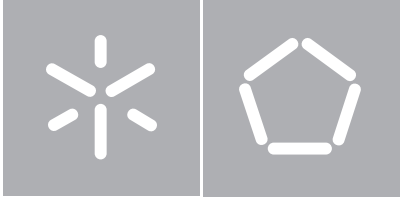

Universidade do Minho
Escola de Engenharia

José Luís Pereira Novais

Benchmark de Bases de Dados de Suporte a Serviços de Informação

Tese de Mestrado
Mestrado em Sistemas de Informação

Trabalho efectuado sob a orientação de
Professor Doutor Leonel Duarte dos Santos

Outubro de 2006

DECLARAÇÃO

Nome: José Luis Pereira Novais

Endereço electrónico: josenovais@josenovais.com Telefone: 966387461

Número de Bilhete de Identidade: 10524512

Título tese: *Benchmark* de bases de dados de suporte a serviços de informação

Orientador: Leonel Duarte dos Santos Ano de conclusão: 2006

Designação do Mestrado: Sistemas de Informação

Universidade do Minho, 23 de Outubro de 2006

_____

José Luis Pereira Novais

Agradecimentos

A conclusão deste trabalho não teria sido possível sem o apoio de várias pessoas.

Gostaria de expressar um agradecimento muito especial à minha família pelo apoio incondicional e paciência demonstrados desde o primeiro momento.

Gostaria de deixar um reconhecimento especial ao Manuel Portelinha, que terá sido das pessoas mais persistentes e empenhadas em que eu conseguisse atingir o objectivo, e de quem, tal como com o Doutor Leonel Santos, sempre foi possível obter aconselhamento e ajuda nos momentos mais difíceis. A ambos, muito obrigado.

Um agradecimento também a todos os meus amigos que sempre me incentivaram e com os quais sempre pude contar.

Por último, um agradecimento a todos que de um forma ou de outra colaboraram neste trabalho.

Benchmark de bases de dados de suporte a serviços de informação

Resumo

Os serviços de informação podem ser considerados como aplicações especialmente vocacionadas para a recolha, armazenamento, tratamento e disseminação de informação, que podem ser facilmente associados à Internet ou similar, e cuja utilização é feita por parte de públicos variados. O uso crescente da Internet como forma de comunicação e a consequente necessidade de disponibilização de informação em cada vez maiores quantidades requer aplicações eficazes, capazes de responder às solicitações de um grande número de utilizadores. Desta forma, um aspecto crucial para o bom desempenho das aplicações, como os serviços de informação, é a forma como armazenam a informação, mais concretamente o modelo que usam para o fazer.

A utilização de bases de dados é sem dúvida a forma mais comum para o armazenamento de informação necessária ao funcionamento das aplicações, sendo o modelo relacional, já com muitos anos de utilização, o mais conhecido. No entanto, uma nova abordagem para a representação de informação é o XML, o qual tem ganho uma cada vez maior aceitação.

Uma vez que estas duas abordagens para o armazenamento de informação, o modelo relacional em oposição ao XML, são extremamente relevantes, neste trabalho é feita uma análise comparativa de desempenho, sendo definido um *benchmark*, com o objectivo de identificar situações onde o uso de uma poderá ter vantagens relativamente ao uso da outra num contexto dos serviços de informação. Isto é feito com o recurso a um sistema de testes, baseado em sistemas existentes mas construído de raiz com vista a dar resposta às necessidades deste trabalho.

Os resultados obtidos apontam para um desempenho superior do modelo relacional. No entanto, conclui-se que há situações eventualmente mais favoráveis para o uso de XML, onde o modelo relacional poderá ser inferior, pelo que selecção do modelo a usar terá de ser feita tendo consciência das situações onde cada um é potencialmente mais indicado.

Benchmark for databases supporting information services

Abstract

Online information services can be considered as applications specially oriented to retrieving, storing, treating and disseminating information, that can be easily associated with the internet (or similar) and which utilization is done by many types of public (general public or more restricted public). The growing use of internet as a mean of communication and the resulting need for greater amounts of information requires efficient applications able to respond to the solicitations of a potentially great number of users. Because of this, a crucial aspect to the good performance of applications like online information services is the way the information is stored, more precisely the model used to do this.

The use of databases is the most common way to store information needed by the applications and the relational model, with many years of utilization, is the most well known. A new approach to information representation is the XML that has been gaining a growing acceptance.

Because these two approaches for information storage, the relational model and XML, are very important, in this work it was made a comparative analysis of performance, by defining a benchmark. This is done with the goal of identifying the situations where each of this approaches are more adequate in the context of online information services. This is done by using a test system inspired in existing systems, but built from scratch with the goal of satisfying the needs of the present work.

The results obtained show a superior performance of the relational model. However, one concludes that there are situations more favorable to XML where the relational model could have an inferior performance. Because of this, the selection of a model should be done with the knowledge of these situations where one model has a potential superior performance than the other.

Índice

Índice de figuras

Índice de tabelas

Índice de gráficos

Índice de exemplos

Capítulo 1

1 Introdução

1.1 Motivação

Actualmente, com o aumento da utilização da Internet e tecnologias associadas como um meio de comunicação privilegiado e com a diversificação das aplicações que vão sendo colocadas *online*, torna-se necessária a disponibilização de cada vez maiores quantidades de informação a um cada vez maior número de utilizadores. O armazenamento e recuperação de forma eficiente de grandes quantidades de informação assumem assim um papel extremamente importante no bom desempenho das aplicações.

Existindo no projecto *DeGóis – Plataforma Nacional de Ciência e Tecnologia*, uma grande quantidade de informação referente a currículos de investigadores portugueses, é necessário armazená-la e torná-la acessível *online*. Tendo uma noção clara do que é um sistema que permite executar esta tarefa, é importante investigar a forma mais eficaz de o fazer. Nesta perspectiva, torna-se importante conhecer a tecnologia existente e qual o seu comportamento de forma a que sejam construídos sistemas o mais eficientes possível.

Podendo à partida os conceitos envolvidos não ser muito claros para o público em geral torna-se necessário defini-los. É necessário definir o que é um serviço de informação e quais os seus objectivos. Da mesma forma, é necessário definir o que são repositórios de informação. Questões como os componentes que os constituem, se se trata apenas de uma base de dados (BD) com capacidades de pesquisa melhoradas ou se será algo mais e também inclui interface com o utilizador, etc., deverão ser clarificadas. É ainda necessário definir a relação entre serviços de informação e repositórios, e acima de tudo, averiguar qual a forma mais eficaz de armazenamento da informação nos repositórios, através do estudo dos modelos existentes para esta tarefa.

1.2 Enquadramento

Num projecto como aquele onde se enquadra o presente trabalho de dissertação, que lida com grandes quantidades de informação, esta pode ser disponibilizada de várias formas. Para este efeito, a disponibilização de informação, podem ser utilizadas BDs onde a informação é representada segundo vários modelos como o orientado por objectos, hierárquico ou relacional. O modelo relacional [Codd 1970], onde a informação é organizada em tabelas, registos e atributos, é talvez o modelo mais divulgado actualmente para armazenar informação. Existem vários Sistemas de Gestão de Bases de Dados (SGBD) que suportam este modelo, sendo os mais conhecidos o Oracle, SQL Server da Microsoft e o DB2 da IBM [Bourret 2006]. Esta informação pode também ser disponibilizada em XML [W3C 2004a]. Uma vez que este tem assumido um papel cada vez mais relevante na representação da informação, deverá ser dada importância a repositórios com suporte para informação representada desta forma.

O XML, que está vocacionado para a troca de informação estruturada, é um padrão que pode desempenhar um papel importante na troca de informação entre aplicações, isto é, na sua interoperabilidade. Permite criar um mecanismo a nível aplicacional para produzir informação que as outras aplicações conseguem utilizar sem necessidade de conhecer nada sobre a aplicação que transmite essa informação (Figura 1). [Linthicum 2001]

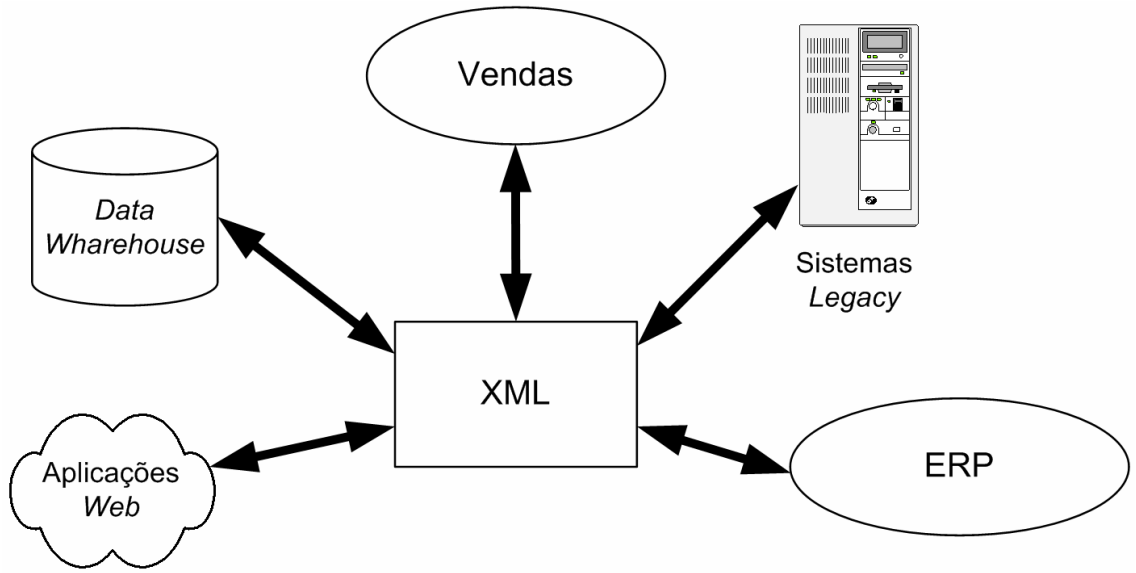

Figura 1: Comunicação com base no XML (baseado em [Linthicum 2001])

Este mecanismo comum para troca e gestão de informação é relevante no âmbito da integração de aplicações, nomeadamente em duas áreas de estudo fortemente relacionadas entre si: EAI (*Enterprise Application Integration*) e integração de aplicações B2B (*Business to business Application Integration*). Apesar deste forte relacionamento e partilharem técnicas e ferramentas, o domínio do problema que abordam é diferente. Por um lado EAI foca-se habitualmente na integração de aplicações e fontes de dados ao nível local de uma organização. Por outro lado, a integração de aplicações B2B foca-se num domínio inter-organizacional, isto é, na integração de sistemas de diferentes organizações com o objectivo de satisfazer as necessidades inerentes ao negócio, como a partilha de informações entre parceiros comerciais [Linthicum 2001]. Neste contexto, o XML fornece a base para a criação de padrões, utilizados no comércio electrónico, como o ebXML [OASIS 2004] ou cXML [cXML 1999], que assumem grande importância actualmente [Linthicum 2001].

Com a utilização do XML, a necessidade de o manter persistente surge naturalmente, sendo as BDs ferramentas que podem ser utilizadas nesta tarefa. O XML pode ser utilizado em BDs de duas formas [Steegmans et al. 2004]. Numa primeira perspectiva, os documentos XML são completamente externos à BD sendo apenas armazenada a informação do documento em si e não qualquer informação sobre a sua estrutura (como elementos e atributos). As BDs que lidam com o XML desta forma são denominadas

BDs com suporte para XML[1] [Steegmans et al. 2004]. Na outra perspectiva cada documento é preservado como um todo. Cada documento pode igualmente ser traduzido para o modelo interno utilizado pela BD, mas preservando ainda assim toda a informação respeitante à sua estrutura, como elementos e atributos. Às BDs que utilizam o XML desta forma dá-se o nome de nativas XML [Steegmans et al. 2004].

Existe ainda um aspecto relevante quando se fala de XML. Este aspecto refere-se ao tipo de documento, o qual pode ter dois modelos distintos [Obasanjo 2001; Bourret 2005b]: centrado nos dados e centrado no documento. Este aspecto pode ter alguma influência na escolha da BD para manipular XML. No modelo centrado nos dados inserem-se documentos que utilizam o XML como forma de transporte de informação [Bourret 2005b], sendo destinados predominantemente a consumo por parte de computadores e não a uso por parte de humanos e onde normalmente a ordem pela qual os vários elementos com o mesmo nível na estrutura são representados é irrelevante. No modelo centrado no documento inserem-se documentos destinados ao consumo de pessoas [Obasanjo 2001; Bourret 2005b], os quais são caracterizados por possuir uma estrutura irregular sendo, tratados como um todo e onde a ordem pela qual os vários elementos com o mesmo nível na estrutura são representados é normalmente relevante.

Nas situações onde a aplicação tem uma visão centrada nos dados, a adopção de BDs com suporte para XML tem tendencialmente um melhor desempenho [Vakali et al. 2005]. Por outro lado, numa uma visão centrada no documento, onde existe a necessidade da recuperação dos documentos na sua forma original, as BDs nativas XML são a solução mais adequada nestes casos [Nambiar et al. 2002b; Vakali et al. 2005].

Cada uma das abordagens possíveis para o armazenamento de informação tem as suas particularidades no que diz respeito à consulta e acesso à informação. Nesta perspectiva a selecção entre os modelos, e consequentemente vários produtos e tecnologias existentes deve ser feita de forma consciente, apoiada em medidas de desempenho tendo como base cenários de utilização o mais reais possíveis.

[1] O termo utilizado em língua inglesa é *XML-Enabled*. Neste trabalho irá ser usada a expressão: com suporte para XML.

1.3 Objectivos

Este trabalho tem como base um caso de estudo, mais concretamente a informação referente a currículos de investigadores portugueses no âmbito do projecto DeGóis – *Plataforma Nacional de Ciência e Tecnologia*. Neste contexto, para o armazenamento de informação por parte dos repositórios, um bom ponto de partida será considerar-se modelos tecnológicos baseados nos modelos relacional e XML. No entanto, tendo em atenção uma base teórica que deverá ser construída antes do início do trabalho de índole prática e que o fundamenta, poderiam ser considerados outros modelos tecnológicos. De qualquer das formas, não se pretende tornar este trabalho um estudo exaustivo de todas possibilidades existentes, sendo o seu foco antes os modelos referidos, podendo desta forma ser a base para qualquer outro estudo futuro mais alargado.

Com base nos modelos tecnológicos referidos, pretende-se delinear casos de utilização que se pretendem os mais aproximados da realidade possível definindo-se assim um modelo de testes. Pretende-se da avaliação conduzida com o modelo de testes, descobrir quais o factores que foram mais relevantes no desempenho de cada tecnologia e em que cenários é que estas são mais eficazes em termos de acesso à informação e consultas, uma vez que o objectivo é a disponibilização da informação *online* em serviços com potencialmente elevada carga de utilização.

Desta forma, os principais objectivos deste trabalho passam pelo estudo do comportamento de soluções a nível de tecnológico em vários ambientes de utilização, fazendo-se uma análise comparativa em termos de desempenho, com a definição de um *benchmark*, bem como a obtenção de resultados e conclusões que possam ser tomados em conta na escolha mais adequada de uma solução para cada cenário.

1.4 Metodologia

Na execução de um trabalho como este é usual passar-se por várias fases, desde de índole mais teórica, passando por fases de índole prática e culminando com a análise dos resultados obtidos. Desta forma, inicialmente é feita uma revisão de literatura com os objectivos principais de obter uma visão geral das áreas de estudo em questão e estabelecer de seguida que aspectos são os mais importantes e para os quais deverá estar

centrada a atenção na execução do trabalho. No presente caso, as principais áreas envolvidas são os serviços de informação *online*, o armazenamento de informação, com destaque para o uso do XML para este fim, bem como testes de performance – *benchmarks*.

Com esta visão obtida das áreas envolvidas e dos aspectos mais importantes, torna-se possível construir um modelo para os testes que é necessário realizar. Este modelo será concretizado na fase seguinte, de ordem pratica, através da concepção e construção de um sistema capaz de executar os testes pretendidos. Para este sistema é definida uma arquitectura onde são definidas regras de funcionamento e respectivas peças que o constituem, bem como quais os factores relevantes a medir, procedendo-se posteriormente à sua implementação.

Além da concepção e implementação deste sistema, é necessário também proceder ao tratamento da informação com a qual serão executados testes. Este tratamento, que é fundamental para que a informação seja adequada aos testes que se pretendem realizar, é feito com recurso a ferramentas criadas para o efeito que permitem gerar as quantidades necessárias de informação e converte-la para o formato apropriado.

Após a implementação do sistema e o necessário tratamento da informação, são realizados os testes, cujos resultados são registados para posterior análise. Desta análise final dos resultados obtidos são identificados os factores que mais os influenciaram de forma a serem tiradas conclusões e a serem apontados caminhos de investigação futura.

1.5 Organização

Este trabalho está organizado em 3 partes. Na primeira parte, nos capítulos 2, 3 e 4, é feita uma revisão de literatura que permite construir uma base teórica onde se apoia o trabalho de índole prática desenvolvido. Este é descrito nos capítulos 5, 6 e 7 que abordam o sistema concebido para a realização de testes, aspectos da sua implementação e finalmente os resultados obtidos. Por último, na terceira parte, são extraídas conclusões e apontadas direcções para trabalhos futuros. Os capítulos deste trabalho são descritos de seguida.

Capítulo 1: Introdução

Este capítulo descreve a motivação, o enquadramento e os objectivos associados ao presente trabalho. Existindo no projecto onde este trabalho se enquadra grandes quantidades de informação que é necessário armazenar e tornar acessível *online*, torna-se relevante o estudo de formas de armazenamento de informação que possam servir de suporte eficaz a serviços de informação.

Capítulo 2: Serviços de informação e Repositórios de informação

Neste capítulo são descritos serviços de informação e tenta-se estabelecer uma relação entre estes e os repositórios de informação. São ainda descritos alguns exemplos do uso do termo repositório.

Capítulo 3: Armazenamento da informação

É possível armazenar informação de várias formas, no entanto o uso de BDs é uma solução bastante comum nas aplicações actuais. Este capítulo descreve os modelos mais utilizados para o armazenamento de informação nas BDs. Dado que o XML tem assumido uma relevância crescente na representação de informação, este capítulo aborda também o XML no que diz respeito à representação de informação e as implicações relativas ao seu armazenamento em BDs.

Capítulo 4: *Benchmarks*

Com este capítulo pretende-se identificar os princípios fundamentais aos quais um sistema para a execução de testes deverá obedecer. Isto é feito com o estudo de vários *benchmarks* orientados para os modelos de armazenamento de informação identificados no capítulo anterior e que estarão na base da concepção e implementação de um sistema de testes, descrito no capítulo seguinte.

Capítulo 5: Concepção de um sistema de testes

Este capítulo descreve os cenários que se pretendeu testar com a construção do sistema de testes, bem como as suas características.

Capítulo 6: Aspectos da Implementação

Neste capítulo são descritos 3 aspectos importantes, de ordem prática na implementação, que foi necessário considerar na implementação do sistema descrito no capítulo anterior. Estes aspectos são a criação de documentos XML com base numa BD relacional, as ferramentas utilizadas e que foram criadas especialmente com vista a

satisfazer as particularidades deste trabalho e as BDs seleccionadas que implementam os modelos que era proposto estudar.

Capítulo 7: **Descrição e análise de resultados**

Este capítulo descreve os resultados obtidos com a execução de testes utilizando o sistema descrito no capítulo 5. São apresentados gráficos onde se compara o desempenho das soluções testadas. De seguida é feita uma análise onde se tenta interpretar os resultados obtidos. Estes apontam para um desempenho superior do modelo relacional, no entanto há situações onde o XML poderá ser mais adequado.

Capítulo 8: **Conclusão e trabalho futuro**

Neste capítulo é feita uma análise do trabalho executado, bem como são extraídas conclusões e apontadas direcções para trabalhos futuros. Os resultados obtidos poderão ter sido influenciados por questões a nível da afinação precisa das BDs utilizadas e das características da informação utilizada. Desta forma, existirá interesse em realizar trabalhos futuros que se proponham testar principalmente este último aspecto.

Anexos

Deste trabalho fazem parte 3 anexos. No primeiro, Anexo I, são descritas as pesquisas utilizadas nos testes realizados neste trabalho. Para cada pesquisa existem diferentes versões, uma vez que nos testes foram utilizadas diferentes tipos de BDs. Por este facto, foram utilizados diferentes esquemas: um relacional (cujo diagrama entidades/ relacionamentos, ER, parcial está representado no Anexo II) e um XML (Anexo III). No Anexo IV são apresentados os valores obtidos nos testes realizados.

Capítulo 2

2 Serviços de informação e Repositórios de informação

Neste capítulo são descritos serviços de informação e tenta-se estabelecer uma relação entre estes e os repositórios de informação. São ainda descritos alguns exemplos do uso do termo repositório.

2.1 Serviços de informação

Os serviços de informação *online*[2] são uma área relevante dos sistemas de informação que justificam por si só estudo próprio. Desta forma, a ACM (*Association for Computing Machinery*) no seu sistema de classificação (*Computing Classification System* – CCS)[ACM 1998] inclui, há vários anos, o ramo serviços de informação *online*:

H. Sistemas de Informação

[2] Neste trabalho, à excepção desta secção do presente capítulo, utiliza-se a expressão "serviços de informação", assumindo-se sempre que se trata de "serviços de informação *online*" conforme a definição dada nesta secção.

H.3 Armazenamento e Recuperação de Informação

H.3.5 Serviços de Informação *Online*

Este ramo, por sua vez está dividido em serviços comerciais, partilha de dados e serviços baseados na *web*.

Na literatura existem algumas definições onde se faz a distinção entre serviços de informação e serviços de informação *online*. Um serviço de informação pode ser visto como uma organização especializada, ou um departamento de uma organização cuja principal função é a recolha, armazenamento e transmissão de informação para pessoas, departamentos, ou organizações exteriores ao serviço de informação [Maguire et al. 1994]. Em [Heijden 2002] define-se serviço de informação não como uma organização ou departamento, mas antes como o fornecimento de dados sobre um determinado tema ou conjunto de temas relacionados, por parte de uma organização ou indivíduo com vista à sua utilização por parte de outras organizações ou indivíduos. Além disto, em [Heijden 2002] define-se ainda que um serviço de informação *online* é acessível aos seus utilizadores através de dispositivos tecnológicos como computadores, *browsers*, telefones móveis, PDA ou outros. Nesta perspectiva, um serviço de informação *online* pode ser visto como um sistema electrónico *online* cuja principal função é a recolha, armazenamento, tratamento e disseminação de informação para pessoas ou organizações [Santos 2004].

Existem inúmeros exemplos de serviços de informação *online* destinados a diferentes públicos e sobre os mais variados assuntos. Estes podem variar desde os mais vulgares como viagens, cultura ou tempo, para um público mais genérico aos mais específicos como informação estatal ou legislação destinados a um público mais restrito. Estes serviços podem ser da responsabilidade de instituições públicas ou privadas, organismos do estado, empresas, entre outros.

Em [Heijden 2002] são definidos ainda os objectivos principais dos serviços de informação *online*, bem como, as principais diferenças para os sistemas de informação. A geração de lucro poderá ser um objectivo importante para as organizações que disponibilizam serviços de informação *online*, uma vez que muitos serviços são utilizáveis mediante pagamento ou disponibilizando mensagens publicitárias. Um outro objectivo é centrar as atenções em outros serviços que de outra forma poderiam escapar à maioria dos utilizadores *online*. O serviço de informação *online* poderá também ser

criado com o objectivo de fazer passar para o exterior uma imagem de experiência numa determinada área por parte da organização.

Ao contrário dos sistemas de informação comuns, os serviços de informação podem operar num ambiente competitivo com outros serviços de informação, isto é, podem existir vários com o mesmo objectivo, o que faz com que sejam mais vulneráveis às leis do mercado. Além disto, [Heijden 2002] defende também que os sistemas de informação comuns são concebidos e implementados para desempenharem as tarefas de forma eficaz e eficiente, estando de certa forma associados a questões de trabalho, o mesmo não se passando com muitos serviços de informação, os quais podem estar associados a questões de outros âmbitos. Outro aspecto importante é o seu conteúdo ser fornecido tipicamente através de um sistema de informação.

Apesar de um serviço de informação *online* não ser obrigatoriamente uma aplicação *web*, esta é sem dúvida uma associação que pode ocorrer facilmente quando se fala em serviços de informação *online*. A concepção de aplicações *web* com grandes quantidades de informação é uma tarefa complexa, existindo para tal várias metodologias [Bommel 2003], no entanto o denominador comum é sempre a necessidade de armazenamento de informação. Na Figura 2 é apresentada uma aplicação *web* que é classificada como serviço de informação *online* pela instituição que a disponibiliza.

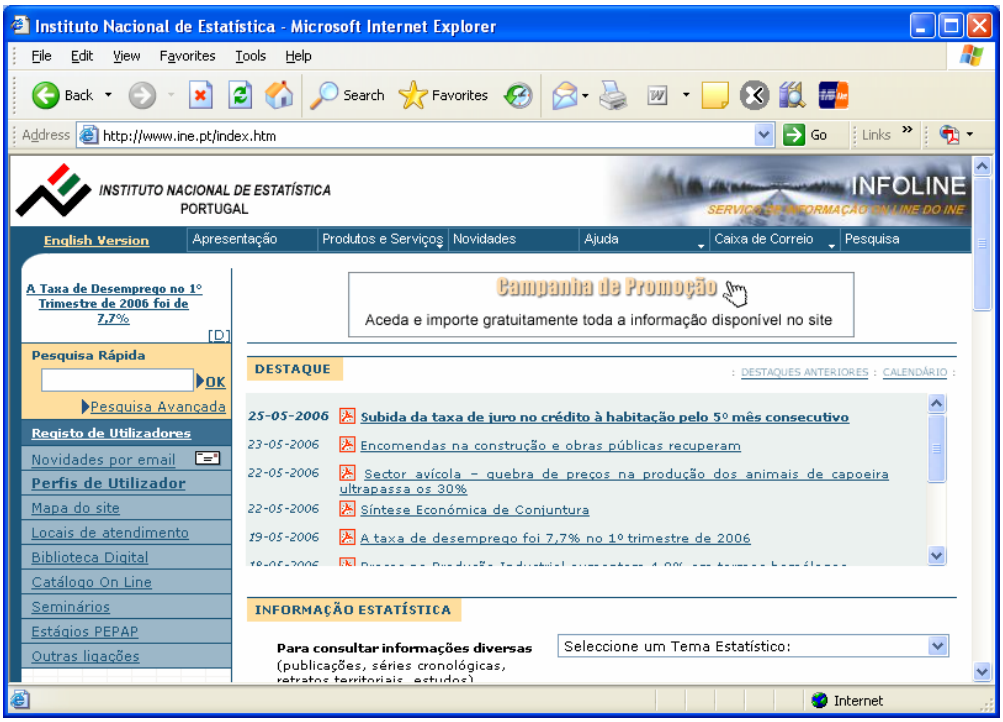

Figura 2: Serviço de informação *online* sobre estatística, disponibilizado pelo INE

2.2 Repositórios de informação

O termo repositório é utilizado de várias formas em vários contextos. Um repositório pode ser visto de várias perspectivas como por exemplo espaço de armazenamento, simples ficheiros com informação, BDs ou sistemas complexos de gestão de informação. Na literatura é possível encontrar várias referências a tecnologias que utilizam o conceito de repositório. De seguida são descritas sucintamente três referências.

