Universidade do Minho
Escola de Engenharia

Nélio Martins Guimarães

Qualidade de Serviço em Sistemas de Data *Warehousing*

Outubro 2009

Universidade do Minho
Escola de Engenharia

Nélio Martins Guimarães

Qualidade de Serviço em Sistemas de Data *Warehousing*

Dissertação de Mestrado em Informática

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

Outubro 2009

Aos meus pais, à minha irmã, à Sónia e à minha avó

Agradecimentos

Quero deixar neste documento, na esperança que prevaleça ao longo do tempo, expressos os meus sinceros agradecimentos a todos aqueles que me apoiaram e contribuíram para a realização desta tese, em particular:

- Ao meu orientador e amigo Professor Doutor Orlando Manuel de Oliveira Belo pelo seu incentivo, e que através das suas constantes criticas e sugestões, tornou possível a elaboração da presente dissertação, tendo ainda contribuído para a evolução do meu espírito crítico e iniciativa científica.
- À Alice Marques pelos anos de trabalho em conjunto e por todas as horas que passamos a discutir os problemas desta dissertação.
- Ao Tomé Faria pelo constante incentivo e disponibilidade para a leitura dos meus trabalhos.
- Aos meus colegas do Departamento de Informática Ana Sofia Duarte, Dave Moderno, Hugo Maia, Rui Pedro Carvalho e Nuno Oliveira, pelo espírito de colaboração e incentivo.
- Aos Metallica, Iron Maiden, Tom Waits, António Vivaldi, Rodrigo y Gabriela e a todos os outros, pelas horas de companhia que as suas musicas me fizeram durante todo este tempo.
- À "Neka" por todo o incondicional apoio, compreensão, carinho e ajuda que sempre me prestou, pois sem ela não tinha conseguido.
- Aos meus pais, irmã e avó por tudo aquilo que até hoje eu consegui.

Resumo

Qualidade de Serviço em Sistemas de *Data Warehousing*

Na sociedade actual, as organizações apostam cada vez mais em manter os seus dados em sistemas de informação centralizados, que permitam uma gestão cuidadosa e um acesso facilitado a toda a informação, a qual deverá estar sempre disponível para a análise em várias perspectivas. No dia-a-dia de uma organização são tomadas decisões, que muitas vezes influenciam a sua camada de negócio. Ora, decisões tão complexas e com elevado factor de risco encontram, normalmente, nos sistemas de *data warehousing* a fundamentação necessária ao processo de tomada de decisão. Estes sistemas, que nem sempre são de fácil manuseamento requerem acima de tudo mão de obra especializada para a sua administração, de forma a que o rigor assim como os níveis de desempenho e a qualidade de resposta sejam, obviamente, o mais elevados possível. O meio de interacção mais comum entre os sistemas de *data warehousing* e os seus utilizadores é através da utilização de *queries* SQL. Estas *queries* podem surgir de diversos locais, como resultado dos processos de navegação em relatórios, como provenientes do refrescamento de cubos OLAP ou até mesmo de eventuais *queries ad-hoc*. Independentemente da sua origem, os utilizadores que as lançam, directa ou indirectamente, requerem tempos de resposta bastante curtos providos de constante elevada qualidade de serviço. De facto, o grande meio de avaliação destes sistemas é efectuado a partir da percepção que os utilizadores têm do desempenho das suas *queries*. Desta forma, avaliar a qualidade de serviço de um sistema de *data warehousing* e consequentemente o desempenho das *queries*, apresenta-se como um excelente meio de estudar os factores, que hipoteticamente, possam estar a interferir na qualidade de um *data warehouse.* Nesse sentido, a presente dissertação de mestrado apresenta um sistema, não intrusivo, que faz o

estudo do desempenho das *queries* submetidas a um sistema de *data warehousing,* com o objectivo calcular um índice que represente a sua qualidade de serviço.

Palavras Chave: *Data Warehouse, Data Webhouse, Profiler, Outliers, Clustering,* Regressão Linear, Modelação Dimensional, Índice de Qualidade de Serviço, Métricas para a Qualidade de Serviço, Qualidade de Serviço, Classes de QoS, Extracção, Transformação, Integração, Mineração de Dados, Previsão Numérica, Desempenho de *Queries*.

Abstract

Quality of Service in *Data Warehouse* Systems

In today's society, organizations are increasing their focus on keeping their data in centralized information systems that allow a careful management and an easy access to all information, which must always be available for analysis on various perspectives. Decisions are made every day in an organization, which usually influences the business layer. Such complex and high risk decisions usually find the necessary foundation for decision making within the data warehousing systems. These systems can be of hard usability, and need specialized personnel for its administration, so that the accuracy, the levels of performance and the answer quality are the best possible. The most used method of interaction between data warehousing systems and its users are the SQL queries. These queries can have various sources, for example a result of a report navigation processes, the refreshing of OLAP cubes or even eventual ad-hoc queries. Regardless of its origin, the users that directly or indirectly throw these queries require short waiting time for their answers and a great quality of service. In fact, the biggest method of evaluation of these systems is done through the perception that users have of the performance of their queries. Evaluating the quality of service of a data warehousing system, and consequently the queries performance, appears as an excellent way of studying all factors that may interfere in a data warehouse's quality. With this in mind, this thesis presents a non intrusive system that studies the performance of the queries submitted to a data warehousing system, with the objective of calculating a index that represents the quality of service.

Keywords: Data Warehouse, Data Webhouse, Profiler, Outliers, Clustering, Linear Regression, Dimensional Modelling, Quality of Service Index, Quality of Service Metrics, Quality of Service, QoS Classes, Extraction, Transformation, Integration, Data Mining, Numerical Prediction, *Query* Performance.

Índice

Índice de Figuras

Índice de Tabelas

Lista de Siglas e Acrónimos

CPU - Central Processing Unit

OLTP - Online Transaction Processing

BI - Business Intelligence

OLAP - Online Analytical Processing

SGBD - Sistema de Gestão de Base de Dados

ETI - Extracção Transformação e Integração

DW - Data Warehouse

SDW - Sistema de Data Warehousing

Capítulo 1

Introdução

1.1 Em Prol da Qualidade

Na sociedade actual os padrões de comportamento da população mudaram. Em tempos não muito remotos, as pessoas mantinham frequentemente uma atitude passiva perante os mercados e acomodavam-se com os serviços que estes lhes prestavam. Contudo, com o aumento (galopante) do custo de vida, o comportamento geral da população alterou. De uma forma geral, a passividade deixou de ser o comportamento mais comum, passando a verificar-se uma atitude mais agressiva que não se contenta com aquilo que lhe é imediatamente fornecido. Na realidade, pretendem um serviço de qualidade e de acordo com aquilo que estão a pagar. É, então perfeitamente entendível, o porquê da competição actual entre organizações. Uma vez que a sua saúde económica é suportada pela confiança que os clientes depositam nos seus produtos, estas tendem cada vez mais a aumentar o nível de qualidade dos seus serviços.

No momento actual, preços competitivos e qualidade elevada são factores preponderantes, mas, no entanto, não suficientes para assegurar o sucesso comercial. A agilidade de processos e a facilidade com que as companhias se adaptam a uma nova tendência, tem sido, cada vez mais alvo de preocupação e interesse, marcando a evolução industrial ao longo dos últimos anos (Figura 1.1). Esta rápida sucessão de custo, qualidade, velocidade de entrega e agilidade como

imperativas estratégicas, reflecte a pesquisa continua por capacidades diferenciadoras que forneçam às organizações vantagens competitivas nos mercados actuais.

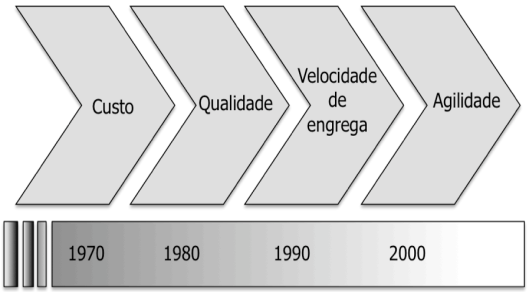

Figura 1.1: Evolução das prioridades competitivas

Com o esforço das organizações para cumprir os desafios de rápida e flexível resposta ao consumidor, novos recursos organizacionais estão a ser envolvidos. A informação deve agora ser expandida a toda a rede da organização, fornecendo oportunidades sem precedentes de construir novos sistemas de recolha de conhecimento de forma a responderem em tempo real às necessidades dos gestores e, assim, adequarem as suas formas de produção aos mercados que se avizinham. Um dos activos mais importantes de uma organização é a sua informação e a forma como é gerida. Geralmente, a informação relevante ao negócio é mantida sobre duas formas: em sistemas operacionais ou em sistemas de *data warehousing* (SDW).

Os sistemas operacionais, são o motor de uma organização nos quais toda a informação transaccional é guardada. Estes sistemas processam encomendas, mantêm o inventário, guardam informação relativa a clientes, registam pagamentos e reclamações, entre outras operações, dependendo daquilo para o qual foram idealizados. Sem estes sistemas, nenhuma companhia moderna consegue sobreviver. Começaram a ser construídos pelas organizações nos inícios dos anos 60 do século passado e neste momento fazem parte do seu dia-a-dia.

Na década de 90 do século passado, a economia e as formas de negócio tornaram-se mais complexas. A globalização tornou-se num marco diário na vida das organizações e a competição empresarial conheceu dias de intensa luta, provocando nos gestores de negócio, uma vontade exacerbada pela busca de informação, que permita a tomada de decisões estratégicas de forma a manterem-se no meio competitivo. Dia após dia, no desenvolvimento das suas tarefas quotidianas, estas organizações vão gerando grandes volumes de informação que, de uma forma ou de outra,

vão sendo armazenadas nos repositórios de dados dos seus sistemas operacionais, muitas vezes com uma organização algo deficitária e precária. Desta forma, o desenvolvimento de processos de exploração e utilização dos dados, contidos nas fontes de informação das organizações, vão sendo cada vez mais de difícil execução, provocando, a curto prazo, a saturação do sistema e dos seus próprios utilizadores. Por outro lado, a incapacidade dos sistemas transaccionais em fornecerem informação adequada à tomada de decisão – informação estratégica – associada à diversidade e heterogeneidade das fontes de dados, tornam muitas vezes os resultados vindos destes sistemas demasiado inconsistentes. Este tipo de inconsistências podem ser atenuadas se a informação relevante aos processos de tomada de decisão se conciliar num sistema único e homogéneo, chamado de SDW.

Em termos gerais, um SDW (Figura 1.2) caracteriza-se por um conjunto de processos de extracção, transformação e Integração (*ETI*) e tecnologias que possibilitam a um leque de utilizadores, desde responsáveis pelas actividades de gestão até analistas e directores de marketing, extrair conhecimento (através da análise de dados consistentes), de forma a proporcionar melhores e mais rápidas decisões. É, precisamente perante este objectivo, que um SDW requer níveis de serviço elevados. Grande parte das decisões de negócio das companhias modernas têm como base a informação retornada por estes sistemas. Os utilizadores procuram, de forma rápida e fiável, saber indicadores de extrema importância para as suas actividades de gestão, isto é, exigem uma resposta imediata ás questões que colocam ao sistema. Ora, quando destas questões depende largamente a saúde económica da companhia, os sistemas que apoiam estas decisões devem conseguir responder às necessidades da forma mais adequada possível.

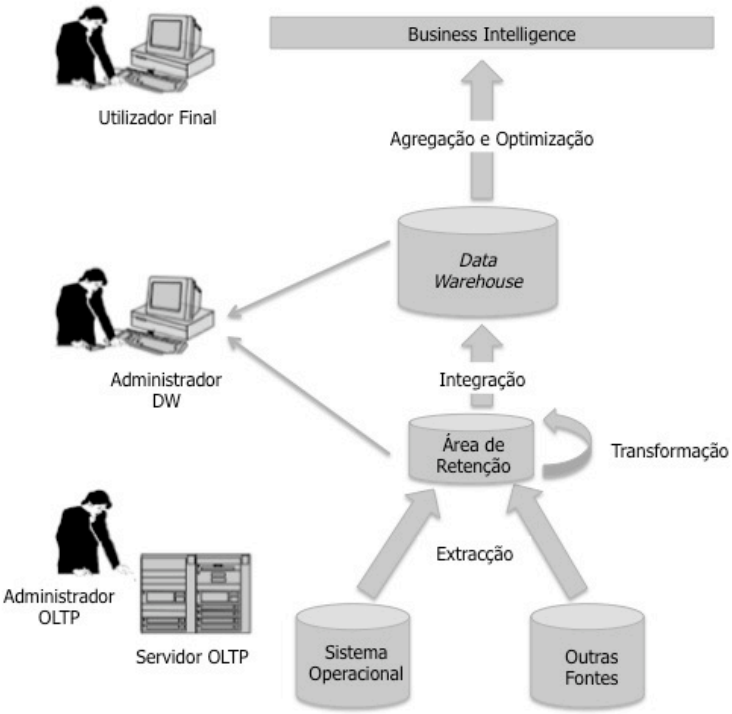

Figura 1.2: Arquitectura genérica para um sistema de *data warehousing*

Contrariamente aos sistemas operacionais, os SDW caracterizam-se essencialmente pela forma como a informação contida neles é manipulada. São construídos de forma a permitirem análises multidimensionais e diminuírem os tempos de resposta de *queries* extremamente complexas, envolvendo acções computacionalmente pesadas (cláusulas *group by*, *order by*) sobre grandes volumes de dados.

Grande parte das questões de negócio podem ser expressas facilmente através da linguagem SQL (*Structured Query Language)*. Qualquer pessoa familiarizada com a sintaxe SQL pode construir interrogações, que, facilmente, retornam as vendas de um produto para um determinado ano e semana específica. Contudo, muitas outras interrogações não são tão facilmente materializadas. *Queries* que requerem comparações, podem ser muitas vezes um enorme desafio para programadores assim como para a própria linguagem SQL. Uma simples questão que compare valores mensais, semanais, trimestrais e até mesmo anuais é das interrogações mais comuns durante o processo de análise, no entanto expressar estas questões sobre a forma de uma *query* representa um tremendo desafio, tanto para o programador como para a linguagem.

Um dos tópicos frequentemente associados à má prestação ou falta de qualidade dos serviços, é a demora no tempo de resposta aos serviços solicitados, que vai, para além daquilo que os utilizadores concebem como razoável. No entanto, os utilizadores não podem ser culpados por esta atitude. Pretendem uma resposta rápida e que nunca ultrapasse os limites do tolerável. Quando, situações destas acontecem, a primeira questão que se coloca está directamente associada à qualidade do algoritmo que sustenta a *query*. Contudo nem sempre algoritmos pobres e não optimizados estão na origem dos problemas levantados por utilizadores, quase sempre exigentes. Nestas situações, por exemplo, um pico de utilização de um servidor, ou até mesmo a sobrecarga da rede, podem estar na origem dos elevados tempos de resposta, que se traduzem em qualidade de serviço deficiente.

Uma forma de fazer o rastreio e controlo da qualidade de serviço de um SDW, pode ser através monitorização das *queries* enviados ao sistema. Dado que as *queries* são a forma mais utilizada para interagir com o sistema, a monitorização dos indicadores de desempenho destas, retornam um conjunto de parâmetros que, quando relacionados entre si, favorecem a criação de um índice (tal como o índice bolsista) que reflecte a qualidade da resposta do servidor do *data warehouse* (DW), indicando assim a qualidade de serviço do sistema.

1.2 Motivação e Objectivos

Um SDW assume-se, nos dias de hoje, como um factor fundamental nas actividades de gestão das organizações modernas. Sendo assim, tornou-se necessário o desenvolvimento de um repositório de dados que estivesse optimizado para processos de interrogação mais efectivos, independente dos sistemas operacionais, mas que destes obtivesse a sua informação. Como sistema de apoio à decisão um DW, deve constantemente fornecer altos níveis de qualidade de serviço. A coerência, a actualidade, a precisão, a acessibilidade, a disponibilidade e o desempenho são, entre outros factores, os mais requeridos pelos utilizadores finais de um sistema de apoia à decisão. Contudo, devido à não homogeneidade do tipo de utilizadores de um DW os factores de qualidade para cada um deles apresentam-se tão díspares quanto as suas funções (dos utilizadores). Se para um qualquer utilizador *A* precisão é o principal factor de qualidade, para um outro qualquer utilizador *B* a coerência da informação poderá ser o mais relevante. Neste sentido, avaliar o desempenho de um SDW significa, numa primeira instância, avaliar as necessidades dos seus utilizadores. Contudo, a rapidez na resposta às suas *queries* apresenta-se sempre como um factor principal, comum às

necessidades diárias dos utilizadores. Qualquer utilizador, independentemente da sua função na organização, exige que as respostas às suas interrogações sejam o mais breve quanto possível. Assim sendo, avaliar o desempenho de *queries* significa avaliar a qualidade de serviço do servidor que suporta o DW em relação aos pedidos que lhe são endereçados. Desta forma, pretende-se com este trabalho de mestrado o estudo de uma sistema não intrusivo que calcule um índice de desempenho de um SDW.

Temos, então, como objectivos da presente dissertação de mestrado o estudo, análise, especificação, modelação e implementação de um sistema de cálculo para a qualidade de serviço em SDW, mais concretamente em:

1. Apresentar e dissertar sobre os principais factores que influenciam o desempenho de um SDW.
2. Definir um conjunto de métricas avaliadoras da desempenho de um SDW.
3. Elaborar um sistema multidimensional para o armazenamento das métricas definidas de forma a permitir análises multidimensionais.
4. Definir e interpretar modelos de mineração de dados como meio activo para o cálculo de uma equação de regressão que resulte num índice de qualidade de serviço.
5. Desenvolver um protótipo de um sistema para a análise e acompanhamento da evolução da qualidade de serviço que demonstre as técnicas e conceitos apresentados anteriormente.

Não é, contudo, objectivo deste trabalho de mestrado a apresentação de um documento que sirva de guia passo-a-passo para a implementação de um sistema de cálculo para a qualidade de serviço de um SDW. Nesse sentido, pretende-se que este trabalho seja uma primeira abordagem ao estudo da qualidade de serviço neste tipo de sistemas. No entanto o sucesso desse estudo estará sempre dependente do tipo de sistema a analisar.

1.3 Estrutura da Dissertação

Alem do presente capítulo, onde se inclui uma pequena introdução sobre o estudo da qualidade de serviço em SDW, a presente documento de dissertação apresenta-se organizado em mais cinco capítulos, nomeadamente:

- O segundo capítulo faz uma breve abordagem aos SDW, no âmbito dos processos de tomada de decisão, bem como aos seus principais objectivos e componentes. A sua ultima secção enumera os determinantes essenciais no desempenho de *queries* em SDW.
- O capítulo três apresenta o sistema projectado e desenvolvido para fazer a avaliação da qualidade de serviço de um SDW, assim como a arquitectura que suporta o cálculo do índice de qualidade de serviço (QoS). É feita uma descrição do modelo dimensional que suporta o armazenamento de todos os parâmetros monitorizados, assim como as métricas usadas para o cálculo do referido índice. No final do capítulo são descritos os mecanismos de extracção, transformação e integração de dados desenvolvidos para o sistema em questão.
- O capítulo quatro aborda de uma forma genérica o conceito de mineração de dados. Aqui são apresentadas as diversas classes de QoS e realizada uma análise dos dados disponíveis, baseada em *clusters,* no âmbito do cálculo da qualidade de serviço. Por fim, são abordados os métodos de previsão numérica e suas principais características, como meio de cálculo de uma equação de regressão que retorne um índice de QoS.
- No quinto capítulo é feita a validação do sistema desenvolvido e do seu modelo de suporte sobre um sistema real. São também apresentados e discutidos os resultados de todos os processos implementados, terminando-se com uma avaliação critica ao modelo construído.
- Finalmente, no capítulo seis, é efectuada uma súmula das principais vantagens e dificuldades no processo de desenvolvimento de sistemas de controlo de qualidade e previsão de desempenho, sendo retiradas e discutidas as conclusões deste trabalho e indicando alguns dos principais trabalhos que podem vir a ser desenvolvidos num futuro próximo, no âmbito do cálculo de qualidade de serviço.

