por 6 medidas e 13 dimensões.

Dimensões:

- Store
- Store Size in SQFT
- Store Type
- Time
- Product
- Warehouse
- Promotion Media
- Promotions
- Customers
- Education Level
- Gender
- Marital Status
- Yearly Income

Medidas:

- Unit Sales
- Store Cost
- Store Sales
- Sales Count
- Customer Count
- Promotion Sales

O conjunto de dados iniciais deste estudo consistiu num ficheiro de log, sobre o qual foi extraído um total de 57689 *queries* realizadas entre 1 de Janeiro de 2009 e 7 de Março de 2010, por diferentes utilizadores, sobre a estrutura acima referida. Durante o ano de 2009 o número de *queries* realizadas diariamente manteve-se constante, com uma média acima de 150 *queries*/dia. No entanto, nos casos registados em 2010 observou-se uma descida significativa, com uma média de 45 *queries*/dia. Em termos gerais, e de forma mais concreta, foram realizadas, em média, 156 *queries* por dia, perfazendo uma média mensal de 4121 *queries*. Na Tabela 2 podemos observar a grande discrepância entre 2009 e 2010, devido essencialmente ao facto de que em 2010 ter sido registado um total de apenas 178 *queries*, efetuadas nos meses de Fevereiro e Março.

Tabela 2 - Média total de *queries* realizadas por mês

Ano	AVG (Queries/Mês)
2009/2010	4121
2009	4793
2010	89

Após a devida seleção e extração dos dados para a base de dados relacional, foi possível identificar ao todo um total de 157 *queries* distintas realizadas sobre o nosso cubo. Apesar de distintas, é possível identificar grupos de *queries* que utilizam o mesmo conjunto de dimensões e medidas. No entanto, dada a enorme quantidade de dados e a complexidade das próprias *queries* MDX, torna-se complicado extrair informação útil e precisa sem o auxílio de ferramentas analíticas. Focando-nos agora nos perfis de utilização e respetivos intervenientes, foi possível identificar três utilizadores distintos, cuja média de *queries* realizadas por mês não varia muito entre si.

Tabela 3 - Média mensal de *queries* por utilizador

	AVG (Queries/Mês)		
Utilizador	**2009/10**	**2009**	**2010**
[http-8080-1]	1478	1599	49
[http-8080-2]	1510	1594	70
[http-8080-3]	1430	1593	32

É de salientar que durante o ano de 2009, ambos os utilizadores mantiveram uma média de *queries* constante durante todos os meses. No entanto, no ano de 2010, para além da redução significativa da carga de exploração, em Janeiro não se registou qualquer *query,* o que tornou possível observar a discrepância entre os diversos utilizadores nos restantes meses.

Tabela 4 - Total de *queries* realizadas por cada utilizador no ano de 2010

	Total Queries		
Utilizador	**Janeiro**	**Fevereiro**	**Março**
[http-8080-1]	0	66	10
[http-8080-2]	0	70	0
[http-8080-3]	0	0	32

Apesar de, ao longo de 2009, os diversos utilizadores terem mantido uma carga de utilização bastante semelhante, o tipo de exploração efetuado por cada um é claramente variável, como é possível verificar na Tabela 5. O perfil de utilização de cada utilizador/analista pode ser bastante diversificado, uma vez que depende de variáveis como a área de negócio, o processo em que estão inseridos, as suas necessidades específicas e, até as próprias características do analista.

É possível verificar a diversidade de análise ao observar as *queries* realizadas. Para responder a cada *query* foram utilizadas, como sabemos, algumas dimensões e medidas do cubo, e com base nisso é possível identificar para cada utilizador quais as partes mais acedidas das estruturas que costumam consultar. Também neste caso, devido à inúmera quantidade de *queries* que cada utilizador efetuou e à quantidade de dimensões e medidas utilizadas, também a utilização de ferramentas analíticas se torna indispensável. No entanto, já na tabela apresentada é possível verificar que o número de dimensões e medidas utilizadas para responder às *queries* varia consoante o utilizador, indicando à partida que as próprias *queries* variam de utilizador para utilizador, bem como as partes do cubo utilizadas.

Tabela 5 - Características numéricas das *queries* realizadas por cada utilizador

	Dimensões			**Medidas**		
Utilizador	**AVG**	**Min**	**Max**	**AVG**	**Min**	**Max**
[http-8080-1]	5	3	8	4	1	6
[http-8080-2]	3	2	3	5	3	6
[http-8080-3]	4	3	5	4	3	6

Dada a natureza dos dados extraídos, a aplicação de técnicas de mineração de dados, como as regras de associação, cujos estudos demonstraram ser ferramentas bastante úteis na análise de *clickstreams*/*web* logs, e na definição de padrões de utilização *Web*, pode revelar-se uma abordagem interessante no estudo de preferências OLAP.

4.4 Modelação dimensional

Com os dados estruturados e armazenados na base de dados apresentada acima tornou-se mais fácil identificar o que seria importante analisar e quais os dados relevantes para essa análise. Quando estamos a estudar preferências, nomeadamente preferências OLAP, existem diversos pontos onde nos devemos focar para além dos diretamente relacionados com a preferência em si. Por outras palavras, apesar das preferências OLAP serem indicadas através da análise de cada uma das *queries* que foram executadas, ou seja, da instrução MDX propriamente dita, existem outros dados relativos à utilização e exploração do cubo bastante importantes e relevantes neste estudo. Para estabelecer preferências de utilização de cubos, para além de extrair informação

relativa às dimensões e medidas utilizadas, é necessária realizar também uma análise temporal das *queries* executadas, assim como observar o comportamento de cada utilizador.

Assim, tornou-se relevante fazer a implementação de um pequeno *data warehouse* para acolhimento dos dados recolhidos e de forma a facilitar o processamento dos mesmos através de técnicas de mineração. O grão definido para a tabela de factos envolvida permite estabelecer preferências de utilização OLAP com base nas *queries* realizadas, uma vez que representa uma determinada ***query***, constituída por determinadas **dimensões** e determinadas **medidas**, realizada por um determinado **utilizador**, sobre um determinado **cubo** multidimensional de dados, numa determinada **data** e **hora**, inserida num determinado **período** do dia.

Tabela 6 – A matriz de decisão do caso de estudo

Dimensões	Data Mart: Queries MDX (tf_logquery)
Cubo (dim_cube)	X
Utilizador (dim_user)	X
Query (dim_query)	X
Dimensão (dim_dimension)	X
Medida (dim_measure)	X
Data (dim_date)	X
Período (dim_period)	X

A matriz de decisão apresentada na Tabela 6, levou à definição de uma estrutura em estrela, constituída por sete dimensões, uma tabela de factos e duas tabelas ponte. A tabela de factos em si é composta por diversos eixos de análise, representados por chaves estrangeiras para as tabelas de dimensão, e quatros medidas que permitem complementar a caracterização de cada facto, nomeadamente: o tempo de execução da *query*, o número de dimensões utilizadas, o número de medidas e o total de ocorrências. Uma vez que cada *query* pode utilizar mais do que uma dimensão ou medida, e que cada dimensão/medida pode ser utilizada em uma ou mais *queries*, foi necessário a criação de duas tabelas ponte, uma para as dimensões e outra para as medidas, de forma a definir este relacionamento (Figura 5).

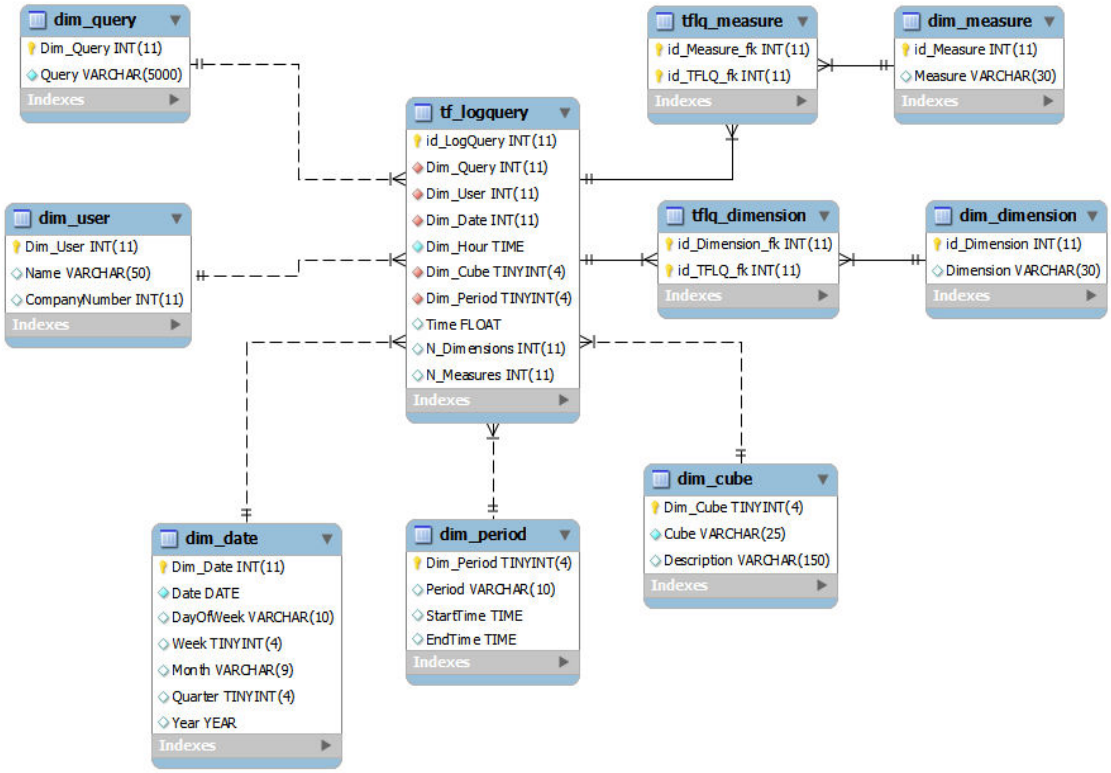


Figura 5 - Esquema do data mart implementado

Passemos agora à caracterização de cada um dos constituintes do nosso *data mart*:

- **Tabela de factos ("tf_logquery"):** tabela onde são guardados os factos que serão utilizados na análise. Cada facto representa uma determinada *query*, constituída por dimensões e medidas, realizada por um utilizador, sobre um determinado cubo multidimensional de dados, numa determinada data e hora, inserida num período específico do dia.

Tabela 7 - Caracterização da tabela de factos "tf_logquery"

Atributo	Descrição	Tipo de dados	Domínio
Dim_Query	Código de identificação da *query* MDX realizada sobre o cubo.	Integer	Número inteiro positivo até 11 algarismos
Dim_User	Código de identificação do utilizador que efetuou a *query*	Integer	Número inteiro positivo até 11 algarismos
Dim_Date	Data em que a *query* foi executada	Integer	Número inteiro positivo até 11 algarismos
Dim_Hour	Hora a que a *query* foi executada	Time	Ex: 19:16:37
Dim_Cube	Código de identificação do cubo sobre o qual foi executada a *query*	Integer	Número inteiro positivo até 4 algarismos
Dim_Period	Período do dia em que foi executada a *query*	Integer	Número inteiro positivo até 4 algarismos
Time	Tempo de execução da *query*	Float	Valor decimal positivo
N_Dimensions	Número de dimensões utilizadas na *query*	Integer	Número inteiro positivo até 11 algarismos
N_Measures	Número de medidas utilizadas na *query*	Integer	Número inteiro positivo até 11 algarismos

- **Dimensão Query ("dim_query"):** utilizada para caracterizar cada uma das *query*. Guarda informação relativa às *queries*, nomeadamente as suas instruções desenvolvida em código MDX.

Tabela 8 - Caracterização da dimensão "dim_query"

Atributo	Descrição	Tipo de dados	Domínio
Dim_Query	Código de identificação da *query* MDX	Integer	Número inteiro positivo até 11 algarismos
Query	*Query* MDX	Varchar	String com um limite de 5000 caracteres

- **Dimensão User ("dim_user"):** utilizada para caracterizar os utilizadores do cubo OLAP.

Tabela 9 - Caracterização da dimensão "dim_user"

Atributo	Descrição	Tipo de dados	Domínio
Dim_User	Código de identificação do utilizador	Integer	Número inteiro positivo até 11 algarismos
Name	Nome do utilizador	Varchar	String com um limite de 50 caracteres
CompanyNumber	Identificação da empresa do Utilizador	Integer	Número inteiro positivo até 11 algarismos

- **Dimensão Cube ("dim_cube"):** contem a informação relativa ao cubo de dados sobre o qual foram realizadas as *queries*.

Tabela 10 - Caracterização da dimensão "dim_cube"

Atributo	Descrição	Tipo de dados	Domínio
Dim_Cube	Código de identificação do cubo	Integer	Número inteiro positivo até 11 algarismos
Cube	Nome do cubo OLAP	Varchar	String com um limite de 25 caracteres
Description	Descrição do conteúdo do cubo	Varchar	String com um limite de 150 caracteres

- **Dimensão Date ("dim_date"):** esta dimensão vai permitir a análise temporal das *queries* realizadas sobre o cubo de dados, permitindo estudar preferências de utilização sobre diferentes escalas temporais (dia, mês, ano, etc.).