2.2.1 Gestão de metadados - *repository technology*

Em qualquer processo produtivo as actividades envolvidas favorecem a produção de metadados relacionados com os objectos ou informação manipulados ou produzidos. Como exemplo, actividades como a criação de *software* OO (orientado por objectos) ou reengenharia de processos [Bernstein e Dayal 1994] poderão levar à criação de metadados que é necessário gerir de forma eficaz. Estes metadados podem ser utilizados por várias ferramentas que manipulam os objectos em causa pelo que a sua partilha e gestão é um aspecto de extrema importância.

A gestão dos metadados requer a criação de uma camada de serviços de controlo suportada num SGBD denominado *Repository Manager*. A integração deste com várias ferramentas constitui um *Repository System* [Bernstein e Dayal 1994]. Nesta perspectiva, um repositório é definido como sendo uma BD partilhada de informação sobre artefactos projectados[3], produzidos ou usados por uma instituição como sejam, *software*, documentos, sistemas (como circuitos electrónicos ou automóveis), etc. [Bernstein e Dayal 1994]

Associados a estes artefactos e ao longo do seu tempo de vida vão sendo definidos, criados, manipulados e geridos objectos com recurso a várias ferramentas que necessitam de partilhar informação. Os objectos em si podem não ser armazenados no repositório, mas de várias formas, como num sistema de ficheiros ou sistemas de BDs, sendo as suas descrições e eventualmente informações adicionais como o seu historial ou localização, armazenadas no repositório. Um repositório pretende desta forma fomentar e facilitar a partilha de informação entre ferramentas que estejam integradas com ele.

A Figura 3 mostra a arquitectura de um *Repository System* onde se evidencia o modelo de informação que especifica o modelo para a estrutura e semântica dos objectos que são armazenados no repositório. O *Repository Manager* fornece funcionalidades de acesso e gestão do repositório, bem como da informação que ele armazena. Na camada superior estão as ferramentas que funcionam de forma integrada, partilhando a informação do repositório.

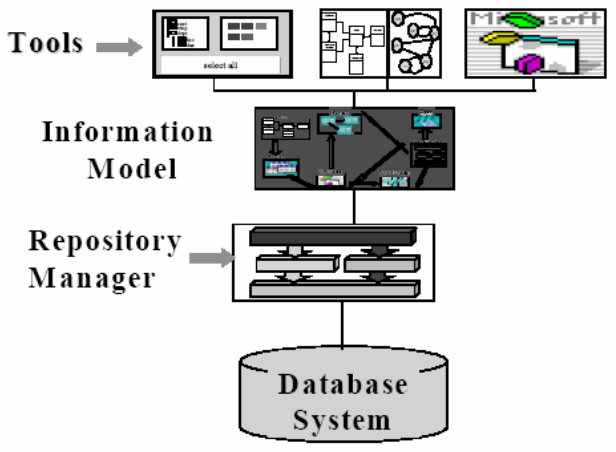

Figura 3: Estrutura do *Repository System* [Bernstein 1998]

[3] O autor utiliza o termos *engineered artifacts* numa perspectiva de artigos que passaram por um processo produtivo.

Sendo um repositório considerado também como uma base dados, o *Repository Manager* deve oferecer características usuais num SGBD como um modelo de dados, integridade, concorrência ou controlo de acessos, no entanto terá que disponibilizar outras [Bernstein e Dayal 1994]:

• ***Checkout/checkin***: esta característica, particularmente útil em actividades de grande duração, permite copiar um dado objecto para a área de trabalho do utilizador (*checkout*) permitindo que este faça as alterações necessárias, as quais serão comunicadas ao repositório após a sua conclusão (*checkin*).

• **Controlo de versões**: permite manter um histórico de versões do objecto para que seja possível consultar a sua evolução ao longo do tempo, mantendo várias versões consistentes do objecto obtidas durante o seu período de vida.

• **Controlo de configurações:** no repositório podem existir objectos compostos que consistem em colecções hierárquicas de outros objectos. Os objectos compostos, bem como os objectos pelos quais estes são constituídos podem ter várias versões. Nesta perspectiva, uma configuração pode ser vista como uma ligação entre uma dada versão do objecto composto e uma versão dos objectos que o constituem.

• **Gestão de contexto:** permite definição de um contexto que é uma visão particular dos objectos no repositório focando apenas os necessários para uma dada tarefa.

• **Notificação**: uma vez que muitos objectos no repositório estão interligados, alterações num podem implicar alterações nos restantes. Desta forma, esta característica permite o despoletar de operações nos restantes objectos ou o envio de eventos apropriados.

• **Controlo de *workflow*:** um *repository manager* deverá implementar um mecanismo de controlo do estado dos objectos relativo ao ciclo de vida pelo qual o artigo projectado progride, de forma a ser possível conhecer a que fase corresponde (analise de requisitos, desenho, etc).

Existem algumas soluções implementadas de repositórios como o Microsoft Repository [Bernstein et al. 1997]. Inicialmente era um componente do Microsoft Visual Basic 5.0 constituído por um conjunto de controlos ActiveX com os quais é possível definir modelos de informação e um motor de repositório que é o mecanismo de armazenamento e gestão associado e que se apoia num SGBD relacional. A tecnologia do Microsoft Repository evoluiu para *Meta Data Services*, sendo posteriormente incluída no Microsoft SQL Server 2000 [Microsoft 2000].

2.2.2 Armazenamento digital de documentos - Bibliotecas digitais

Uma área onde é utilizado o conceito de repositório é a do armazenamento digital de documentos, nomeadamente bibliotecas digitais. Esta área tem sido alvo de atenção nos últimos tempos, existindo actualmente algumas soluções implementadas.

O conceito básico de funcionamento destas soluções pode ser encontrado em [Kahn e Wilensky 1995]. Neste documento, os autores descrevem como é possível executar a gestão de documentos digitais num ambiente de repositórios distribuídos. Esta arquitectura baseia-se em 3 conceitos fundamentais: objectos digitais, identificadores únicos (*handlers*) e repositórios.

À informação digital disponibilizada por alguma entidade ou indivíduo, pode ser associado um identificador global único, denominado *handler* (obtido de uma entidade com competência para o criar), bem como metadados, constituindo-se assim um objecto digital. Estes objectos são depositados em repositórios de forma a ficarem disponíveis para consulta. Após esta operação, o identificador do objecto bem como uma referência ao(s) repositório(s) são armazenadas conjuntamente num sistema de servidores específicos para esse fim. Desta forma, executando uma pesquisa por um dado objecto (pelo seu identificador) nesse sistema é possível obter o repositório onde este se encontra armazenado. Isto permite uma grande flexibilidade na gestão dos objectos pois possibilita o acesso aos objectos sem se conhecer à partida a sua localização.

Neste contexto, um repositório é definido como sendo um sistema de armazenamento acessível em rede, no qual é possível armazenar os objectos digitais e que permitem também o seu acesso e recuperação. A interacção com o repositório é feita utilizando um protocolo de acesso (RAP[4]). Desta forma, qualquer repositório deverá suportar um protocolo que implemente pelo menos as operações mais básicas como o depósito de objectos digitais e acesso aos objectos digitais por identificador. Em [Arms et al. 1997] é proposto um sistema para uma biblioteca digital baseado nestes conceitos, constituído por 4 componentes principais (Figura 4).

[4] *Repository Access Protocol.*

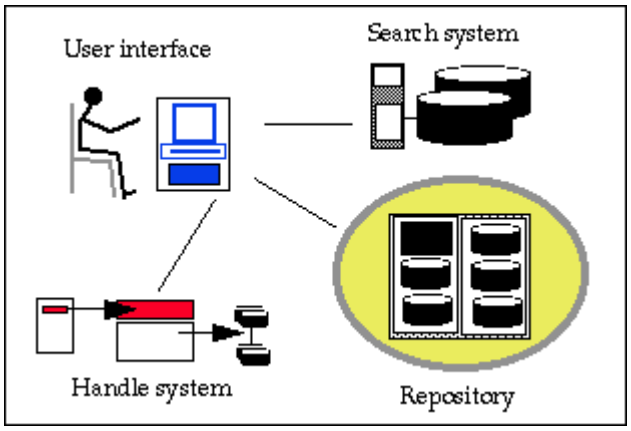

Figura 4: Componentes do sistema

A interface com o utilizador (com duas versões, uma para utilizadores da biblioteca e outra para administradores do sistema) é baseada na utilização de um *browser* (para a interacção com o utilizador) que permite aceder a serviços que interagem com as restantes partes do sistema.

Um outro aspecto deste sistema é o repositório. Uma biblioteca digital de grandes dimensões pode utilizar vários repositórios e de vários tipos. Todos os repositórios devem são acessíveis utilizando um protocolo específico (RAP) pelo que os pedidos originados pelos utilizadores terão de ser traduzidos de forma a serem processados pelo protocolo de acesso. O sistema de identificadores únicos (*handle System*)[5] permite fazer a gestão dos identificadores a atribuir aos objectos digitais sendo possível fazer uma pesquisa por um dado identificador, em resposta à qual é devolvida a referência ao repositório onde o objecto está armazenado. A biblioteca digital implementada por este sistema considera que existem catálogos (administrados de forma independente) que é possível pesquisar de forma a obter a informação desejada, a qual pode ser posteriormente obtida do repositório. Isto constitui o sistema de pesquisa.

Actualmente, um sistema com bastante notoriedade nesta área é o DSpace [DSpace 2004]. Este sistema, que como é referido em [Tansley et al. 2003] tem alguma inspiração de vários trabalhos anteriores entre os quais [Kahn e Wilensky 1995] e [Arms et al. 1997], começa a ser utilizado por um cada vez maior número de instituições para a preservação digital de documentos entre as quais a Universidade do Minho com o seu "RepositóriUM"[6].

[5] No sistema proposto por [Arms et al. 1997] é utilizado o sistema de *handles* da CNRI - http://www.handle.net

[6] Repositório institucional da Universidade do Minho. https://repositorium.sdum.uminho.pt/

O DSpace é definido [Tansley et al. 2003] como sendo um sistema *open source* que tem como objectivo funcionar como repositório para material de investigação e educacional produzido por organizações ou instituições. É visto como um repositório institucional. Apesar do termo repositório estar aqui a ser usado numa perspectiva algo diferente da apresentada anteriormente para as bibliotecas digitais (repositório como sendo um sistema completo *vs* repositório como parte integrante de um sistema que interage com outras partes como uma interface de utilizador), o sistema apresenta uma arquitectura comparável à apresentada anteriormente, onde são utilizados conceitos como objectos digitais e *handles*.

O Dspace é constituído por 3 camadas [Tansley et al. 2003] (Figura 5). A camada superior, de aplicação, faz a interacção do sistema com o mundo exterior, por exemplo com os utilizadores através uma interface *Web*. A camada intermédia implementa toda a lógica necessária na gestão do sistema, servido de ponte entre a camada superior e a camada inferior, de armazenamento. Nesta camada é feito o armazenamento físico dos objectos e respectivos metadados, quer em BD (relacional, neste caso) quer em sistema de ficheiros.

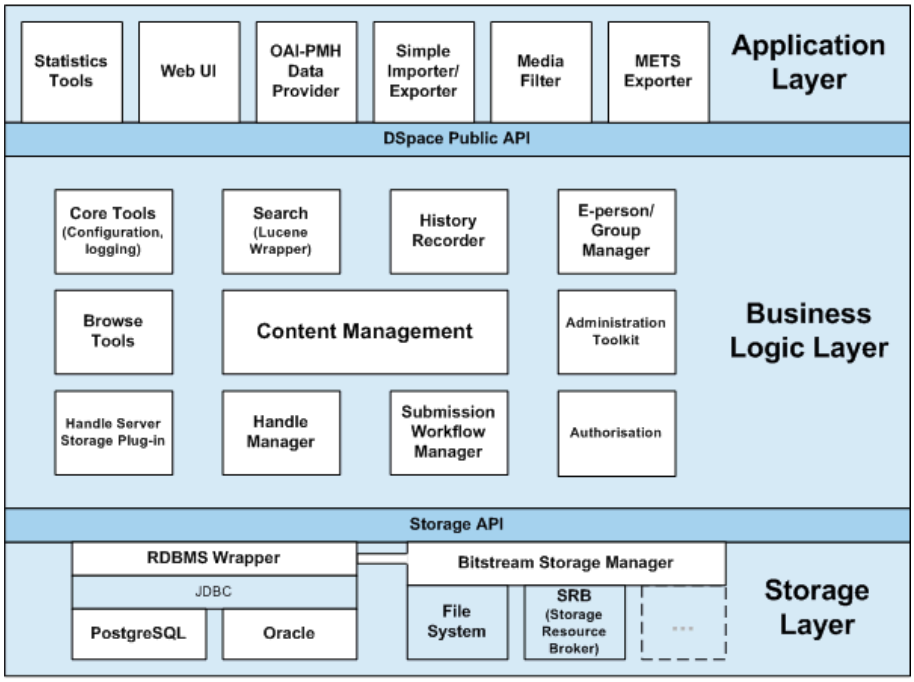

Figura 5: Arquitectura do DSpace

Um outro sistema com relevância actualmente é o *Flexible and Extensible Digital Object and Repository Architecture* (FEDORA) [The Fedora™ Project 2004]. Este sistema disponibiliza funcionalidades de administração e acesso aos objectos digitais

armazenados no repositório sob a forma de *web services*. Desta forma, pode ser usado para construir aplicações orientadas a qualquer realidade na área de armazenamento digital de documentos e não apenas aplicações com fins específicos como repositórios institucionais (como o caso do DSpace) vocacionados para atender às necessidades específicas de um certo tipo de instituições. Pode ser considerado de uma forma mais genérica como um repositório digital.

2.2.3 Comércio electrónico

O conceito de repositório está presente em outras áreas como o comércio electrónico. Um dos factores de sucesso nesta área é a eficácia da comunicação entre os vários parceiros de negócio. Nesta perspectiva têm sido criados padrões de comunicação baseados em XML. A principal vantagem do XML é o facto de tornar possível a troca de informação entre aplicações de uma forma eficaz. Um sistema pode produzir e disponibilizar informação no formato XML (por exemplo a informação com origem numa BD). Esta informação pode ser transmitida a qualquer outro sistema, cujo único requisito é conseguir perceber e tratar convenientemente a informação. Esta característica do XML é importante no B2B uma vez que facilita a comunicação entre organizações.

Um padrão bastante divulgado nesta área é o ebXML, sendo um dos aspectos que o caracterizam o favorecimento da reutilização de componentes na sua implementação por parte das organizações. Com o objectivo também de facilitar a comunicação entre parceiros, o ebXML define o conceito de *registry*/repositório. Como é referido na especificação [ebXML 2001], o *registry* fornece um conjunto de serviços que permitem a partilha de informação entre parceiros de negócio, nomeadamente especificações ebXML. Isto tem como finalidade permitir a integração de processos, servindo como BD para a partilha de informação sobre as organizações necessária às transacções ebXML tais como capacidades das organizações, processos de negócios, facturação, etc.[Chiu 2002]. Desta forma, o *registry* pode ser visto como fornecendo a interface entre utilizadores (humanos ou aplicações informáticas) e o repositório que armazena a informação partilhada.

Os repositórios armazenam informação como perfis de organizações (que são consultados pelas outras organizações com vista a encontrar um possível parceiro de

negócio), definições de processos, componentes básicos, etc. Esta informação, que não tem de estar obrigatoriamente representada em XML, é necessária às fases de implementação do ebXML pelas quais as organizações devem passar antes que possam iniciar a fase de troca de mensagens[7] com os seus parceiros de negócio. O conteúdo do repositório é gerido (os itens são criados, actualizados ou removidos) através de pedidos feitos ao *registry*, pelo que pode ser considerado como uma API para acesso aos itens armazenados [Chiu 2002].

2.3 Conclusão

O termo repositório pode ter várias utilizações em vários contextos. De uma forma geral, um repositório pode ser visto como um depósito de dados constituindo a camada inferior onde assentam todas as outras que podem constituir as aplicações. Isto poderá ser considerado como a camada de armazenamento.

Outra característica importante é o facto de não ser acedido directamente por utilizadores (humanos ou aplicações informáticas). O seu acesso e gestão são feitos exclusivamente através de uma camada superior, a qual abarca toda a lógica da aplicação, atribuindo significado à informação armazenada. A Figura 6 apresenta este conceito.

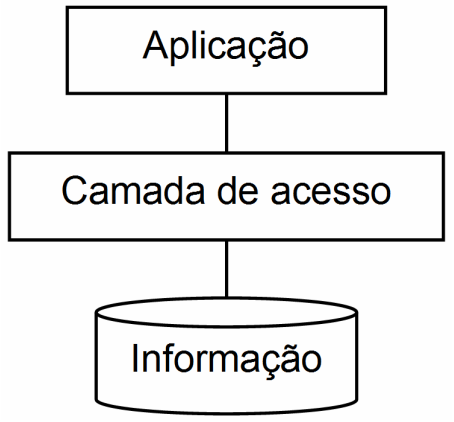

Figura 6: Acesso à informação

A abrangência do termo pode variar, podendo ser considerado apenas como um depósito de dados ou conjuntamente com quaisquer outras camadas. No primeiro caso

[7] Mensagens com o formato devidamente estabelecido nas fases de implementação.

poderá ser encarado como uma BD, mas não só. Poderá ser considerado como uma solução híbrida, utilizando BD e armazenamento em sistema de ficheiros, por exemplo. No segundo caso poderá incluir camadas de interface com o utilizador final, podendo ser considerado por si só como um aplicação completa. Desta forma a abrangência depende do contexto em que o autor define o repositório.

Considerando um repositório por si só como uma aplicação completa, a distinção entre serviço de informação e repositório não é clara, pelo que neste trabalho torna-se necessário fazer uma distinção entre ambos. Desta forma, podendo um serviço de informação, tal como referido anteriormente, ser considerado como uma aplicação *web*, o repositório pode ser visto como uma parte integrante da aplicação, responsável pelo fornecimento de informação necessária ao funcionamento do serviço de informação.

Para a execução desta tarefa, a informação terá de estar armazenada de forma permanente utilizando um qualquer modelo. Desta forma, existe todo o interesse em estudar o modelo mais eficaz para a manipulação da informação que irá alimentar o serviço de informação, sendo este o objectivo do presente trabalho.

Capítulo 3

3 Armazenamento da informação

É possível armazenar informação recorrendo a várias soluções, como sistemas de ficheiros, directórios ou BDs. Uma das soluções mais comuns para o armazenamento de informação nas aplicações actuais é o uso de BDs. Existem vários modelos, no entanto o relacional, que conta já com décadas de utilização, tem uma grande aceitação.

Recentemente o XML tem assumido uma grande relevância, não se tratando de uma forma de armazenamento por si só, mas uma forma de representar informação e que influencia a forma como esta é armazenada. A utilização do XML como forma de representação de informação tem implicações a nível das BDs. Por um lado, as BDs actuais para poderem armazenar e manipular eficientemente a informação representada neste formato necessitam de adaptações. Por outro lado, abriu-se espaço para o aparecimento de novas BDs concebidas de raiz especificamente para o armazenamento e manipulação eficientes de XML.

Além da descrição dos modelos mais comuns utilizados pelas BDs, este capítulo aborda também o XML no que diz respeito à representação de informação e as implicações relativas ao seu armazenamento em BDs.

3.1 Bases de dados

Uma BD é uma colecção de informação relacionada, isto é, factos que têm um valor implícito e que podem ser armazenados [Elmasri e Navathe 2003], de forma persistente e a serem manipulados [Graves 2002], podendo esta ser utilizada para modelar uma organização ou processo organizacional [Hernandez 2003].

Associado à BD existe o sistema de gestão de base de dados (SGBD) que é o conjunto do *software* que permite aos utilizadores criar e manter a BD, podendo a sua informação estar representada segundo uma grande variedade de modelos. De seguida são classificados e descritos os modelos mais comuns.

3.1.1 Modelos de dados

Um modelo de dados pode ser visto como um conjunto de conceitos que podem ser usados para descrever a estrutura e operações suportadas por uma BD. Por estrutura entende-se os tipos de dados, relações e restrições que definem o modelo da BD que irá receber a informação [Navathe 1992; Elmasri e Navathe 2003]. Esses modelos podem ser classificados de acordo com a forma como descrevem a estrutura da BD [Elmasri e Navathe 2003]. Podem ser considerados modelos de informação de alto nível (ou conceptuais) que fornecem uma visão da informação mais aproximada da visão dos utilizadores em oposição a modelos de baixo nível (ou físicos), que descrevem o armazenamento da informação nos sistemas informáticos ou modelos intermédios entre estes, representacionais (de implementação), que apesar de mais facilmente perceptíveis pelo utilizador são de certa forma aproximados a modelos físicos [Elmasri e Navathe 2003].

Em [Navathe 1992] os modelos são também classificados segundo outra perspectiva. Podem ser classificados em baseados em registos[8], semânticos ou baseados em objectos em termos de flexibilidade e expressividade. A primeira característica refere-se à capacidade dos modelos serem utilizados por aplicações complexas ao posso que a segunda refere-se à capacidade dos modelos exibirem diferentes abstracções e relações numa dada aplicação.

[8] *Record-based*

Os modelos baseados em registos como o de rede e hierárquico são pouco flexíveis e expressivos sendo largamente utilizados em SGBD nas décadas de 60 e 70 do século XX. A chegada do modelo relacional veio oferecer outro tipo de possibilidades que não eram possíveis com esses modelos como uma maior independência de dados relativamente ao aspecto físico. Os modelos semânticos dos quais o entidade-relacionamento[9] [Chen 1976], baseado nos conceitos de entidades, atributos e relacionamentos é um exemplo, são mais apropriados para uma definição conceptual da BD. Os modelos baseados em objectos, que podem ser aplicados a nível de implementação e conceptual, são mais apropriados a aplicações emergentes de maior complexidade.

Os modelos de implementação podem ser implementados por SGBD [Navathe 1992; Elmasri e Navathe 2003], como sejam os modelos hierárquico, de rede ou relacional. Os modelos por objectos podem ser vistos como de implementação, mas algo mais aproximados a modelos conceptuais [Elmasri e Navathe 2003].

Modelos hierárquico e rede

O modelo hierárquico (cujo primeiro SGBD foi o IMS da IBM [McGee 1977]), anterior ao modelo relacional, organiza a informação numa estrutura hierárquica em árvore de múltiplos níveis, onde cada nó agrupa os registos de um dado tipo. Cada um desses registos possui um dado número de campos, cada um com o seu tipo (*strings*, inteiros, etc). Cada tipo de registos está associado a outros por relações de 1 para N, em que do lado N das relações estão os filhos, as quais podem ser vistas como relações "pai-filhos". Neste modelo não existem ligações entre elementos da árvore ao mesmo nível ou níveis diferentes nem com elementos em diferentes ramos. Apenas existem ligações entre cada elemento e o seu superior (ou pai). Neste modelo, se um dado registo tiver de pertencer a mais que um ramo da árvore, terá que ser duplicado, podendo isto causar inconsistência na informação e uma redundância de dados.

Apesar de ter caído em desuso, este modelo pode ainda hoje ser encontrado em sistemas mais antigos utilizados por instituições como bancos ou seguradoras [Elmasri e Navathe 2003]. A Figura 7 ilustra a este modelo.

[9] Vulgarmente referido como E-R.

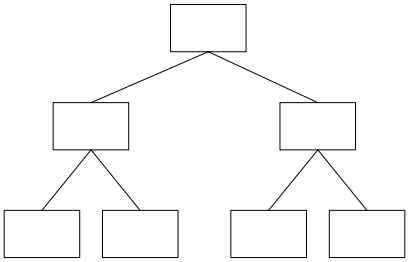

Figura 7: Modelo hierárquico

O modelo de rede foi desenvolvido aproximadamente na mesma altura do modelo relacional tendo sido utilizado em produtos comerciais antes deste último [Jackson 1999]. Neste modelo, a informação é armazenada de forma semelhante ao modelo hierárquico, no entanto, ao contrário deste, cada elemento que constitui a estrutura pode ter ligações com vários elementos ao mesmo nível ou em níveis diferentes, existindo relações um para muitos, muitos para um ou muitos para muitos. Este modelo permite que a navegação até chegar a um determinado elemento não necessite de passar por todos os níveis, podendo tomar atalhos. Desta forma, a informação está organizada num grafo. Tal como o modelo hierárquico, este modelo caiu agora em desuso, existindo no entanto ainda em aplicações mais antigas. A Figura 8 representa o modelo de rede.

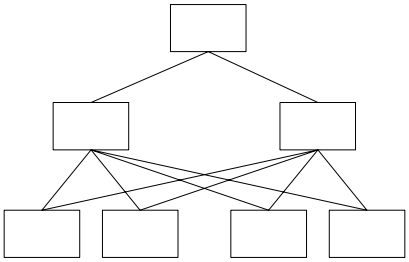

Figura 8: Modelo de rede

Modelo relacional

O modelo Relacional, introduzido por Codd em 1970, e que apresenta uma base teórica sólida de conceitos matemáticos[10], é o modelo mais divulgado actualmente nos SGBD. O seu aparecimento foi motivado por uma série de factores, entre os quais se destacam a necessidade de aumentar a independência dos dados nos SGBD, isto é, esconder certos pormenores respeitantes ao armazenamento da informação, pela necessidade de definir uma abordagem matemática à problemática do armazenamento e recuperação de

[10] Teoria de conjuntos e lógica de predicados de primeira ordem.

informação e também a necessidade de suporte para *queries ad-hoc* e não apenas aqueles predefinidos na implementação da BD como acontecia em modelos anteriores [Jackson 1999]. Este modelo apresenta várias vantagens relativamente aos modelos anteriores, como seja uma representação da informação mais simples e a sua facilidade em suportar *queries* ainda mais complexos [Ramakrishnan e Gehrke 2001].

No modelo relacional toda a informação está representada como relações matemáticas, as quais pode ser vistas como tabelas. Uma BD relacional pode ser vista como um conjunto de relações (tabelas) cada uma com um nome distinto. Uma tabela é constituída por um ou mais atributos (campos), que traduzem o tipo de dados a armazenar. O nome da tabela e das colunas são utilizados para facilitar a interpretação dos valores armazenados em cada linha da tabela (registos) os quais são a instanciação da relação em causa. Cada um dos registos das tabelas é identificado de forma única, pelo que é necessária a existência de um atributo que é a chave de acesso aos registos da tabela. Uma tabela também pode ter atributos cujo conteúdo é chave noutra tabela, denominados chaves estrangeiras permitindo assim a ligação lógica entre tabelas. Isto são restrições que permitem preservação da consistência da informação da BD.