Capítulo 2

Sistemas de *Data Warehousing*

2.1 Objectivos de um Sistema *Data Warehousing*

Compreender e fundamentar a necessidade de um SDW, exige um conhecimento profundo sobre o tema quando se pretende abordar as principais vantagens que levem à sua implementação. Perceber as questões dos utilizadores é um passo fundamental para a justificação da implementação de um sistema deste tipo. Em [Kimball and Ross 02], Ralph Kimball, usa precisamente as necessidades diárias dos gestores de negócio de forma a enumerar os principais objectivos de um DW, dos quais resultam:

- **Um DW deve manter a informação de uma organização sempre consistente:** acima de tudo um DW deve ser um sistema credível, em que os seus utilizadores possam confiar; toda a informação em si contida deve ser cuidadosamente recolhida e manipulada a partir de todas as fontes; deve ser um sistema livre de ambiguidades de modo que a informação nele contida esteja actualizada de acordo com o definido e consistente.
- **Um DW deve servir como principal fundamento à tomada de decisão**: um DW quando construído, deve, por obrigação ser a principal fonte no processo de tomada de decisão; assim, o processo de suporte à decisão deve encontrar no sistema toda a informação necessária que fundamenta a escolha resultante.
- **Um DW deve manter a informação facilmente acessível:** num DW, acessibilidade não deve ser entendida apenas como facilidade de acesso à informação, mas também

como interpretabilidade e legibilidade desta; como tal toda a informação deve ser intuitiva, não só para o administrador do sistema mas também para os seus utilizadores.

- **Um DW deve ser adaptativo e não um entrave à mudança:** as necessidades dos utilizadores, condicionantes de negócio, informação e tecnologia são entidades extremamente expostas à mudança, sendo assim, um DW deve ser pensado de forma a se adaptar a estas alterações; no entanto, não deve tornar obsoleta toda a informação e tecnologia já existentes em alturas de mudança.
- **Um DW deve ser um sistema seguro que proteja a informação que nele se encontra:** a informação é um dos principais activos de uma organização, como tal, quando guardada num DW, esta deve ser mantida o mais segura possível; uma percentagem dos dados contidos no sistema são informação potencialmente útil, que quando acedida de forma indevida poderá revelar informação crítica sobre o negócio e suas formas, assim como informação acerca de clientes que deve ser privada.

2.2 Componentes de um Sistema de *Data Warehousing*

A arquitectura de um SDW assenta essencialmente num sistema de base de dados, um DW, que funciona como repositório central de dados. Este repositório central de informação é complementado por uma série de componentes fundamentais destinados a tornar o ambiente do sistema mais acessível, flexível e funcional, quer por parte dos utilizadores, quer por parte das fontes de dados na altura do seu povoamento.

De uma forma geral os componentes de um SDW podem ser agrupados em cinco conjuntos: fontes de dados, processos ETI, ferramentas de *Business Intelligence*, DW e área de retenção, podendo ser acrescido de mais um ou outro módulo dependendo dos objectivos para o qual se destina o sistema.

As fontes de dados podem ser dividas em dois grandes grupos: internas e externas à organização. As fontes internas são usualmente os sistemas operacionais, que registam as transacções do negócio, assim como outros sistemas de armazenamento contendo informação relevante. Por outro lado as fontes externas representam a informação vinda do exterior, como por exemplo, informação sobre clima ou até mesmo informação sobre a banca.

Os processos de extracção, transformação e integração são um dos componentes mais importantes num SDW. Este conjunto de processos são responsáveis pelas tarefas mais delicadas em sistemas deste tipo. Um sistema de ETI bem estruturado deve garantir, para além, da correcta extracção e consolidação dos dados a consistência e conformidade da informação. Apesar de bastante complexos, a missão destes processos é facilmente compreendida por quase todos os utilizadores: extrair de todas as fontes de dados a informação necessária para a área de retenção (extracção); uma vez extraídos, os dados devem ser limpos de forma a corrigir quaisquer ambiguidades que possam levar a inconsistências, assim como a execução de outras operações elementares (transformação); por fim os dados devem ser carregados para o DW (integração) [Kimball and Caserta 04].

As ferramentas de *Business Intelligence* são um conjunto de tecnologias que permitem, interrogar, analisar e apresentar a informação necessária ao processo de tomada de decisões. Um conjunto mínimo destas ferramentas deve consistir numa ferramenta de execução de *queries*, folhas de cálculo, ferramenta de *reporting* e, frequentemente, um servidor OLAP para análises multidimensionais.

A mais importante estrutura de um sistema de apoio à decisão, o DW, foi definida por William Inmon em 1990 como sendo uma base de dados orientada ao assunto, não volátil, contendo informação histórica de suporte ao processo de tomada de decisão [Inmon 05]. O seu princípio de modelação segue a ideia dimensional, ligeiramente diferente do conceito de modelação entidade-relacionamento (ER), uma vez que a desnormalização de dados é uma prática comum num DW.

Contrariamente às bases de dados operacionais (OLTP), os DW caracterizam-se pela forma como a informação neles contida é manipulada. De facto, as interrogações a estas bases de dados dimensionais caracterizam-se pela sua complexidade, assim como pelo seu tipo. Geralmente, o objecto de trabalho destes sistemas são operações de leitura sobre enormes volumes de dados, contendo funções de agregações, de forma a proporcionar análises de dados sobre várias vertentes. A Tabela 2.1 apresenta as principais diferenças entre os sistemas.

Sistemas OLTP	SDW
• Pequenas transacções • Dimensões na ordem do MB-TB • Operações de actualização • Dados elementares • Utilizadores de aplicações operacionais • Dados que reflectem o estado presente	• Interrogações complexas e longas • Dimensões na ordem do GB-TB • Operações de leitura • Dados agregados ou consolidados • Agentes de decisão e analistas • Dados históricos

Tabela 2.1: Sistemas OLTP vs SDW

A área de retenção (Tabela 2.2) deve ser vista como um local estacionário de dados no fluxo de informação. Este fluxo, de origem nas fontes de dados até ao DW, pode ser entendido de forma análoga à cozinha de um restaurante onde todos os ingredientes são recolhidos, preparados e transformados na ementa final. Em rigor, a área de retenção deve ser o local onde, pela primeira vez toda a informação a ser consolidada no DW se reúne e sobre a qual um conjunto de processos de transformação processa os dados de forma a serem correctamente integrados [Kimball and Caserta 04]. Como tal, e devido aos complexos processos que actuam nesta área, esta deve ser acedida apenas por profissionais especializados.

Armazena	Processamento	
• *Flat Files* • Dados de sistemas OLTP • Dados externos • Outros	• Limpa • Combina • Actualiza • Remove duplicados • Arquiva • Agrega • Exporta para o DW • Outros	**Manipulada por profissionais especializados**

Tabela 2.2: Principais acções a realizar na área de retenção durante o fluxo de povoamento

2.3 Factores Determinantes no Desempenho de *Queries* em Sistemas de *Data Warehousing*

O desempenho de uma SDW, nem sempre é uma questão de fácil abordagem. Desde a qualidade e a quantidade dos dados armazenados, até à própria arquitectura do sistema, passando pelo esquema lógico, entre outros, são factores que influenciam largamente o desempenho e a qualidade de serviço do sistema quando confrontado com *queries* complexas. O aumento dramático do volume de informação ao longo do tempo, misturado com dados desactualizados ou até mesmo dados demográficos de suporte a actividades de previsão, podem causar num curto espaço de tempo a degradação de *queries* mais simples, e, a um nível superior, alguns estrangulamentos no processamento de pedidos. Desta forma o sistema diminui a sua qualidade de resposta e consequentemente o seu desempenho. Assim, compreender algumas técnicas de melhoramento de desempenho e refinamento em SDW pode ser uma preciosa ajuda na sua administração com vista a conseguir o desempenho desejado.

Quando são pensadas alterações de modo a melhorar o desempenho de um sistema, faz todo sentido em antecipadamente recolher os factores que mais influenciam o seu desempenho. Frequentemente, o estudo dos factores determinantes no desempenho do sistema, permitem maximizar o retorno (melhor desempenho de *queries*), minimizando o esforço, isto quando bem conceptualizados. Esta abordagem segue o principio de *Vilfredo Pareto* (regra dos 80 - 20), dizendo que 80% dos benefícios advêm de 20% do esforço [Vilfredo and Schwier 71]. Desta forma, o conhecimento sobre os principais factores responsáveis pela oscilação de desempenho, apresenta-se como uma mais valia nas actividades de melhoramento de desempenho em SDW. De forma a compreender melhor os factores, que directa ou indirectamente, influenciam o desempenho de um SDW, é apresentada nas secções seguintes a súmula dos factores mais importantes em questões de desempenho em base de dados.

2.3.1 Vistas Materializadas

A noção de vista materializada é um conceito essencial em bases de dados [Ullman 88] e tem captado cada vez mais a atenção dos seus peritos com o aumento de popularidade dos DW, precisamente devido às enormes potencialidades que uma vista materializada pode trazer em termos de desempenho. Uma vista materializada, resulta da materialização de uma *query* [Chaudhuri et al 95] numa tabela física, isto é, coloca numa tabela o resultado de uma *query*, de forma a transformar uma *query* com tempo de processamento elevado numa simples selecção à tabela resultante da materialização. Se idealizarmos o cenário de uma *query* que calcula o volume de vendas por produto, e acede a tabelas como *item* e *orders*, é fácil concluir que o desempenho desta *query* está seriamente comprometido pelo tamanho das tabelas a que esta acede. Desta forma, uma vista materializada pode reduzir consideravelmente o tempo de resposta da interrogação.

O seguinte exemplo ilustra a criação de uma vista materializada, usando-se a linguagem PL\SQL do Oracle [Urman and Rinaldi 98]:

```
create materialized view vendorOutstanding
build immediate
refresh complete
enable query rewrite
as
      select item.description, sum(orders.quantity*item.price)
      from orders, item
      where orders.itemnum = item.itemnum
      group by item.description;
```

Para que se tenha algum tipo de vantagens na utilização desta técnica, é importante considerar alguns aspectos relacionados com a manutenção [Gupta et al 95] de vistas. Dado que a materialização é feita a partir de dados provenientes das tabelas originais, o que acontece se estas forem alteradas? É aqui que surge o factor mais importante a considerar. Existe um compromisso entre a rapidez de processamento das *queries* e o custo de manutenção das vistas materializadas. Cada vez que existe alguma alteração em alguma tabela usada pela *query*, e cujo resultado é materializado, é necessário propagar essa alteração a todas as suas dependências. Caso contrário

um problema de inconsistência de dados acabaria por surgir. Desta forma chega-se facilmente à conclusão que um grande número de vistas materializadas poderá não ser tão benéfico como seria de esperar, caso o tempo de manutenção ultrapasse o ganho em termos de tempo de processamento. Este facto poderá ser agravado com a possibilidade de existir dependência entre vistas materializadas.

2.3.2 Modelação Dimensional

Na comunidade de profissionais de bases de dados, e consequentemente no grupo de pessoas que diariamente lida com sistemas de suporte à decisão, a preponderância na modelação lógica de qualquer base de dados afirma-se com um factor de enorme importância para o seu bom desempenho. Na realidade a criação de um bom modelo de dados é uma forma proactiva de assegurar um bom desempenho inicial do sistema. Se imaginarmos um modelo lógico demasiado complexo, é fácil concluir que a sua complexidade será reflectida na complexidade da *query* e consequentemente no seu desempenho.

Nos SDW, aquando da construção do modelo dimensional, alguns factores devem ser considerados de forma a permitir um esquema saudável sem grandes problemas de complexidade. Os seguintes factores, apresentam-se como questões que devem ser estudados de forma a assegurar um bom desempenho:

- **Granularidade.** Este factor deve ser visto como um dos principais assuntos a ser considerado antes da construção do sistema. Dado que um DW faz parte de um sistema de apoio à decisão, o nível de detalhe não precisa ser tão refinado como nos sistemas OLTP. Por exemplo, questões como "Será vantajoso um relatório com nível de detalhe até à chave ou basta até ao departamento?" devem ser colocadas aos utilizadores de negócio de forma a definir os seus requisitos. Um DW não precisa de dados até ao nível do documento em vendas ou encomendas como nos sistemas OLTP. Nesses casos os dados devem ser agregados antes de serem carregados pelo sistema. Definir a informação estritamente necessária pelos utilizadores é determinante para um bom desempenho no futuro.
- **Cardinalidade.** Cardinalidade significa o número de possíveis entradas numa tabela. Facilmente se percebe que em tabelas com elevada cardinalidade o tempo de resposta de

uma *query* aumenta consideravelmente. Fazendo um estudo a esse nível, podemos definir um índice sobre a tabela de forma a diminuir o tempo de pesquisa e consequentemente melhorando o desempenho na resposta.

- **Modelação Dimensional.** Apesar de discordarem em alguns aspectos, a modelação dimensional é a abordagem mais seguida pelos principais autores e investigadores na área. Os esquemas seguem, essencialmente, duas vertentes principais: esquema em estrela ou esquema em floco de neve. Como seria de esperar as duas vertentes possuem um conjunto de prós e de contras, que consoante o caso prático podem influenciar mais ou menos a seu desempenho. Na realidade, ambos os esquemas, como seria de esperar, estão preparados para acolherem enormes volumes de informação Contudo, apesar de permitirem menos especificidade e provocarem maior redundância de dados, os esquemas em estrela, favorecem a rapidez de execução de *queries* [Kimball and Ross 02]. A Tabela 2.3 apresenta, comparativamente, os prós e contras de ambos os esquemas, e como podemos verificar, o esquema em estrela dá mais garantias em termos de desempenho. No entanto, dependendo do problema esta avaliação não deve ser fixa.

	Esquema em Floco de Neve	Esquema em Estrela
Tipo de DW	Bom para DW ou *Datamarts* pequenos	Bom para grandes DW
Normalização	Terceira Forma Normal	Desnormalizado
Facilidade de interrogação	Consultas mais complexas, consequentemente menos fáceis de entender	*Queries* menos complexos, consequentemente mais fáceis de entender
Manutenção	Menor redundância, torna-se mais simples de manter	Maior redundância de dados, torna-se mais difícil de manter
Query performance	Mais chaves estrangeiras, consequentemente mais tempo de processamento	Menos chaves estrangeiras, consequentemente menos tempo de processamento

Tabela 2.3: Esquema em Floco de Neve vs. Esquema em Estrela

2.3.3 Particionamento

O particionamento de tabelas é um importante aspecto no processo de construção de um DW, principalmente no que diz respeito a questões de desempenho. O processo de particionamento visa decompor grandes tabelas (tabelas de facto, vistas materializadas, dimensões, etc.) em tabelas mais pequenas e consequentemente facilitar o acesso às mesmas.

Dois tipos de particionamento são possíveis para decompor uma tabela: particionamento vertical e particionamento horizontal. O particionamento vertical [Navathe et al 84] (Figura 2.1) de uma tabela *T* fragmenta a tabela original em duas ou mais tabelas, chamados fragmentos verticais, em que cada uma contém um conjunto de colunas da tabela original *T*. Assim, as *queries* apenas necessitam de aceder a um pequeno conjunto de colunas reduzindo o volume de dados a ser percorrido para responder à *query*. De salientar que as chaves primárias são duplicadas em cada fragmento.

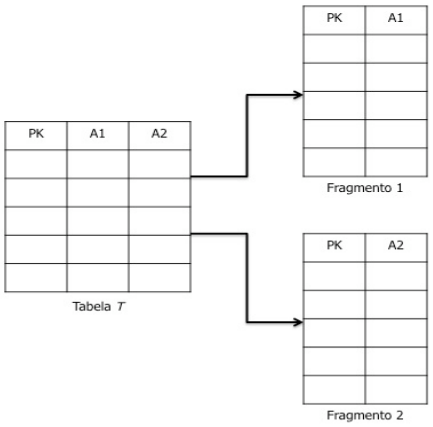

Figura 2.1: Exemplo de um particionamento vertical

Ao contrário, do particionamento vertical, o particionamento horizontal [Ceri et al 82] (Figura 2.2) de uma tabela *T* fragmenta em duas ou mais tabelas, chamados fragmentos horizontais, em que cada fragmento contem um conjunto de linhas da tabela original. Desta forma não há a necessidade de percorrer toda a tabela em pesquisas exaustivas para encontrar o registo necessário, para isso basta percorrer o fragmento que o contém.

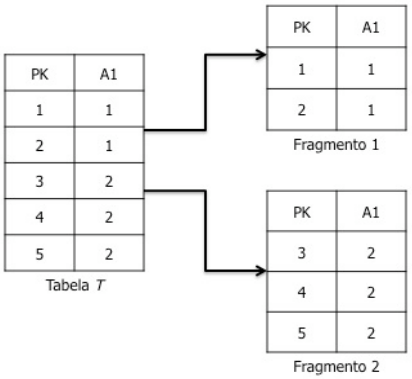

Figura 2.2: Exemplo de um particionamento horizontal

2.3.4 Índices Bitmap

Os índices bitmap são um dos métodos de indexação disponíveis mais eficientes para *queries* multidimensionais. A ideia é usar operações lógicas elementares sobre bits, que são melhor suportadas pelo *hardware* de servidores, para facilitar as operações de pesquisa nas tabelas. Para um dado atributo, com *c* valores distintos, um índice bitmap gera *c* bitmaps com *n* bits cada, onde *n* é o número de registos na tabela. Cada bit no bitmap é colocado a 1 se o atributo no registo é de um valor específico, caso contrario o bit é colocado a 0. A Figura 2.3 ilustra um índice bitmap com seis bitmaps, em que cada bitmap representa um valor para cada atributo.

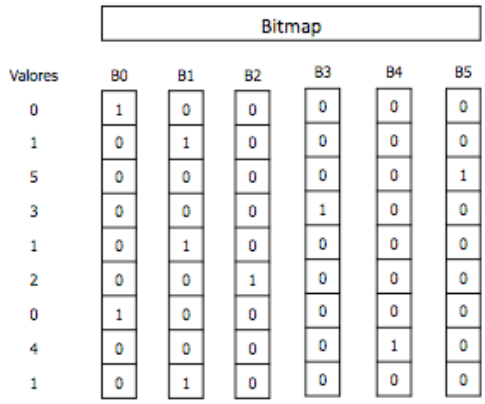

Figura 2.3: Exemplo de um índice bitmap

Dadas as características destes índices, as tabelas que raramente são alvo de *inserts* ou *updates*, são boas candidatas a ter os seus atributos indexados através de um índice bitmap, pois devido ao facto de não sofrem muitas alterações nos seus registos estas não obrigam a um refrescamento constante do índice. Também é perceptível que estes índices funcionam bem para atributos de baixa cardinalidade [Chan and Ioannidis 99], caso contrário o tempo de pesquisa no índice, devido ao aumento do espaço necessário para o guardar, aumentaria exponencialmente tornando assim os acessos menos eficientes. Contudo, existem na literatura actual vários mecanismos que visam colmatar o problema da eficiência do índice para atributos de alta cardinalidade, a saber:

- Encoding [Chan and Ioannidis 99] [Chan and Ioannidis 98] [Wu and Buchmann 98]
- Binning [Rotem et al 05] [Stockinger et al 04] [Koudas 00]
- Compression [Antoshenkov 04] [Johnson 99] [Wu el al 05]

2.3.5 Paralelismo

A ideia de computação em paralelo [Almasi and Gottlieb 88] [Kumar el al 94] resume-se à execução de várias operações em simultâneo, atendendo ao principio de que um grande problema pode ser dividido em pequenas partes a serem resolvidas concorrentemente. Desta forma o processamento em paralelo tem vindo a captar a atenção da comunidade de administradores de bases de dados, como forma de aumentar o desempenho na execução de pedidos [DeWitt and Gray 92] [Valduriez 93] [Graefe 93], isto é, diminuir o tempo de espera e aumentar a escalabilidade da *query*. Sistemas de multiprocessamento simétrico, sistemas *clustered*, entre outros, encontram nas técnicas de processamento em paralelo grandes benefícios em termos de eficiência de execução, visto que as operações podem ser fragmentadas pelos vários *CPUs* do sistema.