Tabela 11 - Caracterização da dimensão "dim_date"

Atributo	Descrição	Tipo de dados	Domínio
Dim_Date	Código de identificação da data	Integer	Número inteiro positivo até 11 algarismos
Date	Data em que foi realizada a *query*	Date	Ex:' 2009-01-12'
DayOfWeek	Dia da semana	Varchar	String com um limite de 10 caracteres Ex: 'Monday'
Week	Semana do ano em que ocorreu a consulta	Integer	Número inteiro positivo Ex:' 3'
Month	Mês em que foi realizada a *query*	Varchar	String com um limite de 9 caracteres Ex: 'January'
Quarter	Trimestre do ano	Integer	Número inteiro positivo Ex: '1'
Year	Ano da data em que foi realizada a *query*	Integer	Número inteiro positivo Ex: '2009'

- **Dimensão Period ("dim_period"):** esta dimensão é utilizada para caracterizar os períodos do dia (manhã, tarde, etc.). Como a exploração OLAP pode variar ao longo do dia, revelou-se importante para o estudo a criação deste eixo complementar de análise.

Tabela 12 - Caracterização da dimensão "dim_period"

Atributo	Descrição	Tipo de dados	Domínio
Dim_Period	Código de identificação do período	Integer	Número inteiro positivo até 11 algarismos
Period	Período do dia	Varchar	String com um limite de 10 caracteres Ex: 'Morning'
StartTime	Hora de início do período	Time	Ex: '08:00:00'
EndTime	Hora de fim do período	Time	Ex:' 12:59:59'

- **Dimensão Dimension ("dim_dimension"):** utilizada para caracterizar as dimensões utilizadas na *query* MDX.

Tabela 13 - Caracterização da dimensão "dim_dimension"

Atributo	Descrição	Tipo de dados	Domínio
id_Dimension	Código de identificação da dimensão	Integer	Número inteiro positivo até 11 algarismos
Dimension	Nome da dimensão	Varchar	String com um limite de 30 caracteres

- **Dimensão Measure ("dim_measure"):** utilizada para caracterizar as medidas utilizadas na *query* MDX.

Tabela 14 - Caracterização da dimensão "dim_measure"

Atributo	Descrição	Tipo de dados	Domínio
id_Measure	Código de identificação da medida	Integer	Número inteiro positivo até 11 algarismos
Measure	Nome da medida	Varchar	String com um limite de 30 caracteres

- **Tabelas ponte Query-Dimension ("tflq_dimension") e Query-Measure ("tflq_measure"):** criadas de forma a representar o relacionamento n:m entre um facto e as respetivas dimensões e medidas utilizadas na *query* em causa. Cada uma destas tabelas é composta pela chave da tabela de factos que identifica a *query* realizada e a chave da respetiva dimensão.

Após criado o *data mart* foi necessário proceder ao seu povoamento. Uma vez que os dados já se encontravam devidamente selecionados e tratados no sistema operacional, apenas foi necessário adaptá-los e importá-los para a nova estrutura, não sendo necessário qualquer processo de tratamento complementar. A primeira fase do povoamento consistiu na importação dos dados do

sistema operacional para uma área de retenção, onde estes foram transformados de forma a adaptarem-se ao nosso modelo multidimensional. A segunda fase passou pelo carregamento desses dados para a nossa estrutura, iniciando-se o povoamento pelas tabelas de dimensão, seguindo-se a tabela de factos e só depois as tabelas ponte.

Uma vez que grande parte dos registos que tínhamos eram dados históricos, a dimensão Date ("dim_date") foi definida com datas compreendidas entre '2008-01-01' e '2018-01-01'. Para o povoamento da dimensão Period ("dim_period") optou-se por definir apenas três registos, correspondendo aos três períodos do dia: manhã (08:00:00-12:59:59), tarde (13:00:00-19:59:59) e noite (20:00:00-07:59:59).

Todo este processo, desde a seleção e extração dos dados para uma base de dados relacional, a criação do *data warehouse* e a posterior adaptação e importação dos dados para o mesmo foi realizado através da ferramenta MySql Workbench 5.2 (OracleCorporation, 2012). Uma vez devidamente povoado o *data warehouse*, ficámos com a nossa base de estudo pronta para aplicação de técnicas de mineração e extração de preferências.

4.5 Aplicação de mineração para extração de preferências

Quando analisamos os dados deste problema em concreto, verificamos que é possível definir as preferências como um conjunto de dados relacionados que são pedidos várias vezes, ao longo do tempo, ou seja, que são frequentes. Existem diversos algoritmos de mineração de dados que permitem encontrar um conjunto de itens frequentes entre os dados e, com isto, encontrar interessantes associações e relacionamentos entre estes. Deste ponto de vista, ao encontrarmos um conjunto de itens frequentes e suas regras de associação correspondentes, conseguimos obter um conjunto de preferências.

4.5.1 Abordagem de OLAP Mining

Como já foi referido, existem diversas abordagens possíveis para a extração de conhecimento de estruturas multidimensionais, desde a utilização de sistemas OLAM, à adaptação das técnicas

tradicionais de mineração de dados, ou mesmo dos próprios modelos de dados. Neste caso, devido a uma maior diversidade de ferramentas *open-source* de mineração de dados que lidam com modelos unidimensionais do que ferramentas de mineração de estruturas multidimensionais, optou-se pela última abordagem. Desta forma, os dados oriundos da estrutura multidimensional passaram por uma fase de pré-processamento, na qual foram selecionados, extraídos e transformados de forma a permitir a aplicação de algoritmos de mineração de dados tradicionais.

4.5.2 Ferramentas utilizadas

A abordagem escolhida abriu-nos um leque bastante diversificado de escolhas no que diz respeito a ferramentas *open-source* para mineração de dados. Uma vez que se optou pela extração e transformação dos dados, a escolha da ferramenta a utilizar teve em conta não só critérios relativos à eficiência na extração de conhecimento, mas também na exploração e análise dos dados. Tendo por base estes critérios, optámos por utilizar a ferramenta RapidMiner, versão 5.1, devido, em grande parte, à extraordinária gama de operadores que apresenta. O RapidMiner contém desde operadores de importação de dados, a operadores de modelação, avaliação de modelos e transformação de dados. Este último grupo de operadores revelou-se bastante útil pois permitiu a extração e transformação dos dados oriundos do *data warehouse* diretamente através da ferramenta, sem ser necessário efetuar alterações no modelo de dados.

4.5.3 Análise e escolha do algoritmo

A utilização de algoritmos de mineração de regras de associação pode tornar-se bastante eficaz na extração e definição de preferências. A escolha de um algoritmo adequado ao caso em estudo pode fazer toda a diferença nos resultados que serão obtidos. Desta forma, procedeu-se à análise e comparação de dois algoritmos de mineração de regras de associação, o Apriori e o FP-Growth.

De forma a verificar qual o algoritmo que melhor se adaptava ao caso em estudo, para além de ponderar apenas as características, vantagens e limitações de cada um, foi feita uma análise de desempenho aos dois algoritmos, executando cada um deles sobre uma amostra com 57689 transações do conjunto de dados em análise. Para efetuar essa comparação foi utilizada a ferramenta *Rapidminer*, numa plataforma com as seguintes especificações: processador Intel Core

i3-370M Dual Core, 4GB de memória RAM, Windows 7 de 64bits. Cada um dos algoritmos foi executado sobre o mesmo conjunto de dados, variando, em cada teste, o valor de suporte mínimo. O principal objetivo era analisar o desempenho de cada um dos algoritmos na geração dos itens frequentes e das regras de associação.

Primeiramente, foi realizada uma análise de escalabilidade a cada um, verificando-se a forma como estes se comportavam em termos de tempo de execução para valores de suporte mínimo compreendidos entre 0.1 e 1. Na Figura 6 podemos ver que o tempo de execução do algoritmo FP-Growth diminui à medida que o valor de suporte mínimo aumenta. Já o Apriori tende a ter um tempo de execução estável. No entanto, apresenta tempos de execução piores que o FP-Growth para valores de suporte superiores a 0.5.

Seguidamente foi analisada a variação de itens frequentes e regras de associação gerados por cada um dos algoritmos para os diversos valores de suporte mínimo. Para este caso específico verificou-se que, para qualquer valor de suporte, o FP-Growth gera mais itens frequentes e mais regras que o Apriori, como é possível ver na Figura 7 e Figura 8.

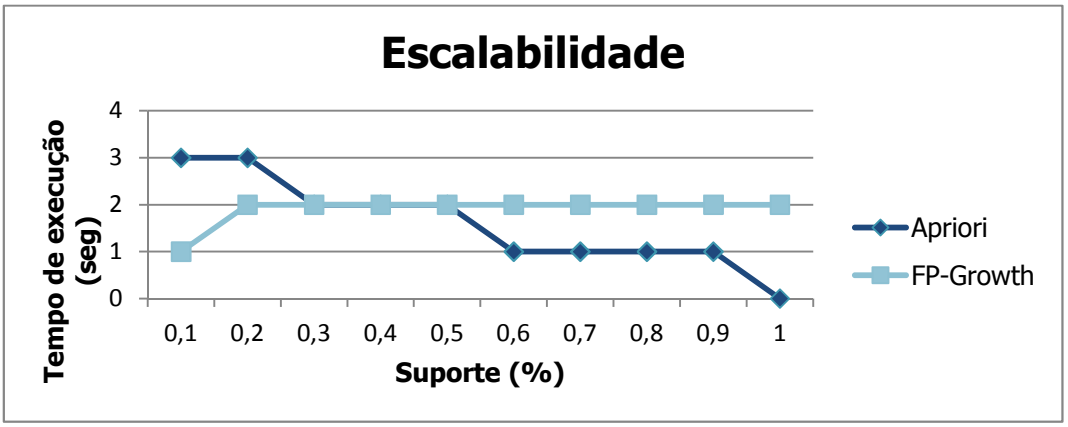


Figura 6 - Análise de escalabilidade (Apriori vs. FP-Growth)

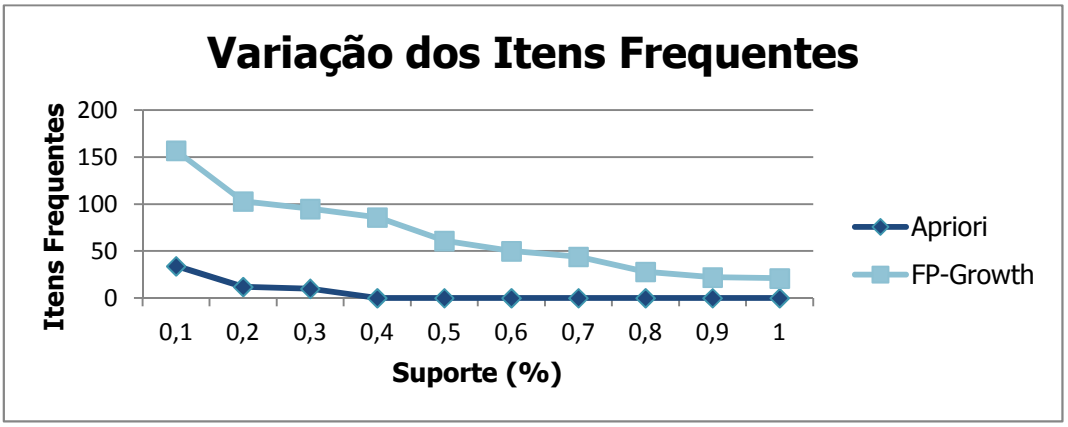


Figura 7 - Variação dos itens frequentes (Apriori vs. FP-Growth)

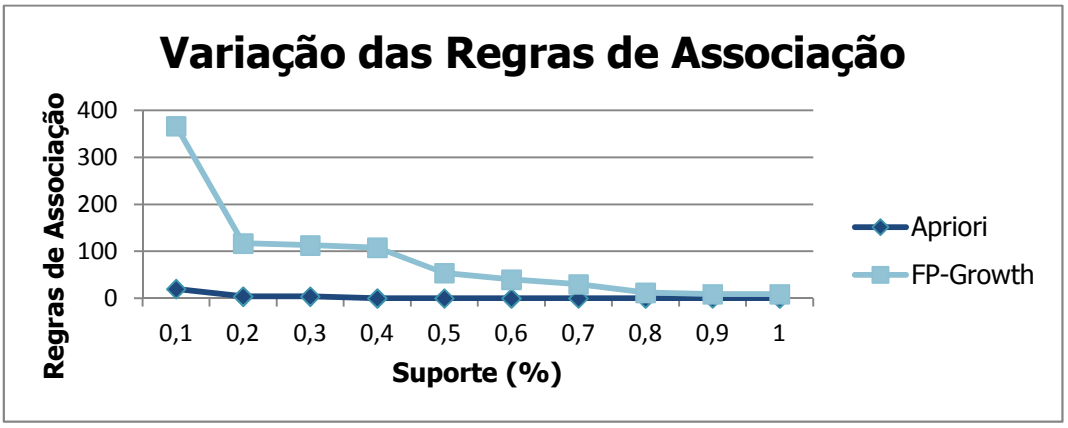


Figura 8 - Variação das Regras de Associação (Apriori vs. FP-Growth)

O tempo de execução levado por cada algoritmo e o número de itens/regras geradas não nos conseguem dar uma perspetiva clara do seu desempenho nos dois processos. Desta forma, foi estimado o tempo médio que cada um dos algoritmos levava por cada item frequente gerado, assim como por cada regra gerada. A Figura 9 revela que quanto menor for o valor de suporte, menor é o tempo de execução por item frequente para o FP-Growth. Devido à falta de resultados do algoritmo Apriori, para valores de suporte superiores a 0.3, torna-se difícil fazer uma comparação. No entanto, para os valores de 0.1 a 0.3 conseguimos verificar que o seu tempo de

execução é mais elevado que no algoritmo FP-Growth. A Figura 10 apresenta um comportamento semelhante ao da Figura 9.