Numa BD relacional um aspecto importante é o desenho da estrutura das tabelas. O desenho ineficiente pode levar à redundância de dados. De forma a ser construído um modelo consistente da informação que será armazenada é necessário proceder a um processo de normalização, que utiliza 6 formas normais, o qual reduz a redundância de informação e a possibilidade desta ficar inconsistente.

A Figura 9 ilustra o modelo relacional.

Tabela Autores

ID_autor (chave)	**Nome**
1	Carl Sagan
2	Stephen Hawking
3	Richard P. Feynman

Tabela Livros

ID_livro (chave)	**ID_autor**	**Nome**
1	2	O fim da física
2	1	Os dragões do Éden
3	3	Uma tarde com o Sr. Feynman

Figura 9: Modelo relacional

Modelos orientados por objectos

Os modelos OO surgiram como resposta às necessidades de aplicações mais complexas às quais modelos anteriores, como o relacional, não davam resposta convenientemente, como sejam aplicações de CAD/CAM, engenharia, multimédia, etc [Jackson 1999; Elmasri e Navathe 2003].

Estes modelos permitem a definição não só da estrutura de objectos complexos, mas também das operações que podem ser aplicadas aos objectos, bem como as relações entre eles. Estes objectos podem apresentar uma estrutura de complexidade arbitrária de forma a ser possível conter toda a informação que descreve o objecto, o que em modelos tradicionais como o relacional poderia implicar uma dispersão da informação por vários registos e relações, deixando de existir uma correspondência directa entre objectos do mundo real e os representados na BD [Elmasri e Navathe 2003]. Esta forma mais realista de representação da realidade é uma das vantagens deste modelo face a outros, nomeadamente o relacional [Zand et al. 1995].

Os conceitos OO nos quais estes modelos se baseiam, como a noção de objectos e classes, encapsulamento, herança, etc, são importados das linguagens de programação OO. Exemplos deste tipo de linguagens são o Smalltalk, C++, Java. Em [Atkinson et al. 1989] são apresentadas as características que um sistema de BD OO deve possuir dividindo-as em obrigatórias, como suporte para objectos complexos, identidade de objectos, classes e outros, e opcionais como herança múltipla ou suporte de versões e abertas que podem ser definidas livremente.

O que é armazenado na BD são objectos que têm associado a si informação, os quais dispõem de métodos para que possam ser efectuadas operações sobre eles. Desta forma é possível que objectos com uma existência transiente utilizados nas linguagens de programação OO possam ter uma existência física, sendo transferidos para uma armazenagem permanente [Elmasri e Navathe 2003]. O facto de os conceitos OO serem cada vez mais usados no âmbito da programação é um incentivo à utilização destes modelos nas BDs com vista a uma integração mais harmoniosa entre ambos [Elmasri e Navathe 2003].

Outros modelos

Existem actualmente outros modelos de dados, tais como o modelo objecto-relacional[11]. Este modelo não consiste numa tecnologia inteiramente nova, consistindo antes no aproveitamento das melhores características dos modelos por objectos e relacional. Desta forma é possível utilizar conceitos da tecnologia OO nos seus sistemas actuais baseados no modelo relacional sem que sejam necessárias alterações radicais. Existem vários outros modelos como o multimédia, que foram desenvolvidos mas que no entanto nunca tiveram grande impacto a nível comercial.

3.2 XML

Com a massificação da utilização da Internet, o XML tem-se assumido como um padrão com um papel preponderante na troca de informação. O XML é vocacionado para a troca de informação estruturada. Um exemplo da sua crescente importância é a sua adopção como base na criação de padrões utilizados para o comércio electrónico. Estes padrões são aceites por cada vez mais organizações sendo reconhecidos como um meio de relacionamento eficaz com base na Internet. O XML facilita também a interoperabilidade entre aplicações.

Desta forma, a tendência de futuro será uma maior utilização do XML para representar informação que circula na Internet. Nesta perspectiva, é obrigatório que seja dada atenção ao XML reconhecendo a sua importância como forma de representação de informação. Esta tendência actual leva a que actualmente muitos fabricantes de *software* estejam a incorporar nos seus produtos suporte para XML, como é o caso das BDs.

Esta nova forma de representação de informação estruturada apresenta novos desafios para o seu armazenamento exigindo conversões para a utilização com os modelos convencionais.

3.2.1 Introdução ao XML

O XML é uma linguagem de marcação, criada com o objectivo de descrever dados com

[11] *Object-relational*

forte estruturação. Isto é, fornece um conjunto de regras para descrever o conteúdo dos documentos, assim como a sua estrutura lógica, de modo a que estes possam ser interpretados e/ou manipulados. É importante salientar que um documento XML não contém apenas os dados, mas também informações sobre esses mesmos dados (nome, atributos, etc.) – os metadados. O seguinte exemplo (Exemplo 1) apresenta um documento XML bem formado. Para um documento ser considerado bem formado, deve respeitar as normas de escrita de documentos XML, nomeadamente a existência de apenas um elemento raiz, todas as *tags* devem estar devidamente fechadas (para cada `<elemento>` deve existir o respectivo `</elemento>`) e correctamente aninhadas, etc.

```
<?xml version="1.0"?>
<livro>
      <titulo>Cosmos</titulo>
      <autor>Carl Sagan</autor>
</livro>
```

Exemplo 1: Documento XML bem formado

Características do XML

As características do XML acabam por se tornar também nas suas grandes vantagens como tecnologia. Entre as principais constam [Ozu et al. 2001]:

- **Extensibilidade:** permite criar as etiquetas que se achar necessárias e com isso consegue-se adaptar continuamente a estrutura do documento. Enquanto o HTML usa um conjunto fixo de marcadores, o XML possibilita a utilização de marcadores de uma forma livre e adaptável;
- **Descrição de dados:** através de marcadores semânticos os documentos XML permitem que a informação que contém seja compreensível tanto para humanos como para os computadores;
- **Fácil processamento:** os documentos XML podem ser processados automaticamente, recorrendo a *software* preparado para tal;
- **Flexibilidade:** é possível representar dados estruturados ou semi-estruturados, bem como o seu transporte, mantendo a sua integridade;

- **Padrão aberto:** esta é certamente a característica do XML que se torna indiscutivelmente numa das suas grandes vantagens. Foi em grande parte devido a esta particularidade que o XML teve a aceitação e o crescimento que teve. Padrão aberto, significa que os documentos XML são totalmente independentes de aplicações, sistemas operativos, etc., isto é, independentes das plataformas de *software* e *hardware* que os utilizam;
- **Validade:** os documentos XML podem ser validados recorrendo a um *Schema* (documento XML que descreve a estrutura de outro documento XML, mais ou menos como a documentação da estrutura de uma BD) ou a um DTD – *Document Type Definition*;
- **Apresentação isolada:** uma vez que um documento XML não tem possibilidades de formatação – é necessário recorrer a uma folha de estilo (XSL). O mesmo documento XML pode ser apresentado de formas distintas.

Utilização do XML

Pode considerar-se que existem duas grandes áreas de utilização do XML. A primeira é a sua utilização como uma linguagem universal para definição de documentos, com sintaxe específicas para os mais variados domínios do conhecimento (matemática, química, comércio, etc.), isto é, permite criar linguagens específicas para determinadas áreas, como é o caso do *MathML – Mathematical Markup Language,* do CML – *Chemical Markup Language*, SVG – Gráficos Vectoriais, etc. A segunda é a possibilidade de se tornar numa forma de intercâmbio por excelência de documentos entre ambientes com plataformas completamente distintas, actuando como elemento integrador entre aplicações electrónicas como A2A (*aplication–to–aplication*), B2B (*business-to-business*). Em qualquer dos casos, o propósito do XML é representar e descrever os dados de um documento, de uma forma compacta e flexível, com o objectivo de trocar e manipular esses mesmos dados.

Tecnologias associadas ao XML

Existe um grande número de tecnologias associadas ao XML com vários fins como a validação ou navegação e que poderão ser úteis no âmbito da implementação de repositórios de dados. De seguida são descritas algumas dessas tecnologias.

Validação de documentos

Um dos aspectos fundamentais no XML é a validação de documentos. Um documento é válido se é bem formado e respeita restrições de validade impostas por um DTD[12] ou XML *Schema* [Ozu et al. 2001]. Isto permite que um documento XML possa servir de base à troca de informação entre várias entidades uma vez que desta forma será interpretado sempre da mesma forma por todos os intervenientes.

Cada ficheiro XML poderá estar associado a uma estrutura específica que o define. Esta definição é feita através de um DTD[13]. Este define as regras para especificação de uma classe de documentos, isto é, que tipo de elementos podem existir num documento, que atributos esse elementos podem ter e como as instâncias desses elementos estão hierarquicamente relacionados. Pode-se dizer então que o ficheiro DTD define o formato e a sintaxe de cada *tag* que foi criado no ficheiro XML. O DTD pode ser utilizado por programas que precisem efectuar validações na estrutura do documento, geralmente programas que envolvem a inserção ou actualização de informação numa BD, ou ainda ser utilizado como documento de referência a outros utilizadores que queiram utilizar a mesma estrutura para partilha de informação.

Um XML *Schema* [W3C 2004b] é um documento XML que define a estrutura permitida para um outro documento XML. Isto é, estipula para esse documento:

1. Os elementos reconhecidos;
2. Os atributos autorizados;
3. Os elementos que por sua vez são *child elements*;
4. O número de *child elements*;
5. A ordem dos *child elements*;
6. Os tipos de dados para os elementos e atributos;
7. Se um elemento é vazio ou não;
8. Valores por defeito ou constantes para os elementos ou atributos.

Existem 2 tipos de *schemas* – internos ou externos. Isto é, se o *schema* é interno vem incorporado num documento XML (ficheiro *.xml*), se é externo vem referenciado no

[12] Não confundir com declaração de tipo de documento, isto é, declaração `DOCTYPE`.

[13] Os DTDs são parte integrante da especificação do XML

documento XML e ficheiro de *schema* correspondente tem o sufixo *.xsd*. Neste último caso, o mesmo documento XML pode referenciar vários *schemas* e o mesmo *Schema* pode ser referenciado por vários documentos.

O XML *Schema* apresenta uma versatilidade que os DTD's não conseguem combater e são amplamente mais utilizados na manipulação de dados estruturados. Eis algumas razões que justificam este facto:

- *XML Schemas suportam tipo de dados* – desta forma, é mais fácil descrever o conteúdo, validar se a informação está correcta, utilizar restrições e padrões e converter dados de diferentes tipos;
- *XML Schemas utilizam sintaxe XML* – o que se torna mais simples de utilizar, uma vez que não é necessário aprender uma nova linguagem, e ainda possibilita que os ficheiros *schemas* sejam interpretados pelos *parsers* de XML;
- *XML Schemas são extensíveis* – o que permite usar um *schema* dentro de outros *schemas*, criar novos tipos de dados a partir dos já existentes e fazer referências a vários *schemas* no mesmo ficheiro XML.

DOM e SAX

O DOM [W3C 2001a] e o SAX [SAX 2004] são APIs (*Application Programming Interface*) que permitem a manipulação de documentos XML através de programação. O DOM – *Document Object Model* – é um padrão do W3C para armazenar e manipular documentos hierárquicos na memória. Um *parser* DOM faz o *parsing* do documento e constrói uma árvore em memória que o representa e que na qual se pode navegar para aceder aos vários nós. Uma vez que a árvore tem de ser completamente construída antes de ser possível navegar na sua estrutura, a manipulação de documentos XML de grandes dimensões pode ser problemática, uma vez que pode causar um grande consumo de recursos.

O SAX – *Simple API for XML* – ao contrário do DOM, utiliza um modelo orientado por eventos. Cada vez que ocorre uma *tag* no processamento do documento a aplicação é notificada, pelo que o não é necessário que seja feito o *parsing* de todo o documento antes que este possa ser manipulado. O SAX tornou-se no padrão mais utilizado uma vez que é mais apropriado para a manipulação de documentos de grandes dimensões.

XPath

O XPath [W3C 1999] – *XML Path Language* – é uma tecnologia que tem como objectivo permitir o acesso a partes de um documento XML, fornecendo uma sintaxe e semântica para esse efeito. A informação é organizada em árvore e pode ser acedida de uma forma hierárquica ou de uma forma arbitrária com base em identificadores únicos que os elementos deverão possuir.

XQuery

O XQuery [W3C 2005] – *XML Query Language* – é uma linguagem de *query* que é utilizada na manipulação de informação em formato XML. Esta informação pode estar fisicamente armazenada em ficheiros XML ou ser originada por qualquer fonte, como BDs relacionais.

O Xquery, que muitas vezes é visto para o XML como o SQL é para a informação relacional, apresenta características que o tornam extremamente poderoso no processamento e criação de XML. Destas suas características destacam-se a localização de nós em estruturas XML com expressões de *path* (p*ath expressions*), a construção de documentos ou fragmentos XML com construtores e a utilização de expressões FLWOR (FOR – LET – WHERE – ORDER BY – RETURN) que possibilitam que a informação seja manipulada de forma a serem criadas novas estruturas XML de uma forma muito semelhante à construção SELECT – FROM – WHERE que é possível encontrar em SQL.

3.2.2 Armazenamento de XML

A forma mais natural de integrar o XML em BDs é a utilização dos modelos convencionais mais divulgados actualmente, como o modelo relacional, permitindo assim a reutilização das BDs actualmente em funcionamento.

A maneira mais simples de armazenar documentos XML numa BD é a utilização de CLOBs (*character large objects*) [Fiebig et al. 2002; Bourret 2005b]. Todo o documento é tratado como texto tirando partido das funcionalidades de manipulação de texto disponíveis na BD. Uma vez que o documento é tratado como um todo, estamos perante uma visão da informação centrada no documento. Se for necessária a manipulação de fragmentos do documento, este terá de ser lido da BD na totalidade e

posteriormente tratado.

Uma outra abordagem é o mapeamento da estrutura de um documento XML para modelos utilizados em BDs actuais, como o relacional. Estas duas formas de representar informação apresentam diferenças substanciais. Como foi referido anteriormente, o modelo relacional é constituído por três partes: a BD é um conjunto de tabelas, que são conjuntos de registos, os quais são constituídos por campos que deverão ser atómicos (não podem ser eles próprios colecções de outros sub-elementos). Os documentos XML apresentam um conceito diferente, no qual a informação pode encarada como árvore de nós[14] [Schmidt et al. 2000]. O exemplo e imagem seguintes (Exemplo 2 e Figura 10) representam este conceito:

```
<?xml version="1.0" encoding="UTF-8"?>
<biblioteca>
      <seccao nome="Ciências" codigo="1A">
            <livro>
                  <nome>Um mundo infestado de demónios</nome>
                  <autor>Carl Sagan</autor>
                  <editora>Gradiva</editora>
                  <isbn>9726625629</isbn>
            </livro>
            <livro>
                  <nome>O último Teorema de Fermat</nome>
                  <autor>Amir D. Aczel</autor>
                  <editora>Gradiva</editora>
                  <isbn>9726625688</isbn>
            </livro>
      </seccao>
      <seccao nome="Arquitectura" codigo="2A">
            <livro>
                  <nome>Arquitectura no século XX</nome>
                  <autor>Peter Gössel</autor>
                  <autor>Gabrielle Leuthäuser</autor>
                  <editora>Taschen</editora>
                  <isbn>3822811653</isbn>
            </livro>
```

[14] Deste que o documento seja bem formado, isto é, que respeite as normas do XML

```
      </seccao>
</biblioteca>
```

Exemplo 2: Documento XML a ser representado como árvore

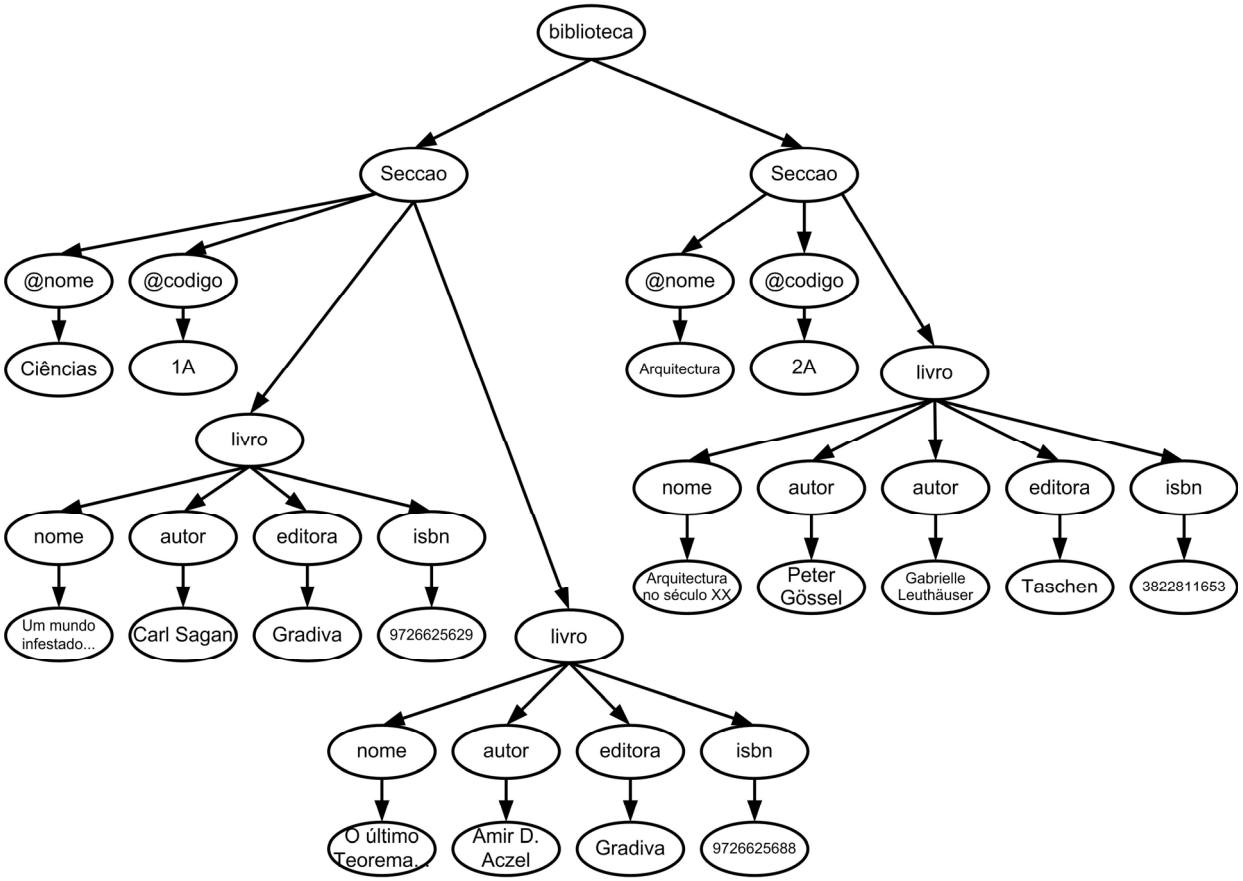

Figura 10: Vista de um documento XML como árvore

Para armazenar um documento XML numa BD torna-se necessário fazer o mapeamento da sua estrutura para o modelo utilizado por esta. No caso do modelo relacional, cujas diferenças para o XML são apresentadas na Tabela 1, isto envolve a criação de várias tabelas e a consequente dispersão da informação do documento original[15]. Esta operação pode trazer um problema associado que é a não preservação do documento na sua forma original quando é feita a sua extracção da BD e posterior reconstrução. Isto poderá ser tolerável para determinados tipos de aplicações, mas não para outros. Um outro problema que pode ocorrer é ser necessária a criação de um grande número de tabelas na BD, tornando-a ineficiente ou a criação de vários campos com valores nulos, o que é desperdício de espaço. A recuperação dos documentos envolve a consulta de várias tabelas envolvendo JOINs pelo que se trata de uma operação intensiva para o servidor.

[15] Existem várias abordagens possíveis para esta tarefa [Bourret 2005b].

Relacional	XML
Representação tabular da informação.	Representação hierárquica da informação.
Cada célula tem um único valor.	Nós tem elementos e/ou valores de atributos
As células contêm valores atómicos.	Os elementos podem conter outros sub-elementos.
Fortemente estruturado com definições estáticas de *schema*. O mesmo *schema* aplica-se a todas as linhas de uma dada tabela.	Semi-estruturado. *Schemas* flexíveis. O *schema* pode ou não existir para algum ou mesmo para todos os documentos e são sendo facilmente estendidos.
Todas as relações são definidas por uma chave primária e chaves estrangeiras.	Um documento contém a informação e a informação sobre a relação que a descreve.
Não existe noção de ordem.	A ordem é relevante. A informação está ordenada em sequências que são ordenadas por definição.
Fortemente tipado. Cada coluna de uma tabela tem o seu tipo.	A utilização de tipos não é obrigatória. Os tipos podem ou não estar definidos para alguns ou mesmo para todos os elementos definidos pelo *schema*.
São necessários *Joins* para recuperar documentos simples.	A recuperação de elementos simples é directa.

Tabela 1: Diferenças entre o modelo relacional e XML (baseado em [Champion 2001])

Apesar de existirem vários estudos sobre a utilização do modelo relacional [Florescu e Kossmann 1999; Shanmugasundaram et al. 1999] há autores que defendem que o modelo orientado aos objectos é mais apropriado à tarefa de manipulação de XML, pois a estrutura em árvore dos documentos XML é mais facilmente adaptável a este modelo [Straube e Özsu 1990; Fegaras e Elmasri 2001]. No entanto, o modelo orientado aos objectos parece ainda não estar suficientemente desenvolvido para suportar *queries* complexos em grandes BDs [Florescu e Kossmann 1999], além do facto de ser pouco

utilizado comparativamente ao modelo relacional.

Actualmente existem várias alternativas na forma e como o XML é armazenado. As soluções mais comuns variam entre sistemas de ficheiros, BDs e servidores de directório. Em [Vakali et al. 2005] é feito um resumo das abordagens mais comuns com as respectivas vantagens e desvantagens (Tabela 2).

Tipo	Formato do armazenamento	Principais vantagens	Principais desvantagens
Sistema de ficheiros	Ficheiros ASCII armazenados em sistema de ficheiros ou em BDs como BLOBs ou CLOBs	Fácil implementação. Adequado a conjuntos pequenos de XML.	Acesso e actualização são difíceis.
SGBD relacionais	Tabelas	Escalabilidade, fiabilidade, fácil implementação.	Requer muitos *joins* devido à factorização dos documentos XML
SGBD objecto-relacionais	Tabelas e objectos	Fácil implementação. Suporte para tipos abstractos.	Factorização de documentos XML.
XML Nativo	Modelos ad hoc ou modelos mais comuns das BDs	Flexibilidade. Bom desempenho a nível de acesso.	Menos maduro que os SGBD convencionais
Servidores de directório	Estrutura em árvore	Optimizado para queries.	Mau desempenho a nível de actualização

Tabela 2: abordagens para o armazenamento de XML (adaptado de [Vakali et al. 2005])

Bases de dados com suporte para XML e nativas XML

O XML pode ser utilizado em BDs de duas formas [Steegmans et al. 2004]. Numa primeira abordagem, o XML é usado para troca de informação entre a BD e as

aplicações ou outras BDs. Quando um documento é armazenado na BD é convertido para o modelo interno utilizado pela BD, sendo apenas a sua informação armazenada. Quando é recuperado é reconstruído com base na informação previamente guardada na BD. Nesta perspectiva o documento XML é completamente externo à BD, uma vez que nenhuma informação sobre o seu *schema* (como elementos e atributos) é armazenada, apenas a informação. Desta forma, a informação é armazenada internamente num *schema* compatível com o do ficheiro, pelo que a cada *schema* da BD corresponde um *schema* diferente de documento XML. Às BDs que utilizam o XML desta forma é dado o nome de BD com suporte para XML [Steegmans et al. 2004].

Na outra abordagem, todo o documento é armazenado na BD, não sendo este apenas utilizado como meio de fornecer ou extrair informação na BD. O documento pode igualmente ser traduzido para o modelo interno utilizado pela BD, mas de forma a que seja preservada toda a informação respeitante à sua estrutura, como elementos e atributos. Neste caso o *schema* interno da BD está apto a armazenar qualquer tipo de documento. Às BDs que utilizam o XML desta forma dá-se o nome de nativas XML. Pode-se dizer que estas BDs têm um novo modelo de dados – o modelo XML – que é uma árvore ordenada com nós que têm associados a si uma descrição e um tipo, e como folhas, nós sem descrição onde é armazenada a informação [Steegmans et al. 2004]. (como é visível na Figura 10).

As BDs com suporte para XML são BDs convencionais que incluem um mecanismo de tradução entre o XML e o seu modelo interno. Em muitos casos o mecanismo é interno, dispensando assim a utilização componentes de *software* adicionais. No entanto, esta tradução pode ser feita com recurso a *software* externo denominado *middleware*. O uso de *middleware* permite estabelecer a comunicação entre a aplicação que irá tratar e utilizar os documentos XML e respectiva BD, retirando da BD e da aplicação a tarefa da tradução do XML para o formato utilizado pela BD. Existem muitas opções nesta área com funcionalidades bastante diferentes, nomeadamente no que diz respeito à bidireccionalidade da comunicação, isto é, não só a capacidade de extrair informação da BD convertendo-a para XML, mas também a capacidade de inserir informação na BD com origem num documento XML.

No que diz respeito às BDs nativas XML, segundo [XML:DB 2004], uma BD deste género:

- Define um modelo (lógico) para um documento XML – em oposição à informação nesse documento – e armazena e recupera documentos de acordo com esse modelo. No mínimo, o modelo deve incluir elementos, atributos, PCDATA e ordem no documento. Exemplos de tais modelos são o modelo de dados Xpath, XML Infoset e os modelos implícitos pelo DOM e os eventos em SAX 1.0;
- Tem um documento XML como a sua unidade (lógica) de armazenamento fundamental, tal como uma BD relacional tem um registo de uma tabela como a sua unidade (lógica) fundamental de armazenamento;
- Não tem obrigatoriamente um modelo físico de armazenamento subjacente. Por exemplo, pode ser construída utilizando uma base da dados relacional, hierárquica, orientada por objectos ou utilizando um modelo proprietário.

Neste tipo de BDs, a entrada, saída e armazenamento de informação é sempre no formato XML, não existindo qualquer tipo de conversões intermédias como nas soluções descritas anteriormente uma vez que a unidade fundamental de processamento é o documento XML. Desta forma, uma das grandes vantagens é o facto de os documentos serem armazenados e recuperados intactos, o que pode ser importante para determinadas aplicações.