Com a popularidade das bases de dados e, consequentemente, com o aumento do volume de informação, os métodos de paralelismo em base de dados apresentam-se como uma interessante alternativa na busca por melhores tempos de execução e desempenho. O método de paralelismo *inter-query* resulta da possibilidade de os SGBD poderem responder a múltiplos pedidos concorrentemente, isto é, várias *queries* podem ser executadas simultaneamente dentro de uma única base de dados, desde que obedeçam às propriedades elementares das transacções (atomicidade, consistência, isolamento, durabilidade). As outras formas de paralelismo (*intra-*

query) são todas baseadas no paralelismo de operações algébricas (selecção, intersecção, união) sobre conjuntos de processamento [Mehta and DeWitt 95] [Rahm 93], dividindo-se em:

- ***Inter-operator:*** execução de diferentes operações de uma *query* em paralelo.
- ***Intra-operator:*** execução de uma única operação do plano de execução da *query* em múltiplos processos, ou seja, fragmentar a operação e executar concorrentemente os fragmentos da operação. É também chamado de paralelismo baseado na fragmentação ou particionamento, visto que o processamento se foca em conjuntos disjuntos de dados.

Apesar das óbvias melhorias no tempo de execução e na escalabilidade do sistema, ambas as técnicas de *intra-query* apresentam problemas que requerem alguma atenção, nomeadamente no que diz respeito ao balanceamento das operações (*inter-operator*) e ao balanceamento do volume de dados de cada partição (*intra-operator*).

2.3.6 Tecnologia de *Caching*

A tecnologia de *caching* [Elhardt and Bayer 84] é uma das tecnologias mais importantes na dualidade entre grandes volumes de dados e desempenho do sistema. A vantagem principal na utilização de uma cache consiste em evitar o acesso ao local de armazenamento primário, muitas vezes demorado, armazenando a informação previamente consultada, já sobre a forma de output, em dispositivos de acesso mais rápido.

Quando uma nova *query* é lançada ao sistema, este guarda o resultado na memória cache. Consequentemente, *queries* idênticas acedem directamente à memória cache em detrimento de acederem de novo à base de dados. Desta forma uma *query* que tinha um tempo de processamento elevado passa a ter uma resposta imediata. As técnicas são de tal forma importantes que grande parte das organizações reservam já blocos de memória especificamente orientados para a implementação de sistemas de *caching.* Contudo, assim como em qualquer outra técnica de melhoramento de desempenho, também as técnicas cache têm os seus contras. Neste caso, estes relacionam-se com o refrescamento da *query*, isto é, sempre que as tabelas acedidas são alteradas, os valores em cache devem ser refrescados de forma a traduzirem essas alterações.

2.3.7 Arquitecturas de Redes e Serviços

Muitos peritos em desempenho de SDW concordam que o esquema lógico do sistema é um dos determinantes mais importantes quando se trata de desempenho de *queries*. Contudo, mesmo um soberbo esquema lógico pode não colmatar os problemas que advêm de um servidor sobrecarregado ou uma infra-estrutura de rede constantemente limitada. Antes de se investigar as questões de desempenho do sistema, há que verificar a saúde da plataforma computacional. Uma verificação alto nível sobre algumas das questões que frequentemente degradam o desempenho computacional de qualquer sistema, pode, muitas vezes ser o ponto de partida para o melhoramento do desempenho do servidor.

Locais de estrangulamento em recursos primários, tais como discos físicos, CPU, entre outros, podem muitas vezes levar a excessos no tempo de processamento de uma *query*, como tal, a identificação destes problemas revela-se crucial para um bom desempenho computacional. Relativamente aos discos físicos, estes apresentam-se como uma questão essencial. De facto, a velocidade de acesso (leitura e escrita) e o tempo de output são importantes condicionantes na rapidez de execução de uma *query*, uma vez que grande parte das suas operações são efectivamente leituras e escritas.

A largura de banda é outro factor essencial no desempenho de uma *query.* Geralmente quando um SDW está implementado numa organização, este é acedido remotamente. Toda a interacção entre servidor e cliente é feita através de ligações intranet (internos à organização) ou internet (externos à organização). Enquanto no segundo caso, os problemas na ligação vão para além dos serviços internos da organização, no primeiro caso é sempre da responsabilidade desta um bom desempenho na estrutura de comunicações de dados. Como tal, a correcta optimização e configuração das infra-estruturas de redes apresenta-se como um determinante essencial para a fluidez e rapidez de resposta de um SDW. Em rigor, quanto maior for a largura de banda associada a uma estrutura de comunicações maior será a capacidade de resposta perante uma dada *query*.

Quando se instala um sistema, e qualquer que seja o seu objectivo, a sua configuração revela-se uma actividade de extrema importância para o bom funcionamento do serviço em causa. De facto, não só configurações de performance têm impacto no desempenho final do sistema, como também pequenas configurações para actualizações do sistema operativo ou até mesmo configurações de segurança podem impedir de alcançar o desempenho desejado. Se pensarmos que o sistema

operativo está configurado para fazer actualizações de *software* todas as terças-feiras ás 9h, podemos imaginar a degradação de desempenho provocada nesse mesmo período.

Conclui-se, então, que um desempenho deficitário pode resultar directamente do estrangulamento de um recurso de *hardware* ou rede, e não propriamente da ineficiência de uma dada *query*. Taxas de utilização de CPU constantemente elevadas, utilização quase completa de memória ou até mesmo baixos níveis de I/O, podem ser fortes indicadores à cerca da saúde deficitária do servidor. Assim, a verificação destes factores pode ser uma forma de rastreio de problemas de desempenho.

Capítulo 3

Um Primeiro Modelo para o Cálculo da Qualidade de Serviço

3.1 Arquitectura

A arquitectura genérica (Figura 3.1) para o cálculo da qualidade de serviço em SDW idealizada apresenta um sistema não intrusivo que de forma transparente monitoriza todas as *queries* enviadas ao sistema alvo, e, examina o seu comportamento de forma a calcular um índice para a qualidade dos seus serviços. A arquitectura do sistema é composta por quatro entidades principais, a saber:

1. Fontes de dados. São talvez a entidade mais fácil de compreender. Na arquitectura apresentada, são unicamente os dados monitorizados pela ferramenta de *profiling* e os indicadores de desempenho do sistema operativo (estes são recolhidos através de uma script, construída para tal).
2. O SDW alvo. Representa o sistema sobre o qual é estudada a qualidade de serviço.
3. O DW que armazena a informação sobre a qualidade de serviço. Este *DW* armazena toda a informação, de forma dimensional, necessária ao estudo sobre a qualidade de serviço. O uso de um DW como suporte ao cálculo da qualidade de serviço justifica-se com as potencialidades de análise que estes sistemas favorecem. Concretamente, permite relacionar a informação sobre a execução das *queries* em vários eixos de análise. Como mantém informação histórica, permite analisar o comportamento do sistema ao longo do tempo.

4. Os modelos de previsão. Executam toda a parte de mineração de dados necessária para o cálculo do índice de qualidade de serviço. As técnicas de mineração escolhidas (*Clustering* e regressão linear) foram aplicadas com base na metodologia CRISP-DM [1]

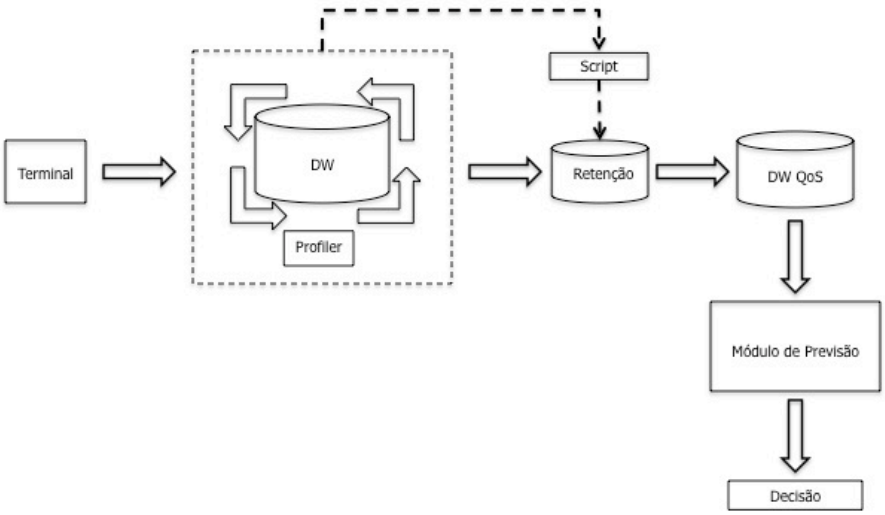

Figura 3.1: Arquitectura genérica de um sistema de cálculo de qualidade de serviço

Em termos gerais, o funcionamento do sistema pode ser visto como um fluxo de dados que é iniciado quando uma *query* é enviada ao DW alvo de estudo. A partir daí, qualquer execução de uma *query* é monitorizada pela ferramenta de *profiling* do SGBD. Paralelamente, um programa recolhe indicadores de desempenho do sistema operativo, no qual se encontra instalado o DW. Todos estes dados são consolidados num sistema intermédio para armazenamento temporário e manipulados de forma a serem integrados posteriormente num DW, construído para armazenar dimensionalmente a informação necessária. Por fim, com a aplicação de técnicas de mineração de dados (algoritmos de *clustering* e algoritmos de previsão), é obtido um índice de qualidade de serviço, que pretende traduzir a qualidade de resposta do servidor de DW às *queries* enviadas ao sistema.

3.2 Métricas e Modelo Dimensional para o Cálculo da Qualidade de Serviço

A modelação dimensional inerente a um sistema de previsão de qualidade de serviço é bastante semelhante à existente em qualquer outro projecto de um DW. Em termos gerais, o objectivo é definir um contexto válido, correcto e compreensível onde os factos a analisar façam sentido. Assim, como em qualquer outra implementação, é crucial definir correctamente a granularidade da tabela de factos, de forma a serem construídas todas as dimensões necessárias às analises pretendidas. Desta forma, a granularidade da tabela de factos poderá ser: o registo de cada *query* lançado a um conjunto de tabelas por um determinado utilizador num dado minuto. Com este nível de especificidade é possível ver respondidas questões como:

- Quais as horas ou dias de maior afluência de *queries*?
- Quais as *queries* que provocam estrangulamento do sistema?
- Quais as tabelas mais acedidas?

Na Figura 3.2 é apresentado o esquema genérico para uma tabela de factos idealizada para a análise de desempenho de um DW. O tipo de estudos conseguidos através do modelo apresentado, serão uma ferramenta especialmente útil para administradores e programadores do sistema alvo.

São cinco as dimensões existentes, complementadas com uma tabela ponte que sai do relacionamento *N:M* entre a dimensão *Tabela,* e a tabela de factos. Isto, porque numa *query* várias tabelas podem ser consultadas. Descrevemos, assim, quem (dimensão *Utilizador*) pediu o quê (dimensão *Query*), quando (dimensões *Tempo* e *Calendário*) a onde (dimensão *Tabela*).

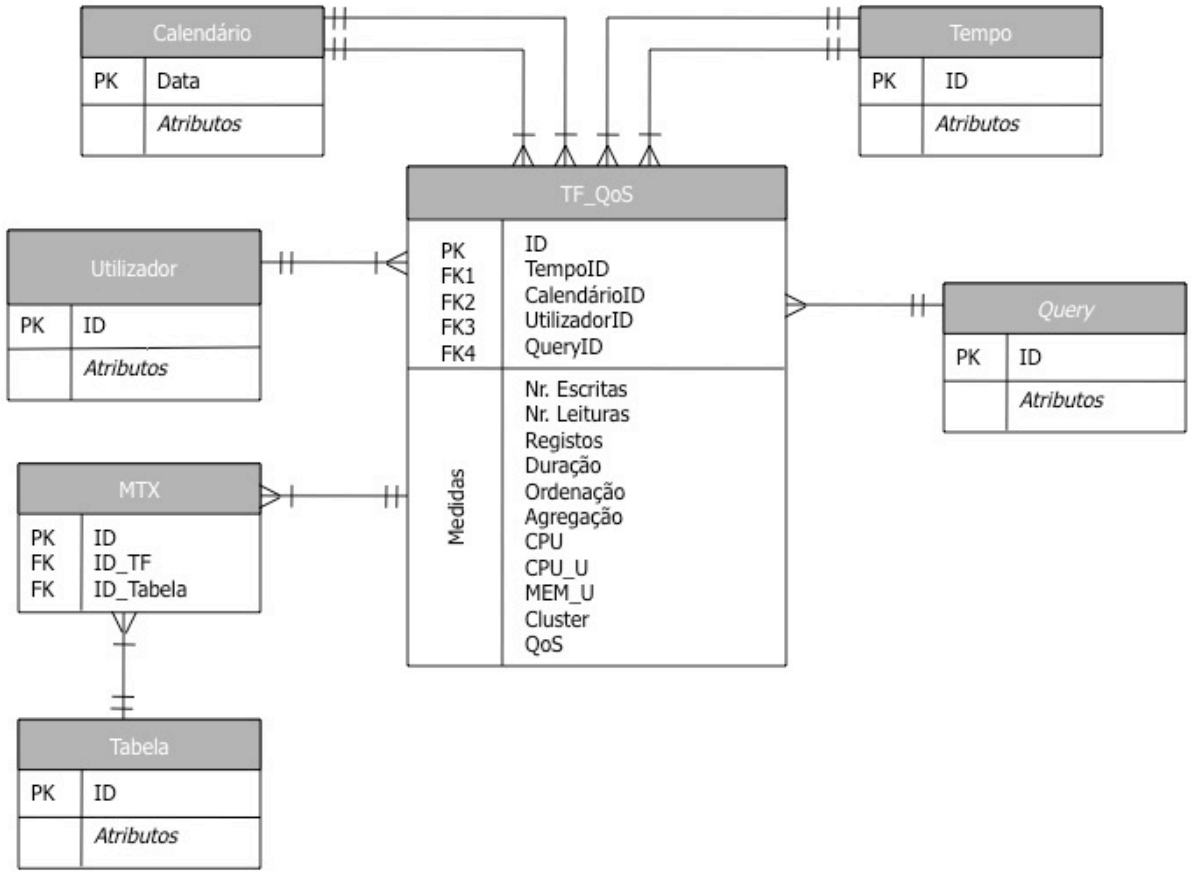

Figura 3.2: Esquema dimensional para análise da qualidade de serviço

A tabela de factos (Tabela 3.1) é composta, para além das habituais chaves (primária e estrangeiras), por onze medidas. Estas podem ser divididas em dois grupos: métricas para qualidade de serviço e atributos objectivo. As métricas para a qualidade de serviço são aquelas que caracterizam a execução da *query*, sendo consideradas indicadores de desempenho. Quanto aos atributos objectivo estes acolhem o resultado do módulo de previsão de desempenho relativamente à qualidade de serviço do sistema. Os valores destes atributos são calculados numa segunda fase do processamento, recorrendo a técnicas de mineração de dados (Secção 4.2 e Secção 4.3), e acolhem a materialização da qualidade de serviço em termos quantitativos.

Coloca-se neste momento uma questão essencial sobre o porquê da utilização destes indicadores como medidas de avaliação para qualidade de serviço. Como referido, estes são indicadores que essencialmente caracterizam a execução de uma *query*. Como tal, fazer a avaliação da qualidade de serviço em SDW, representa avaliar o desempenho das *queries* perante esses sistemas. Assim,

a escolha das métricas referidas, justifica-se pelo facto de elas se apresentarem como os principais factores que influenciam a qualidade de uma *query*.

O que é então uma *query* ideal no âmbito deste estudo? A resposta não se apresenta tão óbvia como seria esperado, dependendo, claramente, da natureza do sistema e do tipo de utilizadores. No entanto podemos definir uma *query* ideal como aquela que cumpre aquilo para o qual foi idealizada, no menor espaço de tempo possível e com o menor uso de recursos. A título de exemplo, não é coerente considerar uma *query* que devolve um milhão de registos e que efectuou dois milhões de leituras num espaço de três minutos como uma *query* ineficiente. Em rigor é preciso encontrar um ponto de equilíbrio entre o que é considerado mau ou bom mediante o problema em questão.

Medida	Descrição
ID	Valores delegados. Chave primária.
TempoID	Chave estrangeira para a dimensão Tempo.
CalendárioID	Chave estrangeira para a dimensão Calendário.
UtilizadorID	Chave estrangeira para a dimensão Utilizador.
QueryID	Chave estrangeira para a dimensão *Query*.
Nr. Escritas	Representa o número de acessos a disco para escrita por operações da *query*.
Nr. Leituras	Representa o número de acessos a disco para leitura por operações da *query*.
Registos	Representa o número de registo devolvidos pela *query*.
Duração	Duração de execução da *query*.
Ordenação	Indica se a *query* possui funções de ordenação.
Agregação	Indica se a *query* possui funções de agregação.
CPU	Tempo de CPU usado para a execução da *query*.
CPU_U	Carga de CPU actual a ser usada.
MEM_U	Memória actual a ser usada.
Cluster	Indica a que *cluster* pertence o registo na tabela de factos.
QoS	Representa o valor quantitativo da qualidade de serviço.

Tabela 3.1: Atributos da Tabela de Factos Qualidade de Serviço

As cinco dimensões representadas visam, como em qualquer outro DW, contextualizar cada facto na tabela de factos. Assim, a dimensão *Utilizador* (Tabela 3.2) caracteriza a entidade que enviou a *query* ao sistema. Esta entidade, é aquela que se autentica perante o sistema de gestão de base de dados.

Atributo	Descrição
ID	Valores delegados para a chave primária
Auth	Utilizador que se autentica no sistema. Exemplo "Pedro_Admin"

Tabela 3.2: Atributos da dimensão *Utilizador*

A dimensão *Query* (Tabela 3.3) faz a caracterização genérica do pedido (*query*) enviado ao sistema, isto é, identifica a *query* submetida, assim como o seu tipo (inserção, agregação, selecção, etc.)

Atributo	Descrição
ID	Valores delegados para a chave primária
Query	Texto integral da *query* enviada ao sistema. Exemplo "Select * From TF"
Tipo	Tipo da *query*. Exemplo "Select"

Tabela 3.3: Atributos da dimensão *Query*

De todas as dimensões existentes, as dimensões temporais, *Tempo* e *Calendário*, são talvez as mais óbvias e intuitivas, visto que qualquer evento dificilmente fará sentido se estiver desprovido de referências desta natureza. Devido ao funcionamento de um ambiente DW, é necessário manter informação ao minuto. Faz agora sentido, distinguir as referências horárias das referências temporais (datas), isto porque, se todos os dados fossem mantidos numa única tabela, toda a informação relativa a horários teria de ser repetida por cada dia, que, levaria ao aumento exponencial do espaço necessário ao armazenamento. Optou-se assim pelo uso de duas dimensões distintas.

A dimensão *Calendário* (Tabela 3.4), caracteriza o dia em que ocorreu o evento, isto é, reflecte informação acerca do ano, semestre, trimestre, mês, dia do mes, semana, dia da semana e dia do ano. Da mesma forma, a dimensão *Tempo* (Tabela 3.5), regista o momento do dia em que ocorreu o evento, guardando a informação relativa à hora, minuto e período do dia.

Por fim, a dimensão *Tabela* (Tabela 3.6), como o próprio nome indica, faz a descrição da tabela, ou tabelas, às quais a *query* acede. Representa a informação acerca do tipo da tabela, do seu grau e se possui ou não índices (os índices relativos ás chaves primárias não são contabilizados). Como

a estrutura de uma tabela pode ser alterada ao longo do tempo, esta dimensão é considerada uma dimensão de variação lenta, como tal, dois outros atributos são necessários, a saber:

- Estado, que indica se a tabela está activa ou não
- Apontador, que faz a referência para o novo registo da tabela

Atributo	Descrição
Data	Data do dia do registo. Chave primária.
Ano	Ano da data do registo. Exemplo "2009"
Semestre	Semestre da data do registo. Exemplo "2"
Trimestre	Trimestre da data do registo. Exemplo "1"
Mês	Mês da data do registo. "Janeiro"
Semana	Semana do ano da data do registo. Exemplo "3"
Dia da Semana	Dia da semana da data do registo. Exemplo "6"
Dia do Mês	Dia do mês da data do registo. Exemplo "23"
Dia do Ano	Dia do ano da data do registo. Exemplo "23"

Tabela 3.4: Atributos da dimensão *Calendário*

Atributo	Descrição
ID	Chave primária. Valores delegados.
Hora	Hora do registo. Exemplo "15".
Minuto	Minuto da hora do registo. Exemplo "32".
Período	Período do dia da hora do registo. Exemplo "Tarde".