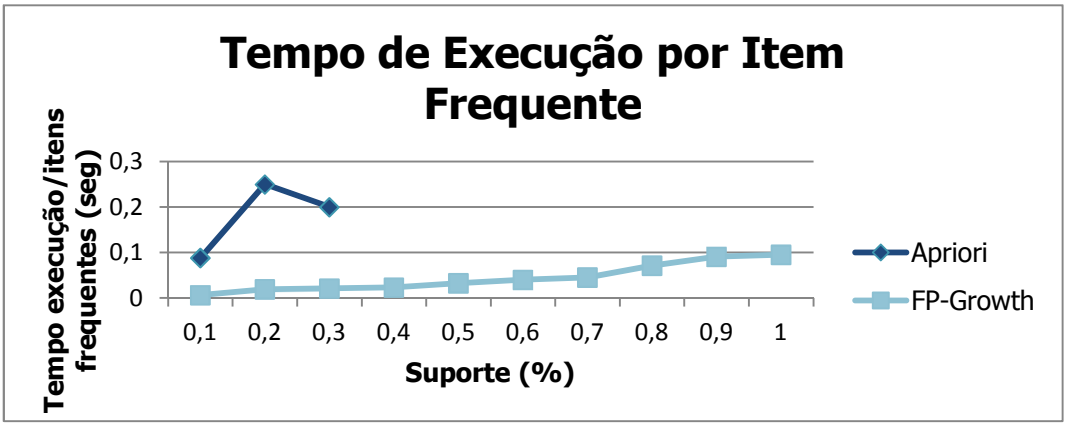


Figura 9 - Tempo de Execução por Item Frequente (Apriori vs. FP-Growth)

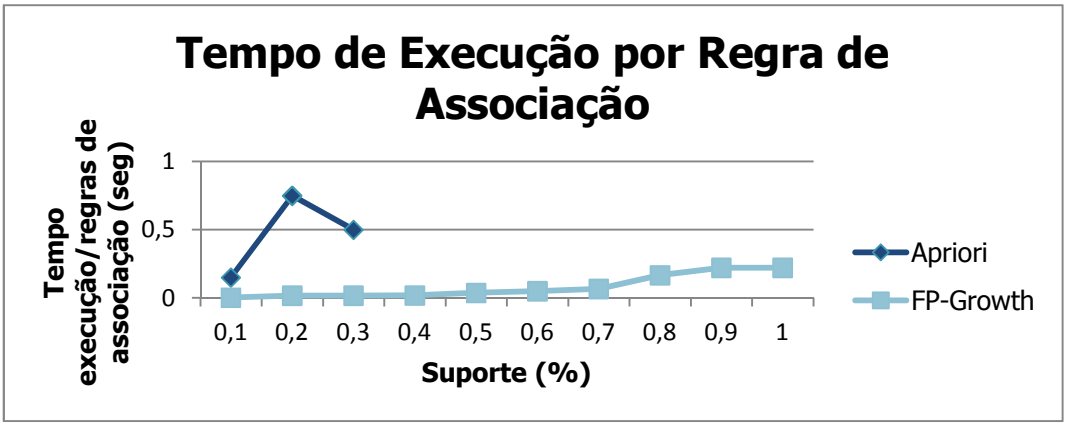


Figura 10 - Tempo de Execução por Regra de Associação (Apriori vs. FP-Growth)

Ao analisar os gráficos das figuras anteriores, e ponderando sobre os seus respetivos dados, é possível verificar que o algoritmo Apriori não se adequa ao caso em estudo, pois, para além de apresentar tempos de execução piores que o FP-Growth, para valores razoáveis de suporte mínimo, gerou poucas ou nenhumas regras de associação e itens frequentes, contrariando um pouco o que seria esperado teoricamente. Desta forma, tendo em conta o funcionamento de cada algoritmo, suas vantagens e limitações, *performance* e os restantes resultados obtidos no estudo comparativo, optou-se pela utilização do FP-Growth para esta fase da definição de preferências.

4.5.4 O algoritmo FP-Growth

Já foi apresentado resumidamente o funcionamento do algoritmo FP-Growth, no entanto, para um melhor entendimento, é explicado, a seguir, mais pormenorizadamente o comportamento deste algoritmo utilizando um exemplo relacionado com o tema em estudo, baseado no exemplo apresentado em (Tan et al., 2005).

Consideremos que temos um conjunto de transações, apresentadas na Tabela 15, na qual cada transação se refere a uma *query* realizada que está identificada por TID. Cada transação é composta por um conjunto de itens, neste caso, identificados como dimensões *di*, com i $\in$ [1..n].

Tabela 15 - Conjunto de transações

TID	Dimensões
1	{d1,d2}
2	{d2,d3,d4}
3	{d1,d3,d4,d5}
4	{d1,d4,d5}
5	{d1,d2,d3}
6	{d1,d2,d3,d4}
7	{d1}
8	{d1,d2,d3}
9	{d1,d2,d4}
10	{d2,d3,d5}

Com a aplicação do algoritmo pretende-se obter as dimensões mais frequentes no conjunto de dados e as associações ou relacionamentos existentes entre estas, por outras palavras, queremos obter, para um determinado conjunto de dimensões, qual ou quais as dimensões que são devolvidas juntamente com este.

Como já foi referido, a primeira fase do algoritmo passa por armazenar todas as transações numa estrutura de dados denominada *frequent pattern tree* (FP-tree). Desta forma, o algoritmo começa por ler a tabela de transações apresentada acima e vai construindo a FP-Tree, como podemos observar através da Figura 11.

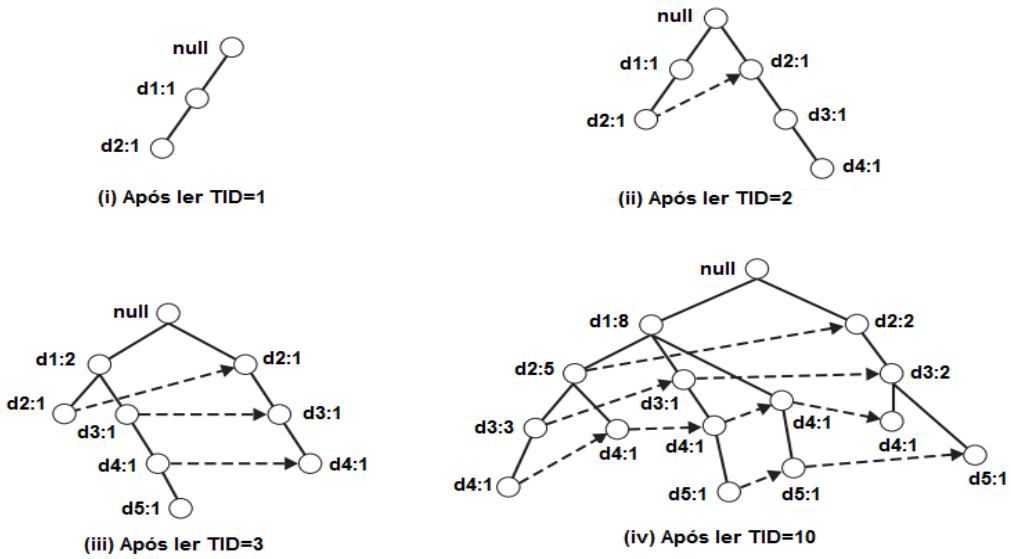


Figura 11 - Processo de construção da FP-Tree

À medida que as transações são lidas, o número de ocorrências de cada item que a compõe é incrementado. Por exemplo, após serem lidas as dez transações podemos observar que o item mais frequente é a dimensão 1, pois ocorre em oito das dez transações.

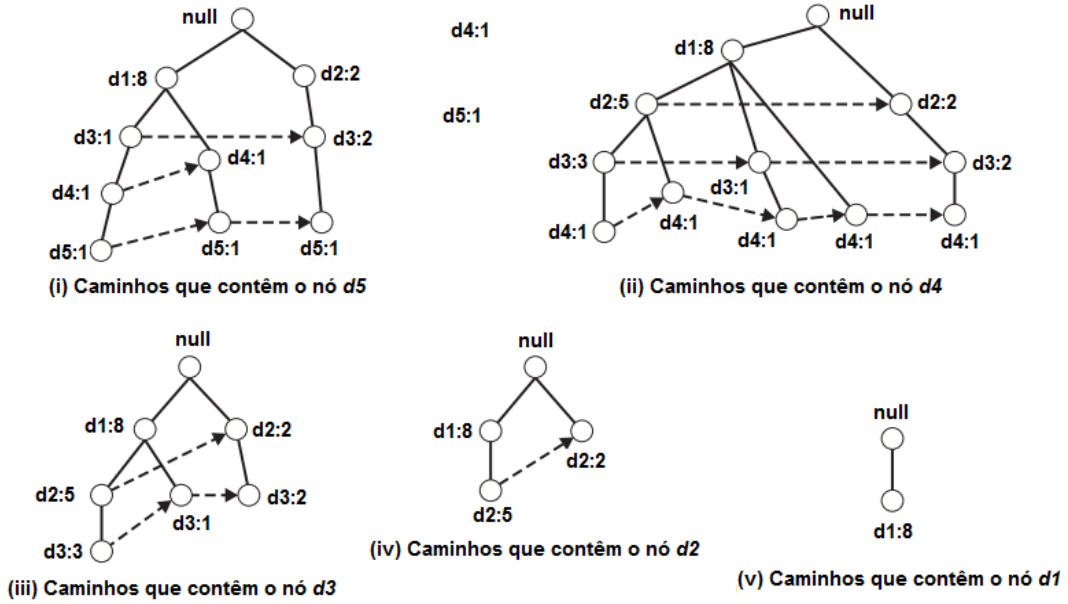


Figura 12 - Decomposição da FP-Tree por cada nó existente

Após a FP-Tree ser construída, é efetuada uma leitura sobre a estrutura numa abordagem *bottom-up*, criando para cada item uma FP-Tree com os caminhos em que este aparece (Figura 12).

Seguidamente, para cada nó considerado frequente, ou seja, que satisfaça um suporte mínimo, são identificados os seus prefixos e criadas as árvores condicionais correspondentes. Uma árvore condicional de um determinado prefixo é uma FP-Tree que permite encontrar conjuntos de itens frequentes que terminam com esse prefixo. Após essa decomposição, o algoritmo é aplicado recursivamente sobre cada árvore condicional. A Figura 13 mostra como são geradas as árvores condicionais que vão permitir determinar os conjuntos de itens frequentes. Consideremos o nó d5 cuja árvore de caminhos desse nó é apresentada na Figura 13(i). Supondo que o valor de suporte mínimo é 2, verificamos que *d5* é frequente, pois a soma das suas ocorrências é igual a 3. Desta forma, é determinada a árvore condicional para *d5* e identificados os seus prefixos, ou seja, (*d4*,*d5*), (*d3*,*d5*), (*d2*,*d5*) e (*d1*,*d5*).

A FP-Tree de um determinado prefixo apresenta para cada nó tanto a ocorrência deste em transações que o incluem, como transações que não o incluem. Desta forma, ao gerar a árvore condicional para esse determinado prefixo, são consideradas apenas as transações onde esse prefixo ocorre, diminuindo consequentemente o suporte de cada nó. Por exemplo, o prefixo *d5* foi

considerado frequente, logo foi gerada a sua árvore condicional, como é possível ver na Figura 13(ii). Na mesma figura, em (i), é apresentada a árvore que inclui todos os caminhos que terminam em *d5*, no entanto apenas as seguintes transações incluem *d5*: {*d1*,*d3*,*d4*,*d5*}, {*d1*,*d4*,*d5*} e {*d2*,*d3*,*d5*}. As ocorrências dos itens pertencentes, por exemplo, à transação {*d2*,*d3*} estão incluídas na FP-Tree, no entanto, essa transação não inclui o item *d5*, logo é necessário diminuir o suporte dos nós inerentes a essa transação.

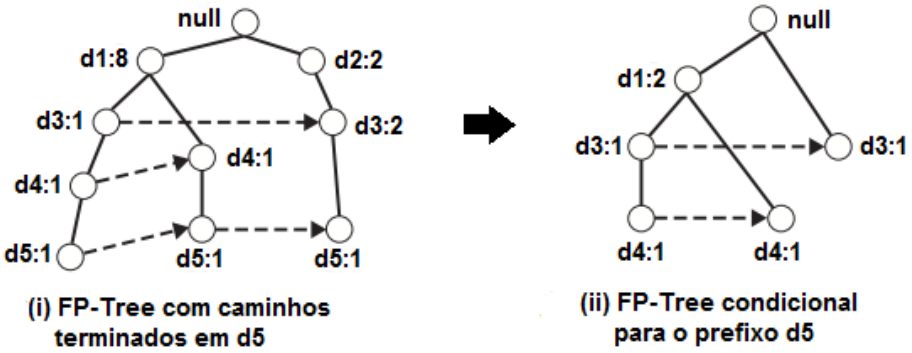


Figura 13- Geração de uma árvore condicional de suporte mínimo igual a 2

Após a diminuição das ocorrências, os nós que não respeitem o suporte mínimo são eliminados. Como pudemos observar, ao passar da FP-Tree em (i) para a FP-Tree em (ii), o nó *d2* foi eliminado, pois o seu suporte era apenas 1. Já o nó *d3* é mantido, pois encontra-se ligado a outro nó *d3*, perfazendo no total um suporte de valor 2 para caminhos terminados em (*d3*,*d5*), logo considerado frequente. A árvore condicional é gerada apenas para prefixos considerados frequentes. Assim, quando a árvore é gerada já não é necessário considerar os nós referentes a esse prefixo.