Este tipo de BD não inclui um modelo físico de armazenamento, podendo ser suportadas um qualquer modelo físico, uma vez que não têm como objectivo a substituição das BDs existentes, mas são apenas ferramentas que permitem o armazenamento e manipulação de documentos XML de uma forma mais robusta.

As BDs nativas XML têm uma camada de armazenamento onde a informação é armazenada fisicamente, a qual pode ser implementada de várias formas, como seja uma armazenamento de informação com base nos modelos comuns em BDs (como o relacional ou por objectos) ou então usando um modelo proprietário. Com a utilização de um modelo proprietário evita-se a conversão do XML para outros modelos e as dificuldades inerentes a esta operação[16], o que poderia levar a limitações em termos de funcionalidades e impacto no desempenho [Schöning 2003].

As BDs nativas XML aceitam documentos de duas formas distintas [Vakali et al. 2005]:

[16] Denominado em [Schöning 2003] como *impedance mismatch*.

em formato de texto ou num formato binário, numa representação em árvore baseado num modelo tal como o DOM. A Tabela 3 ilustra estas duas situações.

Tipo	Método	Armazenamento	Apropriado para
Texto	Baseado em texto	Documento completo (por exemplo em BLOB ou CLOB)	Recuperação de documentos completos ou partes especificas.
Binário, baseado num modelo	Baseado em árvore	Árvore baseada na hierarquia do XML	Aceder a partes específicas de um documento ou conjunto de documentos.

Tabela 3: Modelos de armazenamento de informação em BDs Nativas XML (adaptado de [Vakali et al. 2005])

Independentemente da sua camada de armazenamento, as BDs Nativas XML são concebidas para lidar com documentos XML de qualquer tipo, sendo consideradas por certos autores como a melhor solução para o armazenamento e manipulação de documentos XML [Schmidt et al. 2001].

Apesar de ambas manipularem XML, as BDs Nativas XML e com suporte para XML apresentam diferenças [Bourret 2005b]:

- As nativas preservam a estrutura física dos documentos o que nem sempre é feito nas com suporte para XML;
- As nativas guardam qualquer documento, mesmo sem conhecer o seu *schema*, o que pode não ser possível nas com suporte para XML;
- Nas nativas, a informação apenas é acessível em formato XML, ao passo nas com suporte para XML poderá existir outra forma de aceder à informação.

Nas com suporte para XML, a linguagem de *query* é traduzida para uma linguagem de *query* utilizada pelo motor da BD que no caso de ser relacional é o SQL.

3.2.3 Tipos de documentos e Dualidade dados *vs* documentos

Existem dois modelos de documentos XML [Obasanjo 2001; Bourret 2005b]: centrado nos dados[17] e centrado no documento[18]. Estes modelos influenciam a forma como o XML é armazenado de forma persistente em BD.

No modelo centrado nos dados inserem-se documentos que utilizam o XML como forma de transporte de informação [Bourret 2005b], sendo destinados predominantemente a consumo por parte de computadores e não a uso por parte de humanos. O uso de XML em documentos centrados nos dados é um facto de menor importância [Bourret 2005b], uma vez que o formato escolhido para representar a informação poderia ser qualquer outro eventualmente mais apropriado para a aplicação em causa. Neste modelo, os documentos são caracterizados por possuírem uma estrutura regular e normalmente a ordem pela qual os vários elementos com o mesmo nível na estrutura são representados é irrelevante. Outra característica deste tipo de documentos é também o facto de apresentarem uma pequena granulosidade na informação [Bourret 2005b], isto é, as unidades mais elementares de informação estão ao nível de elementos que suportam apenas dados[19] ou de atributos, não existindo elementos com conteúdo misto de informação e sub-elementos.

Documentos deste tipo são obtidos por exemplo quando se extrai informação de uma BD e cuja informação é convertida para XML. Exemplos deste tipo de documentos podem ser ordens de compra, ficha pessoal de um paciente num hospital, registos de informação científica de vários tipos, etc. O seguinte exemplo (Exemplo 3) representa um documento centrado nos dados:

```
<?xml version="1.0" encoding="utf-8"?>
<EN_RECURSO_HUMANO
      NRO_ID_CNPQ="0327839545905969"
      NME_RH_FILTRO="ANTONIO TESTE"
      COD_SIST_PROCED="999" SGL_PAIS_NASC="POR"
```

[17] *Data-centric* em língua inglesa

[18] *Document-centric* em língua inglesa

[19] Elementos denominados PCDATA

```
      CPF_RH="6667" NME_RH="António Teste"
      TPO_NACIONALIDADE="P"
      COD_SEXO="M" NME_CITACAO_BIBLIOG="T, A"
      TPO_ENDER_CORR="I" STA_PERMISSAO_APRES="N"
      COD_ORIGEM_CURRICULO="1">
      <EN_ENDERECO_RH TPO_ENDER="I"/>
      <EN_PRODUCAO_CIENTIF_TECNOL SEQ_PRODUCAO="1"
            COD_TIPO_PRODUCAO="111"
            ANO_PRODUCAO="1994"
            TXT_TITULO_PRODUCAO="Venir Après: Prendre la Parole Dans
Une Theatre Historique"
            SEQ_ORDEM_AUTORIA="2" TXT_OBSERVACAO=" "/>
      <EN_PRODUCAO_CIENTIF_TECNOL SEQ_PRODUCAO="2"
            COD_TIPO_PRODUCAO="111"
            ANO_PRODUCAO="1994"
            TXT_TITULO_PRODUCAO="Teoria e História: Incompatibilidades
e Reconciliações"
            SEQ_ORDEM_AUTORIA="2" TXT_OBSERVACAO=" "/>
      <EN_PRODUCAO_CIENTIF_TECNOL SEQ_PRODUCAO="3"
            COD_TIPO_PRODUCAO="111"
            ANO_PRODUCAO="1995"
            TXT_TITULO_PRODUCAO="Capuchinho Vermelho em Portugal"
            SEQ_ORDEM_AUTORIA="2" TXT_OBSERVACAO=" "/>
</EN_RECURSO_HUMANO>
```

Exemplo 3: Documento centrado nos dados

No modelo centrado no documento inserem-se documentos destinados ao consumo de pessoas [Obasanjo 2001; Bourret 2005b]. Estes documentos caracterizam-se por possuir uma estrutura irregular e uma maior granulosidade na informação [Bourret 2005b], isto é as unidades mais elementares de informação podem estar ao nível de elementos com conteúdo misto. Estes documentos são tratados como um todo, e a ordem pela qual os vários elementos com o mesmo nível na estrutura são representados é normalmente relevante. Exemplos deste tipo de documentos podem ser documentos escritos à mão e convertidos posteriormente para XML, manuais, paginas *Web* estáticas, etc. O seguinte exemplo (Exemplo 4) representa um documento centrado no documento.

```
<?xml version="1.0" encoding="UTF-8"?>
<artigo>
      <nome>The Entity-Relationship Model-Toward a Unified View of
```

```
Data</nome>
      <abstract>
            <paragrafo>A data model, called the <b>entity-relationship
model</b>, is proposed. This model incorporates some of the important
semantic information about the real world. A special diagrammatic
technique is introduced as a tool for database design. An example of
database design and description using the model and the diagrammatic
technique is given. Some implications for data integrity, information
retrieval, and data manipulation are discussed.</paragrafo>
            <paragrafo>The <b>entity-relationship model</b> can be used
as a basis for unification of different views of data: the network
model, the relational model, and the entity set model. Semantic
ambiguities in these models are analyzed. Possible ways to derive
their views of data from the entity-relationship model are
presented.</paragrafo>
      </abstract>
      <introducao>
            <paragrafo>The logical view of data has been an important
issue in recent years. Three major data models have been proposed: the
network model <referencia>[2, 3, 7]</referencia>, the relational model
<referencia>[8]</referencia>, and the entity set model
<referencia>[25]</referencia>.
            </paragrafo>
      </introducao>
</artigo>
```

Exemplo 4: Documento centrado no documento

Quando existem grandes quantidades de informação representada em XML e é necessário armazená-la em BDs há que ter em conta a forma como esta está estruturada. Podemos ter uma visão da informação armazenada nos documentos XML centrada nos dados ou uma visão mais orientada nos documentos.

Numa visão centrada nos dados o XML pode representar uma informação altamente estruturada, obedecendo a uma determinada estrutura (que pode ser representada recorrendo a DTD ou XML *Schemas*), tal como acontece com a informação representada noutros modelos, como o relacional ou orientado por objectos, a qual obedece a uma estrutura predefinida. De uma forma semelhante ao que acontece à informação representada nesses modelos, a ordenação dos elementos não é considerada relevante. Isto é algo que não acontece numa visão centrada no documento, onde a

informação não é estruturada e em cujos documentos a ordem implícita e explícita dos elementos é importante. A ordem implícita é a ordem pela qual os documentos estão organizados no documento e a explícita pode ser expressa por um atributo ou *tag* no documento [Nambiar et al. 2002a]. Como é referido em [Nambiar et al. 2002a], exprimir a ordem implícita ao converter numa representação relacional um documento XML orientado ao documento é algo complexo (o mesmo não se passando com a explícita) resultando numa transformação dispendiosa em termos de tempo e espaço.

Num armazenamento recorrendo a BDs com suporte para XML, é necessário traduzir a estrutura existente no XML (*schema*) em causa para uma estrutura (um *database schema*) na BD que irá receber essa informação. No caso de uma BD relacional seria a criação de uma estrutura adequada de tabelas, ficando a informação representada em XML dispersa pelas várias tabelas.

Nesta abordagem há que ter em atenção que documentos criados posteriormente a partir desta informação armazenada na BD, podem não ser exactamente os mesmos que os originais (que estiveram na origem da informação que foi armazenada na BD), pois esta abordagem omite a estrutura física e lógica dos documentos. Esta abordagem centra-se nos dados, pelo que a recuperação dos documentos[20], resulta em documentos diferentes, o que não é compatível com uma visão centrada no documento. No entanto, isto é algo que pode ou não ser aceitável dependendo das aplicações que irão utilizar estes documentos. Nestas situações onde a aplicação tem uma visão centrada nos dados, a adopção de BDs com suporte para XML tem tendencialmente um melhor desempenho [Vakali et al. 2005]. De qualquer das formas, mesmo em situações onde se tem uma visão centrada nos dados é possível armazenar a informação em BDs Nativas XML. Desta forma a informação é inserida na BD organizada em documentos.

Para uma visão centrada no documento, é necessária uma solução que possibilite o armazenamento de grandes documentos e que disponha de mecanismos de recuperação dos mesmos de forma eficiente utilizando tecnologias para esse efeito como o XPath ou XQuery, bem como possibilite a recuperação dos documentos na sua forma original. Soluções de armazenamento em sistema de ficheiros ou CLOBs em BDs tradicionais podem ser utilizadas, no entanto as BDs Nativas XML são a solução mais adequada nestes casos [Nambiar et al. 2002b; Vakali et al. 2005].

[20] O termo utilizado por [Bourret 2005b] é *round-tripping* dos documentos.

Capítulo 4

4 *Benchmarks*

Existe uma grande diversidade de sistemas que manipulam e armazenam informação, baseada em diversos modelos como o relacional, orientado por objectos ou XML, com desempenhos e características variados. A selecção do sistema e modelo mais adequados para uma dada aplicação passa inevitavelmente pelo conhecimento das suas características bem como do seu desempenho. A necessidade de comparar desempenhos e características surge assim de uma forma natural na altura em que os gestores de Tecnologias de Informação (TI) devem tomar a decisão do melhor sistema que satisfaça os requisitos de determinada aplicação. Esta necessidade tem como consequência a criação de testes que deverão permitir comparar os sistemas com base em condições semelhantes, tomando em atenção a tecnologia existente e cenários utilizados em produção [Schmidt et al. 2001].

Um *benchmark* é programa utilizado para testar o desempenho de *software*, *hardware* ou um sistema [Collin 2002]. Mais concretamente, um *benchmark* para BDs pode ser visto como um conjunto de instruções utilizadas para medir e comparar o desempenho de dois ou mais SGBD. Isto é feito recorrendo à execução de experiências bem definidas cujas medidas de desempenho serão usadas para prever o desempenho do sistema [Seng et al. 2005].

Desta forma, na especificação de um *benchmark* são considerados 3 componentes principais [Menascé 2002]: o sistema a ser testado (SUT[21]), a carga de trabalho submetida ao SUT (*workload*), que consiste nas operações de teste, e uma ou mais métricas que são resultantes da monitorização e avaliação do desempenho do SUT o qual inclui a BD de teste (Figura 11). Exemplos de métricas são *throughput*, tempo de resposta, tamanho da BD e relação desempenho / custos de manutenção.

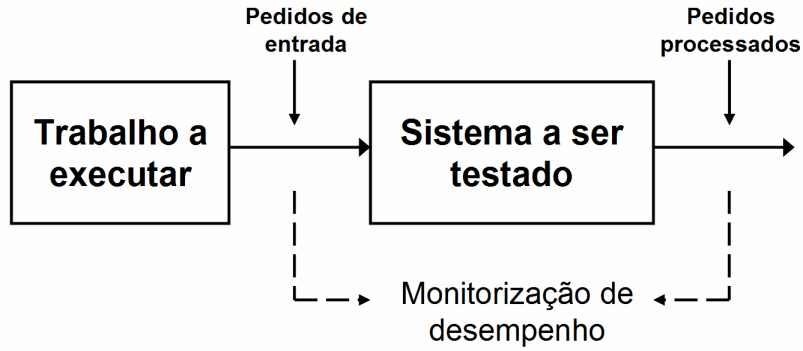

Figura 11: Estrutura de um *benchmark* (adaptado de [Menascé 2002])

Os sistemas, que serão alvos de teste dos *benchmarks*, são construídos de forma a darem resposta a problemas específicos de determinados domínios de aplicação. A utilização de *benchmarks* genéricos, em sistemas de vários domínios de aplicação, não conduz a resultados fiáveis dada a natureza dos sistemas, já que uns estarão mais aptos para certo tipo de tarefas que outros. Desta forma em [Gray 1993] defende-se o uso de *benchmarks* específicos para os diversos domínios, os quais deverão respeitar os princípios básicos:

- **Relevância:** deverá capturar as características do sistema a ser medido, executando operações comuns no respectivo domínio;
- **Portabilidade:** deverá facilmente ser implementado em diferentes sistemas, com diferentes arquitecturas;
- **Escalabilidade:** deverá ser aplicável a sistemas de pequenas e grandes dimensões;
- **Simplicidade:** deverá ser perceptível de forma a ser credível.

Sendo assim, na carga de trabalho submetida aos sistemas e na BD de teste, os *benchmarks* deverão tentar caracterizar aplicações típicas do domínio em causa, isto é deverão ter em conta aplicações reais do domínio, tendo em conta os cenários de

[21] Da língua inglesa: *system under test*.

utilização desse domínio.

Os *benchmarks* antigos estavam orientados para o teste das funcionalidades das BDs a um nível geral e não tinham em consideração nenhum cenário de utilização em particular. Com a evolução das BDs e suas funcionalidades estas começam a ser vistas cada vez mais como uma parte integrante dos sistemas. A sua integração nos diversos cenários de aplicação tornou-se cada vez mais importante, não sendo vistas como elementos isolados mas como parte integrante de aplicações complexas. Daqui, a necessidade de os *benchmarks* modelarem cenários reais de aplicações [Schmidt 2002]. Este aspecto pode ter duas faces. Se por um lado a não modelação de cenários reais pode levar eventualmente à criação de *benchmarks* que não satisfaçam as necessidades de aplicações onde eles sejam utilizados, por outro lado a modelação poderá levar a *benchmarks* demasiado específicos. Este tipo de *benchmarks* é específico de determinadas aplicações e são desenvolvidos com o intuito de as testar com base em critérios próprios. Este é o caso do *benchmark* SAP [SAP AG 2004]. Desta forma, um *benchmark* que se pretende de aplicação geral deverá assim ser relevante no domínio da aplicação que modela, tentando abordar o maior número possível de situações.

Os *benchmarks* podem também ser classificados como sintéticos ou empíricos [Seng et al. 2005] ou uma mistura entre ambos. *Benchmarks* sintéticos emulam aplicações típicas de um determinado domínio, tanto a nível de operações como de BDs de teste, ao passo que os empíricos utilizam operações de testes e informação reais. Estes últimos representam um ambiente realista mas são difíceis de obter e alterar para responder a certas necessidades de implementação e poderão apresentar elevados custos de instalação. Sendo assim, os *benchmarks* sintéticos, que não apresentam estes inconvenientes, são mais usados. No entanto, para este tipo de *benchmarks*, levanta-se uma questão importante que é a da representatividade da aplicação utilizada do domínio em causa, o que pode levar a que os resultados possam não retratar correctamente determinados sistemas.

Para sistemas baseados em modelos mais comuns como o relacional, foram desenvolvidos ao longo do tempo diversos testes (ver secção seguinte) de desempenho que estimularam a comparação de desempenho e consequentemente ao aperfeiçoamento dos sistemas.

Com o aparecimento de sistemas capazes de lidar com XML, foi necessário criar testes

que tenham em consideração novos desafios colocados por este modelo. Estes, em [Schmidt et al. 2002] são especificados como:

- Preservação de ordem textual das várias estruturas que compõem os documentos manipulados;
- Utilização de *strings* como tipo de dados básico cujo armazenamento e manipulação podem levantar problemas aos sistemas e podem entrar em conflito com a forma como os tipos de dados são tratados pelas linguagens de *query*;
- *Queries* que envolvem a manipulação de estruturas hierárquicas complexas e a preservação de ordem requerem a execução de operações dispendiosas principalmente quando o XML está armazenado numa estrutura relacional.
- Utilização de *Schemas* pouco rígidos de validação dos documentos[22]. Isto não só não facilita a tarefa de escrita de *queries* complexos por parte dos utilizadores, como também pode levantar problemas a nível de optimização do armazenamento da informação, introduzindo valores NULL que podem fazer aumentar o tamanho da BD.

De seguida são descritos sucintamente alguns dos *benchmarks* mais divulgados quer a nível do modelo relacional como XML.

4.1 *Benchmarks* utilizados em Bases de dados relacionais

Existem vários *benchmarks* utilizados em BDs relacionais, tais como Wisconsin [DeWitt 1993], AS^{3}AP [Turbyfill et al. 1989], mais orientados para o teste de aspectos importantes do servidor de BD, sendo constituídos por uma BD de teste e alguns *queries*.

[22] Um *schema* rígido (*"tight"*) valida de forma precisa a informação num documento XML. Ele define um modelo rígido e preciso para os elementos do documento apresentando características como sejam, por exemplo, a não permissão de conteúdo misto, o uso rigoroso de sequencias ou escolhas de elementos, definição de enumerações, a definição do número máximo de ocorrências, definição tipos de dados, etc. Por outro lado, um *schema* pouco rígido (*"loose"*) é o oposto do rígido, permitindo conteúdo misto e não definindo nenhuma das destas características (bem como outras) de forma rigorosa. Como é natural, entre estes dois extremos, existem *schemas* intermédios e que são definidos de acordo com as necessidades da aplicação onde se enquadram.

Outros *benchmarks* importantes são os definidos pelo *Transaction Processing Performance Council* (TPC), um consórcio de vários fabricantes de *software* e *hardware* sem fins lucrativos. Este consórcio define *benchmarks* para vários fins, como processamento de transacções *online* (OLTP), comércio electrónico, apoio à decisão, etc. Os resultados da utilização destes *benchmarks* são publicados regularmente.

Dois *benchmarks* definidos pelo TPC são o TPC-C [TPC 2005] e o TPC-W [TPC 2002]. Estes *benchmarks*, que como referem as suas especificações poderiam ser utilizados por qualquer SGBD, não só relacionais, têm grande aceitação na indústria. Por este facto, nas secções seguintes é feita uma breve descrição de ambos.

Um outro exemplo é o orientado por requisitos [Seng et al. 2005], um *benchmark* que ao contrário dos referidos anteriormente é independente de aplicação e domínio. Pretende conseguir melhores resultados que os anteriores na medida em que modela a carga de trabalho de acordo com as necessidades do utilizador.

4.1.1 TPC-W

O TPC-W é um *benchmark* transacional que pretende simular um ambiente de comércio electrónico (*e-commerce*), mais concretamente uma livraria *online*, sendo baseado numa estrutura cliente / servidor. As principais características deste *benchmark* são [Menascé 2002]:

- Sessões múltiplas de clientes (*browsers*);
- Páginas dinâmicas com acesso à BD, para consultas e actualizações;
- Autenticação usando SSL (*Secure Sockets Layer*) e TLS (*Transport Layer Security*);
- Emulação de pagamentos;
- BD com múltiplas tabelas e relações;
- Transacções com a BD que suportam as propriedades ACID (atomicidade, consistência, isolamento e durabilidade);
- Execução de transacções *online* o que leva a restrições no acesso e actualização da informação devido a concorrência.

O ambiente implementado por este *benchmark* é constituído por três partes principais: o

sistema a ser testado, os *browsers* emulados e uma emulação de um *gateway* de pagamentos (Figura 12). O sistema a ser testado consiste em tudo aquilo que é necessário para implementar um sítio *web* de comércio electrónico como um servidor *web*, um servidor de BD que responde às transacções requeridas pela aplicação e ligações de rede entre eles.

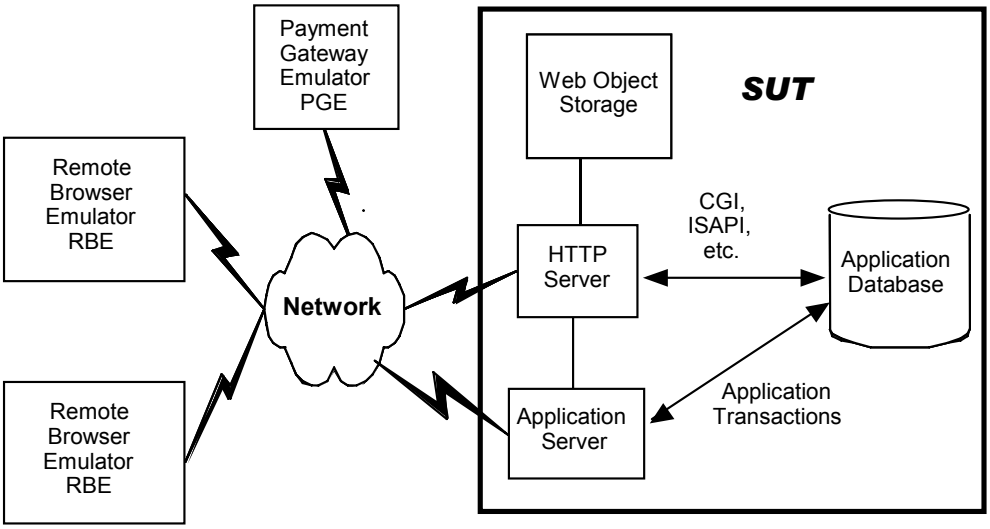

Figura 12: Ambiente do *benchmark* TPC-W[TPC 2002]

Os *browsers* emulados simulam utilizadores humanos, gerando pedidos a intervalos regulares[23] ao SUT usando o protocolo HTTP através da rede, os quais constituem a carga de trabalho a executar pelo SUT. O emulador de *gateway* de pagamentos tem como objectivo simular a última fase da cada compra, ou seja, o pagamento. Desta forma, para terminar uma encomenda, o SUT comunica com o emulador de *gateway* de pagamentos utilizando SSL.

Num ambiente internet a interacção entre os clientes e o sítio *web* é feitas em sessões, as quais são sequências de pedidos consecutivos para a execução de determinadas funções disponibilizadas pelo sítio *web* durante uma determinada visita [Menascé 2002]. Numa sessão o cliente pode navegar por várias páginas podendo o seu padrão de navegação ser representado num grafo (CBMG[24]), onde podem ser representados os vários estados possíveis (interacções), as funções utilizadas e probabilidades de mudança de entre estados.

Este *benchmark* define 14 interacções as quais são caracterizadas segundo três aspectos:

[23] O intervalo de tempo entre dois pedidos consecutivos é denominado por *think time*.

[24] CBMG – *Customer behavior model graph*

o número de imagens da página respectiva, o número de *joins* nas tabelas da BD necessários e o tempo máximo de resposta. Estas 14 interacções são ainda agrupadas em dois grandes grupos: consulta (*browsing*) que envolvem actividades de consulta e pesquisa e compra (*ordering*) que envolvem as actividades relacionadas com a execução de encomendas.

Para melhor caracterizar a utilização do sítio *web*, são definidos 3 perfis de utilização a nível da sessão, com percentagens variáveis de interacções dos tipos referidos:

- Perfil de consulta, com 95% de interacções de consulta e 5% de compra;
- Perfil de compra, com 80% de interacções de consulta e 20% de compra;
- Perfil de encomenda, com 50% de interacções de consulta e 50% de compra.

A Figura 13 representa o grafo dos padrões de navegação de cada sessão definidos pelos *browsers* emulados. Não inclui probabilidades de mudanças de estados dado que estas serão diferentes dependendo de cada perfil adoptado.

O TPC-W define uma BD com 8 tabelas para armazenar informação sobre livros, autores, utilizadores, encomendas e cartões de crédito. Este *benchmark* permite variar o tamanho da BD, o que é feito em função do número de *browsers* emulados e do número de itens no catálogo. As transacções na BD devem respeitar as propriedades ACID (Atomicidade, Consistência, Isolamento e Durabilidade). Em [TPC 2002] estas propriedades são definidas detalhadamente e podem ser descritas resumidamente como:

- **Atomicidade**: o SUT deve garantir que as transacções da BD são atómicas, isto é que as várias operações individuais tão todas executadas ou, em caso contrário, que nenhuma deixe marcas na informação;
- **Consistência**: o SUT deve garantir que após cada transacção a BD passa de um estado consistente para outro igualmente consistente;
- **Isolamento**: não deve existir qualquer interferência entre as várias transacções que podem ser executadas simultaneamente;
- **Durabilidade**: O SUT deve garantir que os efeitos das transacções que são persistentes.

A nível de métricas, o TPC-W define dois tipos: *throughput* e uma razão custo/*throughput*. Existem 3 métricas de *throughput*, correspondentes a cada um dos

perfis de utilização. Estas métricas medem o número interacções válidas executadas (no tempo máximo de resposta que as caracterizam) por segundo num determinado intervalo de tempo. A principal corresponde ao perfil de compra e denomina-se WIPS. As restantes denominam-se WIPSb, para o perfil de consulta e WIPSo para o perfil de encomenda. A razão custo/*throughput* denomina-se $/WIPS e representa a razão entre o custo do sistema a testar (inclui custos de compra e manutenção do *hardware* e *software*) pela a métrica WIPS (perfil de compra).