Tabela 3.5: Atributos da dimensão *Tempo*

Atributo	Descrição
ID	Chave primária. Valores delegados.
Objecto ID	Identificador da tabela no DW. Exemplo "26".
Nome	Nome da tabela. Exemplo "DIM_Tempo".
Tipo	Tipo da Tabela. Exemplo "Dimensão".
Grau	Grau da tabela. Exemplo "12".
Índice	Indica se a tabela possui índices que não resultantes da chave primária. Exemplo "NAO"
Apontador	Apontador para o novo ID do registo. "45"
Estado	Indica se o registo é valido ou não. Exemplo "INV"

Tabela 3.6: Atributos da dimensão *Tabela*

3.3 Fontes de Dados para o Sistema

As fontes de dados de um DW dependem sempre do cariz da informação que se pretende analisar. Essencialmente, um DW é um sistema de "pergunta e resposta", onde as perguntas são questões de negócio e as respostas são fundamentos à decisão. Desta forma, o principal meio de interacção entre os utilizadores e o sistema é através da utilização de *queries* SQL. Consequentemente, a informação a colectar, quando estamos perante um sistema de análise de qualidade de serviço de um DW, é essencialmente aquela que representa directamente a interacção das *queries* submetidas com o sistema.

Os sistemas de gestão de base de dados mais credenciados do mercado[10], apresentam hoje em dia, um vasto leque de ferramentas que permitem, de entre muitas operações, estudar e analisar com rigor todas as acções ou eventos ocorridos numa base de dados. Uma das ferramentas mais utilizadas para a análise de *queries* são os *database profilers*. Estes módulos, permitem dissecar toda a execução de uma *query*, desde um simples tempo de processamento até à contabilização do número de acessos ao disco, passando pelo tempo de CPU usado por cada operação, entre outras acções. Essencialmente, registam grande parte da informação necessária para o povoamento das medidas na tabela de factos. Estas ferramentas, contudo, exigem um forte conhecimento técnico para uma correcta utilização.

Dada a possibilidade de se obter inúmeros parâmetros, qualitativos, sobre a execução das *queries* no sistema, os *database profilers*, apresentam-se como a principal fonte de dados do sistema. Contudo, nem toda a informação fornecida pelos *profilers* é útil, isto é, existem vários parâmetros monitorizados por estas ferramentas que não são de interesse imediato. Há, então que fazer uma cuidadosa selecção sobre os principais parâmetros a recolher. A Tabela 3.7, enumera os atributos fundamentais para o estudo da qualidade de serviço, monitorizados pela ferramenta de *profiling* do Microsoft SQL Server 2008 (Figura 3.4)[11]. Complementarmente, um conjunto de adicional de parâmetros externos podem ser colectados dependendo do âmbito onde o sistema se insere. Nesse sentido, informação relativa aos parâmetros de desempenho do sistema operativo, são indicadores com algum peso neste tipo de estudo. Dentro desses parâmetros, os mais importantes são a carga do CPU e percentagem de memória usada. Estes parâmetros, visto serem externos, devem ser muito bem analisados de forma a evitar imprecisões que influenciem o cálculo do índice de qualidade de serviço global.

Por fim, qualquer outro tipo de informação relevante para o estudo da qualidade de serviço de SDW, deve ser igualmente recolhida e correctamente manipulada. O esquema apresentado na Figura 3.3 ilustra as fontes genéricas de dados.

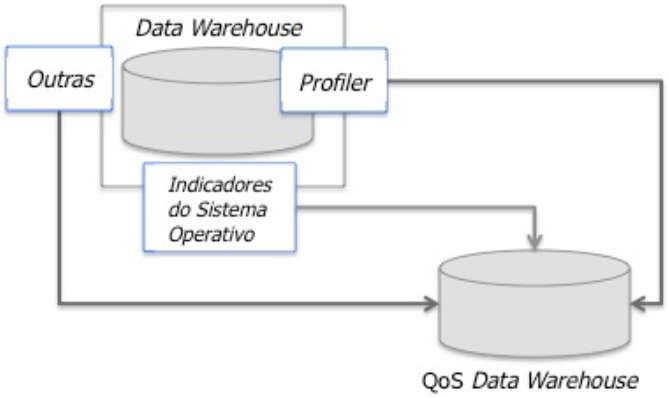

Figura 3.3: Fontes de dados para o estudo da qualidade de serviço.

Parâmetro	Descrição
Nr. Escritas	Representa o número de acessos a disco para escrita por operações da *query*.
Nr. Leituras	Representa o número de acessos a disco para leitura por operações da *query*.
Registos	Representa o número de registo devolvidos pela *query*.
Duração	Duração de execução da *query* em milisegundos.
CPU	Tempo de CPU usado para a execução da *query* em microsegundos.

Tabela 3.7: Parâmetros monitorizados pela ferramenta de *profiling* do Microsoft SQL Server 2008

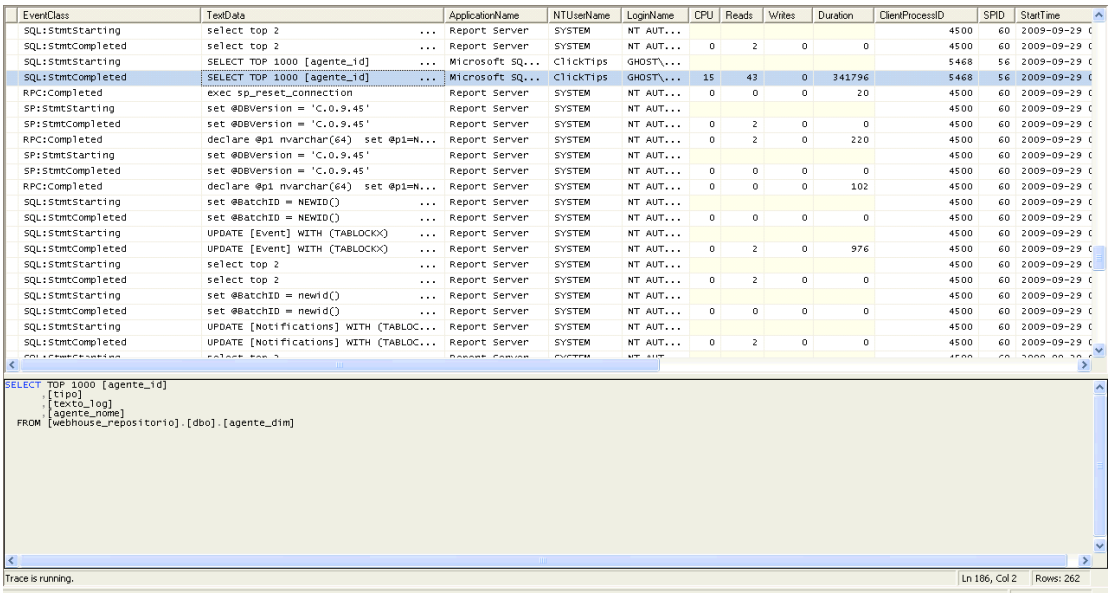

Figura 3.4: Microsoft SQL Server Profiler 2008.

3.4 Processos de Extracção, Preparação e Carregamento de Dados

Apresentado de uma forma simples, os processos de preparação e carregamento de dados visam a recolha, transformação e integração desde as suas fontes primárias até uma zona intermédia de armazenamento, a área de retenção, e por fim o DW final. Este local de armazenamento temporário assume um papel fundamental nos processos referidos. Para além de funcionar, como mencionado, de zona intermédia de armazenamento, é o local onde os dados são trabalhados, isto é, limpos, combinados e transformados, antes de serem integrados no sistema final. Estes processos, que terminam com a integração dos dados no DW, podem ser vistos como um fluxo, cujas actividades dependem da disponibilidade dos dados para a extracção, limpeza e integração, e o sucesso destas estará sempre condicionado pela qualidade e actualidade dos dados [Kimball and Caserta 04].

O processo de extracção de dados genérico para um sistema de previsão de qualidade

Figura 3.5) é em tudo semelhante ao processo de extracção de um qualquer SDW. Neste caso especifico, há que colectar dados sobre a execução das *queries* e informação adicional sobre o

comportamento do sistema operativo durante o tempo de avaliação de desempenho. Na grande maioria, as ferramentas de *profiling*, permitem guardar a informação monitorizada em vários formatos, dos quais se realçam os ficheiros de texto e as tabelas em bases de dados. Qualquer um dos formatos tem as suas vantagens e desvantagens, e dependendo da forma como o sistema a povoar esteja construído, estas podem ser mais ou menos relevantes. Contudo, a manipulação de tabelas é geralmente mais fácil e eficaz do que manipular ficheiros de texto, visto que estes exigem sempre algum tempo de processamento extra. Consoante a estrutura do ficheiro de saída este tempo pode ser mais ou menos demorado. O que se propõe, é guardar a informação gerada pela ferramenta de *profiling* sobre em tabela, desta forma passar os dados para uma tabela auxiliar na área de retenção torna-se trivial.

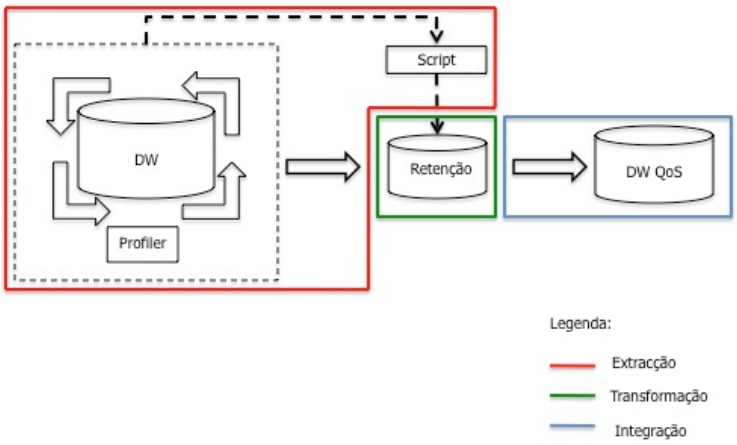

Figura 3.5: Processos de extracção, transformação e integração

A forma de colectar indicadores de desempenho do sistema operativo pode ser feita de duas formas, a saber:

- Utilizar ferramentas apropriadas para tal.
- Utilizar chamadas ao sistema por parte de linguagens de programação que capturam os valores das variáveis do sistema operativo (uma forma simples de fazer chamadas ao sistema é a utilização da linguagem de programação C, através do uso das suas funções).

Em ambos os casos, os valores dos indicadores recolhidos devem ser armazenados da forma mais flexível para posterior transferência para a área de retenção. O exemplo da utilização de ficheiros, pode ser uma forma simples e flexível de armazenar estes parâmetros caso optemos por utilizar linguagens de programação de forma a captar as variáveis de desempenho do sistema. Por fim,

transfere-se a informação para uma tabela auxiliar na área de retenção tal como é feito com a informação de *profiling*. O mesmo deve ser realizado para todas as outras fontes de dados necessárias ao sistema. Desta forma garantimos um local temporário único de armazenamento de informação.

A transformação de dados recai essencialmente sobre informação que será utilizada para povoar as dimensões *Query* e *Tabela*. Nestes casos o processamento de texto será crucial. No caso da dimensão *Query* a identificação do seu tipo apresenta-se como o maior desafio. Propõe-se o processamento da *string* contendo a *query* submetida da seguinte forma:

1. A *string* contém unicamente a cláusula "SELECT", então a *query* é classificada como sendo do tipo "SELECT"
2. A *string* contém funções de agregação ou ordenação, então a *query* é classificada como sendo do tipo da função.
3. A *string* contém a cláusula "INSERT", podendo preceder cláusulas de "SELECT", então a *query* é classificada como sendo do tipo "INSERT".
4. A *string* contém unicamente a cláusula "DELETE", então a *query* é classificada como sendo do tipo "DELETE".
5. A *string* contém a cláusula "UPDATE", podendo preceder cláusulas de "SELECT", então a *query* é classificada como sendo do tipo "UPDATE".

Para o caso da dimensão *Tabela* as principais tarefas recaem essencialmente sobre a identificação do seu tipo, assim como o seu grau e se possui ou não índices. Para estes casos as tabelas de sistema apresentam-se como uma fonte extremamente útil, onde toda esta informação pretendida se encontra armazenada. No caso de estarmos perante o SGBD da Microsoft, o Microsoft SQL Server, as tabelas *SYS.Tables* e *SYS.Indexes* guardam informação genérica sobre tabelas e índices, respectivamente. De forma a identificar-se os tipos das tabelas deve proceder-se a um levantamento manual com os responsáveis do sistema em estudo de forma a estas serem classificadas como "dimensões" ou "tabelas de facto"

Por fim, a fase de integração, representa o carregamento dos dados para DW final. Aqui, a atenção recai, essencialmente, no povoamento da tabela ponte de forma a representar correctamente a relação entre a *query* e tabelas a que ela acede.

Capítulo 4

Previsão de Desempenho em Sistemas de *Data Warehousing*

Para a previsão de desempenho em SDW várias técnicas de mineração de dados podem ser utilizadas, em particular as relacionadas com a descoberta de padrões em sistemas de dados. A definição de mineração de dados pode ser entendida como o processo de descoberta de padrões válidos, inovadores, potencialmente úteis e compreensíveis [Witten and Frank 05] [Han et al 05]. No seu processo estão envolvidas várias áreas e métodos, tais como Estatística, Bases de Dados, DW, Inteligência Artificial, Algoritmia, entre outras. Dos métodos, os mais importantes, podem ser divididos em quatro grupos principais, a saber: Classificação, Regressão, Associação, *Clustering*, e todas as suas possíveis derivações.

O processo de descoberta de conhecimento pode ser entendido como um mecanismo tripartido em que a preparação dos dados, a descoberta de padrões e a interpretação dos padrões obtidos são as principais fases. A preparação de dados é a etapa responsável pela colecta, pela limpeza e pela conciliação da informação. É nesta fase que se remove o ruído dos dados, se faz o tratamento de nulos e *outliers* e se aplicam técnicas de transformação de modo a ser possível aplicar os algoritmos necessários. Em resumo, no fim desta fase deveremos ter acesso a um conjunto de dados sobre o qual possamos aplicar os mais diversos métodos. A etapa de descoberta de padrões consiste na aplicação de algoritmos de mineração sobre o conjunto de dados previamente construído, de forma a encontrar a mais diversa informação.

Por fim, é na fase de interpretação dos padrões obtidos que se verifica o sucesso de todo o processo anterior e se analisa a informação resultante [Chen and Yu 96] [Frawley et al 91] [Fayyad et al 96].

4.1 Classes de QoS

Uma classe pode ser entendida como um conjunto de objectos que de alguma forma possuem um grupo de características, definidas pela classe, semelhantes entre si. Aplicando a noção apresentada ao tema qualidade de serviço, surge a definição de classes de QoS. Estas classes devem ser interpretadas como uma classificação dos níveis de serviço apresentados por um dado sistema. Os níveis são classificados quantitativamente com base em cinco valores distintos (1, 2, 3, 4 e 5) de acordo com a sua importância, na qual a classe cinco representa a maior qualidade de serviço e a classe um representa a menor qualidade de serviço. De forma a serem o mais representativo possível estas classes devem ser actualizadas sempre que o sistema sofra alterações com impacto directo nos seus níveis de serviço, isto é, sempre que o sistema sofra alterações importantes que influenciem o seu desempenho, a classificação das classes devem ser ajustadas de forma a representarem os novos índices de desempenho.

4.2 Análise de *Clusters*

As técnicas de *clustering* têm-se tornado num dos processos mais usados em análise de dados. As suas mais valias têm sido exploradas durante anos em áreas tão distintas como medicina, biologia, marketing, ciências da computação, entre outras. Conceptualmente designa-se por *clustering* o processo de organizar objectos, que de alguma forma são similares, em grupos [Han et al 05] [Berkin 02]. Um *cluster* é um conjunto de objectos com características semelhantes entre si e dissimilares entre objectos de um outro *cluster* distinto. A Figura 4.1 representa um conjunto de dados bidimensional, no qual facilmente se identificam quatro conjuntos (clusters) naturais de dados.

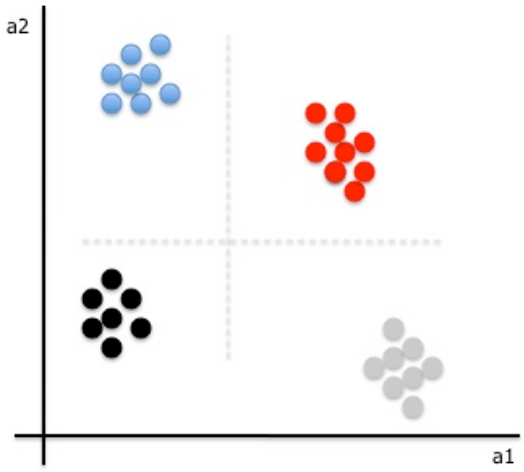

Figura 4.1: Exemplo de quatro *clusters* de objectos de atributos a1 e a2

O uso de técnicas de *clustering* em modelos de previsão para a qualidade de serviço pode ser entendido da mesma forma que o uso dos métodos de *clustering* em estabelecimento de perfis de utilização Web [Liu 07]. No caso dos modelos de previsão a ideia é agrupar *queries* com características semelhantes baseando-se nas suas variáveis de execução (tempo de resposta, número de leituras, número de escritas, etc.) de forma a estabelecer perfis interrogação. A cada perfil, ou seja, a cada *cluster* gerado, é associada uma classe de QoS, tendo em conta os valores médios das variáveis de execução de cada um dos perfis. Desta forma, a classificação usada permite saber o grau de desempenho das *queries* pertencentes a cada um dos perfis. Na prática, a classificação é feita preenchendo o atributo *cluster* (Secção 3.2) de cada um dos registos da tabela de factos, com o valor correspondente à classe ao qual esse registo pertence.

4.2.1 Clusters Hierárquicos vs Clusters Particionados

Das vários tipos de técnicas de *clustering* [Berkin 02] existentes na bibliografia, as técnicas hierárquicas [Murtagh 83] [Jain and Dubes 88] [Kaufman and Rousseeuw 90] e as técnicas particionadas [Boley 98] [Kaufman and Rousseeuw 90] [Jain and Dubes 88] são as mais comuns. Distinguem-se essencialmente pela forma como fazem a divisão dos objectos em grupos. O *clustering* hierárquico, como o próprio nome indica, constrói hierarquias de *clusters*. Por outras palavras, constrói uma árvore de *clusters*, na qual cada nó contém *clusters* filhos, que são por natureza, abrangentes que o *cluster* pai. Por sua vez, as técnicas particionadas fazem a divisão

do conjunto inicial de dados em k subconjuntos, isto é, em k *clusters*. Ao contrário das técnicas hierárquicas, que após a construção dos *clusters* estes não voltam a ser reformulados, os métodos que usam particionamento melhoram de forma gradual os *clusters* construídos através de técnicas de optimização iterativa. Dos métodos mais conhecidos há que salientar os algoritmos Fuzzy [Windham 82], K-Means [Hartigan and Wong 79] [Hartigan 75], a abordagem EM-Clustering [MacLachlan and Krishnan 96] [Dempster et al 77] e todas as suas vertentes conhecidas.

Dos algoritmos mais utilizados nas técnicas hierárquicas realçam-se os algoritmos bottom-up ("aglomerativos") e os top-down (divisão) [Berkin 02], como também as suas diversas abordagens. Estes distinguem-se essencialmente pela forma como constroem os conjuntos. No caso das abordagens bottom-up, estas começam com um *cluster* por item e sucessivamente vão agrupando os vários *clusters* em *clusters* cada vez maiores. Em relação às abordagens que seguem a ideia de divisão, estas fazem o processo inverso das abordagens bottom-up, começando com todos os itens num único conjunto e, sucessivamente, vão-se dividindo até que todos os itens do c*lusters* criados satisfaçam um conjunto de condições [Han et al 05].