Depois, o processo é aplicado sobre a árvore apresentada na Figura 13(ii) para os prefixos de *d5,* acima apresentados, sendo consecutivamente aplicado, recursivamente, sobre cada árvore gerada, identificando os novos prefixos e as árvores condicionais correspondentes.

Ao aplicar o algoritmo ao caso em estudo, vamos obter algo semelhante ao resultado final deste exemplo, por outras palavras, vamos obter os conjuntos de dimensões que são frequentes para cada um dos sufixos, como é apresentado na seguinte tabela.

Tabela 16- Exemplo de conjuntos de itens frequentes obtidos pelo FP-Growth

Sufixo	Conjuntos frequentes
d5	{d5}z, {d4, d5}, { d1, d4, d5}, {d3, d5}, {d1,d5}
d4	{d4}, {d3, d4}, {d2,d3,d4}, {d1,d3,d4}, {d2,d4}, {d1,d2,d4}, {d1,d4}
d3	{d3}, {d2,d3}, {d1, d2, d3}, {d1, d3}
d2	{d2}, {d1,d2}
d1	{d1}

4.5.5 O modelo de mineração

Para a mineração de regras de associação, os dados em análise têm de passar por um conjunto de processos até à obtenção desses relacionamentos. Foi então definido um modelo que passa essencialmente por três fases: a identificação do que se pretende analisar, a seleção e pré-processamento desses dados e a aplicação do algoritmo de mineração.

A Figura 14 apresenta o processo criado para a aplicação do modelo. A primeira e segunda fase do modelo são definidas pelos quatro primeiros operadores, enquanto que o algoritmo FP-Growth, responsável pela geração de itens frequentes e definição de regras de associação, é aplicado pelos dois últimos operadores. Nas secções seguintes é apresentado em pormenor cada uma destas fases.

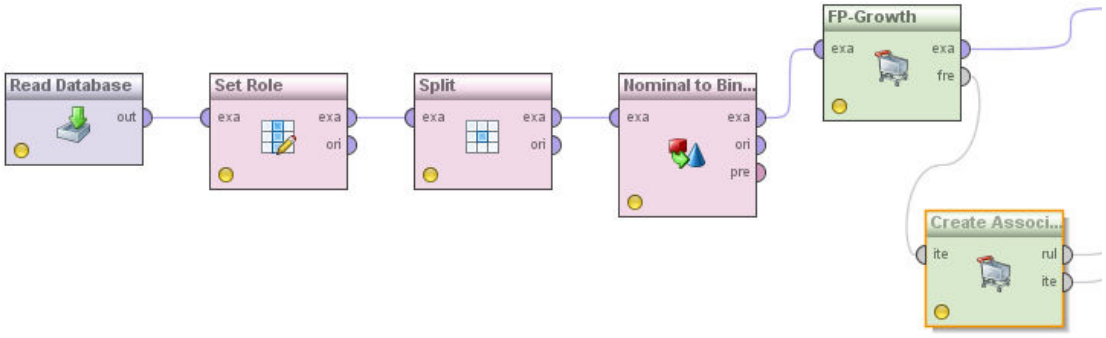


Figura 14 - Modelo de regras de associação no RapidMiner

i. Dados e objetivos a alcançar

O objetivo principal no processo de extração de preferências OLAP é identificar as partes do cubo que são pedidas simultaneamente e diversas vezes ao longo do tempo. Assim, foi necessário identificar quais os dados que seriam importantes para a análise.

Neste caso específico, quando nos referimos às partes do cubo que são pedidas, estamo-nos a referir aos dados que são utilizados para conseguir responder a uma determinada *query*, executada por um determinado utilizador. Precisámos, assim, de identificar os itens que são mais utilizados e os que são devolvidos em conjunto para responder às diversas *queries* executadas sobre o cubo de dados.

Para este tipo de estruturas existem dois tipos de dados que podem ser pedidos numa *query* MDX: medidas, ou seja, valores quantitativos provenientes da tabela de factos, e dados referentes aos eixos em análise, valores qualitativos provenientes das dimensões. No início, o que se pretendeu preferencialmente analisar foram as dimensões, devido ao seu impacto na definição dos cubos *iceberg* a materializar, complementando posteriormente a análise das preferências OLAP com o estudo de padrões sobre as medidas. Focámo-nos, pois, sobre três amostras distintas de dados: uma apenas referente às dimensões pedidas por todas as *queries* MDX realizadas sobre o cubo de dados, outra que continha informação apenas das medidas, e ainda uma outra que incorporava tanto as medidas como as dimensões que eram utilizadas para satisfazer essas mesmas *queries*.

ii. Pré-processamento

Para a aplicação do algoritmo FP-Growth foi necessária uma fase de pré-processamento na qual foi efetuada a extração e transformação dos dados para análise. Como pudemos observar na secção 4.5.4, inicialmente é necessário ter uma lista de transações, para as quais temos um conjunto de itens. Desta forma, uma vez que no *data warehouse* cada registo da tabela de factos corresponde a uma *query* executada num determinado momento, foi necessário que, para cada registo, agrupássemos os itens, neste caso, dimensões ou medidas que a *query* correspondente utilizou.

Esta primeira fase foi realizada diretamente no RapidMiner, através do operador "Read Database" que permite, além de definir uma nova conexão a uma base de dados, executar, entre outras coisas, comandos SQL. Assim, utilizando o comando GROUP_CONCAT() do MySQL foi possível agrupar as dimensões utilizadas por transação. No código XML que se apresenta de seguida

podemos observar os vários operadores utilizados nesta fase e o comando utilizado na amostra referente apenas às dimensões.

```
<operator activated="true" class="process" compatibility="5.1.006" expanded="true"
name="Process">
    <process expanded="true" height="540" width="1123">
    <operator activated="true" class="read_database" compatibility="5.1.006"
    expanded="true" height="60" name="Read Database" width="90" x="45" y="120">
        <parameter key="connection" value="DW_MDXQueries3"/>
        <parameter key="query" value="select lq.id_logquery,
        GROUP_CONCAT(d.id_dimension_fk ORDER BY 1 SEPARATOR &quot;,&quot;) as
        dimensions FROM tf_logquery as lq join tflq_dimension as d on
        id_logquery=id_tflq_fk GROUP BY lq.id_logquery"/>
        <enumeration key="parameters"/>
    </operator>
    <operator activated="true" class="set_role" compatibility="5.1.006" expanded="true"
    height="76" name="Set Role (3)" width="90" x="179" y="120">
        <parameter key="name" value="id_logquery"/>
        <parameter key="target_role" value="id"/>
        <list key="set_additional_roles"/>
    </operator>
    <operator activated="true" class="split" compatibility="5.1.006" expanded="true"
    height="76" name="Split" width="90" x="313" y="120">
        <parameter key="attribute" value="dimensions"/>
        <parameter key="split_mode" value="unordered_split"/>
    </operator>
    <operator activated="true" class="nominal_to_binominal" compatibility="5.1.006"
    expanded="true" height="94" name="Nominal to Binominal" width="90" x="420" y="210"/>
```

Na linha 8, onde é definido o comando SQL, podemos observar que é feito o agrupamento das dimensões pelo "id_logquery", identificador único de uma *query*, executada sobre um determinado cubo, por um utilizador, num determinado momento. Ainda nessa porção de código XML, é possível notar mais três operadores utilizados na fase de pré-processamento, o operador "Set Role", o "Split" e o "Nominal to Binominal". Foi necessário utilizar estes operadores para que fosse possível a aplicação posterior dos operadores de associação. Por sua vez, o "Set Role" foi utilizado para definir o "id_logquery" como atributo identificador da transação, e o "Split" juntamente com o "Nominal to Binominal" foram necessários para converter os dados para um formato binomial, requerido pelo operador do FP-Growth.

Na Tabela 17 é possível ver um exemplo do conjunto de dados, após toda a fase de pré-processamento. Para cada transação, identificada pelo "id_logquery", são indicadas as dimensões que são utilizadas pela *query* correspondente, através da atribuição do valor "*true*" ao atributo referente a essa dimensão, e as que não são utilizadas, através da atribuição do valor "*false*".

Tabela 17 - Exemplo do conjunto de dados após a fase de pré-processamento

id_logquery	dimensions_1	dimensions_10	dimensions_11	dimensions_12	dimensions_13
1	false	False	false	false	False
2	false	False	false	false	False
3	false	True	true	true	False
4	false	True	true	true	False
5	false	True	true	true	False
6	false	True	true	true	False
7	false	True	true	true	False
8	false	True	true	true	False
9	false	True	True	true	False
10	false	True	True	true	False

iii. Extração de itens frequentes e regras de associação

A terceira fase do modelo consistiu na extração dos itens frequentes e das regras de associação a partir do conjunto de dados obtido anteriormente. Nesta fase, foram utilizados dois operadores, o operador "FP-Growth" e o "Create Association Rules". O primeiro é responsável pela extração dos itens e o segundo extrai as regras a partir do *output* do primeiro. Para cada um dos operadores responsáveis pela aplicação do algoritmo FP-Growth foi necessário definir um valor de suporte mínimo e outro de confiança, respetivamente. Por vezes, para determinados tipos de análise, a escolha de um nível de suporte baixo pode revelar resultados bastante interessantes, como é o caso de padrões pouco usuais. No entanto, para este caso escolher um valor de suporte bastante baixo tornar-se-ia desvantajoso. Para além de nos retornar um conjunto enorme de resultados, a própria natureza dos dados ira provocar um nível bastante elevado de redundância, o que iria dificultar a análise. Por outro lado, os próprios requisitos da análise levaram à escolha de níveis de suporte e confiança que nos permitissem obter dados cujo nível de frequência justificasse a materialização seletiva.

Os parâmetros utilizados neste processo podem ser observados no excerto de código XML apresentado de seguida.

```
<operator activated="true" class="fp_growth" compatibility="5.1.006" expanded="true"
height="76" name="FP-Growth" width="90" x="514" y="120">
        <parameter key="min_support" value="0.1"/>
</operator>
<operator activated="true" class="create_association_rules" compatibility="5.1.006"
expanded="true" height="76" name="Create Association Rules" width="90" x="648" y="165">
        <parameter key="min_confidence" value="0.7"/>
        <parameter key="gain_theta" value="2.0"/>
        <parameter key="laplace_k" value="1.0"/>
</operator>
```

4.6 Análise dos resultados

Como foi referido no capítulo anterior, para este estudo utilizámos três amostras distintas: uma apenas referente às dimensões pedidas por todas as *queries* MDX realizadas sobre o cubo de dados, outra que continha informação apenas das medidas, e ainda uma terceira que incorporava tanto as medidas como as dimensões que eram utilizadas para satisfazer essas mesmas *queries*. Com base nessas amostras redirecionámos o nosso estudo em duas vertentes: uma análise mais geral, que consistiu em analisar toda a amostra e determinar quais as partes do cubo que foram mais utilizadas, e uma análise orientada a períodos de tempo específicos. Nesta secção irei apresentar os resultados da aplicação do algoritmo sobre a amostra relativa às dimensões.

4.6.1 Análise geral de itens frequentes

A amostra geral envolveu um total de 57689 *queries* realizadas sobre o cubo Sales ao longo de aproximadamente um ano. Após a aplicação do algoritmo foi possível obter conjuntos de itens frequentemente utilizados durante a exploração multidimensional. Como parâmetro inicial optou-se por um suporte mínimo de 10%, o que levou a um resultado de 65 conjuntos de itens frequentes, de tamanhos variáveis entre 1 e 5 itens. De entre o conjunto de dado obtidos destacamos os seguintes conjuntos de itens frequentes, que apresentaram um suporte superior a 30%. Apesar de ser interessante para análise identificar os diversos conjuntos de itens e a sua frequência (definida como o suporte) sobre o conjunto total de transações, para o nosso caso específico, o objetivo é

identificar conjuntos de itens com uma frequência elevada, pois são esses que nos ajudarão a definir perfis de materialização apropriados. A definição de perfis com base em conjuntos de itens com suporte elevado vai permitir a redução de memória, garantindo, no entanto, um impacto reduzido sobre a performance do sistema.

Tabela 18 – Itens frequentes com suporte > 30% (Análise geral)

Size	Support	Item1	Item2	Item3
1	0.904	Time		
1	0.473	Product		
2	0.473	Time	Product	
1	0.448	Store		
1	0.444	Gender		
1	0.433	Education Level		
1	0.381	Marital Status		
2	0.372	Time	Store	
2	0.349	Time	Gender	
1	0.348	Customers		
2	0.347	Gender	Education Level	
2	0.344	Gender	Marital Status	
2	0.337	Time	Education Level	
2	0.334	Time	Customers	
2	0.323	Education Level	Marital Status	
3	0.323	Gender	Education Level	Marital Status

Ao observar os resultados obtidos conseguimos, para além de identificar conjuntos de itens frequentes, definir tendências de exploração analítica. Com base na frequência de cada item sobre o total de transações realizadas, verificamos que a análise dos utilizadores recai sobre os produtos e as características dos seus consumidores, nomeadamente o género, o estado civil e o nível de escolaridade.

A aplicação da segunda parte do algoritmo FP-Growth permitiu uma análise mais específica destas tendências, através das regras de associação obtidas.