Existem várias implementações e estudos sobre este *benchmark*, tais como [JR et al. 2001; Seng et al. 2002; García e García 2003; Khouja et al. 2004].

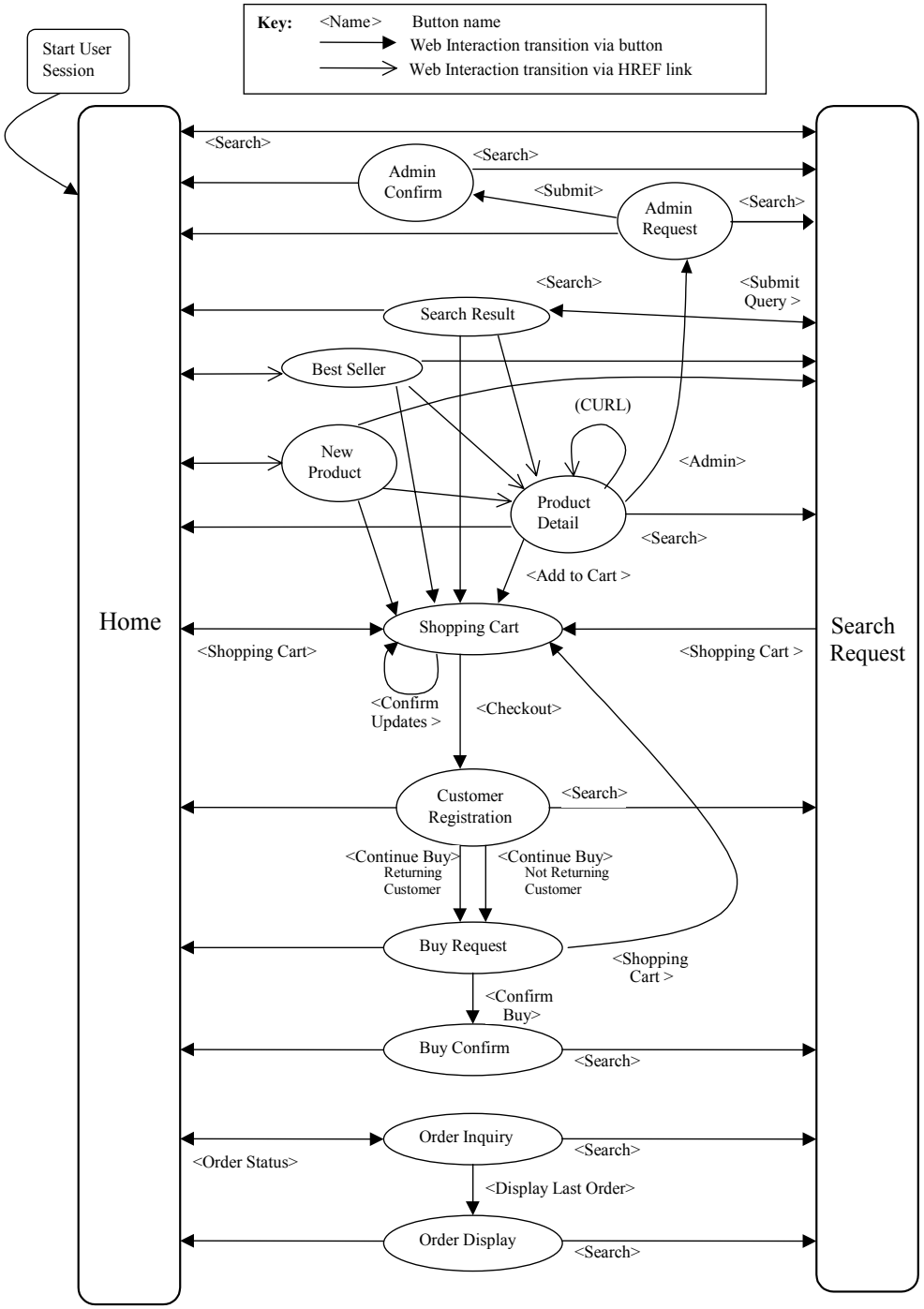

Figura 13: Padrão de navegação no *benchmark* TPC-W

4.1.2 TPC-C

O TPC-C, sucessor do TCP-A, é um *benchmark* para processamento de transacções *online* – OLTP – e é visto como o padrão para sistemas OLTP. Este *benchmark* pretende simular um sistema de OLTP genérico baseando-se para tal num sistema de gestão de inventário de um fornecedor grossista. O ambiente representado neste

benchmark consiste em vários armazéns distribuídos geograficamente. Cada um deles é responsável por um máximo de 10 áreas distintas e cada área tem 3000 clientes. A Figura 14 representa esta hierarquia.

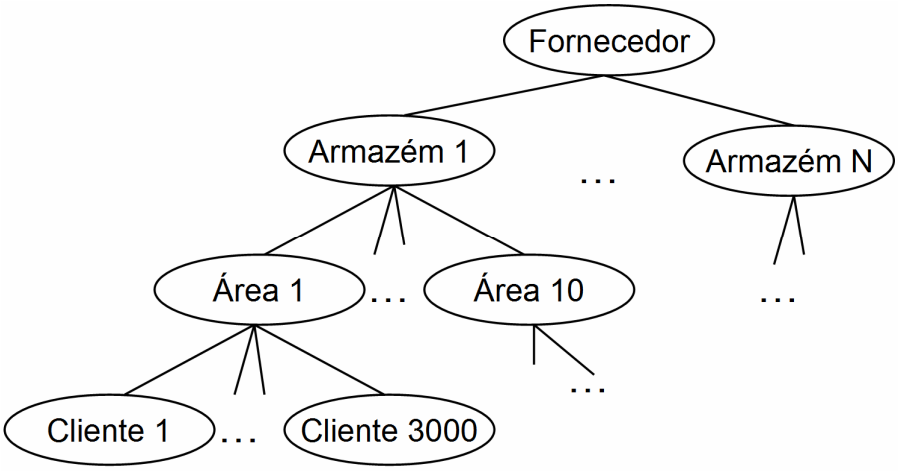

Figura 14: Sistema simulado do TPC-C (adaptação de [TPC 2005])

Neste sistema, operadores de vários terminais utilizam um conjunto de funcionalidades (transacções) representativas de um sistema deste género, as quais implicam a manipulação da BD. Existem 5 tipos de transacções, cada uma com diferentes custos a nível de execução:

- Lançamento de encomendas;
- Entrega de encomendas;
- Verificação do estado das encomendas;
- Registo de pagamentos;
- Verificação dos níveis de stock.

Para cada um destes tipos, o *benchmark* define valores mínimos (em percentagem) para o número de transacções presentes na carga submetida ao SUT num dado período de tempo. É executado o mesmo número de transacções de lançamento de encomenda e pagamento e uma transacção de verificação de estado da encomenda, entrega e verificação de stock por cada 10 lançamentos de encomendas.

Neste *benchmark*, o SUT consiste em todos os componentes necessários para executar a carga de trabalho, como a BDs e todo o *hardware* e *software* necessários, bem como ligações de rede entre os vários componentes. Neste sistema, a BD, que contém 9 tabelas, respeita as propriedades ACID e varia em termos de tamanho com número de

armazéns utilizados na execução do *benchmark*. A carga de trabalho é originada por terminais emulados que emulam a interacção dos utilizadores com o SUT, seleccionado transacções e recebendo mensagens de resposta do SUT.

A métrica utilizada para avaliar a performance do SUT denomina-se *tmpC* e mede o número de transacções de lançamento de encomenda executadas por minuto pelo sistema. Com base nesta métrica é possível calcular outra denominada preço/performance que tem em conta custos de manutenção do SUT e é expressa como preço/*tpmC*.

4.2 *Benchmarks* para Bases de dados XML

Estes *benchmarks* permitem testar sistemas para armazenamento / produção / manipulação de XML, nomeadamente a forma como processam os *queries* expressos numa dada linguagem de *query* XML. O desempenho da implementação por parte de um sistema de uma linguagem de *query* depende em grande medida das funcionalidades que esta oferece [Nambiar et al. 2001].

Existem várias linguagens de *query* XML (como por exemplo LOREL, XSU, XML-QL, XQL), que apresentam características diferentes mas predominantemente orientadas nos dados [Li et al. 2001]. Um exemplo de uma linguagem é o XQuery, que adiciona a estas funcionalidades outras para a manipulação de documentos. Estas linguagens deveriam suportar funcionalidades como permitir *queries* orientados nos dados e nos documentos ou mistos, suporte para manipulação de valores nulos ou utilização de quantificadores ($\forall$,$\exists$ e $\sim$) em *queries* e outras. Em [Carey et al. 1993; Bressan et al. 2003] é apresentada uma lista das funcionalidades desejadas para uma linguagem de *query* XML e que são especificadas em [Chamberlin et al. 2003].

Existem dois tipos de *benchmarks* para BDs XML: *micro-benchmarks* e aplicacionais. Todos eles consistem num conjunto de informação (BD - *data set*) que pode ter várias versões com tamanhos diferentes e sobre o qual serão executados um conjunto predefinido de *queries*.

Os *micro-benchmarks*, como o Michigan *Benchmark* [Runapongsa et al. 2003], são desenhados de forma a testarem componentes específicos do sistema com vista a isolar e ajudar a corrigir certos problemas. Pretendem explorar o impacto na performance do

sistema das características mais importantes do XML (definidas em [Runapongsa et al. 2003] como profundidade e granularidade), dispondo de um *data set* heterogéneo, não inspirado numa qualquer aplicação real, sobre o qual são especificados *queries* especialmente desenhados para testar componentes elementares da linguagem de *query* (como selecção, *joins*, etc). Com um *benchmark* deste tipo torna-se possível aperfeiçoar as operações mais básicas ao nível do SGBD.

Os *Benchmarks* aplicacionais, por seu lado funcionam a um nível mais elevado, pretendendo medir o desempenho do sistema como um todo e não questões específicas. Cada um deles dispõe de um *data set*, que pode ser ou não inspirado numa aplicação real, sobre o qual são definidos *queries* que pretendem abranger o maior número possível de características da linguagem de *query*. Exemplos destes *benchmarks* são o XOO7 [Li et al. 2001], Xmark [Schmidt et al. 2002], XBench [Yao et al. 2004] e o XMach-1 [Böhme e Rahm 2001].

4.2.1 XOO7

Este *benchmark*, desenvolvido na *National University of Singapore*, é baseado no OO7 [Carey et al. 1993], concebido para testar a performance de SGBD OO. O modelo de dados é baseado no do OO7. Em [Li et al. 2001] é apresentado o diagrama ER do modelo de dados que foi convertido para um DTD para uso no XOO7 tendo em atenção as semelhanças entre o modelo de dados XML e OO [Nambiar et al. 2001].

Também os *queries* foram importados e convertidos para linguagem de *query* XML, sendo no entanto adicionados novos *queries* para melhor explorar as características do XML, tentando abranger o máximo das funcionalidades que as linguagens de *query* deveriam suportar (conforme apresentadas em [Carey et al. 1993; Bressan et al. 2003]). Uma vez que os *queries* do OO7 abordam uma perspectiva da informação centrada nos dados, foi necessário expandir os queries do XOO7 de forma a abordarem uma perspectiva centrada no documento. Estes *queries* (18 no total) foram divididos pelos autores, conforme as características que pretendem explorar, em três grupos [Nambiar et al. 2001; Bressan et al. 2003]: *queries* relacionais, para testar funcionalidades mais comuns e já testadas noutros *benchmarks* como selecção/projecção, de navegação, para explorar funcionalidades de navegação na árvore XML e de documento, para explorar o suporte à abordagem centrada nos documentos da informação e ordenação da

informação.

O modelo de dados não pretende modelar nenhuma aplicação em concreto, mas antes objectos complexos relacionados entre si. Com este modelo, os autores construíram para os seus testes [Nambiar et al. 2002b] BDs de teste com três tamanhos distintos: pequeno (4.2MB), médio (8.4MB) e grande (12.8MB).

4.2.2 XMark

O XMark foi desenvolvido no *National Research Institute for Mathematics and Computer Science* (CWI) na Holanda. Ao contrário do XOO7, este *benchmark* baseia o seu modelo de dados numa aplicação real, mais propriamente numa aplicação Internet de leilões. Este modelo apresenta características predominantemente centradas nos dados mas também algumas centradas no documento, nomeadamente com a introdução de descrições textuais associadas a objectos representados no modelo de dados. Com base neste modelo, é gerada a BD de teste que consiste num único ficheiro XML recorrendo a uma ferramenta desenvolvida pelos autores. O documento gerado pode variar entre 10MB (com um factor de escala[25] de 0.1) e os 10GB (com um factor de escala 100) [Schmidt et al. 2002].

Este *benchmark* oferece um conjunto de 20 *queries* para avaliar a capacidade de processamento de *queries* por parte do sistema, não disponibilizado à semelhança do XOO7, qualquer operação de actualização de informação.

4.2.3 XBench

O XBench é uma família de *benchmarks* que pretende abranger vários tipos de aplicações de BD. Após análises estatísticas detalhadas de vários conjuntos de informação XML, os autores classificam as aplicações de BDs segundo duas vertentes.

A primeira refere-se às características da aplicação (tipo de informação da BD): *data-centric* - DC (centradas nos dados), como um catálogo de uma aplicação de *e-commerce* ou *text-centric* - TC (*document-centric*, centrado no documento), como dicionários ou arquivos de notícias.

[25] O tamanho base do documento é 100MB, isto é, factor de escala 1.

A segunda refere-se à forma da BD: *single document* (SD) documento único (toda a informação da BD está armazenada num único documento) ou *multiple document* (MD), documentos múltiplos (a BD é constituída por vários documentos).

Desta classificação surgem quatro classes, DC/SD, DC/MD, TC/SD e TC/MD. O XBench distingue várias classes de aplicações, algo que não acontece com os outros *benchmarks*, os quais consideram apenas um tipo de aplicação que utiliza um dado *schema*.

Para as classes TC, são criados *schemas* da BD baseados em aplicações reais (como a informação do *Oxford English Dictionary* ou arquivo de notícias da *Reuters*), combinando as suas características, tendo em conta dados estatísticos recolhidos sobre os documentos, bem como as suas estruturas. Para as classes DC, é utilizado o *schema* do *benchmark* TPC-W, o qual é adaptado e convertido para XML.

Esta família de *benchmarks* apresenta 20 *queries* (cada uma das classes não os utiliza todos) que cobrem todas as funcionalidades do XQuery tal como são descritas em [Chamberlin et al. 2003]. As BDs de teste utilizadas têm o tamanho de 10MB (pequeno), 100MB (normal), 1GB (grande) e 10GB (enorme).

4.2.4 XMach-1

O Xmach-1 foi desenvolvido na Universidade de Leipzig e publicado no início de 2001, tendo sido o primeiro *benchmark* publicado com aplicabilidade geral [Böhme e Rahm 2003].

Um aspecto que distingue este *benchmark* dos outros, é o facto de na sua arquitectura, baseada numa aplicação *web*, existirem outros elementos além do sistema de armazenamento de informação XML (BD). Considera também um servidor de aplicações (que em conjunto com o servidor de BD[26] constitui o sistema que é alvo de teste (SUT), isto é o sistema do qual se pretende tirar conclusões sobre o seu desempenho), bem como clientes (*browsers* e *loaders* – sistemas de aquisição de informação) para execução de consultas e actualizações de informação (estas tarefas são denominadas no seu conjunto como *workload*) os quais comunicam com o sistema alvo

[26] Ao contrário dos outros *benchmarks* que utilizam uma BD centralizada, neste caso a BD pode ser distribuída, com um número variável de servidores. Da mesma forma, o servidor de aplicações poderá ser distribuído.

de teste utilizando a Internet ou uma intranet. A Figura 15 ilustra esta arquitectura.

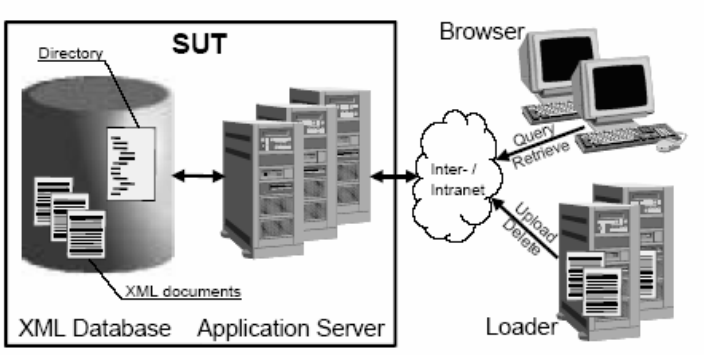

Figura 15: Arquitectura do XMach-1 (extraída de [Böhme e Rahm 2003])

Este *benchmark* é multi-utilizador sendo o seu principal objectivo a medição do *throughput*, definindo para tal uma métrica denominada Xqps (xml *queries* por segundo).

A BD contém documentos centrados nos dados e no documento. Ela é constituída maioritariamente por documentos centrados no documento que reproduzem objectos como livros ou ensaios (onde existem secções, parágrafos, etc) que os autores denominam de documentos geridos e cujo tamanho médio é 16KB. Existe também um documento (centrado nos dados) onde é armazenada meta-informação sobre cada um dos outros documentos, tais como URL, nome, hora de inserção na BD, etc.

Para realizar testes, a BD é iniciada com documentos geridos gerados sinteticamente (bem como com as respectivas entradas no documento que contém a sua meta-informação) que podem variar entre os 10.000 e os 10.0000.000. Sobre esta BD podem ser executados 8 *queries* e 3 operações de actualização.

4.2.5 Conclusão

A seguinte tabela (Tabela 4) apresenta um resumo de algumas das características referidas anteriormente.

	XOO7	XBench	XMach-1	XMark
Tipo de dados da BD de teste (*data set*) (centrados no documento / dados)	Predominan-temente centrado nos dados.	Centrado nos dados/centrado no documento	Predominante-mente centrado no documento.	Predominante-mente centrado nos dados, mas também com algumas características centrado no documento.
Tipo de BD	Uni-documento	Uni/multi-documento	Multi-documento	Uni-documento
Nº de *Queries*	18	20	8 + 3 de actualização	20
Tamanho BD	Desde 4MB	10MB-10GB	10.000 - 10.000.000 de documentos	10MB-10GB

Tabela 4: Resumo das características dos *benchmarks*

Existem várias comparações sobre estes *benchmarks* [Nambiar et al. 2001; Yao et al. 2002; Böhme e Rahm 2003; Bressan et al. 2003] que evidenciam as diferenças de características existentes entre eles. Os *benchmarks* apresentam diferenças por exemplo a nível do domínio de aplicação (por exemplo XMach-1 e Xmark são baseados numa aplicação E-Commerce), no número que *queries* que apresentam e respectiva abrangência no que diz respeito às características das linguagens de *query*, no suporte de BDs de múltiplos documentos ou de documento único, entre vários outros aspectos.

A selecção de um *benchmark* poderá passar pelo conhecimento da natureza das aplicações a que se destinam os sistemas. Desta forma, será possível seleccionar um *benchmark* que apresente características de certa forma semelhantes à aplicação em causa.

Capítulo 5

5 Concepção de um sistema de testes

Para tirar conclusões sobre os modelos mais adequados para a utilização em repositórios de informação *online* torna-se necessária a execução de testes, isto é de *benchmark*.

Trabalhos como [Böhme e Rahm 2001; Li et al. 2001; Schmidt et al. 2002; Yao et al. 2004] abordam questões relacionadas com o processamento de XML nomeadamente o processamento de *queries* cujo objectivo é abranger o maior número possível de características da linguagem de *query*, pretendendo medir o desempenho do sistema como um todo, existindo um foco exclusivo em BDs de XML. Da mesma forma, outros trabalhos como [Turbyfill et al. 1989; DeWitt 1993] abordam questões de processamento de *queries* em sistemas relacionais.

Na presente situação o objectivo não é o mesmo que o dos trabalhos referidos uma vez que não se está particularmente interessado em explorar questões de processamento de XML ou características das linguagens de *query* ou ainda a capacidade de processamento destes pelos sistemas. A atenção foca-se antes na criação de um *benchmark*, que se deseja bastante simples na medida em que não se pretende a criação de sistemas de teste e de um conjunto de regras de funcionamento complexos. O

objectivo é auxiliar na escolha de um modelo, XML ou relacional, para o armazenamento da informação num contexto específico de pesquisa de informação, deixando de parte operações de inserção ou actualização. Esta simplicidade reflecte-se a nível da construção do sistema que permite a execução dos testes, nomeadamente da sua arquitectura e pesquisas utilizadas, bem como nos resultados medidos que deverão ser de fácil interpretação e, preferencialmente, de fácil transposição para uma forma gráfica com o objectivo de favorecer a sua leitura. Desta forma, a escolha dos modelos será feita com base em cenários de utilização previamente definidos e com recurso a um sistema que utiliza diferentes BDs de teste, o qual irá permitir tirar conclusões sobre o desempenho de cada modelo.

Apesar desta simplicidade que é desejada, como qualquer *benchmark*, deverá respeitar, além desta, as restantes características básicas enunciados em [Gray 1993]: relevância, portabilidade, escalabilidade. Apesar de ser considerado um princípio básico importante, a portabilidade não será aqui um aspecto de grande relevância dado que o objectivo não é comparar desempenho entre sistemas diferentes, já que se pretende que o sistema onde são executados os testes seja uma constante e não uma variável. Mesmo assim, os testes poderiam ser facilmente adaptados a outras plataformas, bastando para tal uma nova implementação utilizando tecnologias compatíveis com estas.

No que diz respeito à relevância, nestes testes tenta-se abranger situações mais comuns numa aplicação de consulta de informação. No que se refere à escalabilidade, os testes envolvem BDs com tamanhos distintos, o mesmo acontecendo ao número de clientes simultâneos conectados ao sistema. Nos testes poderiam ser utilizados quaisquer valores para estas duas variáveis, desde que suportados pelo sistema desenvolvido.

O restante deste capítulo descreve os cenários que se pretendeu testar com a construção deste *benchmark*, bem como as suas características.

5.1 Cenários

Como é referido no Capítulo 3, é possível armazenar informação recorrendo a várias soluções. A informação pode estar modelada segundo o modelo relacional, devidamente inserida em registos de tabelas numa BD, sendo este um cenário muito comum. No entanto, pode-se optar por representar esta mesma informação não utilizando o modelo

relacional mas XML. Poderá no entanto levantar-se a questão: porquê representar a informação em XML? Uma resposta possível é que o XML é actualmente um meio privilegiado para a troca de informação. Desta forma, tendo sido a informação devidamente representada em XML, torna-se mais fácil proceder ao intercâmbio de informação entre aplicações, evitando-se o processo de conversão entre modelos. Por exemplo, pode ser possível exportar os resultados de pesquisas no formato XML para outras aplicações estando desta forma a tarefa mais facilitada. Um outro aspecto importante é a facilidade de manipulação do XML. Com as tecnologias associadas ao XML, um documento pode ser facilmente manipulado para originar outros ou ser visualizado de formas distintas.

Sendo o modelo relacional um dos mais utilizados actualmente e estando o XML a ganhar cada vez mais aceitação, existe todo o interesse em estudar o comportamento destes dois modelos. Para tal é necessário seleccionar BDs para estas duas situações. Se para o caso do modelo relacional, a escolha não apresenta qualquer dificuldade, escolhendo-se o único tipo de BD possível (relacional), no caso do XML a escolha tem mais opções.

A informação pode ser encarada de duas formas distintas, tal como referido anteriormente (Capítulo 3): centrado nos dados ou centrado nos documentos. Apesar de não existirem regras rígidas para o armazenamento de XML, cada uma destas hipóteses será mais adequada para utilização num tipo distinto de BD. Desta forma, uma BD com suporte para XML terá melhor desempenho com informação centrada nos dados, ao passo que uma visão centrada no documento é mais adequada para uma BD nativa XML.

A informação que será utilizada nos testes representa currículos (CVs) de investigadores (conforme descrito na secção 5.2.3). Inicialmente esta informação estava armazenada numa BD relacional, tendo sido posteriormente convertida para XML de forma a obter-se informação representada neste formato para utilização nos testes (ver Capítulo 6 os pormenores de implementação). O Exemplo 3 (página 41) representa um dos documentos XML gerados. Estes documentos apresentam características que os classificam como centrados nos dados, como seja inexistência de conteúdo misto ou uma pequena granulosidade na informação.

Uma característica que diferencia documentos centrados nos dados de centrados no

documento é o seu destinatário. Apesar dos documentos envolvidos nos testes apresentarem características claramente centrados nos dados e por conseguinte se poder pressupor que eles serão exclusivamente para uso de computadores, eles apresentam *tags* descritivos, que facilitam a sua leitura e compreensão por parte de humanos. Desta forma, esta não será uma característica diferenciadora neste caso em particular, até porque em última instância, toda a informação contida no CV é destinada a ser consultada por humanos.

Sendo os documentos XML utilizados nos testes claramente classificados como centrados nos dados, pode ser utilizada uma BD com suporte para XML para os armazenar. No entanto, e apesar das BDs Nativas XML poderem ser as mais apropriadas em situações cuja informação envolvida é centrada nos documentos, também poderão ser utilizadas em situações cuja informação é centrada nos dados.

Desta forma, terá interesse estudar cenários onde sejam utilizadas para o armazenamento físico de informação uma BD relacional (para a representação da informação no modelo relacional), com suporte para XML ou nativa XML (para suportar informação representada em XML). Estas três hipóteses são os valores para a primeira variável nos testes – modelo de dados / tipo de BD.

Um outro aspecto importante com interesse em ser testado, é a escabilidade, a qual foi referida no Capítulo 4 como uma das características principais de um *benchmark*. Um ponto em comum a todos os *benchmarks* referidos no Capítulo 4 é o facto de existir a possibilidade da realização de testes em BDs de diferentes dimensões em termos do tamanho em *bytes* ou registos inseridos. Isto é importante na medida em que se torna possível tirar conclusões sobre a influência do tamanho da BD no desempenho do sistema, o que poderá ser relevante na previsão do comportamento da aplicação quando surgir a necessidade de crescimento a nível do armazenamento de informação.

Um outro ponto associado à escalabilidade é a carga a que o sistema está submetido, o que pode ser influenciado pelo número de clientes simultâneos que acedem ao sistema. *Benchmarks* como o TPC-W ou XMach-1 permitem seleccionar o número de clientes, variando assim a carga do sistema. De forma semelhante ao tamanho da BD, a variação do número de clientes poderá ser relevante na previsão do comportamento da aplicação em situações de grande carga no sistema.

Estes dois pontos, tamanho da BD e número de clientes, constituem assim as restantes

variáveis para os testes a serem executados. Desta forma, com cenários que abarcam todas as combinações destas variáveis, é possível abranger um grande número de situações, atribuindo um maior realismo ao estudo a efectuar.