A utilização de *clusters* para o cálculo da qualidade de serviço prevê a utilização de clusters particionados. A justificação prendem-se com as características de ambos os métodos. Os métodos hierárquicos são especialmente lentos em conjuntos de dados de grande dimensão. Contrariamente, os métodos particionados comportam-se relativamente bem com grandes conjuntos de dados. Uma outra característica fundamental tem a ver com a possibilidade de escolher o número de *clusters* à partida. Isto significa que os métodos particionados permitem a escolha do número de *clusters* a serem gerados, característica essencial para o cálculo da qualidade de serviço.

4.2.2 Influencia de *Outliers* no Cálculo da Qualidade de Serviço

Os *outliers* (Figura 4.2) são um dos factores mais importantes a ter em conta na construção de *clusters* e consequentemente no estabelecimento de perfis de *queries*. O aparecimento de *outliers* surge directamente ligado ao tipo de utilizadores e às suas *queries*, que, como sabemos, podem gerar valores para as variáveis de execução muito distantes da média global. Neste sentido, um *outilier* define-se como um ponto no conjunto de dados que não se encaixa na distribuição. As

abordagens mais clássicas na análise de dados utilizam o conceito de profundidade (*depth*) e definem um *outlier* como um ponto de baixa profundidade [Tukey 77]. Alternativamente podem ser interpretados como pontos de ruído no conjunto de dados, pontos esses que apresentam um comportamento anormal.

O problema dos *outliers* tem impacto na construção dos *clusters*, podendo esta ser mais ou menos influenciada consoante o tipo de algoritmo escolhido e, consequentemente o cálculo da média de cada variável para cada *cluster*. De forma a considerarmos um pouco melhor a questão anterior, vejamos o seguinte exemplo sobre o impacto de *outliers* na geração de *clusters*:

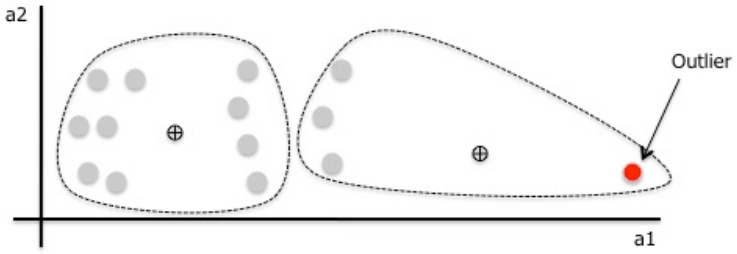

Figura 4.2: Impacto de *outliers* na geração de *clusters*

Como é possível verificar na Figura 4.2, o facto de um elemento do conjunto de dados se distanciar do posicionamento global da maioria do objectos provoca uma discrepância nos elementos do *cluster*, e, consequentemente, nos *clusters* geometricamente próximos do *cluster* afectado pelo *outlier*. Ora, como a ideia de *clustering* é, precisamente, agrupar elementos com características semelhantes entre si, o facto do conjunto de objectos possuir um elemento claramente distante dos restantes provoca resultados finais incorrectos e imprecisos.

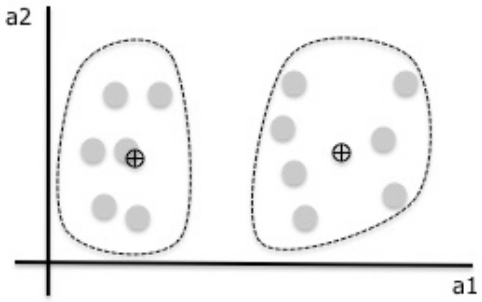

Figura 4.3: Exemplo de *clusters* sem a influência de *outliers*

De forma a analisar o impacto da influência de *outliers* na geração de *clusters* e no cálculo da qualidade de serviço dos seus elementos, consideremos o seguinte cenário com um *cluster* com dois atributos (a1, a2) e seis elementos (Tabela 4.1)

	a1	a2
1	0.2	1.0
2	0.3	1.0
3	0.24	1.2
4	0.32	1.1
5	8.4	1.5
6	0.39	1.2

Tabela 4.1: Exemplo de um *cluster* com dois atributos

Como podemos verificar o objecto 5 para o atributo a1 possui um valor que se distancia dos restantes valores do atributo. Calculando a média para o atributo a1 obtemos o valor de 1.64, que se mostra não ser representativo para o conjunto de valores do atributo a1, uma vez que o esperado seria algo no intervalo [0.28 – 0.31]. Ora, dado que a média dos valores para cada um dos atributos é utilizada para atribuir uma classe de qualidade de serviço a cada um dos *clusters*, é fácil de perceber que usando o valor médio afectado pelo *outlier* este não é uma representação numérica correcta do atributo, influenciando assim a classificação do *cluster*. Existe na bibliografia actual vários métodos para lidar com *outliers* no conjunto de dados. Na realidade este trabalho deve ser feito na fase de pré-processamento, contudo alguns algoritmos já apresentam recursos para lidarem com este tipo de problema. O algoritmo CURE [Guha et al 98] usa o subconjunto

mais representativo do cluster para suprimir o efeito dos *outliers*. O algoritmo CLIQUE [Agrawal et al 98] vai um pouco mais além, este elimina subespaços pouco povoados de forma e contrariar o aparecimento de *outliers*. Já o algoritmo ORCLUS [Aggarwal and Philip 00] produz uma partição com um conjunto de *outliers.*

4.3 Modelos de Previsão Numérica

A ideia de que o ser humano aprende interagindo com o ambiente, baseando-se em acontecimentos passados para evoluir a sua experiencia em situações futuras, é provavelmente a acção mais intuitiva quando pensamos em aprendizagem natural. Numa abordagem computacional a ideia de aprendizagem segue os mesmos princípios, isto é, baseado em acontecimentos passados prever acções futuras. Na realidade, o uso de modelos de previsão têm sido desde há muito utilizados nas mais diversas áreas, um pouco por todo o lado, desde a Genética até a Engenharia ou mesmo na Inteligência Artificial. Suponhamos que um consultor de marketing pretende saber quanto gastará um dado cliente numa nova campanha de equipamentos electrónicos. Esta acção pressupõe um problema de previsão, em particular um problema de previsão numérica, onde o modelo construído prevê um valor numérico através de uma função numérica contínua. O que se pretende neste exemplo é prever o comportamento de um dado cliente tendo em conta todas as suas acções (interacções com a companhia) históricas. Estamos então perante um problema de regressão. Existem no mercado várias ferramentas que permitem trabalhar sobre problemas de regressão, destacando-se o *SAS, SPSS, S-Plus, MatLab, Mathematica*, entre outras [2, 3, 4, 5, 6].

A análise de regressão é uma metodologia estatística que é maioritariamente usada para fazer previsão numérica [Han et al 05]. Em rigor, um modelo de regressão numérico é usado para modelar relacionamentos entre uma ou mais variáveis, chamadas de variáveis de previsão, e uma variável objectivo. O propósito final de um modelo de regressão é construir uma função *f* tal que *f(X) = y*, onde *X* representa o conjunto de variáveis de previsão e *y* a variável objectivo (aquela que pretendemos prever). A Figura 4.4 ilustra uma equação de regressão linear sobre dois atributos.

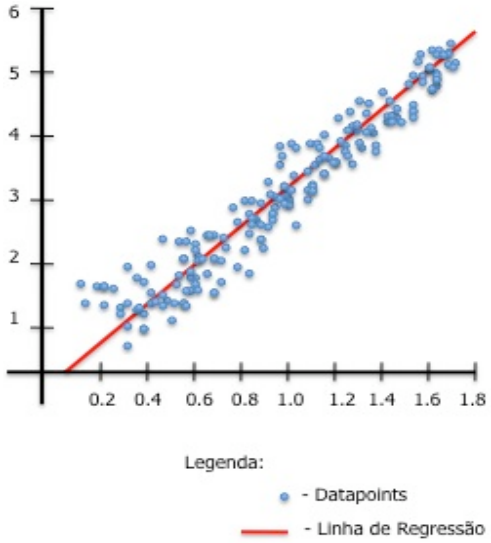

Figura 4.4: Representação gráfica de uma equação de regressão

A utilização de modelos de previsão numérica em qualidade de serviço para um SDW prevê, na nossa opinião, um problema de regressão. De facto, como todas as medidas da tabela de factos (variáveis de previsão) são atributos numéricos não é difícil de perceber que estamos perante um problema de previsão numérica. O objectivo, de uma forma simplista, é construir uma equação matemática que represente a relação de todas as variáveis de previsão, isto é, todas as métricas de qualidade de serviço inseridas na tabela de factos, com a variável objectivo. Desta forma, construímos uma relação matemática que permite saber a influência das variáveis de previsão no valor tomado pela variável objectivo, e assim usá-la como método de previsão.

Como em qualquer outro modelo estatístico, também as técnicas de previsão numérica, nas quais se inserem os modelos de regressão linear, possuem várias medidas de avaliação de desempenho (Tabela 4.2), isto é, valores de erro para avaliarem a precisão do modelo. O erro absoluto faz a quantificação da diferença entre o valor real para um dado registo (y_i) e o valor previsto (y'_i), permitindo assim visualizar a diferença entre a previsão e o valor real. Desta forma, o erro médio absoluto representa a diferença média entre o real e o previsto para o conjunto de treino. Devido à sua natureza, o erro médio absoluto apresenta-se como um bom avaliador mesmo na presença de *outliers,* sendo usualmente um dos indicadores mais utilizados. Contrariamente, o erro médio quadrático tende a exagerar na presença de *outliers*. No caso dos erros relativo absoluto e relativo quadrático, estes apresentam os valores afectados por um simples normalização.

Medida de Avaliação	Fórmula
Erro absoluto	$\left\|y_i - y'_i\right\|$
Erro médio absoluto	$\frac{\sum_{i=1}^{d}\left\|y_i - y'_i\right\|}{d}$
Erro quadrático	$\left(y_i - y'_i\right)^2$
Erro médio quadrático	$\frac{\sum_{i=1}^{d}\left(y_i - y'_i\right)^2}{d}$
Erro relativo absoluto	$\frac{\sum_{i=1}^{d}\left\|yi - y'i\right\|}{\sum_{i=1}^{d}\left\|y_i - \bar{y}\right\|}$
Erro relativo quadrático	$\frac{\sum_{i=1}^{d}\left(y_i - y'_i\right)^2}{\sum_{i=1}^{d}\left(y_i - \bar{y}\right)^2}$

Tabela 4.2:Tabela de erros em modelos de previsão

Todos os indicadores apresentados anteriormente, apesar de poderem ser considerados avaliadores de precisão (em termos de erros), sem um mecanismo de avaliação não são constatados. As técnicas mais comuns para obter um valor representativo da qualidade dos modelos de previsão englobam a validação simples, a validação cruzada, as abordagens *bootstrap*, entre outras. Todos estes mecanismos são baseados no particionamento aleatório do conjunto de treino, usando uma das partições para teste de precisão e as restantes como conjunto de treino.

A validação simples [Han et al 05] (Figura 4.5) é o método mais comum para estudos de precisão em modelos de precisão. Este método uso um particionamento aleatório do conjunto inicial em dois subconjuntos independentes, o conjunto de treino e o conjunto de teste. Tipicamente dois terços do conjunto total de dados são alocados para o conjunto de treino e o restante terço para o conjunto de teste. Desta forma, possuímos um conjunto dados para criar o modelo e um conjunto mais pequeno para estimar a precisão.

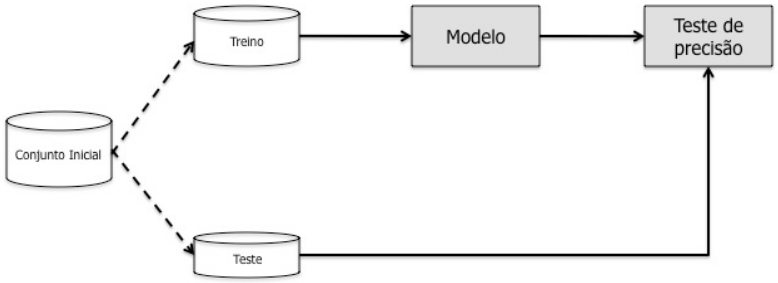

Figura 4.5: Cálculo de precisão com o método de validação simples

Apesar de muito simples e intuitivo, este método apresenta uma grande limitação. A estimativa resultante é sempre pessimista, porque apenas uma partição do conjunto inicial é usada para criar o modelo. De forma a colmatar este problema surgiu uma variação do método de validação simples, onde processo anterior é repetido *k* vezes. Assim a precisão global é dada pela média dos valores ocorridos em cada uma das *k* iterações.

O método de validação cruzada é talvez a abordagem mais seguida neste tipo de estudos. O conjunto inicial de dados é aleatoriamente particionado em *k* subconjuntos chamados de "*folds*", onde cada uma das partições resultantes, P_1, P_2,...,P_k, tem aproximadamente a mesma dimensão. É visto como um método iterativo em que cada iteração *i* a partição P_i é reservada para conjunto de teste e as restantes partições são usadas colectivamente para a construção do modelo. Assim na primeira iteração, a partição P_1 é usada como conjunto de teste e as restantes partições para a construção do modelo. Iterativamente cada uma das partições vai sendo usada como conjunto de teste. Ao contrário dos métodos anteriores, em que cada uma das partições é usada o mesmo número de vezes para conjunto de treino e teste, aqui cada uma das partições é usada apenas uma vez para treino e apenas uma outra como partição de teste. Consequentemente, os erros para avaliação de precisão são calculados pela soma dos valores das *k* iterações dividindo por *k.* Desta forma é garantido uma maior independência dos resultados evitando o sobre ajustamento do modelo ao conjunto de treino (uma das falhas dos métodos anteriores). Apesar de ser um método interessante e amplamente utilizado, a forma como os subconjuntos são gerados é um dos factores mais importantes a ter em conta. O que se pretende são *k* subconjuntos com distribuições o mais semelhante possível com a distribuição do conjunto original. Sendo assim o método de particionamento semi-aleatório mais utilizado é o particionamento estratificado ("*stratified cross-*

validation"). [Kohavi 95] [Han et al 05]. A Figura 4.6 ilustra o método de validação cruzada com *k=3*.

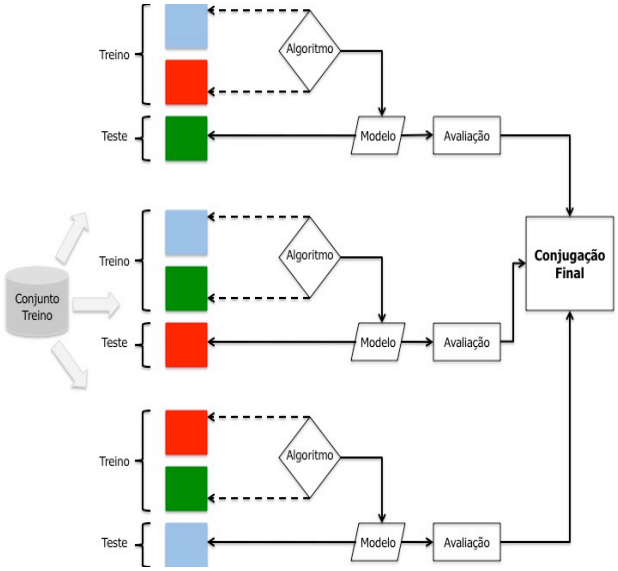

Figura 4.6: Método de validação cruzada

Ao contrário dos métodos de avaliação de precisão mencionados anteriormente, o método *Bootstrap* [Han et al 05] [Kohavi 95] [Efron and Tibshirani 93] particiona o conjunto de inicial de forma uniforme e com substituição. Quer isto dizer que de cada vez que um subconjunto é seleccionado este é igualmente provável de ser seleccionado outra vez e adicionado de novo ao conjunto de treino. Existem vários métodos de *bootstraping*, no entanto o mais usado é o .632 *bootstrap*. O seu processamento funciona da seguinte forma: suponhamos que temos um conjunto de dados particionado em *k* subconjuntos; em cada uma das *k* iterações é seleccionado um subconjunto dos *k* subconjuntos e é adicionado ao conjunto que será considerado como o conjunto de treino final; no fim da primeira execução esse conjunto é reposto juntamente com os *k* subconjuntos iniciais; na iteração seguinte o processo é de igual forma repetido, onde qualquer um dos conjuntos anteriormente seleccionados pode ser de novo escolhido. Desta forma a abordagem .632 *bootstrap* garante um conjunto de treino com o tamanho igual ao conjunto inicial, onde os conjuntos não seleccionados serão parte integrante do conjunto de teste final. O método garante que cerca de 63.2% dos subconjuntos (daí .632 *bootstrap*) vão fazer parte do conjunto de treino final e os restantes 36.8% vão fazer parte do conjunto de teste.

Capítulo 5

Avaliação Experimental

5.1 Descrição do *Data Warehouse* Alvo

O DW sujeito ao estudo sobre qualidade de serviço, acolhe informação sobre dados Web, nestas condições o sistema é denominado por *data webhouse*. Em [Joshi el al 99] [Joshi el al 03] é descrito o recurso a um DW com armazenamento *Relational Online Analytical Processing* (ROLAP) para descoberta, por processos de mineração, de padrões e acesso Web, tendo como única fonte de dados os *logs* de acesso. No caso de estudo, o DW em questão armazena informação sobre os acessos Web a um portal da instituição Universidade do Minho, concretamente ao portal do repositório online [7].

Com cerca de seis utilizadores, desde administradores do sistema até administradores do portal, este sistema processa diariamente cerca de 30 megabytes de texto em ficheiros de *logs* o que se traduz num espaço em disco de cerca de 6 megabytes, o que faz com que a informação diária acrescente ao DW cerca de 6 megabytes. Os acessos são essencialmente feitos por ferramentas de criação relatórios ou ferramentas OLAP, e uma menor percentagem diz respeito a acessos através de *queries ad-hoc.*

O modelo dimensional construído para armazenar os dados provenientes dos ficheiros de Web *log* dos *sites* Web é composto por uma tabela de factos, *TF_Sessions*, seis dimensões (*Calendário, Tempo, Request, Referrer, Agente, Computador Utilizador*) e por uma tabela ponte, *TP_Sessions*, que estabelece a relação entre uma sessão e os pedidos feitos nessa sessão (Figura 5.1).

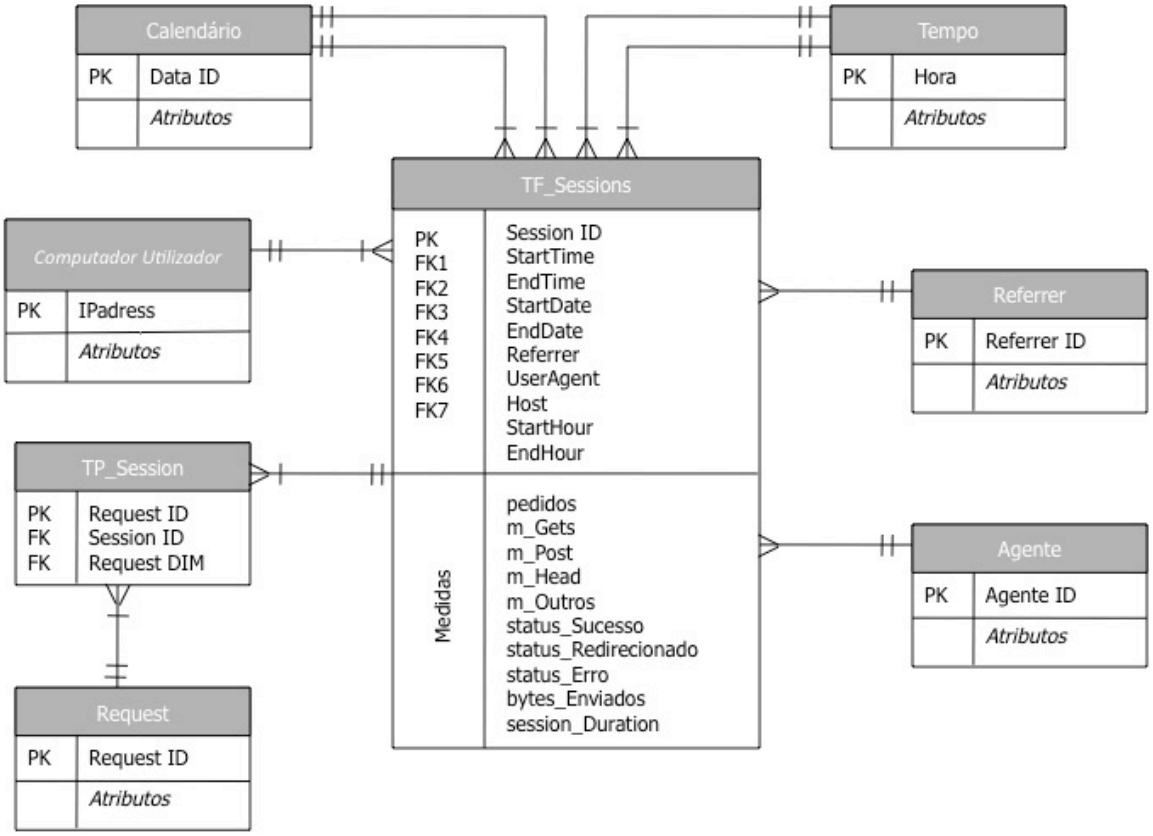

Figura 5.1: *Datamart* para sessões *Web*

A tabela de factos (Tabela A.1) guarda o registo de todas as sessões identificadas nos ficheiros de *log*. Esta é composta por vinte atributos, dos quais sete são chaves estrangeiras para as várias dimensões, e dez são medidas que permitem analisar os dados de cada sessão sobre os mais diversos pontos de vista.