Tabela 19 - Regras de associação com suporte > 30% (Análise geral)

Regras de Associação	Support	Confidence	Lift	Conviction
[Product] ⇒ [Time]	0,473	1	1,106	Infinity
[Gender] ⇒ [Education Level]	0,347	0,781	1,804	2,589
[Education Level] ⇒ [Gender]	0,347	0,802	1,804	2,804
[Gender] ⇒ [Marital Status]	0,344	0,774	2,031	2,74
[Marital Status] ⇒ [Gender]	0,344	0,903	2,031	5,703
[Gender] ⇒ [Education Level, Marital Status]	0,323	0,727	2,25	2,48
[Education Level] ⇒ [Marital Status]	0,323	0,747	1,959	2,442
[Education Level] ⇒ [Gender, Marital Status]	0,323	0,747	2,17	2,588
[Marital Status] ⇒ [Education Level]	0,323	0,848	1,959	3,724
[Marital Status] ⇒ [Gender, Education Level]	0,323	0,848	2,443	4,287
[Gender, Education Level] ⇒ [Marital Status]	0,323	0,931	2,443	8,977
[Gender, Marital Status] ⇒ [Education Level]	0,323	0,939	2,17	9,326
[Education Level, Marital Status] ⇒ [Gender]	0,323	1	2,25	Infinity

Verificou-se que apesar de a dimensão Product e as dimensões utilizadas para caracterizar os consumidores serem as mais frequentes, não existe uma associação relevante entre elas. Através das regras cujo suporte mínimo é de 30% conseguimos identificar duas claras tendências de exploração:

- Análise dos produtos ao longo do tempo.
- Análise do tipo de consumidores.

No entanto, ao baixar o nível de suporte, conseguimos identificar algumas regras interessantes que mostram que numa pequena proporção de transações os analistas tendem a fazer uma exploração com base nos produtos e nas características dos consumidores desses mesmos produtos. De facto, verificamos que apesar do baixo valor de suporte essas regras apresentam valores satisfatórios nas restantes medidas de interesse.

Consideremos, por exemplo a seguinte regra:

- [Product, Gender, Marital Status] ⇒ [Time, Education Level]

Esta regra é bastante interessante pois, tendo em conta os parâmetros de Confidence e Conviction, indica-nos que, sempre que as dimensões Product, Gender e Marital Status forem utilizadas numa query, prevê-se que as dimensões Time e Education Level também o sejam.

Tabela 20 - Regras de associação com confiança >70% (Análise geral)

Regras de Associação	Support	Confidence	Lift	Conviction
[Product, Education Level] ⇒ [Time, Gender]	0,13	0,702	2,012	2,184
[Product, Education Level] ⇒ [Time, Marital Status]	0,13	0,702	2,343	2,349
[Product, Education Level] ⇒ [Time, Gender, Marital Status]	0,13	0,702	2,675	2,473
[Product, Gender] ⇒ [Time, Education Level]	0,13	0,781	2,318	3,033
[Product, Gender] ⇒ [Time, Marital Status]	0,13	0,781	2,609	3,205
[Product, Gender] ⇒ [Education Level, Marital Status]	0,13	0,781	2,419	3,097
[Product, Gender] ⇒ [Time, Education Level, Marital Status]	0,13	0,781	3,237	3,47
[Product, Marital Status] ⇒ [Education Level]	0,13	0,876	2,025	4,59
[Product, Marital Status] ⇒ [Time, Gender]	0,13	0,876	2,513	5,27
[Product, Marital Status] ⇒ [Time, Education Level]	0,13	0,876	2,6	5,364
[Product, Marital Status] ⇒ [Gender, Education Level]	0,13	0,876	2,526	5,284
[Product, Marital Status] ⇒ [Time, Gender, Education Level]	0,13	0,876	3,487	6,058
[Product, Gender, Education Level] ⇒ [Marital Status]	0,13	1	2,624	Infinity
[Product, Gender, Marital Status] ⇒ [Education Level]	0,13	1	2,311	Infinity
[Product, Gender, Education Level] ⇒ [Time, Marital Status]	0,13	1	3,339	Infinity
[Product, Gender, Marital Status] ⇒ [Time, Education Level]	0,13	1	2,966	Infinity
[Product, Education Level, Marital Status] ⇒ [Time, Gender]	0,13	1	2,868	Infinity

4.6.2 Análise temporal dos itens frequentes

Ao falar de perfis de materialização, estamos a falar de perfis de utilização de cubos, e com isso é inevitável falar de períodos de utilização. De facto, podemos fazer diversos tipos de análise de utilização: por utilizador, por sessão e por período. Estas são variáveis essenciais quando pretendemos traçar um perfil de materialização. Uma vez que focámos este estudo na análise da utilização geral do cubo e não das suas sessões ou na classificação de utilizadores, optou-se por

seguir uma análise temporal de utilização. Desta forma, verificou-se pertinente analisar o acesso ao cubo em diversos períodos do dia.

Período da manhã:

O período da manhã, definido entre as 08:00h e as 12:59h, revelou-se o período com mais afluência com um total de 28170 *queries* realizadas sobre o nosso cubo de dados. Através da aplicação da 1ª fase do algoritmo obtivemos os seguintes item frequentes:

Tabela 21 - Itens frequentes do período da manhã (Análise temporal)

Size	Support	Item1	Item2	Item3	Item4	Item5
1	0.870	Time				
1	0.513	Gender				
1	0.506	Store				
1	0.473	Education Level				
1	0.438	Customers				
1	0.424	Marital Status				
2	0.416	Time	Store			
2	0.410	Time	Customers			
2	0.395	Gender	Education Level			
1	0.388	Product				
2	0.388	Time	Product			
2	0.388	Gender	Marital Status			
2	0.383	Time	Gender			
2	0.348	Education Level	Marital Status			
3	0.348	Gender	Education Level	Marital Status		
2	0.343	Time	Education Level			
2	0.323	Time	Marital Status			
3	0.286	Time	Gender	Marital Status		
2	0.285	Education Level	Customers			
2	0.265	Gender	Customers			
3	0.265	Time	Gender	Education Level		
3	0.256	Time	Education Level	Customers		

Tabela 22 - Itens frequentes do período da manhã (Análise temporal) (continuação)

Size	Support	Item1	Item2	Item3	Item4	Item5
3	0.246	Time	Education Level	Marital Status		
4	0.246	Time	Gender	Education Level	Marital Status	
3	0.236	Time	Gender	Customers		
3	0.228	Gender	Education Level	Customers		
2	0.217	Customers	Marital Status			
3	0.217	Time	Customers	Marital Status		
2	0.205	Store	Education Level			

Como pudemos observar, e tal como era esperado, a dimensão "Time" é a mais frequente, com um ratio de 87% sobre as *queries* realizadas, o que indica que esta dimensão foi utilizada em aproximadamente 24507 *queries*. Dentro dos itens frequentes com um valor de suporte superior a 40% encontram-se as dimensões: "Gender", "Store", "Education Level", "Customers" e "Marital Status".

Período da tarde:

Durante o período da tarde, definido entre as 13:00h e as 19:59h, registaram-se 21752 *queries*, das quais mais de 40% utilizaram as seguintes dimensões: "Time", "Product", "Education Level", "Gender" e "Marital Status".

Tabela 23 - Itens frequentes do período da tarde (Análise temporal)

Size	Support	Item1	Item2	Item3
1	0.915	Time		
1	0.548	Product		
2	0.548	Time	Product	
1	0.484	Education Level		
1	0.447	Gender		
1	0.428	Marital Status		
2	0.41	Gender	Marital Status	
2	0.408	Education Level	Gender	
2	0.406	Education Level	Marital Status	
3	0.406	Education Level	Gender	Marital Status

Período da noite:

O período da noite, definido entre as 20:00h e as 07:59h, revelou-se o período com menos afluência, com um total de 7767 *queries* realizadas. Durante este período, a dimensão "Time" foi utilizada em 100% das *queries*, seguindo-se da dimensão "Product" e "Store".

Tabela 24 - Itens frequentes do período da noite (Análise temporal)

Size	Support	Item1	Item2	Item3
1	1.000	Time		
1	0.572	Product		
2	0.572	Time	Product	
1	0.427	Store		
2	0.427	Time	Store	
1	0.189	Gender		
2	0.189	Time	Gender	
2	0.188	Product	Gender	
3	0.188	Time	Product	Gender
1	0.143	Education Level		
2	0.143	Time	Education Level	
2	0.142	Product	Education Level	
3	0.142	Time	Product	Education Level

Com base nos itens frequentes obtidos conseguimos identificar um conjunto de dimensões que são utilizadas para responder às diversas *queries* em mais do que um período do dia, no entanto com ratios de utilização diferentes, como é o caso da dimensão "Gender", das mais utilizadas durante a manhã e a tarde, no entanto pouco utilizada no período da noite, ou a dimensão "Store", que durante o período da tarde tem uma percentagem de utilização mais baixa (<40%).

É possível ainda identificar um conjunto de dimensões com uma taxa de acesso bastante baixa ou inexistente, como é o caso das dimensões "Store Size in SQFT", "Promotions", "Promotion Media", que apenas foram utilizadas no período da manhã e cujo suporte nem chega aos 20%. A percentagem de utilização da dimensão "Warehouse" é inferior aos 10%, não sendo por isso apresentada nos resultados.

Podemos verificar assim que é possível traçar perfis de materialização que permitam responder às necessidades diárias dos utilizadores, assim como também a necessidades específicas em

determinados períodos do dia. Apesar dos próprios resultados do algoritmo, obtidos através do RapidMiner, serem intuitivos, tivemos a necessidade de os tornar um pouco mais simples e flexíveis de forma a adaptá-los melhor ao próprio utilizador/analista.

4.7 Cubes Definer

A utilização de preferências deve ser adaptada ao contexto do problema e às necessidades de análise de cada agente de decisão. Como vimos, anteriormente, na secção 4.6, pareceu-nos pertinente analisar apenas regras e itens com valores de interesse altos. Contudo, pudemos observar também que por vezes é possível obter regras bastante interessantes ao variar as diversas medidas de interesse. De forma a permitir a utilização e adaptação deste estudo não a um caso específico como foi por exemplo este caso das vendas, mas também a diversos cubos, com diferentes propósitos e estruturas, optou-se por criar um seletor de vistas, o *CubesDefiner*. Este seletor permite, para além de apresentar os resultados de forma simples ao utilizador, adaptar os resultados das preferências ou regras de associação obtidas às necessidades requeridas, e gerar a partir do resultado da aplicação dos critérios do agente de decisão, visualizar as vistas do cubo que serão necessárias materializar para responder a um conjunto de preferências com esses determinados critérios.

Esta aplicação, desenvolvida em linguagem Java, permite ao utilizador ver as preferências, ou seja, os relacionamentos entre as diversas dimensões consultadas, adaptar as preferências aos seus critérios de interesse e, além disso, ver a *lattice* do cubo correspondente ao conjunto resultante. A aplicação permite, também, importar os resultados do algoritmo de mineração, isto é, o conjunto de regras, e adaptá-lo com base num conjunto de parâmetros de avaliação (medidas de interesse).

4.7.1 Medidas de interesse

Como pudemos observar, os algoritmos de mineração de regras de associação podem devolver um conjunto bastante vasto de regras. De facto, apesar do enorme progresso registado na obtenção de regras de associação e na melhoria dos algoritmos de mineração, a aplicação destes sobre *datasets* reais pode levar, em alguns casos, à obtenção de milhões de regras (Zheng et al., 2001).

De facto, dada a necessidade de reduzir o número de regras obtidas e da obtenção de regras que de facto respondessem às necessidades do analista, muitos optaram pela utilização de métodos de *pruning* e de medidas de interesse. Saber analisar uma regra pode fazer toda a diferença na qualidade da informação obtido. Porém, com um conjunto extremamente vasto, essa tarefa torna-se inexequível.

A definição de medidas de interesse veio, por um lado, ajudar a reduzir o conjunto de regras obtidas, mas também, a determinar estatisticamente e de forma objetiva a força de uma regra. O *CubesDefiner* permite ao utilizador aplicar um conjunto de medidas de interesse sobre o conjunto de regras de associação importado, permitindo definir limites de suporte, confiança, lift e convicção, e, com isso, obter um conjunto de regras mais específico.

As medidas de suporte e confiança têm sido as mais utilizadas desde que surgiu a necessidade de reduzir o número de regras de associação. O suporte de um determinado conjunto de itens, representa a percentagem de transações do *dataset* que contêm esses itens. Por outras palavras, o suporte de uma regra de associação A⇒B, Sup(A $\Rightarrow$ B), é dado por Sup(A $\cup$ B). Por outro lado, a confiança, pode ser vista como a probabilidade condicional de uma regra, ou seja, representa a percentagem de transações contendo o(s) item(s) antecedente(s), onde se verifica a presença do(s) item(s) consequente(s). A confiança de uma regra A⇒B é defina por *Conf*(A⇒B) = *Sup*($A \cup B$) / *Sup*(A) (Gonçalves, 2005). Apesar destas duas medidas apresentarem bons resultados, não resolvem dois problemas bastante comuns quando lidamos com este tipo de regras: a redundância e regras óbvias ou pouco interessantes.

Para contornar o problema das regras demasiado óbvias, que nada acrescentam à informação útil para o analista, foram definidas novas medidas de interesse que permitem determinar a dependência entre itens. O *Lift* (Factor de Interesse) permite medir o desvio de independência estatística entre itens, isto é, permite determinar se o facto de dois itens aparecem juntos numa transação é provocado por uma relação de dependência entre esses dois itens. O *Lift* é definido por Lift(A $\Rightarrow$ B) = Conf(A $\Rightarrow$ B) / Sup(B) (Gonçalves, 2005). Esta medida pode ter três interpretações:

- *Se Lift*($A \Rightarrow B$) = 1, os itens são independentes;
- *Se Lift*($A \Rightarrow B$) < 1, *os itens são* negativamente dependentes, o que indica que o suporte real desta regra é inferior ao suporte esperado.