5.2 Sistema de testes

As secções seguintes descrevem as características do sistema de testes, nomeadamente o modelo, onde se descrevem as características que se pretende testar, a arquitectura, onde se descreve o sistema de testes implementado e por fim a base de dados, onde se descreve a informação que foi utilizada nos testes.

5.2.1 Modelo

No trabalho actual, existe à partida um contexto específico (a pesquisa de informação) que enquadra os cenários definidos. Além disto, o objectivo principal destes testes é tirar conclusões sobre a utilização de dois modelos, o relacional e XML, para o armazenamento de informação. Esta é uma abordagem diferente de outros trabalhos (por exemplo [Böhme e Rahm 2001; Schmidt et al. 2002; TPC 2002; TPC 2005]) dado que estes apresentam objectivos e contextos diferentes do que é proposto no trabalho actual. Estes trabalhos pretendem simular uma dada aplicação de um determinado domínio com o objectivo de realizar testes de forma a abordar o maior número de aspectos relevantes possível. No trabalho actual, cujo domínio são os serviços de informação, este objectivo mantém-se, mas tendo em consideração o objectivo principal e o contexto específico.

Os modelos a testar, definidos pelo objectivo principal deste trabalho, apresentam características bastante diferentes (ver Capítulo 3), pelo que os aspectos a considerar nos testes a executar deverão abranger características existentes que façam sentido em ambos. Da mesma forma que os modelos a testar limitam os possíveis aspectos a considerar nos testes, o contexto do trabalho, da consulta de informação, também os influencia.

Em [Schmidt 2002] são identificados 10 desafios que segundo o autor os *benchmarks* para BDs XML deverão superar de forma a cobrir todos os aspectos críticos do processamento de XML:

- ***Bulkloading*:** carregamento em massa de informação;
- **Reconstrução:** capacidade de devolver ficheiros XML tal como foram inseridos na BD;
- **Travessias** *(Path traversals)*: navegação em documentos estruturados;
- **Conversões** *(Casting)*: conversões de tipos entre os elementos XML que são texto e outros tipos, como inteiros, reais, etc.;
- **Elementos em falta:** o mapeamento do XML para diferentes formas de armazenamento pode resultar em valores nulos (NULL), nomeadamente quando os elementos não são definidos como obrigatórios pelos DTDs ou *Schemas*, pelo que a manipulação e armazenamento de valores nulos ganha uma grande importância;
- **Acesso ordenado:** a noção de ordem está "omni-presente nos queries de XML e afecta todos os aspectos da gestão da informação" [Schmidt 2002];
- **Referências:** forma de modelar a informação que o XML suporta de uma forma semelhante às restrições estruturais no modelo relacional;
- ***Joins*:** os *queries* que incluem *joins* podem ser dispendiosos para a BDs e constituem uma característica que não era suportada por linguagens de *query* XML anteriores;
- **Construção de resultados de grandes dimensões:** este aspecto aqui é mais visto como o oposto da Reconstrução pois pretende-se testar a capacidade de construir resultados (composição, isto é, construir novos documentos com base na informação da BD) bem como a capacidade de lidar com grandes quantidades de informação.
- **Pesquisa em texto integral** (*Full-text search*).

Desta lista de aspectos essenciais para um *benchmark* para BDs XML facilmente se conclui que a maior parte deles não se adequa a outro modelo nem a um contexto de pesquisa de informação.

Desta forma, estando este trabalho centrado, tal como referido, numa perspectiva de consulta de informação, o carregamento em massa de informação (bem como qualquer outra operação de carregamento de informação para as BDs a utilizar nos testes) sai fora

do âmbito do que é proposto estudar. Para todos os estudos que serão conduzidos a informação estará previamente armazenada nas BDs de teste. No entanto, convém referir que esta operação é de grande importância. No caso das BDs XML esta operação que à partida poderá ser considerada de menor importância, é na realidade dispendiosa para a BD, sendo bastante influenciada pela a forma como o XML é armazenamento internamente pela BD. Existem várias abordagens para o armazenamento interno de XML e vários estudos sobre esta questão (como por exemplo [Florescu e Kossmann 1999; Schmidt et al. 2000; Khan e Rao 2001; Schmidt 2002; Seng et al. 2003]), mas uma abordagem mais aprofundada deste tema sai fora do âmbito deste trabalho.

Dos desafios propostos, o correspondente a *Construção de resultados de grandes dimensões* interessa ser visto na perspectiva da capacidade da BD lidar com resultados de grandes dimensões dado que a reconstrução é um conceito inexistente em BDs relacionais. O desafio correspondente a *Joins* é de bastante relevante em ambos os modelos. Este tópico é bem conhecido das BDs relacionais e assume também grande relevância nas BDs XML. Por último as pesquisas textuais são de grande importância dado que se trata de uma funcionalidade bastante utilizada nas aplicações. Estas pesquisas de informação por palavra ou frase requerem mecanismos optimizados, como indexação, e podem ser bastante exigentes para a BDs.

Em vez de tentar identificar os aspectos comuns aos dois modelos a serem testados, tenta-se neste trabalho uma outra abordagem que é a identificação de funcionalidades básicas que uma aplicação do domínio em causa – serviços de informação – deverá possuir. Esta abordagem justifica-se com a necessidade de uma abordagem de mais alto nível, mais distante de cada um dos modelos. Estas funcionalidades acabarão elas próprias por usar estas ou outras características.

Pode-se desta forma considerar que um serviço de informação com um perfil centrado na consulta de informação deverá permitir as seguintes funcionalidades básicas:

1. Acesso directo aos itens por chave (pesquisa por chave);
2. Obtenção de resultados ordenados segundo determinados critérios;
3. Obtenção de indicadores com base na informação armazenada;
4. Suporte eficiente para resultados de grandes dimensões (elevado número de itens);

5. Pesquisas textuais

Com base na informação utilizada nos testes (ver secção 5.2.3) podem ser dados os seguintes exemplos relativos a estas funcionalidades.

No ponto um pretende-se a recuperação de um item completo (isto é, um CV) com determinado identificador único.

No ponto dois pretende-se que os resultados das pesquisas possam ser ordenados utilizando para tal campos de texto ou numéricos.

No ponto três pretende-se a obtenção de indicadores com base na informação armazenada como por exemplo o número de pessoas com data de nascimento entre duas datas determinadas, o número de pessoas com determinado nome, o número de publicações de determinada pessoa ou a pessoa com mais publicações.

No ponto quatro pretende-se que exista suporte para pesquisas pouco especificas e que podem devolver um grande número de resultados. Um exemplo pode ser uma pesquisa que devolva os CVs de pessoas do sexo masculino/feminino.

No ponto cinco pretende-se que seja possível realizar pesquisas por texto, palavra ou frase, de forma eficiente. Um exemplo poderá ser uma pesquisa por todas as pessoas que têm "Manuel" na constituição do seu nome.

Estas funcionalidades irão ser reflectidas em pesquisas que deverão ser implementadas. Uma vez que se pretende executar testes com modelos distintos, as pesquisas irão envolver *queries* que deverão ter versões adaptadas a cada modelo e que devolvam resultados equivalentes.

A execução de testes pressupõe a medição de parâmetros relevantes. *Benchmarks* como o TPC-C, TPC-W ou o XMach-1 definem métricas baseadas numa medida importante, o *throughput*. Neste caso, pode-se definir o *throughput* do sistema como o número de pesquisas, baseadas nas características propostas, para o qual se obteve uma resposta por segundo. Desta forma os testes deverão ser realizados em intervalos de tempo previamente definidos e contabilizados os resultados com sucesso obtidos nesse período.

Com base no *throughput* pode ser calculado outro parâmetro, o tempo médio de resposta para um determinado período de tempo. Este pode ser calculado com base na lei do tempo de resposta [Menascé e Almeida 2001]:

$$R = \frac{M}{X_0} - Z$$

O tempo de resposta médio (R) pode ser calculado com base nos pedidos feitos ao sistema (M), no *throughput* do sistema (X_0) e no intervalo médio decorrido entre a obtenção da resposta a um pedido e a submissão de novo pedido por parte do mesmo cliente (Z, *think time*).

5.2.2 Arquitectura

Para a elaboração de testes torna-se necessária a criação de um sistema adequado, capaz de executar todos os testes pretendidos. A arquitectura do sistema implementado neste trabalho está representada na Figura 16.

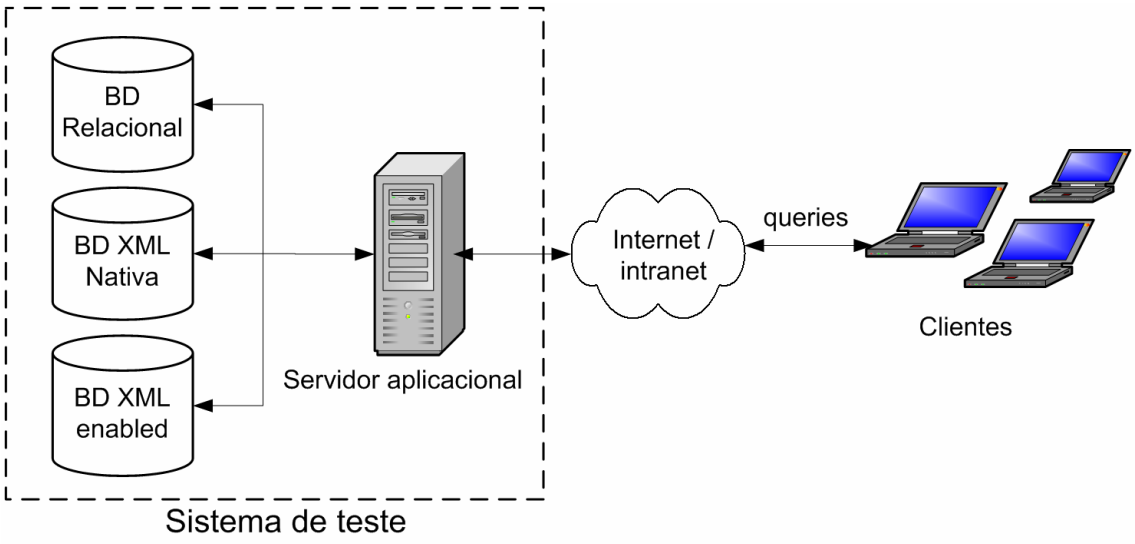

Figura 16: Arquitectura do sistema de teste

Esta arquitectura é inspirada nas propostas em [Böhme e Rahm 2001; TPC 2002] sendo igualmente baseada numa aplicação *web*. É composta por três componentes: BD, servidor aplicacional e clientes, não existindo a necessidade do componentes adicionais externos aos SUT como o que faz recolha e *upload* de informação para a BD (*Loader*) [Böhme e Rahm 2001] ou *gateway* de pagamentos [TPC 2002].

Na presente situação o sistema irá utilizar, em testes distintos, BDs para a manipulação de XML (com suporte para XML e nativa) e relacional, todas elas contendo informação equivalente representada nos respectivos modelos. Estes testes consistem em pesquisas básicas nas respectivas BDs com base nas funcionalidades básicas para um serviço de informação, propostas na secção anterior. Para estas pesquisas, 5 no total, foram criados

queries simples (descritos no Anexo I) que têm três versões, uma para cada tipo de BD utilizada, e que devolvem resultados equivalentes. Um aspecto que se procurou satisfazer com a criação de *queries* muito simples foi o facto de assim se evitarem problemas de desempenho relacionados com a possível má optimização destes ou respectiva tradução para as restantes versões.

O mecanismo de funcionamento das pesquisas está descrito na Figura 17. Os parâmetros de pesquisa são fornecidos à camada de acesso à informação que por sua vez executa a pesquisa, interpreta os resultados e devolve o produto final da pesquisa ao cliente.

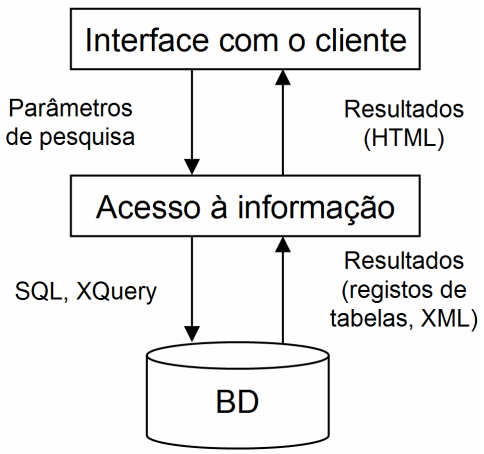

Figura 17: Fluxo de informação numa pesquisa

No servidor aplicacional (que tal como o servidor de BD poderia ser na realidade constituído por vários servidores) é executada uma aplicação *web* com base num servidor *web* (HTTP) bem como outros componentes de *software* necessários para acesso às BDs.

O conjunto do servidor aplicacional e de BD constitui o sistema de teste, sendo excluídos os clientes pelo que não se considera nos resultados dos testes atrasos devidos a comunicações e a processamentos feitos a nível dos clientes. Desta forma, os clientes conectam-se ao sistema de teste via o protocolo HTTP e todos os resultados são medidos a nível do servidor aplicacional tendo apenas em conta o intervalo de tempo entre a chegada do pedido e o final do seu processamento (ignorando o tempo que o resultado demoraria a chegar ao cliente). Para isto, a aplicação *web* mantém contadores que acumulam os números das respostas obtidas para cada tipo de pesquisa implementada, cujo resultado é calculado e enviado para o cliente.

Para simular a utilização do sistema por parte de vários clientes, existe uma aplicação

cujo objectivo é executar pedidos (HTTP) em paralelo à aplicação *web*, simulando assim vários clientes simultâneos. Para cada cliente que a aplicação simula, são executados pedidos e obtidas respostas de forma sequencial, conforme o esquematizado na Figura 18.

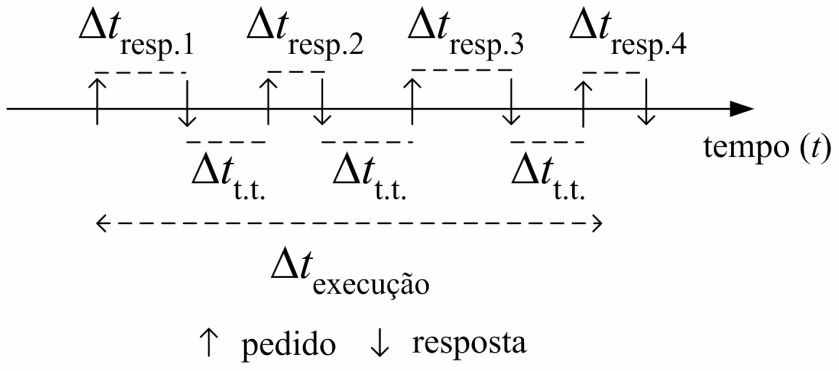

Figura 18: Execução de pedidos e obtenção de repostas de um cliente da aplicação

O intervalo de tempo entre a execução de um pedido e a obtenção da respectiva resposta é variável, dependendo de factores como a sua complexidade e a carga a que o servidor está sujeito. Por outro lado, cada pedido é feito com um determinado período de tempo de atraso após a obtenção da resposta ao pedido anterior (*think time* - $\Delta t_{t.t.}$). Neste trabalho optou-se por tornar este tempo numa constante. A existência deste período de tempo (independentemente de ser constante ou variável) adequa-se melhor ao que seria uma actualização humana, uma vez que um utilizador humano não está apto a lançar pedidos imediatamente após a recepção de resposta ao pedido anterior uma vez que é sempre necessário um período para interpretação de resultados.

Os pedidos dos clientes podem ser feitos durante um período de tempo previamente estipulado, ao fim do qual não serão contabilizadas as eventuais respostas que ainda se obtenham respeitantes a processamentos que tenham ficado a executar. Dado que a contabilização de respostas é feita a nível do servidor aplicacional, isto significa que a aplicação *web* deixa de contabilizar as respostas que calcula e envia para os clientes após o referido período de tempo de execução, podendo no entanto ficar a processar os pedidos ainda pendentes mas não contabilizáveis.

Imediatamente antes de fazer um pedido, cada cliente tem de decidir qual a pesquisa a utilizar das 5 disponíveis (Anexo I). Isto é calculado de forma aleatória onde a probabilidade de cada pesquisa ser utilizada no pedido é a seguinte (Tabela 5):

Pesquisa	Percentagem	Descrição
1	30%	Pesquisa por texto (nome)
2	23%	Acesso directo a um item (por código)
3	17%	Obtenção de indicadores com base na informação armazenada
4	15%	Ordenação dos resultados por um número
5	15%	Ordenação dos resultados por uma string

Tabela 5: Probabilidade de seleccionar as pesquisas

Desta forma, optou-se por um perfil de utilização onde existem mais pesquisas por texto, vários acessos ao item encontrado pelas pesquisas por código e onde foi dada menos importância à ordenação de resultados e obtenção de indicadores com base na informação armazenada. Este perfil de utilização foi mantido em todos os testes.

Para evitar atrasos relativos à rede, os servidores aplicacional e de BDs serão os mesmos, com dados armazenados localmente e com vista a evitar inconsistências nos resultados, não é permitida cache ao nível do servidor da aplicação *web*.

Na realização de testes e para simular diferentes cenários, além da mudança do tipo de BD, são utilizadas diferentes BDs com quantidades de informação distintas, bem como é alterado o número de clientes e consequentemente o número de pedidos ao servidor aplicacional.

5.2.3 Bases de dados de teste

A informação e *schemas* utilizados nas BDs de testes são baseados numa BD relacional já existente. A informação armazenada nesta BD é referente a pessoas, mais concretamente ao CV, sendo esta constituída por um misto de informação real e fictícia. Ao *schema* desta BD foi feita uma simplificação, mas mantendo a estrutura básica de um CV. O Anexo II mostra o diagrama ER (parcial) do *schema* obtido. De seguida, a informação relevante das tabelas utilizadas pelo novo *schema* simplificado foi copiada da BD relacional já existente, dando origem à BD relacional inicial que esteve na origem da criação das BDs utilizadas nos testes.

O XML *Schema* utilizado nas BDs de XML foi criado com base no *schema* relacional simplificado. Apesar de terem sido utilizadas ferramentas automáticas para esta tarefa, a versão final teve uma revisão manual, tendo sido seguidas as considerações do capítulo 6 para a criação de XML *Schemas* (usando atributos para representar as colunas das tabelas e usando os nomes das colunas no nome dos atributos). O Anexo III mostra o *schema* obtido.

Após a criação do XML *Schema* foi feita a conversão da informação relacional para o formato XML, respeitando este *schema*. Cada CV corresponde, a nível do modelo relacional a uma ou mais linhas em algumas ou mesmo todas as tabelas da BD. A nível de XML, cada CV corresponde a um documento. A abordagem é multi-documento, isto é, a cada CV corresponde um documento. *Benchmarks* como XOO7, XBench e XMark têm ou suportam a abordagem oposta, uni-documento, onde com toda a informação é criado um documento de grandes dimensões. A abordagem multi-documento é a mais apropriada para as BDs seleccionadas, uma vez que no caso da BD com suporte para XML, os documentos XML são armazenados numa tabela. Da mesma forma, no caso da BD nativa XML, os documentos são organizados em colecções de documentos.

Para responder aos vários cenários idealizados para os testes é necessário que sejam utilizadas BDs de teste com diferentes números de registos. Desta forma, a BD relacional inicial (com o *schema* simplificado) foi completada, até se obterem as quantidades de informação desejadas para os testes. Isto levou também à criação das respectivas BD de XML com o mesmo número de documentos.

Neste trabalho convencionou-se que irão ser utilizados 3 tamanhos distintos para as BD, quer para o modelo relacional, quer para XML. Estes tamanhos denominados de pequeno, médio e grande, são gerados com base na BD relacional inicial (com o *schema* simplificado), cujo número de registos na sua tabela principal era de 28.768. Desta forma, cada BD é construída em blocos de 28.768 registos que são gerados com base nos registos originais, com a ajuda de ferramentas especialmente desenvolvidas, dando origem a BDs com números de registos em múltiplos do tamanho original.

A Tabela 6 descreve os tamanhos de cada uma das BDs utilizadas nos testes (tanto relacional como de XML), bem como o seu tamanho relativo à BD inicial.

Descrição	Número de registos	Tamanho relativo
Pequeno	115.072	4
Médio	517.824	18
Grande	1.006.880	35

Tabela 6: Tamanhos das BDs utilizadas

Capítulo 6

6 Aspectos da Implementação

No decorrer do trabalho foram surgindo vários aspectos de ordem prática para os quais foi necessário dar resposta. Neste capítulo são abordados 3 aspectos importantes: a criação de documentos XML com base numa BD relacional, as ferramentas utilizadas e que foram criadas especialmente com vista a satisfazer as particularidades deste trabalho e as BDs seleccionadas que implementam os modelos que era proposto estudar.

6.1 Criação de documentos XML com base numa BD relacional

Como foi referido anteriormente, a fonte de dados é uma BD relacional, pelo que se torna necessário converte-los para XML de forma a ser possível inseri-los noutras BDs que suportam XML. Esta tarefa está dividida em duas fases. Em primeiro lugar é necessário obter um esquema XML baseado no esquema relacional e posteriormente, numa segunda fase, irão ser gerados documentos XML com base na informação armazenada na BD relacional e que respeitam o esquema XML criado. Para esta segunda fase existem várias ferramentas, no entanto optou-se por criar uma à medida

das necessidades (ver secção 6.2).

Para a primeira fase, a criação de um esquema XML, pretende-se criar um esquema que reflicta de uma forma credível o esquema relacional, reproduzindo da melhor forma possível a estrutura da BD relacional. A utilização de um XML *Schema* (um ficheiro *.xsd*) em vez um DTD será mais apropriada, uma vez que possuem características que este último não possui, como o ser escrito em XML ou permitir a definição de tipos de dados (ver secção 3.2.1)

Para a conversão do esquema relacional para um XML existem várias abordagens, como:

- A mais óbvia, uma conversão livre, seguindo o senso comum de quem a executa, mas que poderá não obter resultados satisfatórios principalmente em situações mais complexas;
- Abordagem seguida por [Williams et al. 2000] onde são definidas algumas regras simples que possibilitam a escrita de um DTD, nomeadamente a nível da utilização de elementos em oposição a elementos e de estrutura;
- Metodologia seguida descrita em [Du et al. 2001] que permite criar um XML *Schema* com base no enriquecimento semântico do esquema relacional e na aplicação do modelo ORA-SS [Dobbie et al. 2001];
- Algoritmo do algoritmo ConvRel [Duta et al. 2004] que permite para cada relação escolher a estrutura XML mais apropriada.

Neste trabalhou optou-se por uma abordagem simples como é a definida por [Williams et al. 2000], sendo criado no entanto um XML *Schema*.

6.1.1 Atributos *vs* elementos

Pretende-se criar documentos centrados nos dados destinados a serem processados por aplicações, pelo que a condição de serem facilmente legíveis por humanos não é fundamental.

Um ficheiro XML é um conjunto de componentes básicos: elementos e atributos. Para a estruturação de ficheiros XML, para uma abordagem centrada em elementos podem ser facilmente distinguidos os seguintes modelos de conteúdo [Williams et al. 2000]:

- Modelo baseado exclusivamente em elementos, onde os elementos apenas podem conter outros elementos (Exemplo 5):

```
<elemento>
      <subelemento1/>
      <subelemento2/>
</elemento>
```

Exemplo 5: Modelo de conteúdo baseado em elementos

- Modelo que permite conteúdo misto, que permite a intercalação de texto e elementos, como por exemplo (Exemplo 6):

```
<elemento>
      conteúdo de texto
      <subelemento>...</subelemento>
</elemento>
```

Exemplo 6: Modelo de conteúdo misto

- Modelo que permite apenas texto (caso especial do misto). Neste modelo é possível inserir valores num documento centrado nos dados os quais podem ser por exemplo extraídos de colunas de tabelas de BD relacionais (Exemplo 7):

```
<elemento>conteúdo de texto</elemento>
```

Exemplo 7: Modelo de conteúdo de texto

- Modelo EMPTY, onde os elementos não podem conter nada. Este modelo pode ser útil juntamente com a utilização de atributos (Exemplo 8):

```
<elemento/>
```

Exemplo 8: Modelo de conteúdo EMPTY

- Modelo ANY, onde os elementos podem ter qualquer tipo de conteúdo, como outros elementos e em qualquer ordem.

Como é referido em [Williams et al. 2000], os modelos misto e ANY devem ser evitados na estruturação de documentos XML centrados nos dados uma vez que são demasiados permissivos na representação da informação.

A outra abordagem na estruturação de ficheiros XML é o uso de atributos. Esta

abordagem, a par do modelo que permite apenas texto na abordagem centrada em elementos, permite inserir valores num documento centrado nos dados (Exemplo 9):

```
<elemento atributo1="valor1" atributo2="valor2"/>
```

Exemplo 9: Uso de atributos

Ao serem gerados documentos XML com origem numa BD relacional, coloca-se a questão de qual a abordagem a utilizar para representar o conteúdo (isto é, os valores das colunas das tabelas), ou seja, quais as vantagens de usar uma estrutura baseada em elementos (elementos que permitem apenas texto) ou baseada em atributos. Em [Williams et al. 2000], o autor faz uma análise destas abordagens com base nos seguintes critérios:

- Legibilidade;
- Compatibilidade com bases de dados;
- Tipos de dados;
- Complexidade a nível de programação;
- Tamanho do documento

Se a nível de legibilidade não há diferenças assinaláveis pois em documentos criados com cada uma destas abordagens é possível verificar estruturas bem delineadas que na realidade são equivalentes, já a nível dos restantes critérios o autor faz uma distinção clara.

Numa BD relacional há uma divisão clara entre estrutura e conteúdo: a estrutura é expressa nas tabelas e relações entre estas e o conteúdo são os valores das colunas das tabelas. Numa abordagem baseada em elementos, estes são usados para definir a estrutura, bem como para a definir o conteúdo (modelo que permite apenas texto), o que leva à introdução de uma ambiguidade entre estrutura e conteúdo que não existe a nível de BDs relacionais. Numa abordagem baseada em atributos, consegue-se uma separação entre estrutura e conteúdo. Assim, o conteúdo está definido ao nível dos atributos, estando a definição de estrutura reservada aos elementos.

A nível de validação de conteúdo usando DTDs, não existe forma de definir tipos de dados, no entanto a nível dos atributos existe a possibilidade de definição de listas de possíveis valores, o que não acontece a nível do conteúdo dos elementos. Já a nível dos

XML *Schema* é possível definir tipos de dados, pelo que neste trabalho, onde se pretende criar um XML *Schema*, os tipos são uma conversão directa dos da BD relacional para os existentes nos XML *Schemas*.

A manipulação de documentos XML pode ser feita por tecnologias como DOM e SAX. Os passos envolvidos para a recuperação de um determinado valor de um documento com uma abordagem baseada em elementos são diferentes para um com uma abordagem baseada em atributos. Tanto na utilização de DOM como na de SAX, constata-se que é mais fácil aceder aos valores em documentos com uma abordagem baseada em atributos, reduzindo assim a complexidade no desenvolvimento de aplicações.