As dimensões *Tempo* e *Calendário* (Tabela A.1), como em qualquer outro DW, caracterizam o momento em que a sessão ocorreu. Neste caso as dimensões *Calendário* e *Tempo*, caracterizam respectivamente o dia e a hora de ocorrência da sessão. A descrição do local de onde é efectuado o pedido é feita pela dimensão *computador Utilizador* (Tabela A.1), e o seu povoamento utiliza o endereço de IP que fez o pedido ao site. A dimensão *UserAgent* (Tabela A.1) distingue os programas utilizados para aceder ao *site* Web e normalmente identifica qual o browser utilizado. Nesta dimensão é feita a classificação dos utilizadores em *crawlers* e utilizadores regulares. A

dimensão *Referrer* (Tabela A.1), povoada pelo campo do referente contido no ficheiro de logs, caracteriza o *site* que referenciou a página Web relativa ao pedido. A dimensão *Request* (Tabela A.1) descreve o pedido efectuado ao *site*. Esta dimensão é povoada recorrendo ao campo *request* do ficheiro de *logs.* Por fim, a tabela ponte (*TP_Sessions*), estabelece a ligação entre a tabela de factos (*TF_Sessions*) e a dimensão *Request*, dada a necessidade de manter a ordem dos pedidos em cada sessão.

Todo o sistema está implementado numa máquina Pentium IV de CPU 3.0 GHz com 80 GB de disco e 1GB de memória RAM, suportado pelo Microsoft SQL Server 2008 instalado em Windows XP Profissional. Adicionalmente, é importante salientar que nenhuma operação de melhoria de desempenho foi efectuada no sistema - nenhuma tabela possui qualquer tipo de índice alternativo de pesquisa criado, a não ser os relativos a cada uma das chaves primárias das tabelas envolvidas.

5.2 Aplicação das Técnicas de Mineração de Dados

As técnicas de mineração de dados utilizadas para o trabalho prático da presente dissertação assentaram essencialmente sobre duas áreas, nomeadamente as técnicas de *clustering* e técnicas de previsão numérica [Han et al 05] [Witten and Frank 05]. Todo o processo que visa a aplicação das referidas técnicas segue inteiramente a metodologia *CRISP-DM* [Chapman et al 99].

A metodologia *CRISP-DM* foi concebida em 1996 por três especialistas da indústria de mineração de dados e tem sido adoptada por diversas entidades na resolução dos seus projectos de implementação de sistemas de mineração de dados. Em rigor a metodologia *CRISP-DM* apresenta as principais linhas de orientação a serem seguidas aquando da implementação de sistemas de mineração de dados, facilitando em muito o trabalho dos coordenadores deste tipo de projecto.

O ciclo de vida de um projecto de mineração de dados (Figura 5.2) é composto por seis fases principais de ordem não restrita, que isto dizer que, dependendo da tarefa a executar e quais os seus resultados, pode ser necessária uma movimentação dinâmica entre elas. Cada uma das seis fases é descrita da seguinte forma:

1. **Compreensão de Negócio.** Esta é a primeira etapa num projecto de mineração de dados. Foca-se na compreensão dos objectivos e nos requisitos do projecto sobre o ponto de vista do negócio. Mais tarde foca-se na conversão do problema num problema de mineração de dados. Nesta fase são traçados os objectivos a serem cumpridos durante todo o projecto.
2. **Compreensão dos dados.** Esta etapa começa com um conjunto de dados inicial sobre o qual são aplicados um leque de algoritmos de modo a extrair um conhecimento total sobre conjunto de dados. Nesse sentido pretende-se fazer a identificação dos problemas na qualidade dos dados, descobrir os primeiros padrões e detectar subconjuntos de dados que possam fornecer importantes informações.
3. **Preparação dos dados.** A preparação dos dados é uma das fases mais importantes num projecto deste tipo. Nesta fase são feitas todas as transformações necessárias sobre os dados para obter no fim conjunto de treino o mais coeso possível. A preparação de dados pode ser feita em qualquer altura e não tem um ordem específica. Dentro do leque das várias tarefas incluídas na prepa**r**ação temos a selecção de registos, a selecção de atributos, a transformação, a limpeza dos dados e o tratamento de nulos.
4. **Modelação.** Nesta fase várias técnicas de modelação de dados são seleccionadas e aplicadas. Tipicamente, existem várias técnicas que podem ser aplicadas no mesmo problema de mineração de dados, contudo algumas dessas técnicas têm requisitos específicos consoante o seu tipo, o que faz com que seja frequente voltar à fase anterior.
5. **Avaliação.** Neste etapa, um modelo já muito próximo do modelo final deve estar **c**onstruído. Antes de passar para a fase final do projecto é necessário avaliar o modelo e rever todos os passos seguidos até à sua obtenção. Um dos objectivos da revisão é verificar se tudo o que foi planeado foi efectivamente concluído de forma a avaliar se os modelos devem ou não serem utilizados.
6. **Implementação**. A criação do modelo não é a última tarefa num projecto de mineração de dados. Mesmo quando o objectivo do modelo é aumentar o nível de conhecimento, é necessário organizar a informação obtida de modo a poder ser usada pelos clientes. Dependendo dos requisitos, a fase de implementação pode ser tão simples como a geração de um relatório ou tão complexo como um processo de mineração de dados que se vai repetindo.

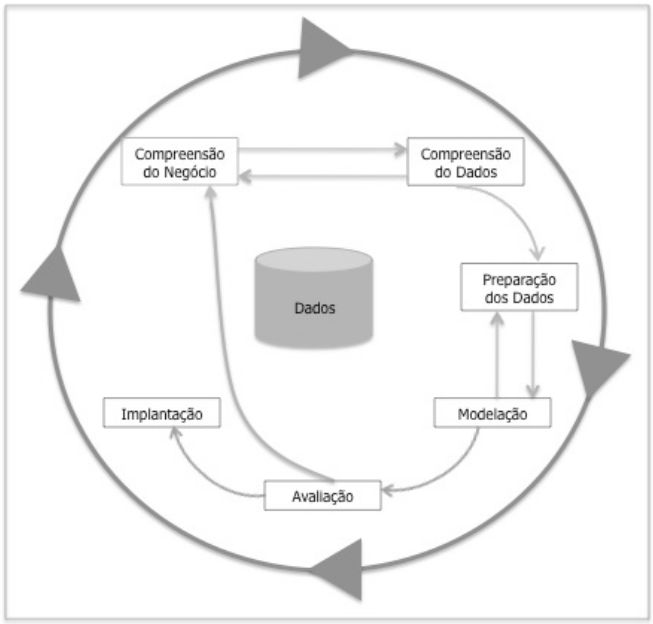

Figura 5.2: Fases da metodologia *CRISP-DM* [1]

Actualmente existem no mercado várias ferramentas para aplicação de técnicas de mineração de dados. Contudo, a ferramenta escolhida para a elaboração dos testes práticos foi a versão 4.4 do *RapidMiner* [8]. Esta ferramenta apresenta-se como uma excelente opção no conjunto global de ferramentas disponíveis, uma vez que:

1. A ferramenta está disponível numa versão *open-source*.
2. O elevado número de algoritmos implementados para as diferentes técnicas (Associação, Regressão, Classificação e *Clustering*).
3. Possui bons módulos gráficos que possibilitam uma melhor visualização de resultados.
4. A possibilidade de poder ser instalada em diversos sistemas operativos (de referir que a ferramenta está programada em JAVA [9]).
5. A interface gráfica é bastante intuitiva, possibilitando a aprendizagem e a rápida evolução.
6. A ferramenta possui uma boa cotação entre a comunidade científica.

5.2.1 O Conjunto de Treino

O conjunto de treino, utilizado para a aplicação das técnicas mencionadas, é composto por 2079 registos, armazenados num DW construído para esse fim. O número de registos é directamente

proporcional ao número de *queries* enviadas ao sistema que foram efectivamente monitorizados pela ferramenta de *Profilling* do Microsoft SQL Server 2008. Todos os atributos do conjunto de treino são atributos numéricos ou binários, não possuindo qualquer valor nulo (Tabela 5.1) - o que facilita o seu processamento.

Atributo	Tipo	Intervalo	Média	Desvio Padrão	% Nulos
Nr. Escritas	Numérico	[0 – 10,457.000]	58.751	+/- 558.328	0%
Nr. Leituras	Numérico	[1.000; 1,503,811.000]	69,450.380	+/- 176,102.121	0%
Registos	Numérico	[1.000; 1,503,811.000]	28,231.044	+/- 121,534.087	0%
Duração.	Numérico	[0.000 - 882.164]	7.523	+/- 39.199	0%
Ordenação	Binário	[0.000 - 1.000]	0.534	+/- 0.499	0%
Agregação	Binário	[0.000 - 1.000]	0.596	+/- 0.491	0%
CPU_U	Numérico	[0.000 - 100.000]	28.761	+/- 34.257	0%
MEM_U	Numérico	[472.890 - 961.740]	879.868	+/- 55.509	0%

Tabela 5.1: Caracterização dos atributos do conjunto de treino

De forma a construir um conjunto de treino mais coeso foram aplicadas algumas técnicas e realizados alguns estudos, tais como estudos de correlação e de remoção de *outliers.* Como mencionado anteriormente (secção 4.2.2) a existência de *ouliers* pode influenciar em larga medida a representação fiel dos *clusters* e, consequentemente, a qualidade de serviço atribuída às *queries* pertencentes aos *clusters* afectados. De forma a contrariar este problema, é feita uma inspecção visual aos atributos na tentativa de encontrar registos com atributos claramente distanciados da média global de cada atributo.

No estudo efectuado verificou-se a existência de três registos considerados como *outliers*. A Figura 5.3 apresenta um dos três *outliers* encontrados. Neste caso em particular, o atributo em questão é o tempo de resposta. Como podemos verificar existe um valor que se distancia claramente dos restantes, neste caso, assim como nos outros, a abordagem seguida leva a remoção dos registos em causa do conjunto de treino.

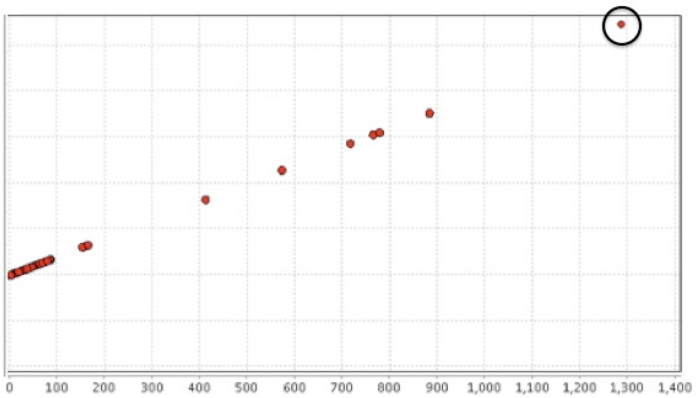

Figura 5.3: Representação gráfica de um *outlier* no atributo Tempo de Processamento

Os estudos de correlação de atributos apresentam-se como uma abordagem especialmente útil na hora de definir o conjunto final de atributos a processar. Em rigor, estudos de correlação permitem descobrir fortes dependências lineares entre atributos, podendo assim excluir aqueles que são fortemente correlacionados, facilitando assim o mecanismos de *clustering*. Assim, a construção de uma matriz de correlação [Johnson and Wichern 82] permitiu descobrir as dependências entre vários atributos levando a uma nova selecção. O limite considerado para o qual todos os valores de correlação entre dois atributos superiores a esse valor são considerados linearmente independentes foi 0.8, ou seja, qualquer valor de correlação entre dois atributos superior a 0.8, esses atributos são considerados, no âmbito deste estudo, linearmente dependentes. A Tabela 5.2, apresenta os resultados do estudo de correlação entre os atributos do conjunto de treino.

Nome	N_writes	N_reads	Registos	Duração	ORDN	AGG	CPU_U
Nr. Escritas	1	0.812	-0.024	0.089	-0.113	-0.125	0.802
Nr. Leituras	0.812	1	-0.05	0.103	0.006	0.04	0.782
Registos	-0.024	-0.05	1	0.111	-0.217	-0.238	0.12
Duração	0.089	0.103	0.111	1	0.064	0.058	0.03
Agregação	-0.113	0.006	-0.217	0.064	1	0.881	0.67
Ordenação	-0.125	0.04	-0.217	0.064	0.881	1	0.56
CPU_U	0.802	0.782	0.12	0.03	0.67	0.56	1

Tabela 5.2: Matriz de correlação entre os atributos

Da análise anterior, conclui-se que os pares de atributos (Nr. Escritas, Nr. Leituras), (Agregação, Ordenação), (Nr. Escritas, CPU_U) são linearmente dependentes entre si, podendo assim excluir um de cada par. O atributo a eliminar foi escolhido tendo em conta os coeficientes de correlação com os restantes atributos. Por exemplo, o atributo Nr. Leituras comparativamente com o atributo Nr. Escritas possuiu coeficientes de correlação com os outros atributos de maior valor do que o atributo Nr. Escritas, ou seja, existe uma maior dependência entre o número de leituras e os restantes atributos do que o número de escritas para com os mesmos atributos. Desta forma o conjunto final de atributos é composto pelas seguintes variáveis: Nr. Escritas, Registos, Duração e Agregação.

5.2.2 Técnicas de *Clustering*

Na ferramenta de mineração utilizada existem várias implementações de algoritmos de *Clustering*, quer implementações que seguem abordagens particionadas que implementações que seguem abordagens hierárquicas. O estudo comparativo entre as potencialidades das duas técnicas, apresenta as técnicas particionadas como a melhor opção para o caso de estudo em particular. A justificação vai de encontro às características do conjunto de treino e dos próprios algoritmos. Neste caso o tamanho do conjunto de treino e a quantidade de atributos provocam um tempo de processamento elevado, comparado com outros algoritmos de particionamento, isto é, a escalabilidade dos algoritmos hierárquicos é menor do que a escalabilidade dos algoritmos particionados. No entanto, apesar das questões de desempenho, os algoritmos hierárquicos que acompanham a distribuição do *RapidMiner* utilizada apresentam uma outra debilidade no âmbito deste estudo, debilidade esta relacionada com o número de *clusters* que estes produzem. De facto, o que é pretendido com a criação de *clusters* é a possibilidade de gerar perfis (determinados a partida) de *queries*. Uma vez que os algoritmos hierárquicos não permitem como parâmetro de entrada o número de *clusters* a serem construídos não é possível criar o número exacto de *clusters* pretendido. A Tabela 5.3 apresenta um pequeno estudo comparativo entre alguns algoritmos com implementações no *RapidMiner.*

Algoritmo	Tipo	Registos	Atributos	Tempo de Execução
K-Means	Particionado	2079	4	10 segundos
K-Medoids				55:12 minutos
EMClustering				3:45 minutos
AgglomerativeClustering	Hierárquico			22:57 minutos
TopDownClustering				S / N

Tabela 5.3: Tabela comparativa para os vários algoritmos de *clustering*

O algoritmo de *clustering* utilizado para a realização dos testes práticos foi o *EMClustering*. A sua implementação utiliza o paradigma do K-Means, com a diferença de que o EMClustering faz uso de probabilidades para atribuir um determinado objecto a um dado *cluster* . Na realidade a escolha do *EMClustering* prende-se com a dualidade entre desempenho e qualidade dos resultados obtidos. Uma vez que os algoritmos hierárquicos são excluídos à partida (por razões já mencionadas) apenas o K-Means, K-Medoids ou *EMClustering* seriam alternativas viáveis - o algoritmo que apresenta melhores resultados em tempo útil é o EMClustering.

Para a criação dos *clusters* passamos como parâmetros ao *RapidMiner* o conjunto de treino através de uma ligação a tabela de factos e seleccionando os atributos definidos. O parâmetro mais importante a ser definido é número de *clusters* que pretendemos como parâmetros de saída. Aqui o número de *clusters* é igual ao número de classes de qualidade de serviço definidas. Neste caso serão cinco as classes definidas. Por questões de optimização foram alterados os seguintes parâmetros de execução do algoritmo:

- MAX_RUNS: 100. Número de iterações do algoritmo
- MAX_OPTIMIZATION_STEPS: 1500. Número de iterações de optimização por cada iteração do algoritmo

Os itens por *cluster* ficaram distribuídos da seguinte forma:

- Cluster0: 1655 itens, P = 0.796
- Cluster1: 140 itens, P = 0.067
- Cluster2: 243 itens, P = 0.117
- Cluster3: 39 itens, P = 0.019

- Cluster4: 2 itens, P = 9.620E-4

Onde a *P* representa a probabilidade de um dado item pertencer ao *cluster*. Como seria de esperar, dado o número de itens que cada *cluster* possuiu, o *cluster0* apresenta-se como o mais provável e o *cluster4* o menos provável.