- *Se Lift*($A \Rightarrow B$) > 1, os itens são positivamente dependentes, o que indica que o suporte real desta regra é superior ao suporte esperado.

Quanto maior o valor de *lift*, maior a dependência entre os itens, e, consequentemente, mais interessante a regra.

A última medida de interesse utilizada neste caso foi a Convicção. Esta medida permite medir a força da implicação da regra, uma vez que é unidirecional, e tal como o *Lift* permite determinar a dependência entre itens. A convicção é dada pela seguinte equação: *Conv*($A \Rightarrow B$) = *Sup*(A)×*Sup*($\neg B$) / *Sup*($A \cup \neg B$). Tal como o *Lift*, esta medida também pode ter várias interpretações:

- *Se Conv*($A \Rightarrow B$) = 1, o antecedente é completamente independente do consequente.
- *Se Conv*($A \Rightarrow B$) > 1, o antecedente é dependente do consequente. Quanto maior o valor de convicção maior a dependência.
- *Regras onde o antecedente nunca aparece sem o consequente, ou seja, onde Conf($A \Rightarrow B$)=100%, o valor de convicção será* ∞.

As imagens seguintes demonstram como é possível através do *CubesDefiner* aplicar as diversas medidas de interesse acima referidas sobre um dado conjunto de regras de associação.

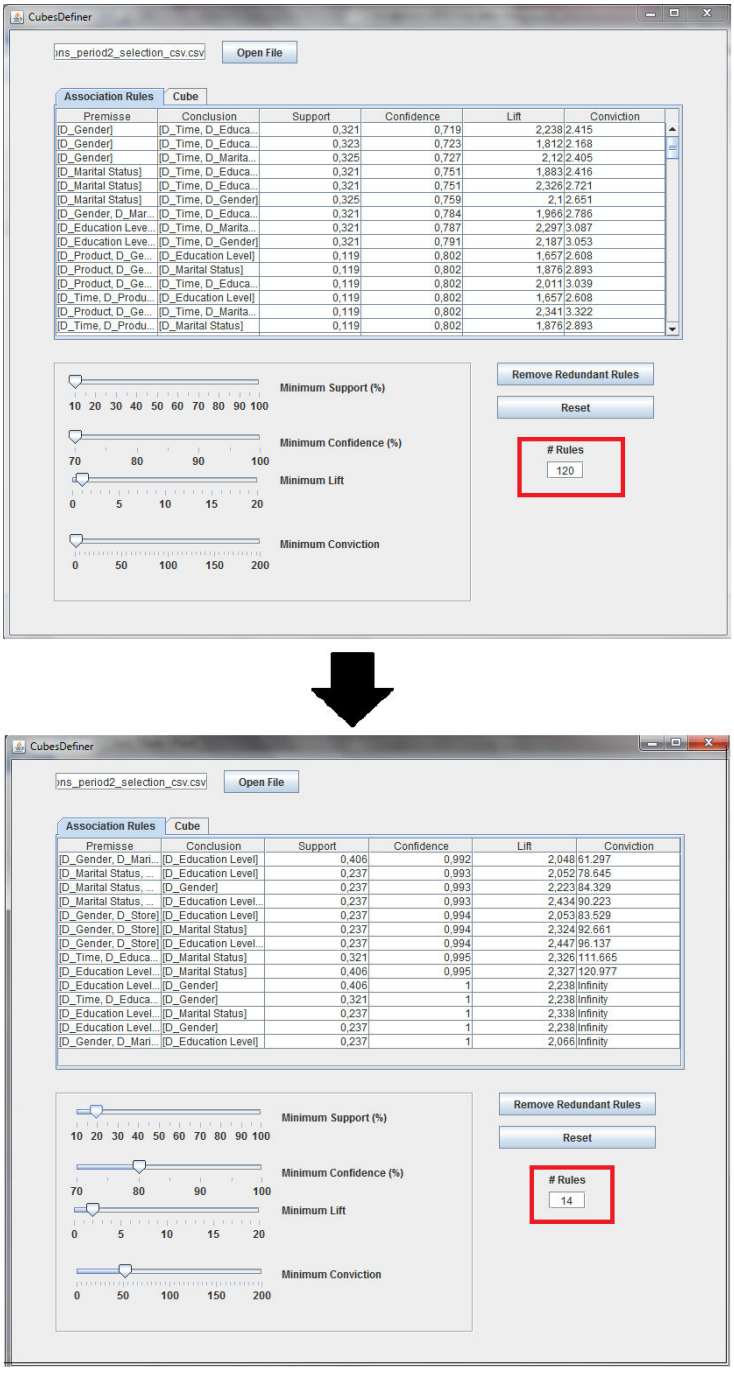


Figura 15 - Aplicação de medidas de interesse através do CubesDefiner

Nas imagens apresentadas pudemos verificar como a aplicação destas medidas sobre o conjunto de dados inicial permitiu diminuir significativamente o conjunto de regras apresentado.

4.7.2 Pruning de preferências

A aplicação de medidas de interesse permitiu melhorar a qualidade das regras de associação, nomeadamente em termos de relevância ou força da regra. No entanto, dentro do conjunto de regras obtido foi possível verificar a existência de várias regras redundantes como por exemplo:

1. [Product, Marital Status] ⇒ [Time, Gender]
2. [Product, Marital Status] ⇒ [Time, Gender, Education Level]

Assim, revelou-se importante que a aplicação permitisse aplicar técnicas de *pruning* sobre o conjunto de regras, de forma a melhorar o conjunto de dados apresentados e com isso fornecer uma melhor experiência ao utilizador. Para executar essa tarefa, foi escolhida uma técnica de *pruning* baseada no conceito de *rule covers*. Este conceito, mencionado em alguns estudos relacionados com *pruning* de regras de associação (Jaroszewicz and Simovici, 2002, Toivonen et al., 1995), defende que se existir uma regra X⇒Y que cobre o mesmo conjunto de dados que uma regra mais específica XZ⇒Y, a regra mais específica pode ser eliminada, desde que não adicione informação relevante para análise. Já em (Bayardo Jr. et al., 1999), por outro lado, os autores defendem que se a força da regra mais específica XZ⇒Y for maior que a da X⇒Y, a regra mais geral poderá ser eliminada, pois a mais específica irá aumentar a capacidade preditiva. Tal como em (Jaroszewicz and Simovici, 2002), também estes autores utilizaram um *threshold* mínimo, neste caso com base em medidas de interesse como a confiança e o suporte, para determinar se uma regra trazia um maior benefício em relação a outra.

Como para este estudo não era necessariamente importante determinar o conjunto de regras que permitissem aumentar o poder preditivo, mas sim definir um conjunto de dimensões mais específico, não reduzindo no entanto o poder analítico, optou-se então por utilizar um algoritmo que eliminasse todas as regras mais específicas que não fossem relevantes. Por outras palavras, todas as regras específicas cuja confiança não respeitasse um *threshold* mínimo de confiança, i.e., se conf(XZ⇒Y)≤conf(X⇒Y) ou se conf(XZ⇒Y)−conf(X⇒Y)<*minconf*, seriam eliminadas. Como *threshold* mínimo, definiu-se minconf=0.01.

4.7.3 Das regras à lattice

Após definidas as regras que melhor satisfazem as necessidades do utilizador, foi finalmente possível determinar qual a estrutura multidimensional resultante, através da construção da *lattice* do cubo definida pelos itens dessas mesmas regras. Como pudemos ver, as regras podem ser constituídas por dimensões e medidas. A construção da *lattice* passou assim pela aplicação da definição de conjunto *Group-by*, apresentada anteriormente no capítulo 2, sobre as dimensões constituintes das regras resultantes. A representação destas regras, em forma de *lattice,* permite ao utilizador ter a percepção da redução em termos de memória que poderá obter, uma vez que, como poderemos ver a seguir, em alguns casos o número de agregações verificadas reduziu significativamente as partes do cubo que necessitará materializar e que melhor respondem às suas necessidades, bem como as operações OLAP que a estrutura resultante lhe permitirá efetuar. Se analisarmos a Figura 16, podemos observar a *lattice* gerada, correspondente ao conjunto final de regras de associação.

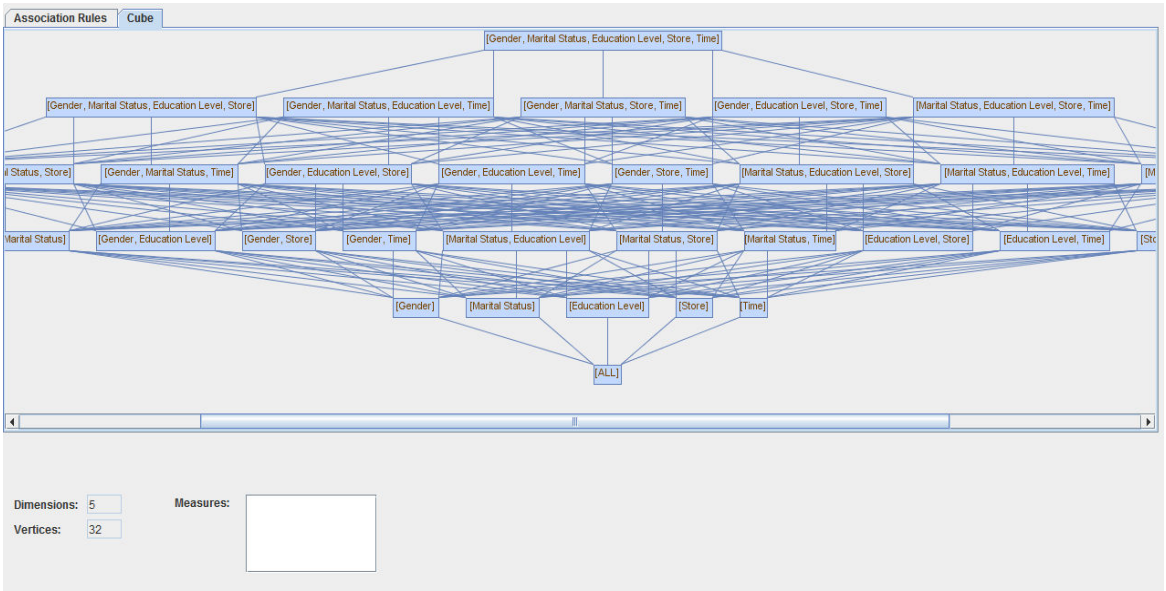


Figura 16 - Lattice do cubo no CubesDefiner

4.7.4 Como utilizar o CubesDefiner

Ao se abrir a aplicação pode-se ver no canto superior esquerdo o botão 'Open File', que permite ao utilizador selecionar o ficheiro com as regras de associação e importá-las para a tabela disponível sob a tab 'Association Rules' (Figura 17).

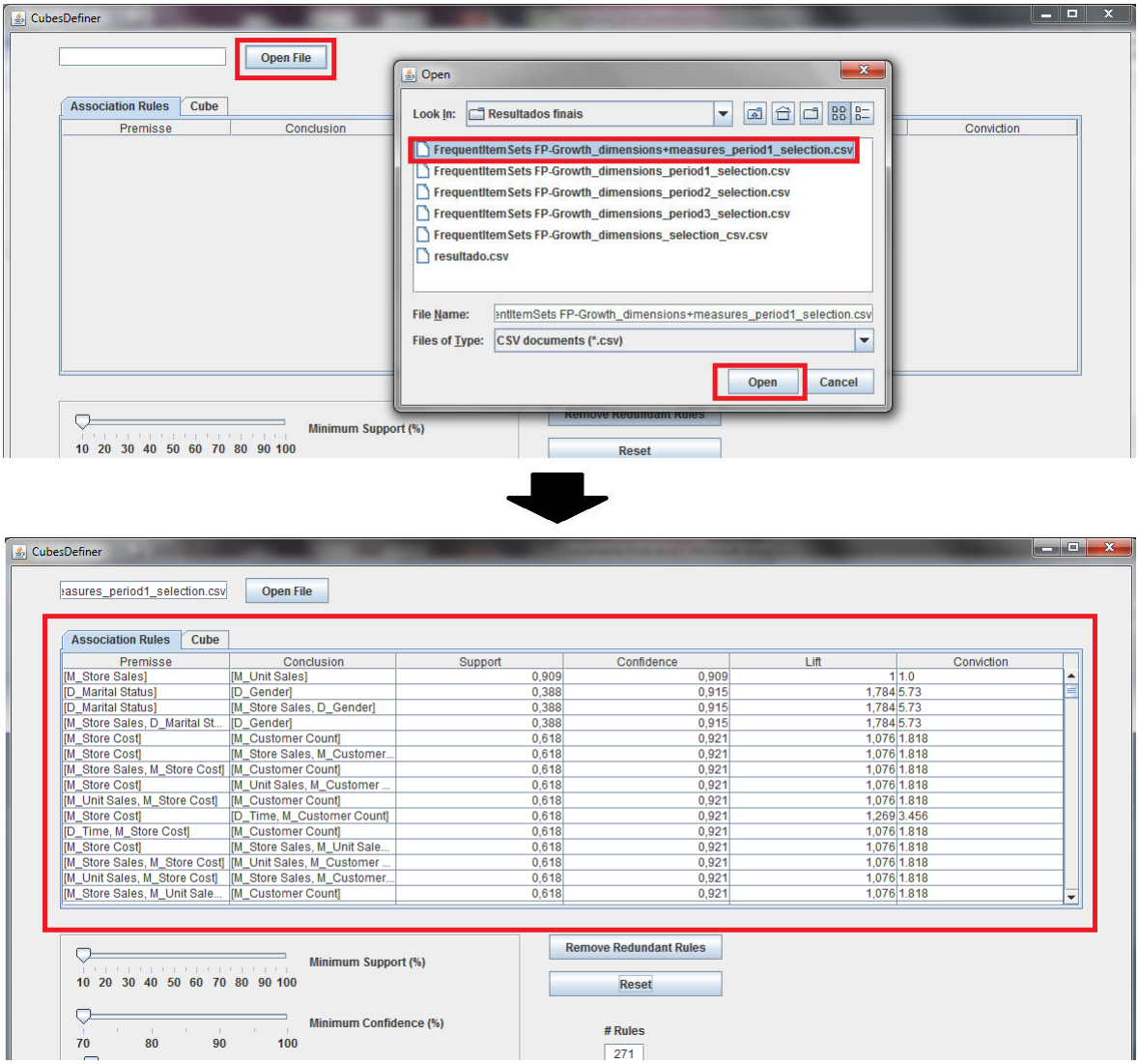


Figura 17 - CubesDefiner: Importação das regras de associação

Após se importar os dados do ficheiro, um conjunto de cursores permite aplicar *thresholds* mínimos de diferentes medidas de interesse sobre o conjunto de regras (Figura 18).