A abordagem baseada em atributos tem uma outra vantagem importante. Com esta abordagem o tamanho do documento resultante poderá ser mais pequeno uma vez que não é necessário armazenar *tags* de início e fecho de elemento para cada valor armazenado no ficheiro. Documentos mais pequenos facilitam a sua transmissão e o seu armazenamento persistente como por exemplo em BDs nativas XML.

Por tudo isto, em [Williams et al. 2000], o autor recomenda a utilização de uma abordagem baseada em atributos, a qual foi seguida no presente trabalho.

6.1.2 Estrutura

Como foi referido anteriormente, numa BD relacional a estrutura é expressa nas tabelas e relações entre estas. Ao converter a informação para XML é conveniente preservar esta estrutura. Os registos de uma dada tabela devem corresponder a elementos XML e tal como referido anteriormente, poderá ser usada uma abordagem baseada em atributos para representar os diversos valores das colunas da tabela.

Outro aspecto importante na estruturação de documentos XML com base numa BD relacional é as relações entre as tabelas. A melhor forma de representar este aspecto nos ficheiros de XML é através do aninhamento de elementos [Williams et al. 2000]. Por exemplo, a informação da relação da Figura 19 (muitos-para-muitos entre `Entidade1` e `Entidade2` com recurso a `Entidade3`) poderia ser representada da seguinte forma (Exemplo 10):

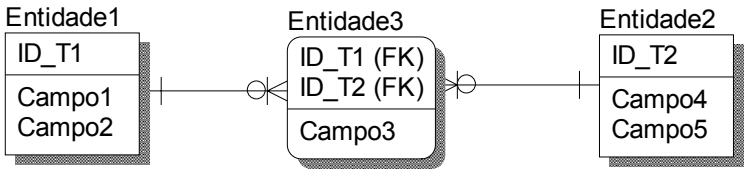

Figura 19: Relação muitos-para-muitos

```
<Entidade1 Campo1="valor1" Campo2="valor2">
      <Entidade3 Campo3="valor3">
            <Entidade2 Campo4="valor4" Campo5="valor5">
            <Entidade2 Campo4="valor41" Campo5="valor51">
      </Entidade3>
</Entidade1>

<Entidade1 Campo1="valor11" Campo2="valor21">
      <Entidade3 Campo3="valor31">
            <Entidade2 Campo4="valor4" Campo5="valor5">
      </Entidade3>
</Entidade1>

(...)
```

Exemplo 10: Relação em XML

No caso particular desta relação, poderia ser utilizada outra abordagem que consiste na utilização de apontadores (IDREFs). A vantagem desta abordagem é que permite a criação de documentos mais pequenos uma vez que evita a repetição de informação, como acontece no exemplo anterior (`Entidade2` está repetida). No entanto, a utilização de apontadores deve ser feita com cautela uma vez que torna o processamento dos documentos mais lento e mais complexo.

6.2 Ferramentas utilizadas

Para a execução dos testes realizados neste trabalho foi necessário implementar algumas ferramentas para satisfazer determinadas necessidades, quer ao nível de preparação da informação bem como ao nível da execução de testes.

Como foi referido anteriormente, a base da informação utilizada neste trabalho é uma BD relacional. Dado que esta BD continha um pouco menos de 29000 registos na sua

tabela principal, a primeira necessidade que surgiu foi a criação de uma quantidade de informação que fosse considerada como suficiente para a realização de testes.

Para esta tarefa foi desenvolvida uma ferramenta (Figura 20) capaz de duplicar as vezes que fossem necessárias, uma determinada informação. Esta ferramenta aceita como parâmetros uma listagem com as chaves primárias dos registos a duplicar (na prática foram fornecidas todas as chaves da BD inicial), bem como listas de nomes próprios e apelidos. Por cada registo (que podemos denominar de modelo) cuja chave primária esteja na lista fornecida, a ferramenta cria um novo registo na tabela principal com uma nova chave primária e de seguida cria (usando essa nova chave) o mesmo número de registos nas tabelas auxiliares do que aqueles que existem relativos ao registo modelo. A nova chave utilizada é calculada somando 1 ao valor convertido para inteiro da maior chave existente, uma vez que as chaves são *strings* contendo apenas dígitos. Os campos dos novos registos criados que representam nomes de pessoas são criados com base em combinações aleatórias de nomes constantes nas listas de nomes próprios e apelidos fornecidas à ferramenta.

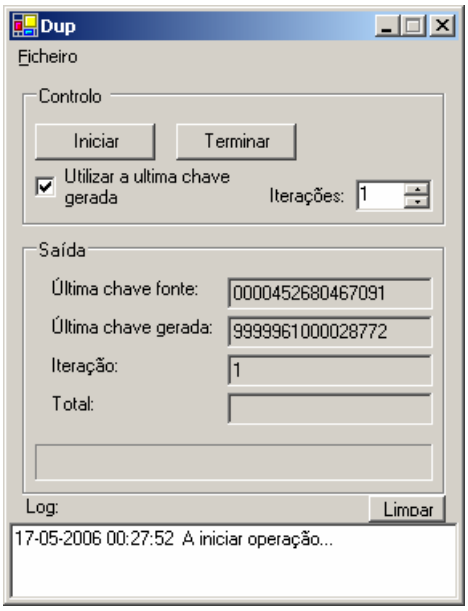

Figura 20: Ferramenta de duplicação de informação

Após a geração de informação relacional considerada suficiente para realizar testes, surge a necessidade de converter esta mesma informação para o formato XML para inserção nas BDs respectivas. Isto leva à criação de uma ferramenta específica para esta função (Figura 21).

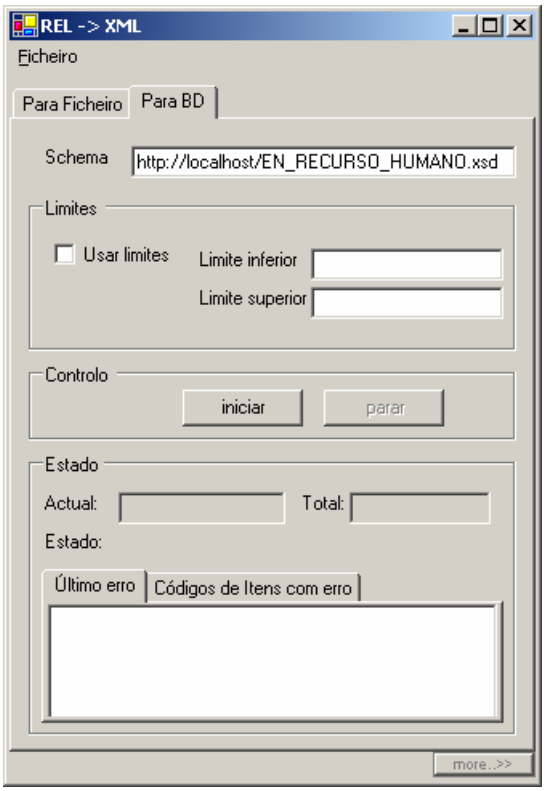

Figura 21: Ferramenta de geração de XML

Esta conversão é feita respeitando um XML *Schema* entretanto criado (Anexo III) e o seu resultado pode ter 2 destinos: a criação de ficheiros ou inserção directamente numa BD. Estas duas opções foram criadas com vista a facilitar a interacção com as BDs seleccionadas para o trabalho. Desta forma, chegou-se à conclusão que a solução mais conveniente para a inserção da informação em XML na BD com suporte para XML, era a inserção de documentos gerados directamente na BD. Por outro lado, a solução mais conveniente para a inserção dos documentos XML gerados na BD nativa XML era criando ficheiros contendo os documentos gerados e que seriam posteriormente inseridos na BD.

Uma vez que estes ficheiros gerados não se apresentam no formato reconhecido pela BD nativa XML, uma nova ferramenta foi criada para fazer esta tarefa específica (Figura 22) de conversão de formato.

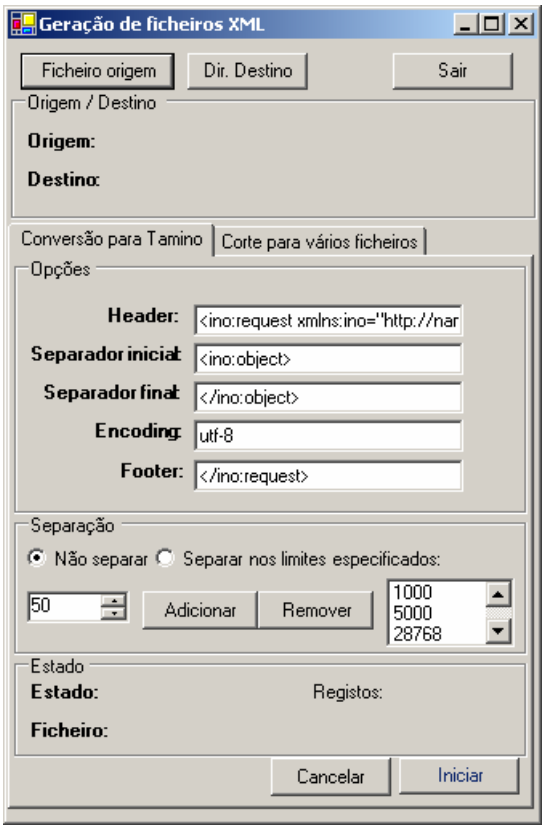

Figura 22: Ferramenta de criação de ficheiros XML

A nível da execução dos testes, existiu a necessidade de simular vários clientes simultâneos. Para tal foi criada uma ferramenta (Figura 23) onde é possível definir para cada teste, o número de clientes, bem como o tempo de execução, o tipo de BD que está activa na aplicação *web* e o tempo entre a recepção de uma resposta a um pedido e a execução do pedido seguinte.

Cada cliente executa pedidos à aplicação *web* cujos parâmetros (nomes ou chaves) são obtidos de listas fornecidas à ferramenta. Assim, a execução de um pedido consiste na chamada a uma página referente à pesquisa a ser feita (existe uma por cada pesquisa implementada) com os respectivos parâmetros correctamente passados à aplicação *web*.

A aplicação *web* fornece também uma página (Figura 24) que permite visualizar os valores contabilizados de pesquisas que são feitas.

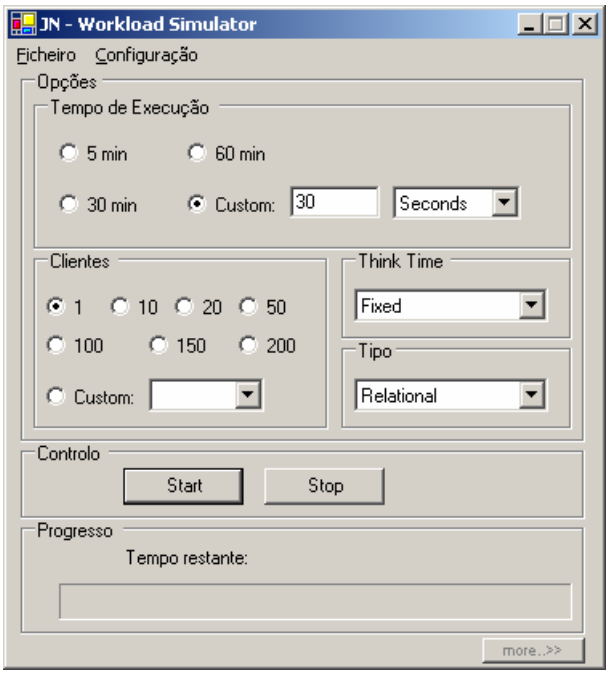

Figura 23: Ferramenta de simulação de carga com clientes simultâneos

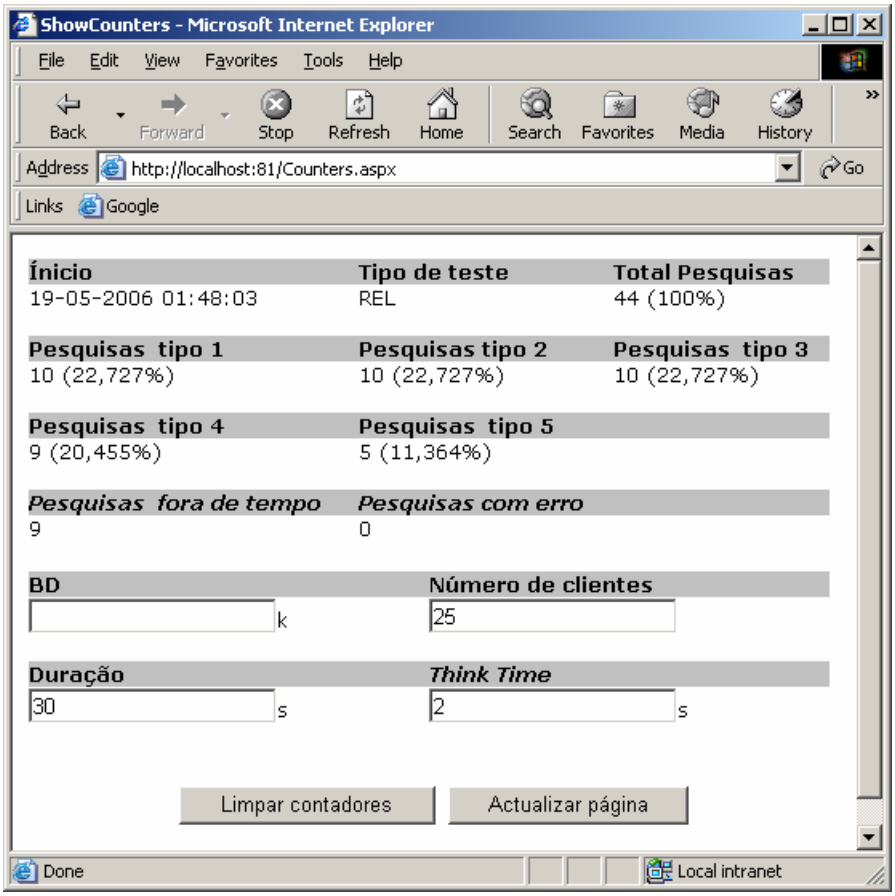

Figura 24: Aplicação *web* - contadores

6.3 Bases de dados utilizadas

Uma das tarefas iniciais na execução deste trabalho, foi a selecção de BDs que implementassem os modelos que se pretendia estudar. Nesta selecção foram considerados relevantes os seguintes critérios:

- Boa documentação. É fundamental a existência de documentos (artigos, livros, entre outros) que exponham claramente as características da BD bem como forneçam exemplos de utilização, em especial para utilizadores com pouca ou nenhuma experiência;
- Suporte eficiente. No decorrer da utilização da BD é extremamente importante existirem fontes de informação adicionais, além da documentação, a que seja possível recorrer em caso do surgimento de problemas. Estas fontes podem ser grupos de utilizadores, fóruns de discussão ou mesmo, no caso de BDs comerciais, um sistema eficiente de resolução de problemas e apoio ao utilizador eficiente;
- Provas dadas na área. É importante que a BD seja reconhecida como sendo uma referência na área. Este aspecto leva consequentemente a uma maior utilização e à criação de uma comunidade alargada de utilizadores, o que poderá favorecer os dois pontos anteriores;
- Possibilidade de funcionar na plataforma previamente escolhida. Na execução deste trabalho, existia uma maior familiaridade com ambientes Microsoft, quer a nível de sistemas operativos e linguagens de programação. Desta forma, um ponto importante na selecção de BDs para este trabalho em particular foi a possibilidade de estas funcionarem nestes ambientes;
- Facilidade na obtenção de licenças e respectivo *software*. No caso das BDs seleccionadas não serem de distribuição livre, este aspecto é bastante relevante, pois poderá inviabilizar qualquer decisão que se tenha tomado com base nos aspectos anteriores.

Um nome de referência no que diz respeito a BDs é Oracle, tendo lançado a primeira BD relacional comercial em 1979, contando assim com quase 3 décadas de experiência nesta área. Ao longo dos anos foram sendo introduzidas inovações e em 1999 foi

lançada a primeira BD com suporte para XML (versão 8*i*). Esta é uma BD que facilmente respeita todos os critérios descritos anteriormente. Dispõe de uma vasta documentação, existindo por exemplo dezenas de livros escritos sobre o tema, e é usada por uma vasta comunidade de utilizadores permitindo assim um intercâmbio de experiências. Existem inúmeras versões desta BD para diversas plataformas como Microsoft Windows, Unix e Linux, e é possível a sua utilização com várias linguagens de programação como Java, PHP, .NET entre outras. O *software* é de fácil obtenção de utilização livre para fins educacionais.

A informação fornecida para a execução deste trabalho encontrava-se armazenada numa BD relacional Oracle. Desta forma, por uma questão de simplicidade e atendendo ao facto que esta BD satisfaz plenamente os critérios de selecção, optou-se por manter este produto para o modelo relacional. No entanto, dado que a informação fornecida estava armazenada numa versão já relativamente antiga, optou-se por utilizar a última versão disponível na altura da realização deste trabalho (10g r2).

No que diz respeito ao modelo XML, a escolha não foi tão directa como aconteceu para o modelo relacional. Em [Bourret 2006] são classificados e descritos vários produtos que manipulam XML. Esta lista de produtos permitiu obter uma visão bastante completa dos produtos existentes actualmente no mercado nesta área.

A nível de XML pretende-se seleccionar uma BD com suporte para XML e uma nativa. Em [Bourret 2006] o Oracle, devido às suas capacidades de manipulação de XML, é classificado como uma BD com suporte XML. Desta forma, a selecção de uma BD deste tipo recai novamente no Oracle, já que satisfaz os critérios de selecção e faz com que não seja necessário mudar de BD para utilizar XML, poupando assim tempo na adaptação a outro produto.

No que diz respeito à selecção de uma BD nativa XML, a escolha não foi tão imediata como no caso da BD com suporte para XML. Da lista disponibilizada em [Bourret 2006] verifica-se que existem várias opções mas que não satisfazem os critérios de selecção. Algumas não têm suporte para a plataforma pretendida, outras têm uma documentação bastante pobre e não existe um grande número de utilizadores, pelo que se surgissem problemas de utilização, estes poderiam ser de difícil resolução.

Após a consideração de várias hipóteses, chegou-se à conclusão que a melhor solução, e a que respeita os critérios de selecção é BD comercial Tamino [Schöning e Wäsch

2000], da SoftwareAG. Esta BD disponibiliza uma documentação bastante completa, com diversos exemplos que permitem a utilizadores com pouca ou nenhuma experiência começar a utilizar a BD muito rapidamente. Associada a esta BD, existe uma comunidade activa de utilizadores que permite a troca de opiniões com vista à resolução dos problemas que vão surgindo, existindo também um suporte a clientes registados bastante eficaz[27]. Esta BD está disponível para uma grande variedade de plataformas e pode ser utilizada com as linguagens de programação mais comuns actualmente. O *software* e respectiva licença de utilização para fins educacionais estão acessíveis através de protocolos estabelecidos entre a Empresa e a Universidade.

[27] Que chegou a ser utilizado no decorrer do trabalho.

Capítulo 7

7 Descrição e análise de resultados

Após a definição e implementação de um sistema capaz de executar os testes desejados (capítulos 5 e 6) e a execução dos mesmos, é necessário interpretar e tirar conclusões sobre os resultados obtidos. Neste capítulo é feita uma descrição dos resultados obtidos, sendo estes apresentados de uma forma gráfica com o objectivo de facilitar a sua interpretação. De seguida é feita uma análise onde são descritas as situações onde cada modelo envolvido nos testes é o mais apropriado.

7.1 Descrição

Com um sistema de testes implementado, estes foram executados e os seus resultados registados. Estes valores foram traduzidos em gráficos uma vez que estes permitem uma mais fácil visualização dos resultados obtidos.

Para a execução dos testes existiam vários factores que poderiam influenciar os seus resultados. Os factores considerados mais importantes e que interessou testar neste trabalho foram os descritos na Tabela 7. Cada um destes factores pode assumir

determinados valores, que são denominados de *níveis*. Todos os outros potenciais factores (tais como os relacionados com o servidor: sistema operativo utilizado, quantidade de memória disponível ou outros) foram mantidos constantes.

Factor	Níveis
Tamanho da BD	3 (pequena, média, grande)
Tipo de BD	3 (relacional, nativa XML, com suporte para XML)
Número de clientes	5 (25, 50, 75, 100 e 125 clientes)

Tabela 7: Factores da experiência

O método seguido neste trabalho, que consiste em testar todas as combinações dos factores considerados, é semelhante ao seguido por trabalhos como [García e García 2003; Khouja et al. 2004] e é denominado de factorial completo[28]. Neste caso é factorial completo 3x3x5, uma vez que existem 3 factores, dois deles com 3 níveis e o restante com 5 níveis. Este método permite estudar a influência dos vários factores no resultado final e as várias replicações das experiências permitem estudar a influência do erro experimental [Antony 2003].

Na execução dos testes optou-se por fazer 7 medições de cada experiência, sendo uma experiência considerada como uma determinada combinação de diferentes níveis de cada factor. Desta forma são usados 315 valores (3x3x5x7), que podem ser analisados no Anexo IV. Para os valores de cada gráfico deste capítulo são utilizadas médias dos 7 valores obtidos para cada combinação dos factores considerados.

Inserindo os valores obtidos (Anexo IV) num *software*[29] especialmente desenhado para o efeito, é possível rapidamente fazer uma análise dos resultados. O *software* produz uma tabela de análise de variância – ANOVA – que pode ser calculada de acordo com o descrito em [Mason et al. 2003], capitulo 6, e que é reproduzida na Tabela 8.

[28] *Full factorial* em língua inglesa.

[29] O *software* utilizado neste trabalho foi o *Design-Expert® (DX), Version 7*. Versão de teste disponível em http://www.statease.com.

Fonte	Soma de quadrados (SS)	Graus liberdade	Média dos quadrados	Valor F	p-value Prob > F
Model	44867,70	44,00	1019,72	1970,41	< 0.0001
A-Tamanho BD	15085,10	2,00	7542,55	14574,47	< 0.0001
B-Tipo BD	14387,31	2,00	7193,66	13900,30	< 0.0001
C-Número de clientes	2387,09	4,00	596,77	1153,14	< 0.0001
AB	7062,97	4,00	1765,74	3411,94	< 0.0001
AC	1135,57	8,00	141,95	274,28	< 0.0001
BC	3249,74	8,00	406,22	784,93	< 0.0001
ABC	1559,91	16,00	97,49	188,39	< 0.0001
Error	139,73	270,00	0,52		
Total	45007,43	314,00			

Tabela 8: Análise de variância

Da análise desta tabela conclui-se que os termos do modelo são significativos. Isto é, A (tamanho da BD), B (tipo de BD), C (número de clientes), AB, AC, BC e ABC (estes últimos representam a intereacção entre os factores) são termos estatisticamente significativos. Isto acontece uma vez que a probabilidade associada ao valor F calculado para cada um (última coluna da Tabela 8) é inferior a 0.05, pelo que os termos do modelo têm um efeito significativo na resposta (valores maiores que 0.1 indicam que os termos do modelo não são significativos). Valores F maiores levam à rejeição da hipótese nula de não existir influência de factores no resultado [Mason et al. 2003].

Nesta tabela é possível também verificar os termos que mais influenciam os resultados. Como é observável na Tabela 8 o cálculo da ANOVA determina que a variação total da resposta é devida na maior parte ao tamanho da BD (factor A), com 33,52% (isto é, SS_A/SS_T, que representa o contributo deste factor na soma de quadrados total) e ao tipo de BD (factor B) com 31,97% (SS_B/SS_T). Por outro lado, a influência do número de clientes (factor C) é pequena, com 5,3% (SS_C/SS_T), ao passo que a interacção entre o Tamanho da BD com o tipo da BD tem uma influência considerável de 15,69% (SS_{AB}/SS_T).

Desta forma, se uma grande parte da variação é explicada pelo efeito de determinado factor, então este à partida terá um efeito considerável na variável de resposta. Isto

acontece na presente situação, onde 2 dos factores por si só, tamanho da BD e tipo de BD, explicam quase 2/3 da variação.

Esta informação também pode ser visualizada graficamente. Assim, nos gráficos 1, 2 e 3 é possível verificar que o aumento do número de clientes não produz aumentos significativos na resposta, existindo mesmo situações onde mais clientes não é sinónimo de um valor superior na resposta. Verifica-se também que a alteração do tamanho da BD produz valores na resposta com diferenças significativas (observável na sequência dos gráficos 1, 2 e 3), o mesmo acontecendo com o tipo de BD (observável em cada gráfico). Outro aspecto, observável na sequência dos gráficos 1, 2 e 3, é o facto de com o aumento do tamanho da BD os valores da resposta vão diminuindo de uma forma geral. De qualquer das formas, nas BDs do tipo relacional os valores são sempre superiores.