De forma a classificar todos os *clusters* com uma a respectiva qualidade de serviço, criou-se manualmente uma árvore de decisão (Figura 5.4) - pode também ser vista como uma árvore de classificação para este trabalho - de forma a classificar os *clusters* com a respectiva qualidade de serviço. Para tal, usa-se a média dos atributos dentro de cada *cluster* e percorrendo a árvore é feita a decisão sobre qual o ramo a escolher consoante o valor médio dos atributos do *cluster.*

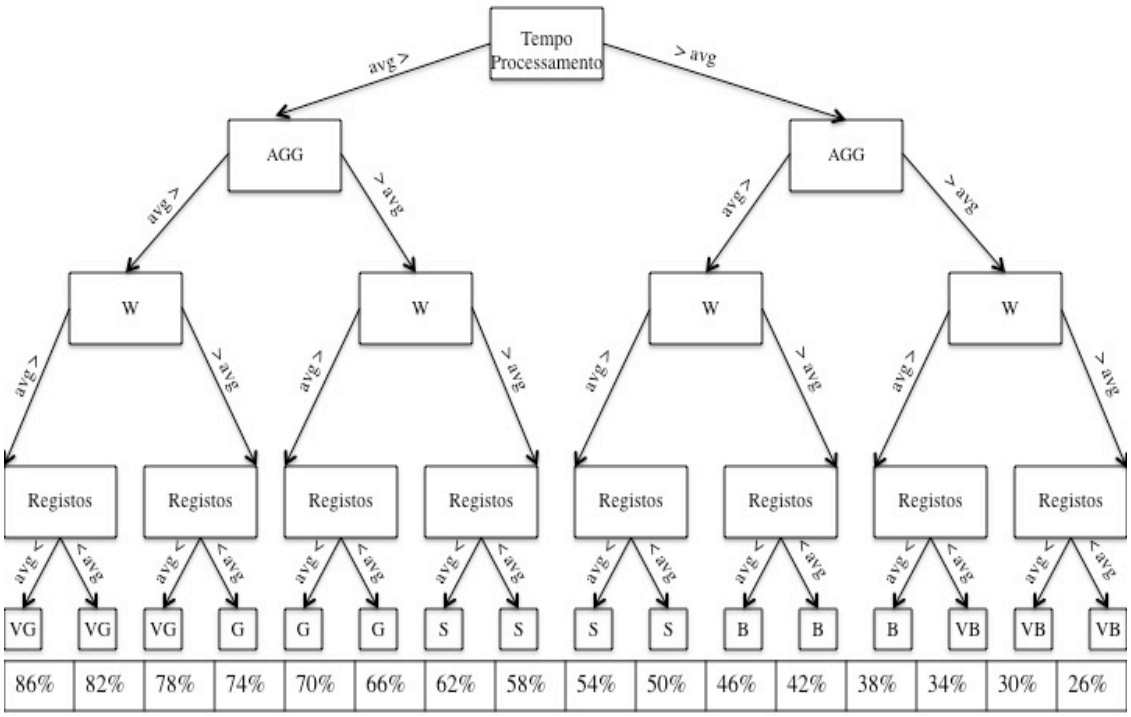

Figura 5.4: Árvore de classificação para a qualidade de serviço dos *clusters*

De forma a exemplificar melhor o mecanismo de decisão, vejamos o seguinte exemplo:

A média geral para todos os atributos do conjunto de treino é:

- Nr. Escritas = 58.75
- Registos = 28,231.04

- Duração = 7.52
- Agregação = 0.60

Para o *cluster0* os valores médios por atributo são:

- Nr. Escritas = 73.79
- Registos = 1,079.66
- Duração = 4.31
- Agregação = 0.69

Executando os passos de decisão obtemos o seguinte ramo:

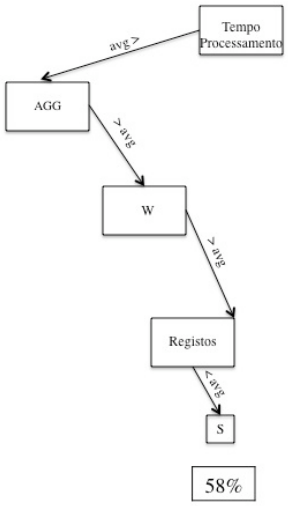

Figura 5.5: Ramo de classificação de um *cluster*

Desta forma o cluster0 é classificado como *Satisfatório* com o valor de *58%*. Os restantes *clusters* são classificados da seguinte forma:

- Cluster1 – *Muito Bom – 86%*
- Cluster2 – *Satisfatório – 54%*
- Cluster3 – *Satisfatório – 54%*
- Cluster4 – *Satisfatório – 54%*

Em rigor, a tabela de factos é agora actualizada com os valores calculados, sendo actualizado o atributo *cluster* com a respectiva classe, para cada registo pertencente a cada *cluster*, bem como o atributo QoS com a representação numérica da classe atribuída.

5.2.3 Técnicas de Regressão

Assim como para as técnicas de *clustering* também para as técnicas de regressão existem, no *RapidMiner* várias implementações de algoritmos que seguem as abordagens mais comuns em regressão. No fundo, o que se pretende com a aplicação de algoritmos de regressão linear a um conjunto de treino é a construção de uma fórmula de regressão, dando origem a uma equação matemática que prevê o valor de uma variável objectivo. Na presente dissertação, o objectivo é a previsão do índice de qualidade de serviço que o sistema apresenta na resposta a cada *query* que lhe é lançada. Contudo, dificilmente fará sentido analisar um modelo sem que este apresente qualquer estimativa de precisão. Desta forma a análise de erros (secção 4.3) pode ser vista como a análise de indicadores de desempenho em previsão numérica, possibilitando assim representar o quanto se distancia o modelo do valor real. De forma a fazer essa quantificação o método de avaliação utilizado faz uso da técnica de validação cruzada, particionando o conjunto de dados inicial em dez subconjuntos com o método de particionamento estratificado, garantindo assim uma distribuição, por cada subconjunto, mais aproximada do conjunto de dados inicial. Regra geral, o método de validação cruzada de dez subconjuntos usando o particionamento estratificado é o mais recomendado para estimar a precisão do modelo construído (mesmo que o computacionalmente seja possível mais subconjuntos) devido ao baixo valor relativo de *bias* e *variance*.

A aplicação do algoritmo de regressão ao conjunto de treino, utiliza a implementação do algoritmo de regressão linear oferecido pelo *RapidMiner,* usando o método de validação cruzada*.* Desta forma a resultado obtido, para o cálculo de um índice de desempenho de um DW, apresenta a seguinte equação de regressão:

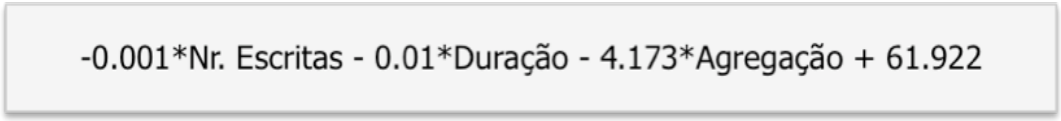

Figura 5.6: Equação de regressão para a qualidade de serviço do sistema

Como podemos verificar através da expressão obtida, esta só retorna valores menores ou iguais a 61.922, o que seria de esperar uma vez que 79.6% dos itens são classificados como *Satisfatório* com um índice de 58% pertencentes ao *cluster 1*. A diferença entre o valor de 61.922 e 58 deve-se aos 140 itens classificados como *Muito Bom* com um índice de 86%, fazendo subir ligeiramente o valor de 58 para 61.922.

Analisando de forma mais aprofundada constatamos que a variável que menos influencia a expressão final é a variável para o Tempo de Processamento (*Duração*). Esta conclusão retira-se pelo facto da gama de valores que a variável *Duração* toma serem mais baixos e menos dispersos que a gama de valores que a variável *Nr. Escritas* toma. Desta forma, a maior variância de valores na variável *Nr. Escritas* torna-a mais influenciadora para o resultado na expressão final. Contudo, esta análise tem um outro ponto a ponderar relativo ao valor dos factores pelos quais as variáveis *Duração* e *Nr. Escritas* são multiplicadas. Ora, como podemos verificar o factor que é multiplicado pela variável *Duração* é 10 vezes maior (0.001*10 = 0.01) que aquele pelo qual a variável *Nr. Escritas* é multiplicada. Esta diferença, entre os factores multiplicadores, atenua de certa a diferença de variação entre as variáveis, no entanto não se apresenta suficiente para a eliminar totalmente. Assim, concluímos que a expressão *factor * variável* para o caso *Nr. Escritas * 0.001* tem uma maior influencia no resultado final do que a expressão *Duração * 0.01.* Por fim, o cálculo da qualidade de serviço global do sistema é efectuado através de uma média aritmética de todos os índices de qualidade de serviço de todas as *queries*.

No seguimento do estudo de erros abordado anteriormente a Tabela 5.4 apresenta os valores para os erros do modelo construído, onde o erro relativo estrito representa a média do desvio absoluto do valor previsto relativamente ao valor real. Como podemos verificar, analisando o valor médio para o erro absoluto, a previsão feita pela equação de regressão desvia-se em média, aproximadamente, 3.602 unidades do valor real. Na prática, o que poderia ser verificado era uma *query* que na realidade pertencesse à classe 3 (Satisfatório), com um índice de qualidade de serviço de 64%, seja classificada como pertencente à classe 4 (Bom), com um índice de 67.602%.

Erro	Valor
Erro Médio Absoluto	3.602 +/- 0.291
Erro Médio Relativo	5.41% +/- 0.38%
Erro Relativo Estrito	6.15% +/- 0.48%
Erro Médio Quadrático	49.025 +/- 7.964

Tabela 5.4: Tabela de comparação de erros

5.3 Considerações Finais Acerca dos Resultados Obtidos

Durante a avaliação experimental foram aplicadas duas técnicas de mineração de dados (*Clustering e* Previsão Numérica) de modo a conseguir calcular um índice de qualidade de serviço para um SDW. Com a utilização das técnicas anteriormente descritas é possível, em primeiro lugar, criar perfis de interrogação (perfis de *queries*) para posterior classificação mediante vários atributos (métricas para a qualidade de serviço) e, em segundo lugar, construir uma equação de regressão que permita o cálculo de um índice de satisfação.

Algumas das análises possibilitadas pelo sistema estudado, para além das habituais relações com as dimensões temporais (onde podemos verificar períodos do tempo com degradação de desempenho no sistema), são os estudos de acessos a tabelas, mediante o índice de satisfação que as *queries* que a elas acedem retornam. Desta forma poderemos identificar problemas de desempenho em determinadas tabelas e assim adequar, por exemplo, um índice consoante o tipo de acessos maioritário. Uma outra análise que este sistema possibilita tem a ver com a ordem de execução das *queries*. Para tal, basta imaginar uma *quer*y que é lançada diariamente a uma hora onde o índice global de satisfação é pequeno e, consequentemente, o grau de satisfação da *quer*y é baixo. Assim, faz sentido analisar o mapa de interrogações segundo a qualidade de serviço do sistema e por sua vez reajustar a ordem de execução de *queries* que diariamente são enviadas. Consequentemente, uma *query* que quando executada num dado instante possuía um índice de satisfação fraco, poderá, quando executada num outro instante qualquer, retornar um nível de satisfação mais adequado.

Como é sabido, os processos computacionalmente mais pesados num SDW são os processos de povoamento da base de dados (ETI), o processamento de cubos OLAP e relatórios, bem como a

criação de cópias de segurança (backups). Segundo informações recolhidas através dos administradores do sistema sabemos que o este tem um pequeno "mapa" para os seus processamentos típicos, a saber:

- [4h – 5h] – Processos de ETI
- [5h – 6h] – Processamento de Relatórios Diários
- [9h – 16h] – *Queries ad-hoc*
- [17h – 18h] – Backup Diário

Ora como podemos verificar pela Figura 5.7, é precisamente nas horas definidas para esses processos que a carga efectiva do sistema aumenta.

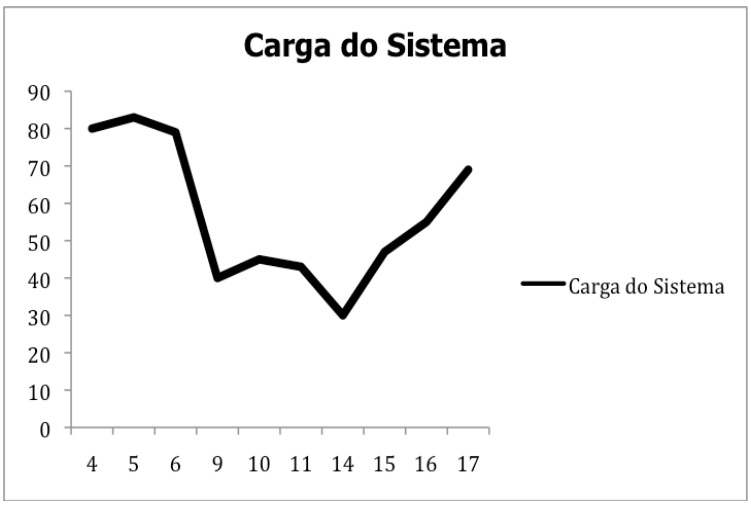

Figura 5.7: Carga do sistema por hora

Como seria de esperar a qualidade de serviço global é inversamente proporcional à carga do sistema (Figura 5.9), isto é, quando a carga do sistema aumenta a seu desempenho, perante as *queries* enviadas, diminui.

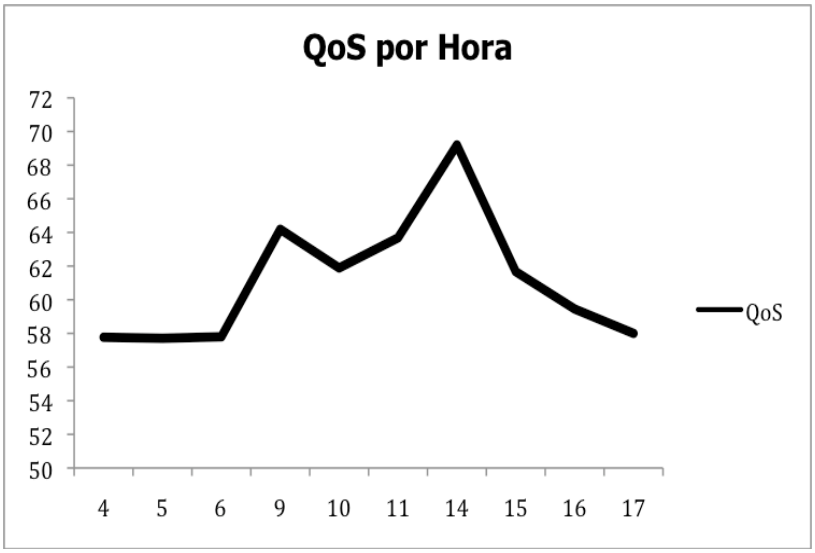

Figura 5.8: Qualidade de serviço por hora

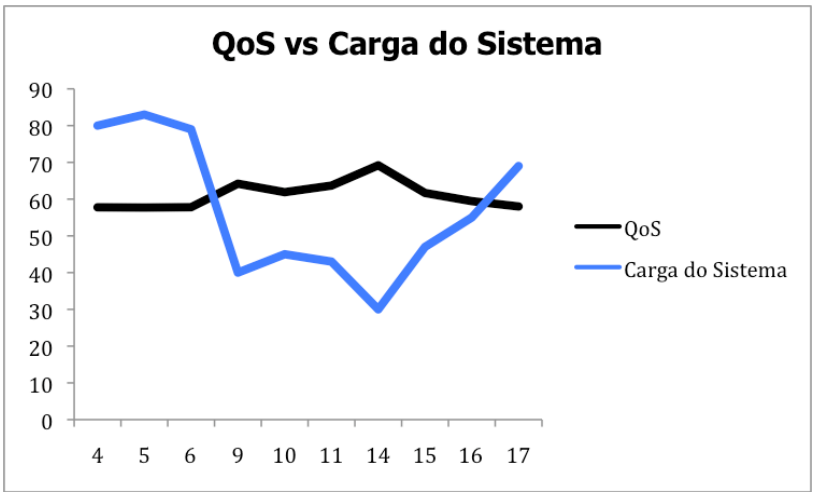

Figura 5.9: Qualidade de Serviço vs. Carga do Sistema

Desta forma podemos identificar alguns pontos de estrangulamento do sistema e assim alocar mais recursos para alguns processos que possam estar a causar esses pontos de degradação com a intenção de tornar o sistema mais fluído.

Capítulo 6

Conclusões e Trabalho Futuro

6.1 Síntese do Problema

Os SDW tornaram-se num dos principais repositórios de informação relevante aos processos de tomada de decisão, sendo actualmente aliados poderosos na competição entre organizações no mundo real. Esta importância mantêm-se à custa da capacidade que estes sistemas têm em relacionar dados e disponibilizar, para os seus utilizadores, a informação necessária. Contudo, quando a disponibilidade da informação não se apresenta tão rápida quanto os utilizadores pretendem, algumas questões de desempenho são imediatamente equacionadas. Na realidade, estas questões de desempenho nem sempre se relacionam com os algoritmos que representam a *query*. Por vezes, tabelas não optimizadas ou até mesmo estrangulamento de recursos são a justificação para tempos de resposta (disponibilidade de informação) elevados. É perante esta questão que se fundamenta a necessidade de construir um sistema de avaliação de desempenho (cálculo da qualidade de serviço), precisamente para avaliar de forma pertinente os factores que possam estar a degradar o desempenho. Nesse sentido a elaboração de técnicas de analise de desempenho constitui uma necessidade fundamental, justificada por três grandes motivações, a saber:

- **Análise diferenciada do desempenho de *queries*.** Na actualidade, as *queries* submetidas aos sistemas são de tal forma complexas, por força da complexidade do negócio, que muitas vezes os programadores não dão conta das questões que podem estar por de trás da execução. Em diversas situações, a implantação de um índice

apropriado numa tabela ou o seu particionamento poderá facilmente resolver as questões relacionadas com o desempenho de *queires*. Contudo, estas conclusões devem ser extraídas após a análise dos valores de execução relacionados com o índice de desempenho das *queries*.

- **Detecção de situações de degradação de desempenho.** À parte dos problemas de desempenho das *queries*, a detecção de situações de degradação do sistema é constantemente uma preocupação dos administradores de SDW. Desta forma, pretendemos que o sistema estudado permita avaliações segundo essas directrizes.
- **Avaliação global do sistema alvo de estudo.** Por fim, a avaliação de desempenho global do sistema apresenta-se como uma questão impossível de não ser abordada, um vez que ao analisarmos a execução de *queries* e analisarmos situações de degradação do de desempenho do sistema abrimos as duas portas principais com vista a avaliarmos o comportamento global do sistema.

Cada uma das motivações acima descritas justifica e valoriza por si só, quaisquer esforços na análise de desempenho de SDW. De facto, todas evidenciam a necessidade e a premência da implementação de um sistema de avaliação de desempenho.

6.2 Conclusões Acerca da Abordagem Seguida

A expêriencia adquirida ao longo da realização desta dissertação permitiu abordar o problema de uma forma prática, concisa e integrada no cenário dos sistemas de apoio à decisão. O sistema entretanto desenvolvido representa, no nosso ponto de vista, uma solução flexível e modular, que, apesar de não estar perfeita, assegura numa primeira abordagem a análise sobre a qualidade de serviço de um SDW. Durante o estudo e concepção, na qual resulta esta dissertação, grande parte do tempo e esforço foram investidos no desenvolvimento, na estruturação e na análise dos resultados obtidos. Apesar de uma cota parte do esforço inicialmente dispendido na pesquisa de bibliografia adequada, concluímos que quase nada de relevo científico tem sido feito no que diz respeito à qualidade de serviço em SDW. Mesmo os trabalhos de [Jarke and Vassiliou 97] [Calero et al 01] [Vassiliadis 00] que abordaram a temática da qualidade em SDW, na nossa opinião, não se enquadram naquilo que foi pretendido na elaboração desta dissertação. Assim a escassez de trabalhos desenvolvidos no tema dificultou o arranque inicial do trabalho prático e teórico. Contudo, usando o título do presente documento como mote, começamos a inferir aquilo que

realmente pode influenciar o desempenho do servidor do sistema perante as *queries*, apresentando assim os principais determinantes no desempenho destes sistemas. Do estudo feito, conclui-se que muitos factores podem não estar directamente relacionados com a execução de *queries* ou até mesmo com a administração do sistema que as recebe, podendo num último caso ser um problema infra-estrutural.

Uma vez que o principal ponto de estudo é a execução de *queries* e posteriormente o desempenho do DW quando executa as operações necessárias para a sua resposta, concluímos que a ferramenta de *profiling* oferecida pelo sistema de gestão de base dados utilizado - Microsoft SQL Server 2008 - é talvez a melhor forma de colectar e monitorizar as variáveis de execução do sistema perante as *queries.* Nesta fase surge um incongruência relativa ao parâmetro CPU monitorizado pelo *profiler*. O problema está relacionado com o tempo de processamento do CPU que se apresenta ser bastante maior que o tempo de processamento global da *querie*. Sendo assim, optámos por excluir essa variável do conjunto de parâmetros a estudar no caso prático. Concluímos, também, através da análise dos determinantes no desempenho dos SDW que alguns parâmetros adicionais, relativos ao desempenho da infra-estrutura que suporta o sistema deveriam ser colectados para uma análise mais rigorosa. Apesar da influência que a camada de comunicações exerce na avaliação do desempenho de qualquer sistema, por questões técnicas, não foi possível recolher informação a esse nível. Aproveitando as capacidades de análise que o modelo dimensional oferece, optou-se por construir um *datamart* como meio de armazenamento da informação monitorizada acerca da execução das *queries* submetidas ao sistema alvo. Desta forma, construímos um primeiro modelo dimensional para um DW vocacionado para análises de desempenho, que possa permitir análises segundo várias directrizes. Por exemplo, cruzar informação relativa a execução de *queries* com informação relativa a tabelas ou utilizadores. A construção do modelo dimensional apresentou-se, sem dúvida, como um interessante e complexo desafio. Como em qualquer implementação de um DW, onde questões acerca do que recolher, do que armazenar e do que analisar se colocam, também na fase de modelação deste sistema todas essas dúvidas acabaram por surgir.