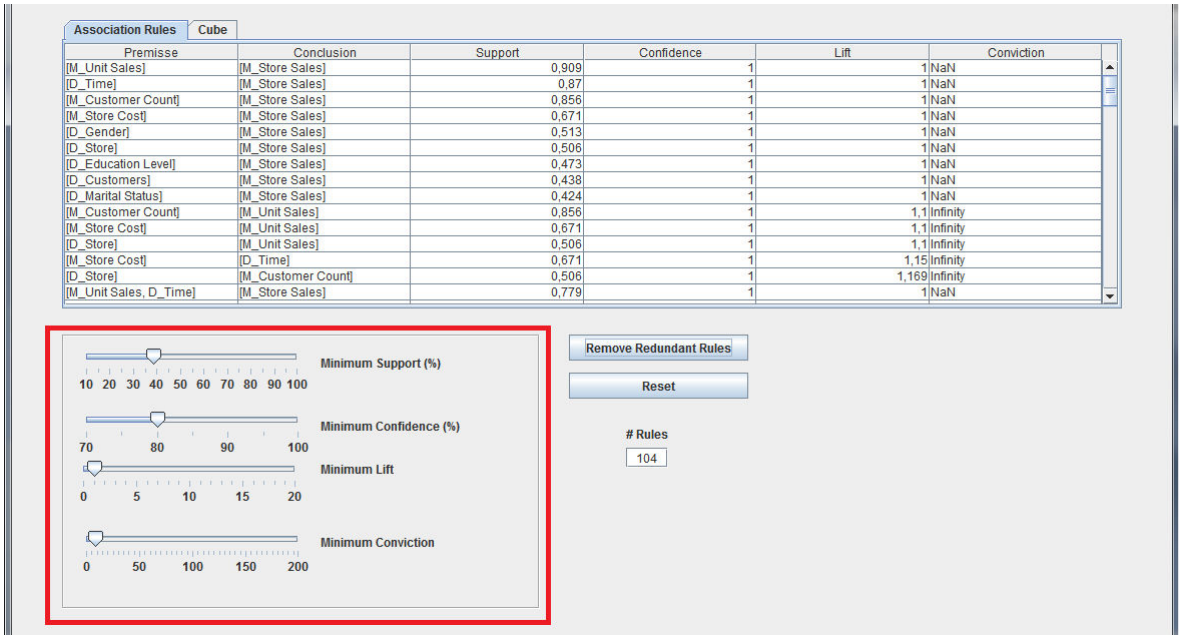


Figura 18 - CubesDefiner: Aplicação de medidas de interesse

Ao carregar no botão 'Remove Redundant Rules' o algoritmo apresentado na secção 4.7.2 será aplicado sobre o conjunto atual de regras (Figura 19).

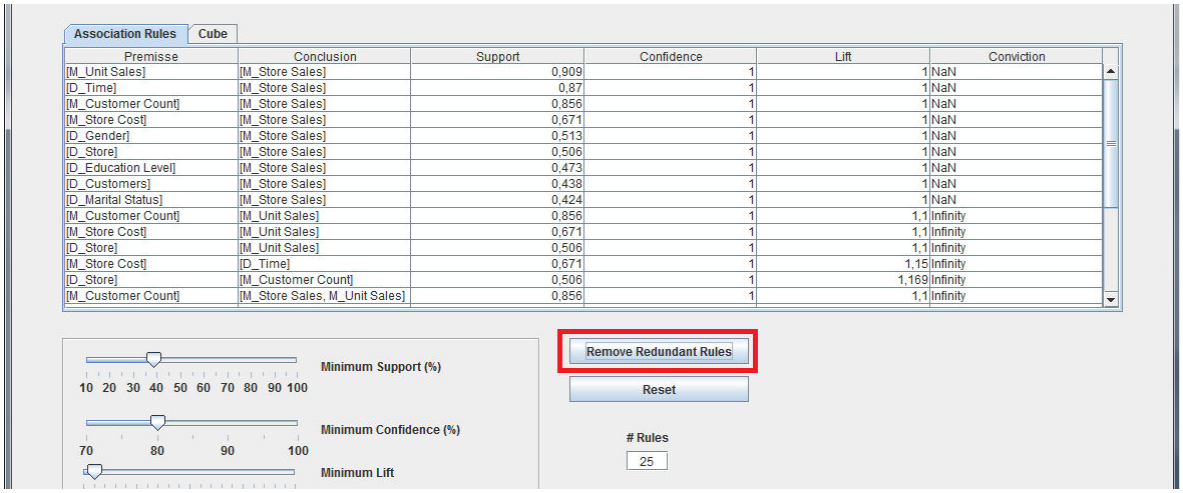


Figura 19 - CubesDefiner: Remoção de regras redundantes

A qualquer momento o utilizador poderá reverter todas as ações efetuadas e voltar ao conjunto inicial de regras, bastando para isso carregar no botão 'Reset' (Figura 20).

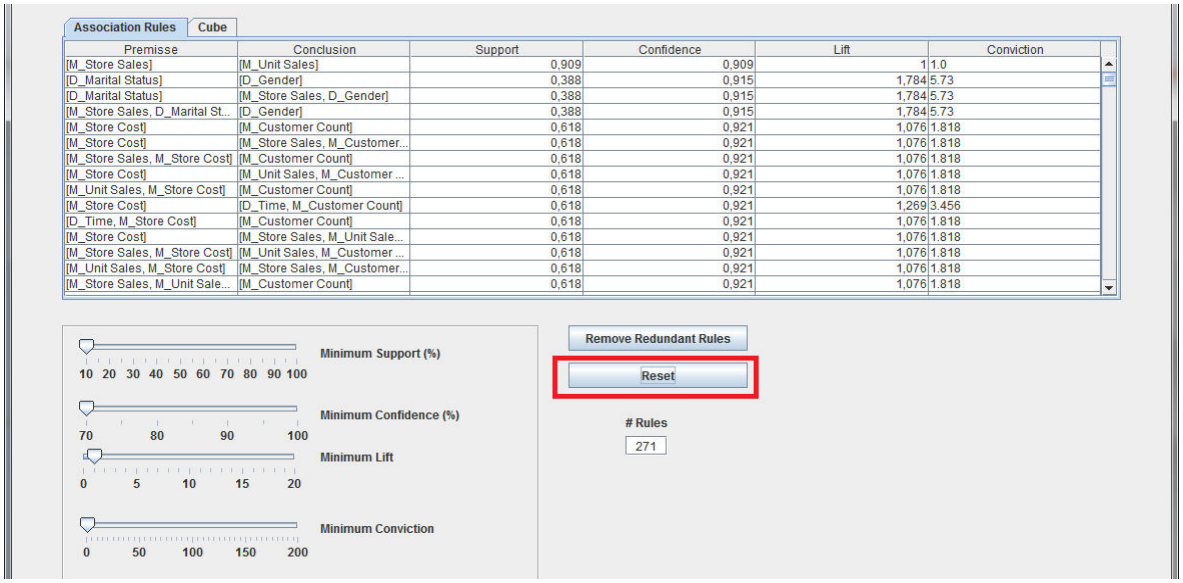


Figura 20 - CubesDefiner: Reiniciar conjunto de regras de associação

Ao alterar para a tab 'Cube' (Figura 21), a lattice do cubo irá ser construída com base na lista atual de regras apresentadas na tabela de 'Association Rules'. São também apresentadas o número de dimensões e vértices da *lattice*, assim como as medidas constituintes das regras que são necessárias materializar.

Figura 21 - CubesDefiner: Lattice correspondente às regras de associação

Capítulo 5

Conclusões e Trabalho Futuro

5.1 Conclusões

A utilização de sistemas de *data warehousing* tem tido um crescimento significativo nos últimos anos. A necessidade de armazenar grandes quantidades de dados para posterior análise e extração de informação tem levado várias organizações a optarem por este tipo de sistemas. No entanto, estas estruturas multidimensionais são complexas e acarretam, usualmente, custos de computação e armazenamento significativos para qualquer organização. Uma análise preliminar no início deste trabalho revelou que, não raras vezes, é feita uma utilização ineficiente deste tipo de estruturas. O armazenamento de cubo esparsos, a fraca qualidade dos dados ou mesmo a inadaptação do cubo às necessidades diárias de cada utilizador/analista pode ter um impacto significativo no desempenho do sistema.

Já vários autores se tinham debruçado sobre a necessidade de uma utilização mais eficiente deste tipo de estruturas e introduziram o conceito de preferências em sistemas OLAP. Neste trabalho, o que se pretendeu foi analisar a utilização de cubos OLAP por parte de utilizadores, ou seja, as *queries* MDX realizadas e definir preferências recorrendo à aplicação de técnicas de mineração de dados conhecidas. A escolha dos algoritmos de associação para a descoberta de preferências deveu-se essencialmente ao facto de ser possível definir preferências como um conjunto de itens frequentes, ou seja, um conjunto de dados relacionados que são pedidos várias vezes ao longo do tempo. Algoritmos como o Apriori ou o FP-Growth demonstravam bons resultados na descoberta

de conjuntos de itens frequentes e regras de associação. No entanto, possivelmente devido ao facto de ter sido utilizado um conjunto de dados de testes, no qual a realização de *queries* MDX sobre o cubo não foi constante ao longo do tempo, o algoritmo FP-Growth revelou-se mais eficaz para o nosso caso de estudo, contrariando um pouco o que teoricamente seria esperado.

Para aplicação do algoritmo em si, foi necessário efetuar a limpeza e o tratamento dos dados iniciais. Desta forma, as *queries* MDX foram previamente extraídas de um ficheiro de log de exploração e armazenadas numa base de dados relacional, para que fosse possível manter a informação estruturada e normalizada. A segunda etapa passou pela implementação de um pequeno *data warehouse* para acolhimento dos dados recolhidos, como é o caso das dimensões e medidas utilizadas, e que veio facilitar a aplicação dos algoritmos de mineração sobre os mesmos.

Para uma melhor análise dos resultados dividimos o nosso conjunto de dados em três amostras distintas: uma apenas referente às dimensões pedidas por todas as *queries* MDX realizadas sobre o cubo de dados, outra que continha informação apenas das medidas, e uma terceira que incorporava tanto as medidas como as dimensões que eram utilizadas para satisfazer essas mesmas *queries*.

Uma análise preliminar ao conjunto de dados iniciais demonstrou que a taxa de utilização do cubo era variável consoante o utilizador ou mesmo o período do dia. Desta forma, revelou-se pertinente realizar uma análise temporal de utilização do cubo e extrair as preferências OLAP correspondentes. De facto, o nosso conjunto de dados permitia-nos realizar diversos tipos de análise de utilização: por utilizador, por sessão ou por período. No entanto, o propósito deste estudo não era a análise de sessões OLAP ou a classificação de utilizadores. A análise temporal era a que melhor se adaptava aos novos objetivos: traçar perfis de materialização que permitissem uma utilização mais eficiente de recursos e, por sua vez, do próprio cubo.

Como foi possível observar, a aplicação do algoritmo FP-Growth sobre estas amostras revelou dados bastante interessantes e comprovou que é possível definir preferências de utilização com base nos resultados obtidos, ou seja, com base nos itens frequentes e nas regras de associação resultantes. No entanto, o conjunto de resultados obtido revelou-se bastante vasto o que dificultou um pouco a análise. Uma das opções para atenuar tal situação, poderia ser a redução do número de regras obtidas com base em medidas de interesse. Porém, dessa forma, estaríamos a restringir o resultado com base nos nossos próprios critérios, fugindo a um dos objetivos principais deste trabalho em obter um perfil de utilização que melhor se adapta ao utilizador.