A previsão de desempenho é, talvez o módulo mais interessante do sistema implementado. Pretendia-se, primeiramente, construir perfis de interrogação com base nos parâmetros de execução das *queries*, através da utilização de algoritmos de *clustering*. Contudo, devido ao tamanho do conjunto de dados (relativamente pequeno para definir perfis) e ao escasso número

de utilizadores (problemas directamente relacionados com o sistema utilizado para a avaliação prática) tornou-se para nós inviável definir tais perfis. Uma primeira solução acaba por surgir se pensarmos em replicar o conjunto de treino, no entanto esta opção apresenta-se como uma ideia inconsistente, uma vez que ao fazer a replicação só estaríamos a aumentar o tamanho do conjunto de treino e não a criar uma maior heterogeneidade nos dados. Isto resultaria, obviamente, em *clusters* maiores mas com os mesmo itens (neste caso replicados). A replicação com heterogeneidade não se apresenta como uma opção válida, uma vez que estaríamos a falsear demais o conjunto de treino e a adapta-lo cada vez mais ao problema em causa.

Por fim, o cálculo de um índice de qualidade de serviço do sistema na respostas às *queries* submetidas não apresentou graves problemas. Uma vez que todo o trabalho até esta última fase foi concluído e, todos os problemas com maior ou com menor dificuldade foram ultrapassados, só nos restava então aplicar uma função de regressão que relacionasse todas as variáveis com vista a uma variável objectivo de forma a pudermos calcular um índice de desempenho.

6.3 Contribuições e Limitações do Trabalho Desenvolvido

6.3.1 Síntese dos Principais Contributos

A abordagem proposta para o cálculo da qualidade de serviço representa um primeiro contacto com o tema qualidade de serviço em SDW. Nesse sentido, esta dissertação representa somente um ponto de partida para a investigação dos factores que condicionam o desempenho de um DW. No entanto, o simples facto de ser ter concebido e implementado uma abordagem que, na nossa opinião, se pensa adequada ao problema representa um contributo científico e prático importante.

Actualmente, o sistema de cálculo de qualidade de serviço disponibiliza uma solução, que apesar de não ser completamente autónoma da intervenção humana, se apresenta não intrusiva, económica e interessante na problemática da análise do desempenho em SDW. De entre os diversos factores científicos e práticos da abordagem seguida, os seguintes destacam-se pela seu grau de inovação, a saber:

- Uma das primeiras abordagem ao tema da qualidade de serviço em SDW
- Um modelo de DW direccionado para a análise de desempenho

- Um conjunto de processos de tratamento e recolha de dados genérico cuja execução se mantém independente do servidor em questão
- Um módulo de previsão de desempenho

6.3.2 Síntese das Principais Limitações

As principais limitações da abordagem seguida advêm sobretudo de questões operacionais e problemas cuja resolução foi apenas equacionada a médio ou longo prazo. Desta forma destacam-se as seguintes limitações, a saber:

- A necessidade de optimização dos vários processos
- A expansão para outros sistemas de gestão de base de dados
- A optimização dos vários módulos, concretamente o módulo que fazem a colecta de dados com vista à expansão para novas métricas de avaliação, e o módulo de avaliação de desempenho
- Uma maior diversidade dos casos de estudo, como volumes de informação mais significativos e necessidades de análises mais especificas
- O cálculo trivial do índice global da qualidade de serviço

6.4 Linhas de Trabalho Futuro

Como seria de esperar, as linhas de trabalho futuro assentam na resolução das limitações actuais que o sistema apresenta. A constante evolução das organizações e necessidades, e consequentemente a evolução tecnológica, fazem com que o trabalho desenvolvido se encontre constantemente em fase de melhoramento devido aos diversos condicionantes. Nessa condição qualquer um dos módulos apresentados pode sofrer alterações importantes consoante a evolução dos sistemas.

A consolidação, a expansão e escalabilidade do sistema, assim como a aquisição de novo conhecimento acerca do desempenho de *queries*, constituem as linhas mais importantes de trabalho a realizar no futuro. Contudo, outras tarefas devem ser realizadas na tentativa de melhorar alguns aspectos pertinentes. Nesse sentido, pretende-se: a optimização dos processos já consolidados, garantindo a flexibilidade da solução perante possíveis extensões; a expansão do

sistema para interagir com outros sistemas de gestão de base de dados lideres de mercado; o refinamento do módulo de *clustering* de modo a produzir grupos de *queries* mais homogéneos; o refinamento do módulo de previsão de forma a garantir uma maior precisão nos valores retornados; o refinamento do módulo de monitorização assim como a sua expansão para a aquisição de novas métricas.

Por fim, não sendo resultado directo de limitações do sistema actual, como linha de trabalho futuro sugere-se a construção de uma plataforma OLAP de forma a aproveitar as potencialidades do modelo dimensional construído e a construção de um módulo de aconselhamento de execução de *queries* diárias.

Bibliografia

[Aggarwal and Philip 00] Aggarwal C.C., Philip S.Yu., "Finding Generalized Projected Clusters in High Dimensional Spaces", SIGMOD Record, 2000.

[Agrawal et al 98] Agrawal R., Gehrke J., Gunopulos D., Raghavan P., "Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications", In Proceedings of ACM SIGMOD, Washington, USA, 1998.

[Almasi and Gottlieb 88] Almasi GS., Gottlieb A., "Highly Parallel Computing", Benjamin-Cummings Publishing Company, USA, 1998.

[Antoshenkov 04] Antoshenkov G., "Byte-aligned Bitmap Compression", Technical Report, Oracle Corporation, USA, 2004

[Berkin 02] Berkhin P., "A Survey of Clustering Data Mining Techniques", Springer, 2002

[Boley 98] Boley D.L., "Principal Direction Divisive Partitioning", Data Mining and Knowledge Discovery, 1998.

[Calero et al 01] Calero C., Piattini M., Pascual C., Serrano M., "Towards Data Warehouse Quality Metrics.", In Proceedings of DMDW, 2001.

[Ceri et al 82] Ceri S., Negri M., Pelagatti G, "Horizontal Data Partitioning in Database Design", Proceedings of the 1982 ACM, Florida, 1982.

[Chan and Ioannidis 98] Chan CY., Ioannidis E., "Bitmap Index Design and Evaluation", Washington, ACM Press, 1998.

[Chan and Ioannidis 99] Chan CY., Ioannidis E., "An Efficient Bitmap Encoding Scheme for Selection Queries", SIGMOD Conference, ACM Press, 1999.

[Chapman et al 99] Chapman P., Clinton J., Khabaza T., Reinart T., Wirth R., "The CRISP-DM Process Model and User Guide", [Available Online]: http://www.crisp-dm.org/CRISPWP-0800.pdf, 1999.

[Chaudhuri et al 95] Chaudhuri S., Krishnamurthy R., Potamianos S., Shim K., "Optimizing Queries with Materialized Views", Proceedings of the 5th International Conference on Data Engineering, 1995.

[Chen and Yu 96] Chen M.S., Yu P. S., ”Data mining: An overview from a database perspective”, IEEE Transactions on Knowledge and Data Engineering, 1996.

[Dempster et al 77] Dempster A., Laird N., Rubin D., "Maximum Likelihood from Incomplete Data Via the EM Algorithm", Journal of the Royal Statistical Society, 1977.

[DeWitt and Gray 92] DeWitt DJ., Gray J., "Parallel Database Systems: The Future of High Performance Database Systems", Communications to ACM, 1992.

[Efron and Tibshirani 93] Efron B., Tibshirani R., “An Introduction to the Bootstrap”, Chapman and Hall, 1993.

[Elhardt and Bayer 84] Elhardt K., Bayer R.,"A Database Cache for High Performance and Fast Restart in Database Systems", ACM Transactions on Database Systems, New York, 1984

[Fayyad et al 96] Fayyad U.M., Piatetsky-Shapiro G., Smyth P., “From Data Mining to Knowledge Discovery: An Overview”, Advances in Knowledge Discovery and Data Mining, 1996.

[Frawley et al 91] Frawley W. J., Piatetsky-Shapiro G., C. J. Matheus C. J. "Knowledge Discovery in Databases: An overview", Knowledge Discovery in Databases, 1991.

[Graefe 93] Graefe G., "Query Evaluation Techniques for Large Databases", New York, ACM, 1993.

[Guha et al 98] Guha S., Rastogi R., Shim K., "CURE: A Clustering Algorithm for Large Databases", International Conference on Management of Data, 1998.

[Gupta and Mumick 95] Gupta A., Mumick IS., "Maintenance of Materialized Views: Problems, Techniques, and Applications", IEEE Computer Society, Vol. 18, No. 2, 1995.

[Han et al 05] Han J., Kamber M., Pei J., "Data Mining: Concepts and Techniques", 2ª Edição, Morgan Kaufmann, 2005.

[Hartigan 75] Hartigan J.A., "Clustering Algorithms", Wiley, 1975.

[Hartigan and Wong 79] Hartigan J.A., Wong M., "Algorithm as136: A K-Means Clustering Algorithm", Applied Statistics, 1979.

[Inmon W 05] Inmon W., "Building the data warehouse", Wiley, 2005.

[Jain and Dubes 88] Jain A., Dubes C., "Algorithms for Clustering Data", Prentice-Hall, NJ, 1988.

[Jarke and Vassiliou 97] Jarke M., Vassiliou Y., "Data Warehouse Quality: A Review of the DWQ Project", Proceedings of the Conference on Information Quality, Cambridge, MA, 1997.

[Johnson and Wichern 82] Johnson R.A., Wichern D.W., "Applied Multivariate Statistical Analysis", Englewood Cliffs, NJ Prentice-Hall, 1982.

[Johnson 99] Johnson T., "Performance Measurements of Compressed Bitmap Indices", International Conference on Very Large Data Bases, Morgan Kaufmann, 1999.

[Joshi el al 03] Joshi K.P., Joshi A., Yesha Y., "On Using a Warehouse to Analyze Web Logs", Jornal of Distributed and Parallel Databases, Kluwer Academic Publishers, Vol. 13, No. 2, 2003.

[Joshi el al 99] Joshi K.P., Joshi A., Yesha Y., Krishnapuram R., "Warehousing and Mining Web Logs", Proceedings of the 2th ACM CIKM Workshop on Web Information and Data Management, 1999.

[Kaufman and Rousseeuw 90] Kaufman L., Rousseeuw P.J., "Finding Groups in Data: An Introduction to Cluster Analysis", Wiley, New York, 1990.

[Kimball and Caserta 04] Kimball R., Caserta J., "The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming and Delivering Data", 1ª Edição, Wiley, 2004

[Kimball and Ross 02] Kimball R., Ross M., "The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling", 2ª Edição, Wiley, 2002

[Kohavi 95] Kohavi R., "A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection", In Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, San Francisco, 1995.

[Koudas 00] Koudas N., "Space Efficient Bitmap Indexing", Conference on Information and Knowledge Management, USA, ACM Press, 2000

[Kumar et al 94] Kumar V., Grama A., Gupta A., Karypis G., "Introduction to Parallel Ccomputing: Design and Analysis of Algorithms", Redwood City, Benjamin-Cummings Publishing Co., 1994.

[Liu 07] Liu B., "Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data", 1ª Edição, Springer, 2007.

[MacLachlan and Krishnan 96] McLachlan G.J., Krishnan T., "The EM Algorithm and Extentions", Wiley, 1996.

[Mehta and DeWitt 95] Mehta M., DeWitt DJ., "Managing Intra-operator Parallelism in Parallel Database Systems", Proceedings of the 21th International Conference on Very Large Databases, 1995.

[Murtagh 83] Murtagh F., "A Survey of Recent Advances in Hierarchical Clustering Algorithms", Computer Journal, 1983.

[Navathe et al 84] Navathe S., Ceri S., Wiederhold G., Dou J., "Vertical Partitioning Algorithms for Database Design", ACM Transactions on Database Systems, 1984.

[Rahm 93] Rahm E., "Parallel Query Processing in Shared Disk Database Systems", SIGMOD Records, 1993.

[Rotem et al 05] Rotem D., Stockinger K., Wu K., "Optimizing I/O Costs of Multi-Dimensional Queries using Bitmap Indices", International Conference on Database and Expert Systems Applications, Denmark, Springer Verlag, 2005.

[Stockinger et al 04] Stockinger K., Wu K., Shoshani A., "Evaluation Strategies for Bitmap Indices with Binning", International Conference on Database and Expert Systems Applications, Spain, Springer-Verlag, 2004.

[Tukey 77] Tukey J.W., "Exploratory Data Analysis", Addison-Wesley, 1977.

[Ullman 88] Ullman J., "Principles of Database & Knowledge-Base Systems Vol. 1: Classical Database Systems", Computer Science Press, 1ª Edição, 1988.

[Urman and Rinaldi 98] Urman S., Rinaldi W., "Oracle PL/SQL Programming", Osborne/McGraw-Hill Berkeley, CA, 1997.

[Valduriez 93] Valduriez P., "Parallel Database Systems: The Case for Shared-Something", Proc. 9th Int. Conf. on Data Engineering, 1993.

[Vassiliadis 00] Vassiliadis P., "Data Warehouse Modeling and Quality Issues", PhD Thesis, NTUA, Athens, 2000.

[Vilfredo and Schwier S71] Pareto V., Schwier S., "Manual of Political Economy", M. Kelley, 1971.

[Windham 82] Windham M.P., "Cluster Validity for the Fuzzy C-Means Clustering Algorithm", IEEE, 1982.

[Witten and Frank 05] Witten IH., Frank E., "Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations", 2ª Edição, Morgan Kaufmann, 2005.

[Wu et al 05] Wu K., Otoo E., Shoshani A., "An Efficient Compression Scheme for Bitmap Indices", ACM Transactions on Database Systems, 2005.

[Wu and Buchmann 98] Wu MC., Buchmann A., "Encoded Bitmap Indexing for Data Warehouses". International Conference on Data Engineering, Florida, IEEE Computer Society Press, 1998.

Referências WWW

[1] http://www.crisp-dm.org/

Este *site* fornece acerca do processo de implementação de sistemas de mineração de dados, sendo possível encontrar toda a documentação sobre todas as fases de implementação.

[2] http://www.sas.com/

O *site* da ferramenta SAS apresenta uma elevada quantidade de informação acerca da ferramenta em si, assim como de todos os seus componentes. É possível encontrar uma vasta gama de recursos que vai desde as versões passando por grupos de discussão.

[3] http://www.spss.com/

O *site* da ferramenta SPSS apresenta uma elevada quantidade de informação acerca da ferramenta em si, assim como de todos os seus componentes. É possível encontrar uma vasta gama de recursos que vai desde as versões passando por grupos de discussão.

[4] http://www.insightful.com/

O *site* da ferramenta S-PLUS apresenta uma elevada quantidade de informação acerca da ferramenta em si, assim como de todos os seus componentes. É possível encontrar uma vasta gama de recursos que vai desde as versões passando por grupos de discussão.

[5] http://www.mathworks.com/

O *site* da ferramenta MatLab apresenta uma elevada quantidade de informação acerca da ferramenta em si, assim como de todos os seus componentes. É possível encontrar uma vasta gama de recursos que vai desde as versões passando por grupos de discussão.

[6] http://www.wolfram.com/

O *site* da ferramenta Mathematica apresenta uma elevada quantidade de informação acerca da ferramenta em si, assim como de todos os seus componentes. É possível encontrar uma vasta gama de recursos que vai desde as versões passando por grupos de discussão.

[7] http://repositorium.sdum.uminho.pt/

O RepositóriUM é o repositório institucional da Universidade do Minho, constituído com o objectivo de armazenar, preservar, divulgar e dar acesso à produção intelectual da Universidade do Minho em formato digital. O RepositóriUM pretende reunir, num único sítio, o conjunto das publicações científicas da UM contribuindo desse modo para o aumento da sua visibilidade e impacto e garantindo a preservação da memória intelectual da Universidade.

[8] http://rapid-i.com/

O *site* da ferramenta *RapidMiner* onde é disponibilizada toda a documentação para uma boa utilização dos seus componentes. Para além de toda a documentação é disponibilizada uma versão gratuita assim como a respectiva verão paga.

[9] http://www.java.sun.com/

Este *site* fornece uma elevada quantidade de informação acerca da linguagem orientada aos objectos Java. É possível encontrar uma vasta gama de recursos que vai desde as versões mais actuais do Java, passando por uma extensa documentação, até grupos de discussão e investigação.

[10] http://www.gartner.com/technology

Site de uma das empresas lideres mundiais de consultoria em tecnologias de informação. O seu conteúdo abrange dezenas de publicações e estudos sobre os mais diversos temas, incluindo os famosos quadrantes mágicos que possibilitam a comparação entre os diversos lideres de mercado nas diversas áreas.

[11] http://msdn.microsoft.com/en-us/library/ms181091.aspx

Site da ferramenta de *profiling* do Microsoft SQL Server 2008. O seu conteúdo fornece a mais diversa gama de recursos que vai desde a documentação até exemplos práticos.

ANEXOS

ANEXO I Caracterização das tabelas do *data webhouse.*

ANEXO I - Caracterização do *data webhouse*

A informação representada na Tabela A.1 descreve todas as tabelas do *data webhouse* alvo do estudo sobre a qualidade de serviço.

Tabela	Atributos	Descrição	Tipo
Tabela de Factos	Session ID	Identificador de Sessão.	Chave Primária
	StartTime	Data e Hora do inicio da sessão	
	EndTime	Data e Hora do fim da sessão	
	StartDate	Data do inicio da sessão	Chave Estrangeira
	EndDate	Data do fim da sessão	
	StartHour	Hora do inicio da sessão	
	EndHour	Hora do fim da sessão	
	Referrer	Site que referenciou a página	
	Agente	Programa utilizado para efectuar os pedidos	
	Host	IP do utilizador que efectuou o pedido	
	Pedidos	Número de pedidos feitos na sessão	Medidas
	m_gets	Número de pedidos com o método Get	
	m_post	Número de pedidos com o método Post	
	m_head	Número de pedidos com o método Head	
	m_outros	Número de pedidos com outros métodos	
	Status_Sucesso	Número de pedidos respondidos com sucesso	
	Status_Redirecionado	Número de pedidos redireccionados	
	Status_Erro	Número de pedidos cuja resposta foi erro	
	Bytes_Enviados	Número de bytes enviados pelo servidor	
	Session_Duration	Tempo de duração da sessão	
Tempo	Hora	Chave Identificadora da dimensão tempo. Indica a hora. Exemplo "8".	
	Período	Período do dia. Exemplo "Manha"	
Calendário	Data ID	Chave identificadora do dia. Exemplo "01-01-08"	
	Dia Semana	Nome do dia semana. Exemplo "Terça"	
	Dia_Semana_nr	Número do dia da semana. Exemplo "3"	
	Dia_mes_nr	Número do dia no mês. Exemplo "1"	
	Dia_ano_nr	Número do dia no ano. Exemplo "1"	

	Tipo_Dia	Fim-de-semana ou dia de trabalho. Exemplo "Semana".
	Semana_nr	Numero da semana no ano. Exemplo "1".
	Mês	Nome do mês. Exemplo "Janeiro".
	Mês_nr	Número do mês. Exemplo "1"
	Quarter	Trimestre correspondente a data. Exemplo "1"
	Semestre	Semestre corresponde a data. Exemplo "1"
	Ano	Ano da data. Exemplo "2008".
Host	IPadress	IP do cliente que fez o pedido. Exemplo "89.84.45.174"
	CodigoISOPais	Código do pais do cliente. Exemplo "PT"
	Pais	Pais de onde foi feito o pedido. Exemplo "Portugal".
Referrer	Referrer ID	Atributo identificador do referenciador. Exemplo "1"
	Tipo	Diz se o referenciador é uma página do próprio site, ou uma página externa. Exemplo "externa".
	URI	Endereço WWW do site que referenciou. Exemplo "www.google.pt"
	Motos Pesquisa	Indica se o referente é ou não um motor de busca. Exemplo "y".
Request	Request ID	Atributo identificador do pedido. Exemplo "1".
	request	Texto do pedido que ficou registado no ficheiro de *log.* Exemplo "/".
	Tipo	Tipo de ficheiro pedido. Exemplo "html"
	Classe	Classe a que pertence o ficheiro pedido
Agente	Agente ID	Atributo identificador do agente. Exemplo "1".
	Tipo	Tipo de agent. Exemplo "crawler".
	Agent_Name	Nome do agent. Exemplo "googleBot".
	Texto_Log	Texto existente no ficheiro de log referente ao agente

Tabela A.1: Caracterização das Tabelas do *data webhouse*