Assim, foi desenvolvido o *CubesDefiner* que veio permitir, para além de apresentar os resultados de forma simples ao utilizador, adaptar os resultados das preferências ou regras de associação obtidas às suas necessidades. Com esta pequena aplicação desenvolvida através da linguagem Java, pretendemo-nos relevar quatro aspetos essenciais: apresentar os resultados de forma simples, permitir ao analista aplicar medidas de interesse de forma a adaptar as preferências às suas necessidades, remover regras redundantes que não trouxessem informação relevante para a análise e que apenas tornavam o conjunto de resultados mais confuso, e, por fim, mas não menos importante, visualizar as vistas do cubo que serão necessárias materializar para responder a um conjunto de preferências com esses determinados critérios.

5.2 Trabalho futuro

Medidas de redução de custos são de extrema importância para qualquer tipo de organização. Como já foi referido, com o aumento de sistemas de informação, fatores como custos de memória e computações tornaram-se relevantes na gestão estratégica de custos organizacionais. Este trabalho de dissertação demonstrou ser possível reduzir custos de materialização e computação ao analisar os componentes do cubo presentes nas *queries* MDX realizadas sobre o mesmo, e apresentar uma estrutura mais específica, que melhor se adaptada às necessidades da organização e seus analistas. No entanto, o estudo de preferências apresentado focou-se apenas nas linhas orientadoras deste trabalho e, representa apenas uma pequena parcela da quantidade de estudos de preferências que é possível realizar através dos dados obtidos através das *queries* MDX.

Para além de informação relativa aos componentes do cubo utilizados, as *queries* MDX contêm dados relativos aos próprios valores categóricos e numéricos assumidos por esses componentes para responder às questões do utilizador. Assim, seria interessante conjugar as técnicas apresentadas, com o conceito de preferências sobre atributos e medidas, apresentado no capítulo 2 desta dissertação, de forma a obter novas perspetivas no estudo de preferências e, quem sabe, proporcionar, para além de redução de custos, um maior poder preditivo às organizações.

Bibliografia

AGRAWAL, R. & SRIKANT, R. 1994. Fast Algorithms for Mining Association Rules in Large Databases. *Proceedings of the 20th International Conference on Very Large Data Bases.* Morgan Kaufmann Publishers Inc.

ALVES, R. & BELO, O. 2003. An OLAM Approach to Analyze e-commerce Clickstreams. *1st Simposio Doutoral do Departamento de Informatica, SDDI 2003*.

BAYARDO JR., R. J., AGRAWAL, R. & GUNOPULOS, D. Year. Constraint-based rule mining in large, dense databases. *In:* Proceedings 15th International Conference on Data Engineering, 1999 Sydney, NSW.

BLAZEWICZ, J. 2003. *Handbook on Data Management in Information Systems*, Springer-Verlag New York, Inc.

CHAUDHURI, S. & DAYAL, U. 1997. An overview of data warehousing and OLAP technology. *ACM SIGMOD Record*.

CHEN, Q. 1999. *Mining Exceptions And Quantitative Association Rules In Olap Data Cube.* University of Science and Technology of China.

CHMELAR, P. & STRYKA, L. 2008. Iterative, Interactive and Intuitive Analytical Data Mining. Faculty of Information Technology, Brno University of Technology.

FUNG, J. & WONG, H. K. 2002. Online Analytical Mining of Path Traversal Patterns for Web Measurement. IGI Global.

GIACOMETTI, A., MARCEL, P. & NEGRE, E. 2008. A framework for recommending OLAP queries. *Proceedings of the ACM 11th international workshop on Data warehousing and OLAP.* Napa Valley, California, USA: ACM.

GIACOMETTI, A., MARCEL, P. & NEGRE, E. 2009. Recommending Multidimensional Queries. *Proceedings of the 11th International Conference on Data Warehousing and Knowledge Discovery.* Linz, Austria: Springer-Verlag.

GOETHALS, B. 2010. Frequent Set Mining. *In:* MAIMON, O. & ROKACH, L. (eds.) *Data Mining and Knowledge Discovery Handbook.* Springer US.

GOIL, S. & CHOUDHARY, A. 1998. High performance multidimensional analysis and data mining. *Proceedings of the 1998 ACM/IEEE conference on Supercomputing (CDROM).* San Jose, CA: IEEE Computer Society.

GOLFARELLI, M. & RIZZI, S. 2008. Preferences on OLAP Datacubes. DEIS, University of Bologna, Viale Risorgimento 2, Bologna, Italy.

GOLFARELLI, M. & RIZZI, S. 2009. Expressing OLAP Preferences. *Proceedings of the 21st International Conference on Scientific and Statistical Database Management.* New Orleans, LA, USA: Springer-Verlag.

GONÇALVES, E. C. 2005. Regras de Associação e suas Medidas de Interesse Objetivas e Subjetiva. *INFOCOMP Journal of Computer Science***,** 26 - 35.

GYŐRÖDI, C., GYŐRÖDI, R. & HOLBAN, S. 2004. A Comparative Study of Association Rules Mining Algorithms. *SACI 2004, 1st Romanian-Hungarian Joint Symposium on Applied Computational Intelligence.* Timisoara, Romania.

HAN, J. 1997. OLAP Mining: An Integration of OLAP with Data Mining. *In In Proceedings of the 7th IFIP 2.6 Working Conference on Database Semantics (DS-7)***,** 1-9.

HAN, J. 1998. Towards on-line analytical mining in large databases. *SIGMOD Rec.,* 27**,** 97-107.

HAN, J., CHEE, S. H. S. & CHIANG, J. Y. 1998. Issues for On-Line Analytical Mining of Data Warehouses. In SIGMOD'98 Workshop on Research Issues on Data Mining and Knowledge - Discovery (DMKD'98).

HAN, J., CHIANG, J. Y., CHEE, S., CHEN, J., CHEN, Q., CHENG, S., GONG, W., KAMBER, M., KOPERSKI, K., LIU, G., LU, Y., STEFANOVIC, N., WINSTONE, L., XIA, B. B., ZAIANE, O. R., ZHANG, S. & ZHU, H. 1997a. DBMiner: a system for data mining in relational databases and data warehouses. *Proceedings of the 1997 conference of the Centre for Advanced Studies on Collaborative research.* Toronto, Ontario, Canada: IBM Press.

HAN, J. & KAMBER, M. 2006. *Data Mining: Concepts and Techniques, 2nd ed,* San Francisco, Morgan Kaufmann Publishers.

HAN, J., KOPERSKI, K. & STEFANOVIC, N. 1997b. GeoMiner: a system prototype for spatial data mining. *Proceedings of the 1997 ACM SIGMOD international conference on Management of data.* Tucson, Arizona, United States: ACM.

HAN, J., PEI, J. & YIN, Y. 2000. Mining frequent patterns without candidate generation. *Proceedings of the 2000 ACM SIGMOD international conference on Management of data.* Dallas, Texas, United States: ACM.

HU, X. & CERCONE, N. 2002. An OLAM framework for Web usage mining and business intelligence reporting. *In Fuzzy Systems, 2002. FUZZ-IEEE'02. Proceedings of the 2002 IEEE International Conference,* Vol. 2**.,** 950 –955.

IVÁNCSY, R. & VAJK, I. 2006. Frequent pattern mining in web log data. Budapest, Hungary: Department of Automation and Applied Informatics, and HAS-BUTE Control Research Group, Budapest University of Technology and Economics.

JAROSZEWICZ, S. & SIMOVICI, D. A. 2002. Pruning Redundant Association Rules Using Maximum Entropy Principle. *Proceedings of the 6th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining.* Springer-Verlag.

JOSHI, K. P., JOSHI, A., YESHA, Y. & KRISHNAPURAM, R. 1999. Warehousing and mining Web logs. *Proceedings of the 2nd international workshop on Web information and data management.* Kansas City, Missouri, United States: ACM.

KIEBLING, W. 2002. Foundations of preferences in database systems. *Proceedings of the 28th international conference on Very Large Data Bases.* Hong Kong, China: VLDB Endowment.

KOZMINA, N. & NIEDRITE, L. 2010. OLAP Personalization with User-Describing Profiles. *In:* FORBRIG, P. & GÜNTHER, H. (eds.) *BIR.* Springer.

KUMAR, B. S. & RUKMANI, K. V. 2010. Implementation of Web Usage Mining Using APRIORI and FP-Growth Algorithms. *Int. J. of Advanced Networking and Applications,* 01**,** 400-404.

MAHAR, B. 2009. Data Cubing Algorithms - Comparative Study of Data Cubing Algorithms. *In Proceedings of the 3rd National Conference*.

MESSAOUD, R. B., BOUSSAÏD, O. & LOUDCHER-RABASEDA, S. 2006a. Mining association rules in olap cubes. *In International Conference on Innovations in Information Technology (ITT 06), Dubai*.

MESSAOUD, R. B., BOUSSAID, O. & RABASÉDA, S. 2004. A new OLAP aggregation based on the AHC technique. *Proceedings of the 7th ACM international workshop on Data warehousing and OLAP.* Washington, DC, USA: ACM.

MESSAOUD, R. B., BOUSSAID, O. & RABASÉDA, S. 2006b. A Data Mining-Based OLAP Aggregation of Complex Data: Application on XML Documents. *IJDWM - International Journal of Data Warehousing and Mining,* 2**,** 1-26.

MOODY, D. L. & KORTINK, M. A. R. 2000. From Enterprise Models to Dimensional Models: A Methodology for Data Warehouse and Data Mart Design. *Proc. of 2nd Int. Wksp on Design and Management of Data Warehouses. CEUR-WS.org*.

NESTOROV, S. & JUKIC, N. 2003. Ad-Hoc Association-Rule Mining within the Data Warehouse. *Proceedings of the 36th Annual Hawaii International Conference on System Sciences (HICSS'03) - Track 8 - Volume 8 %@ 0-7695-1874-5.* IEEE Computer Society.

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.* Springer-Verlag.

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.* Atlanta, Georgia, USA: ACM.

NOLAN, C. 1999. Manipulate and Query OLAP Data Using ADOMD and Multidimensional Expressions. *Microsoft Systems Journal***,** 51-59.

PEDERSEN, T. B. & TRYFONA, N. 2001. Pre-aggregation in Spatial Data Warehouses. *Proceedings of the 7th International Symposium on Advances in Spatial and Temporal Databases.* Springer-Verlag.

PINTO, H., HAN, J., PEI, J., WANG, K., CHEN, Q. & DAYAL, U. 2001. Multi-dimensional sequential pattern mining. *Proceedings of the tenth international conference on Information and knowledge management.* Atlanta, Georgia, USA: ACM.

SARAWAGI, S., AGRAWAL, R. & MEGIDDO, N. 1998. Discovery-driven exploration of OLAP data cubes.

SEIFERT, J. W. 2004. Data Mining: An Overview. *Resources, Science, and Industry Division, Congressional Research Service, Library of Congress. http://digital.library.unt.edu/govdocs/crs/permalink/meta-crs-6059 Computer Sci.*

STONEBRAKER, M. & HELLERSTEIN, J. M. 1998. Readings in database systems (3rd ed.). Morgan Kaufmann Publishers Inc.

TAN, P.-N., STEINBACH, M. & KUMAR, V. 2005. *Introduction to Data Mining*, Addison-Wesley Longman Publishing Co., Inc.

TJIOE, H. & TANIAR, D. 2004. A Framework for Mining Association Rules in Data Warehouses. *In:* YANG, Z., YIN, H. & EVERSON, R. (eds.) *Intelligent Data Engineering and Automated Learning – IDEAL 2004.* Springer Berlin Heidelberg.

TOIVONEN, H., KLEMETTINEN, M., RONKAINEN, P., HATONEN, K. & MANNILA, H. 1995. Pruning and grouping discovered association rules. *ECML'95 MLnet workshop on statistics, machine learning, and knowledge discovery in databases.*

VIJAYALAKSHMI, S. & RAJA, S. S. 2005. Multidimensional frequent pattern mining using association rule based constraints. *Proceedings of the Second international conference on Distributed Computing and Internet Technology %@ 3-540-30999-3, 978-3-540-30999-4.* Bhubaneswar, India: Springer-Verlag.

WIXOM, B. H. & WATSON, H. J. 2001. An empirical investigation of the factors affecting data warehousing success. *MIS Q.,* 25**,** 17-32.

ZAIANE, O. 1999. Principles of Knowledge Discovery in Databases, Chapter 1: Introduction to Data Mining. CMPUT 690, University of Alberta.

ZAÏANE, O. R., HAN, J., LI, Z.-N., CHEE, S. H. & CHIANG, J. Y. 1998a. MultiMediaMiner: a system prototype for multimedia data mining. *Proceedings of the 1998 ACM SIGMOD international conference on Management of data.* Seattle, Washington, United States: ACM.

ZAÏANE, O. R., XIN, M. & HAN, J. 1998b. Discovering Web Access Patterns and Trends by Applying OLAP and Data Mining Technology on Web Logs. *Proceedings of the Advances in Digital Libraries Conference %@ 0-8186-8464-X.* IEEE Computer Society.

ZHENG, Z., KOHAVI, R. & MASON, L. 2001. Real world performance of association rule algorithms. *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining.* San Francisco, California: ACM.

Referências WWW

HAN, J. 2000. DBMiner Enterprise Version E0.8 - DBMiner Tutorial [Online]. Available at: http://www.cs.sfu.ca/CourseCentral/459/han/tutorial/tutorial.html [Accessed 2012].

MICROSOFT. 2012. The Basic MDX Query (MDX) [Online]. Available at: http://msdn.microsoft.com/en-us/library/ms144785.aspx [Accessed 2012].

ORACLE CORPORATION. 2012. MySQL Workbench 5.2: Design, Develop, Administer [Online]. Available at: https://www.mysql.com/products/workbench/ [Accessed 2012].