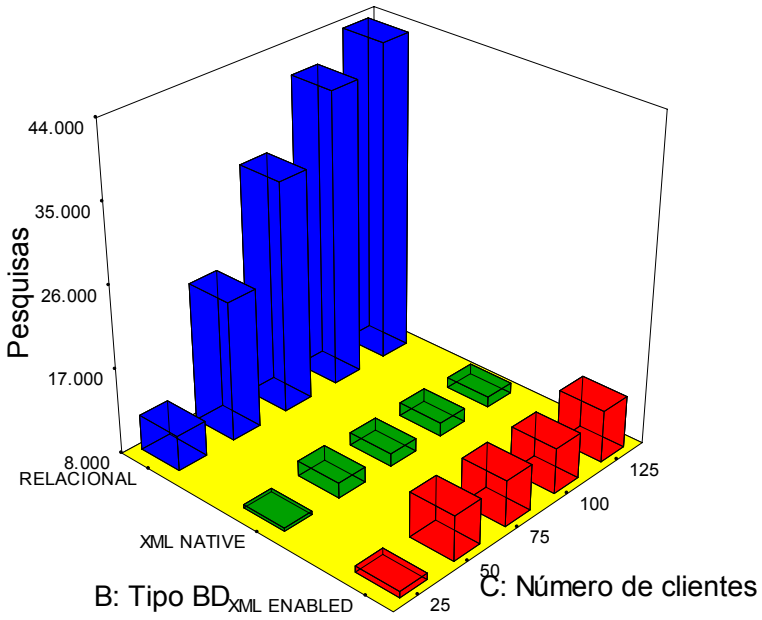

Gráfico 1: Análise de factores no resultado das experiências (com o menor tamanho para as BDs)

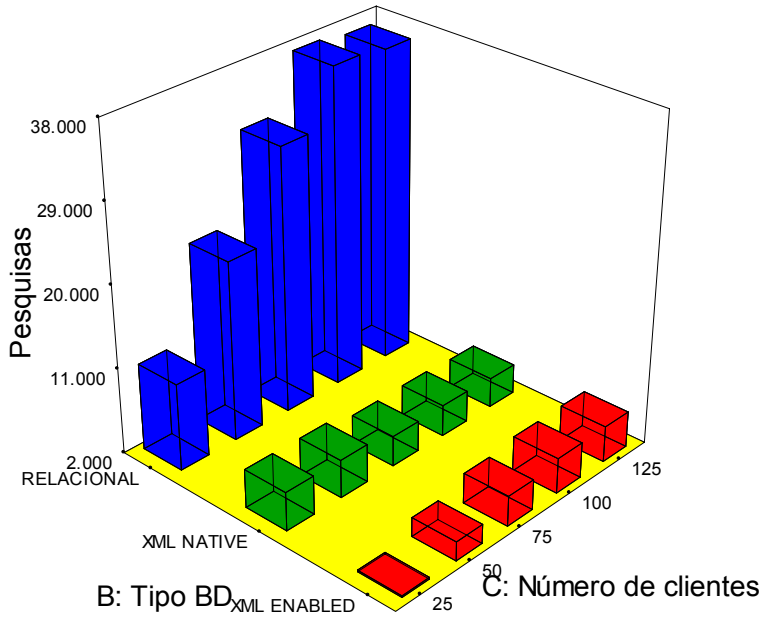

Gráfico 2: Análise de factores no resultado das experiências (com o tamanho médio para as BDs)

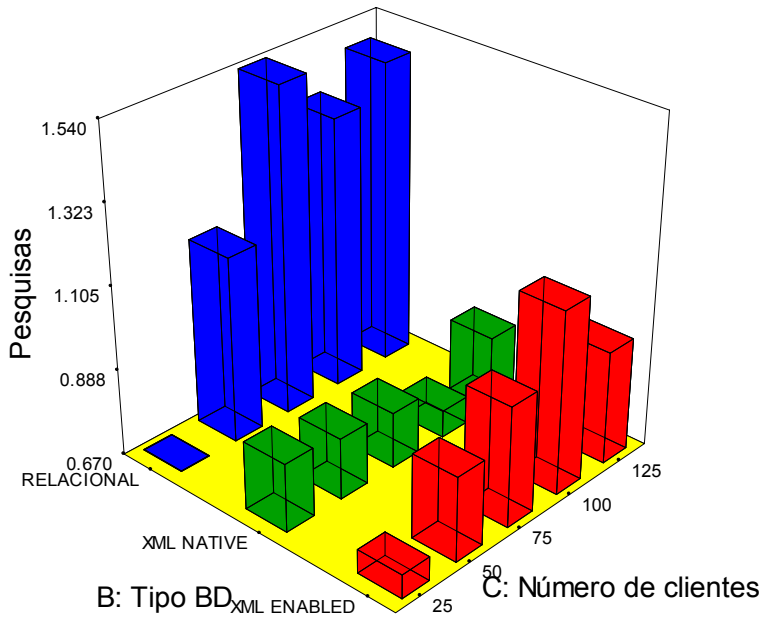

Gráfico 3: Análise de factores no resultado das experiências (com o maior tamanho para as BDs)

Com uma noção dos factores mais importantes na resposta obtida, procede-se a uma análise do desempenho com recurso a gráficos. Desta forma, facilmente se observa nos

gráficos 4, 5 e 6, que a BD relacional apresenta um comportamento quase sempre superior na carga a que foi submetida. Isto apenas não acontece na BD de maiores dimensões com o menor número de clientes por uma margem pequena relativamente às outras BDs, pelo que se poderá considerar pouco significativo.

No caso das BDs pequena e média, a diferença de desempenho entre a BD relacional e as de XML é significativa chegando a ser mais de 6 vezes superior no caso da BD média. Isto é mais relevante com o aumento de clientes. Nota-se claramente dois grupos de desempenho: alto para a relacional e baixo para as BD de XML. Entre as BDs de XML, nota-se um desempenho que se pode considerar idêntico no caso das BDs médias. No caso da BD pequena a com suporte para XML leva vantagem chegando a ter um desempenho cerca de 50% superior à nativa XML.

Na BD grande esta diferença de desempenho relativa entre os vários tipos de BDs não é tão acentuada, estando todos os valores relativamente próximos, mas novamente com a relacional em vantagem. Nota-se no entanto uma queda de desempenho bastante acentuada relativamente às BDs média e pequena.

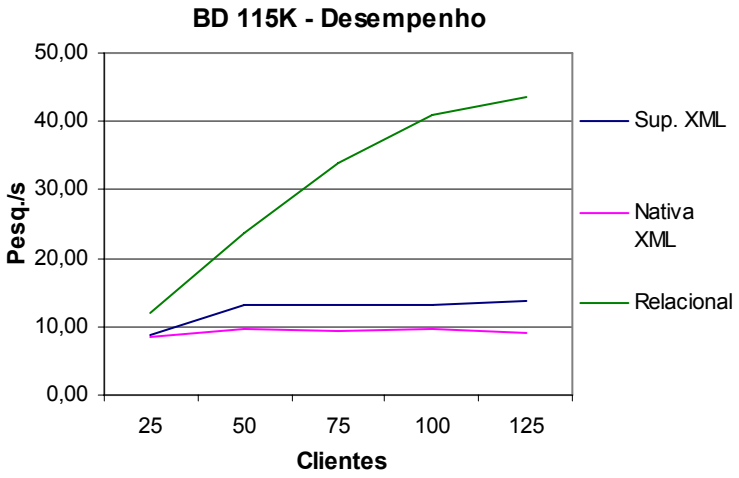

Gráfico 4: Desempenho das bases de dados com 115.000 elementos

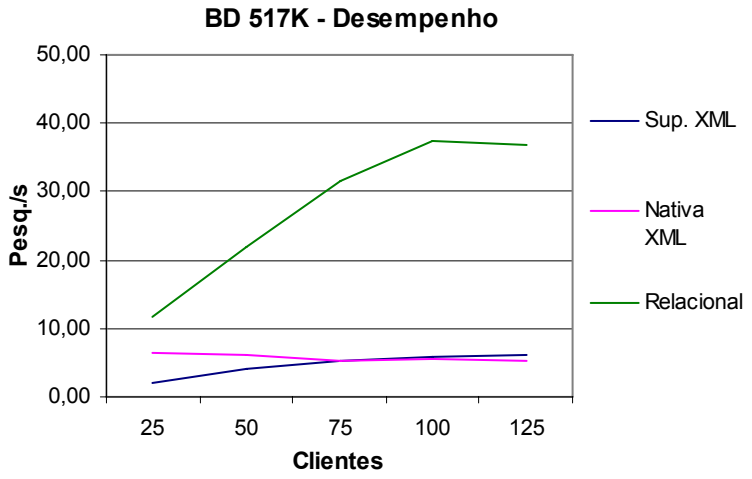

Gráfico 5: Desempenho das bases de dados com 517.000 registos

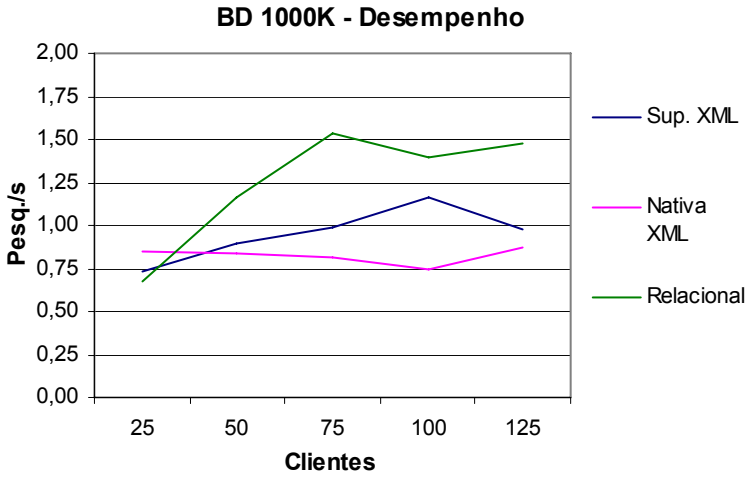

Gráfico 6: Desempenho das bases de dados com um milhão de elementos

Estes valores podem ser vistos de uma outra perspectiva: tempo de resposta. Em praticamente todas as situações as BDs relacionais apresentam valores mais baixos, como se pode ver nos gráficos 7, 8 e 9. Desta forma a BD relacional favorece a escalabilidade das soluções uma vez que um aumento de clientes implica aumentos do tempo de resposta menos acentuados que nas outras BDs.

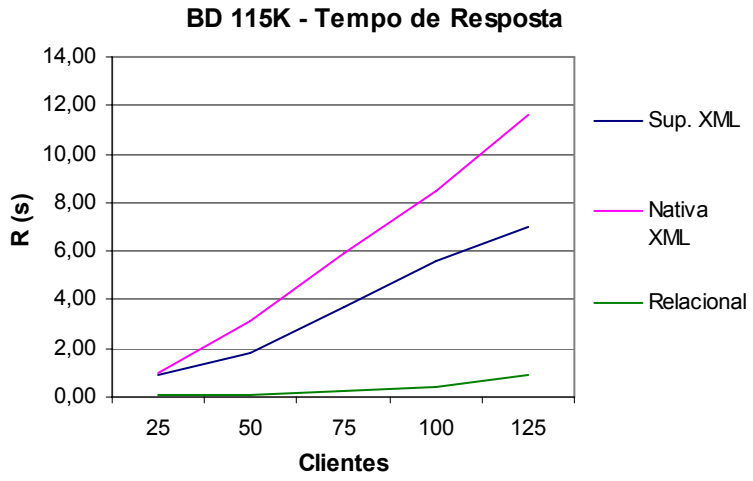

Gráfico 7: Tempo de resposta para as bases de dados de 115.000 elementos.

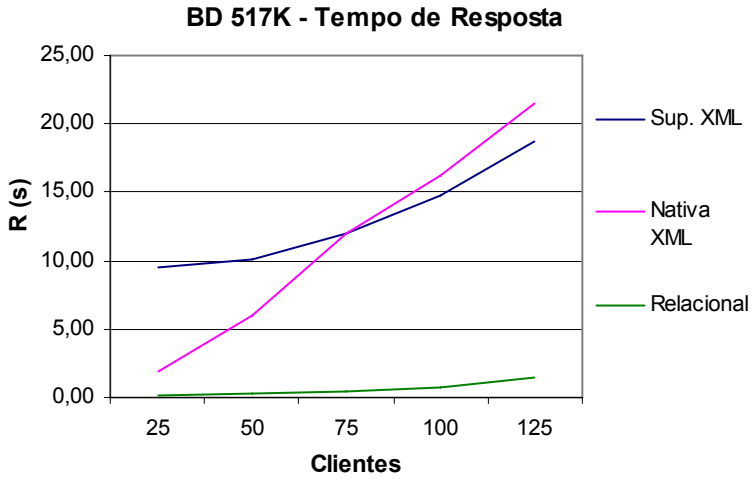

Gráfico 8: Tempo de resposta para as bases de dados de 517.000 elementos

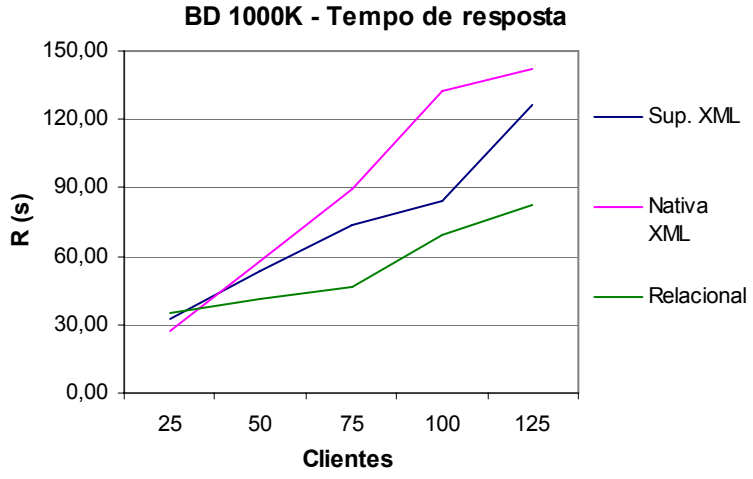

Gráfico 9: Tempo de resposta para as bases de dados de um milhão de elementos

A indexação nas BDs pode ter um grande impacto no desempenho. Das pesquisas que

são utilizadas na carga submetida às BDs no decorrer dos testes, 3 são pesquisas textuais (por nomes) nas BDs. Estas operações podem ser bastante penalizadoras para o desempenho da BD, pelo que a indexação assume um papel importante.

Nos gráficos 10, 11 e 12 são apresentados os resultados dos testes sem a indexação para as BDs relacionais e com suporte para XML (são deixados, para servir de termo de comparação, os resultados das BDs Nativas XML que estão devidamente indexadas). Comparativamente aos valores apresentados nos gráficos 4, 5, e 6, constata-se um aumento significativo no desempenho das BDs relacionais, em especial nas BDs pequena e média, onde são registados ganhos que chegam a ser mais de 7 vezes superiores.

Apesar da BD com suporte para XML aproveitar os mesmos mecanismos de indexação da relacional em que se baseia, os ganhos de desempenho não foram tão significativos como na relacional. Esta diferença poderia ser eventualmente atenuada com o armazenamento do XML seguindo outras abordagens aproveitando ainda mais as capacidades da BD.

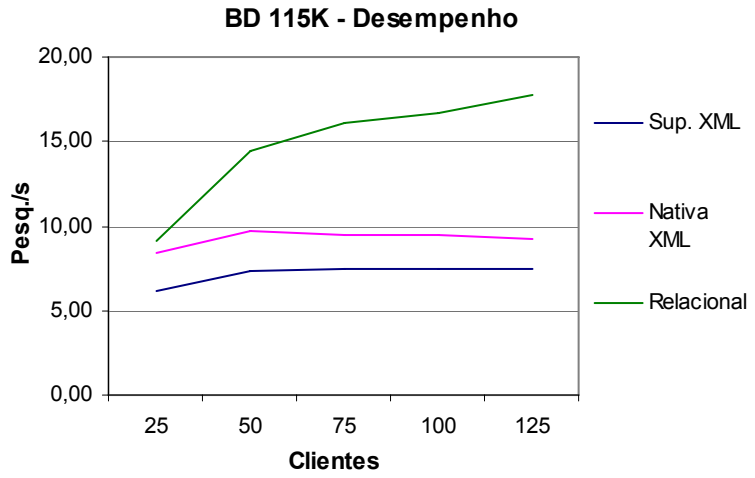

Gráfico 10: Desempenho das bases de dados com 115.000 registos (sem indexação)

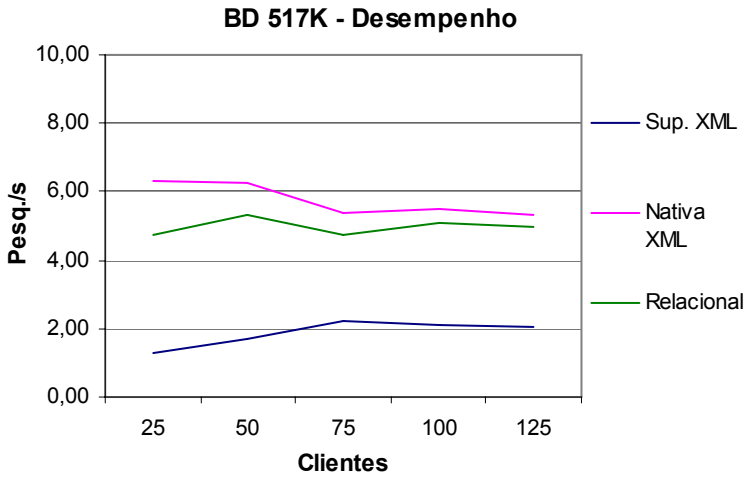

Gráfico 11: Desempenho das bases de dados com 517.000 registos (sem indexação)

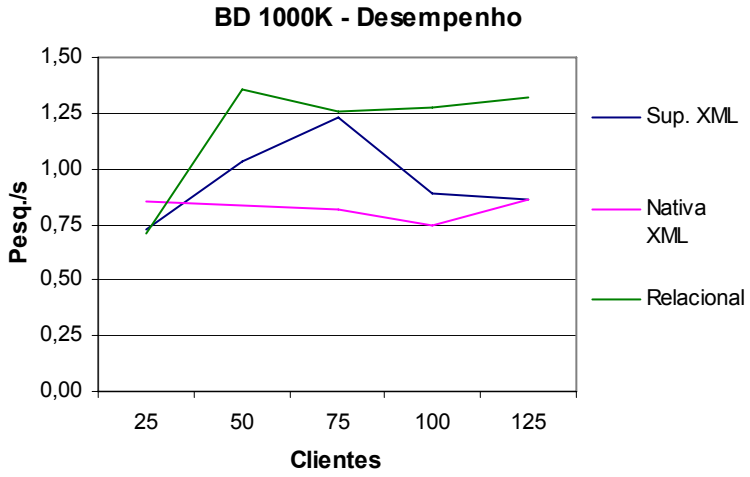

Gráfico 12: Desempenho das bases de dados com um milhão de registos (sem indexação)

No caso da BD grande, ao contrário do que aconteceu nas outras BDs e do que seria de esperar, o ganho não foi significativo, existindo até situações contrárias (perda de desempenho). O Gráfico 13 mostra em pormenor o sucedido nesta BD. A razão deste poderá estar relacionada com a quantidade de memória do servidor. Eventualmente, com mais memória o desempenho de todas as BDs seria superior, em particular da relacional. No entanto, não foi possível executar testes que permitissem comprovar esta hipótese.

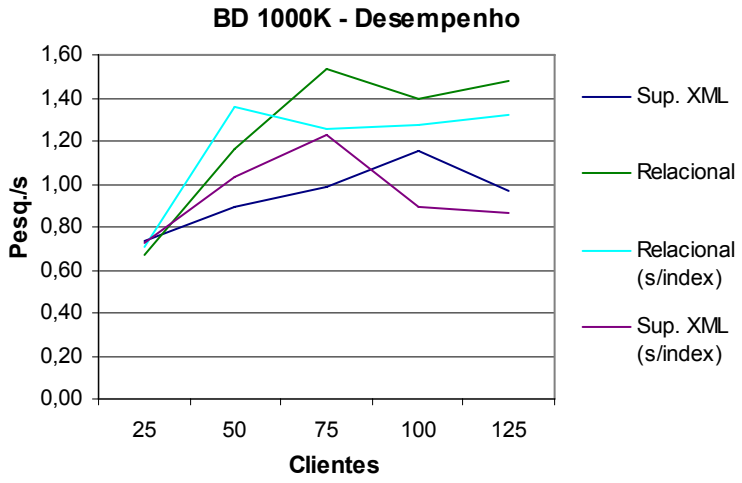

Gráfico 13: Desempenho das bases de dados com um milhão de registos (pormenor)

7.2 Análise

Na fase inicial do desenho de uma aplicação deve ser tomada a decisão de qual o modelo para a representação da informação que deverá ser adoptado. Esta poderá não ser uma tarefa simples.

Para aplicações onde se tem à partida informação que já utiliza um determinado modelo, a questão que se coloca é se se deve manter o formato inicial ou eventualmente mudar para um mais conveniente para os requisitos da aplicação, ou ainda usar uma solução híbrida entre os dois modelos. Neste caso, está-se a falar concretamente num cenário que começa a ser mais frequente hoje em dia que é a introdução de XML em situações onde era utilizado anteriormente o modelo relacional (em substituição ou conjuntamente), que é sem dúvida o modelo mais divulgado actualmente para armazenamento de informação. Um exemplo de uma solução híbrida poderá ser uma BD relacional usada como fonte de informação para a geração de documentos XML que capturam a realidade da BD num determinado momento. No caso de uma aplicação que esteja a ser construída de raiz, a questão é idêntica, isto é, qual o modelo a utilizar e eventualmente se vale a pena utilizar uma solução híbrida entre modelos.

Os modelos relacional e XML são sem dúvida os mais divulgados actualmente e apresentam grandes diferenças entre si, como foi referido anteriormente, o que faz com que a selecção de um modelo para o armazenamento de informação seja feita considerando algumas questões.

Em [Champion 2001] refere-se que BDs relacionais são mais apropriadas para manter a integridade dos dados, ao passo que BDs XML são mais apropriadas para manter documentos. Os dados são proposições sobre o mundo para as quais o modelo relacional fornece uma metodologia para manter a sua consistência lógica [Champion 2001]. Numa BD relacional devidamente normalizada após qualquer alteração numa tabela, irá ser mantido um estado consistente de toda a BD. Com a informação normalizada garante-se a não existência de redundâncias (cada dado deve ser armazenado uma única vez e numa única localização) e a consistência da informação. Qualquer operação de manipulação da informação (Inserção, Alteração, Destruição) deve afectar uma só ocorrência de um dado. Apesar do XML ter mecanismos para minimizar o armazenamento de informação redundante (como o XLink [W3C 2001b]), não existe uma metodologia bem definida para esta tarefa ao contrário do que acontece com as BDs relacionais. Desta forma, é preferível manter informação *mission-critical* em BDs relacionais uma vez que o modelo relacional garante a consistência lógica entre as várias proposições da realidade de negócio existente na BD [Champion 2001]. Actualmente, as BDs XML (principalmente as nativas), devido à sua imaturidade, poderão fornecer em certos casos um suporte incompleto ou no caso de ser completo, ainda pouco robusto, para as propriedades ACID (atomicidade, consistência, isolamento e durabilidade) [Snelson 2005]. Estas propriedades são fundamentais para o armazenamento e manipulação de informação *mission-critical*.

Em oposição a dados, os documentos podem ser produzidos para e por humanos pelo que terão tendencialmente estruturas de difícil tradução para o modelo relacional [Champion 2001]. Exemplo disto é o conteúdo misto nos documentos XML. O benefício de uma solução nativa é o facto de não ser necessária qualquer tradução para outro modelo. A informação é inserida em formato XML e é recuperada em formato XML.

Em [Lapis 2005] são resumidas as considerações que levam a optar por um armazenamento da informação no modelo relacional ou recorrendo a XML. De uma forma geral, pode-se optar pelo modelo relacional quando acontece uma ou mais das seguintes situações [Lapis 2005]:

- A informação enquadra-se naturalmente num formato tabelar;
- A informação destina-se a ser manipulada por aplicações que utilizam o modelo relacional ou quando será processada conjuntamente com outra informação

relacional;

- É necessária a melhor desempenho possível no processamento da informação, uma vez que a informação XML requer mais tempo de CPU para a sua manipulação;
- Na informação não existem relacionamentos do tipo pai / filho, tal como existe no XML.

Da mesma forma, poderá optar-se por armazenar a informação em XML e eventualmente numa combinação entre relacional e XML quanto acontece uma das seguintes situações [Lapis 2005]:

- A informação enquadra-se naturalmente num formato hierárquico. Este tipo de informação pode ser difícil de traduzir no modelo relacional. Neste caso, é possível mesmo assim, em BDs que o permitam, combinar o uso de XML com informação relacional utilizando SQL e Xquery nos mesmo *queries*;
- O *schema* da informação sofre alterações. Isto pode acontecer devido a vários factores como a alteração de processos de negócio. O *schemas* XML são mais flexíveis, pelo que poderão surgir situações em que para evitar alterações aos *schemas* relacionais poderá ser mais vantajoso armazenar XML no seu formato nativo conjuntamente com a informação relacional. Desta forma, poderá ser mais vantajoso optar por XML se a evolução do *schema* se revelar mais fácil em XML, já que no modelo relacional isto poderá implicar algo mais complexo que a simples criação de colunas em tabelas, mas sim a normalização de tabelas em várias outras;
- Se a informação tem uma quantidade significativa de atributos dispersos, resultando em atributos que são vazios ou nulos;
- Quando uma determinada informação apenas faz sentido num determinado contexto, existindo o risco da normalização da informação no modelo relacional ser feita de uma forma que a sua recuperação implica o *join* de um elevado número de tabelas;
- Quando a tradução para o modelo relacional conduz a *schemas* relacionais complexos que requerem um acréscimo no esforço de gestão da BD.

Existe bastante informação XML que não pode ser facilmente convertida para o modelo relacional. Além disto, o XML fornece bastante flexibilidade para informação não estruturada, para a partilha de informação e para *schemas* que vão mudando com o

tempo. Desta forma, em situações que se enquadram neste cenário, não existirá grande dúvida na escolha do modelo a utilizar para o armazenamento de informação, o qual será à partida XML. No entanto, há casos onde o modelo relacional é a escolha mais acertada, dependendo dos requisitos das aplicações. Existem também casos onde é necessária uma abordagem mista, utilizando os dois modelos.

A principal diferença entre estes dois modelos, é o facto de no modelo relacional a informação ser fortemente estruturada e tipada, ao passo que em XML isto não acontece (o XML é por vezes chamado de informação semi-estruturada). Numa tabela relacional cada linha tem exactamente o mesmo número de colunas e cada coluna tem um tipo de dados bem definido. Isto poderá ser bastante limitativo, mas possibilita um processamento eficiente da informação. No entanto, isto poderá ser bastante restritivo para certas situações, pelo que a escolha do XML será a escolha mais indicada, uma vez que é bastante mais flexível. Por exemplo o *schema* XML poderá permitir a ocorrência opcional de elementos ou diversas vezes, poderá definir a estrutura e tipos de dados para algumas partes do documento deixando outras partes indefinidas, nomeadamente os elementos poderão não ter o tipo de dados definido. Estas características apresentadas pelo XML poderão ser de difícil, ou mesmo impossível tradução para o modelo relacional. Desta forma, na construção de uma aplicação, se a informação está em bruto e se enquadra no modelo XML, ou mesmo já em XML, e necessita de características que o XML está mais apto a satisfazer, a escolha do modelo deverá recair no XML.

Em oposição a este cenário, na construção de uma aplicação, se a informação está em bruto e se enquadra no modelo relacional, ou está mesmo já em formato relacional, e onde se prevê que não venha a existir a necessidade de alterações significativas de *schema* ao longo do tempo e onde poderão ser feitos *queries* que aproveitam os mecanismos de optimização oferecidos, a sua tradução e manipulação em XML poderá não ser vantajoso. Isto é demonstrado pelos resultados obtidos nos testes efectuados.

Nos testes, a informação inicialmente relacional foi convertida para XML. Nesta informação foram executados *queries* simples que aproveitaram os mecanismos de indexação das BDs envolvidas. No conjunto de *queries* executados, grande parte deles são de acesso directo a itens que é algo particularmente eficaz no modelo relacional existindo também *queries* de pesquisas textuais e ordenação de resultados que são sempre penalizados a nível de desempenho, em particular no modelo XML.

No contexto dos testes, o modelo relacional revelou-se superior. A escalabilidade de uma solução baseada no modelo relacional é bastante superior comparativamente com uma solução baseada em XML. O impacto do aumento do tamanho das BDs e do aumento do número de clientes é bastante menos negativo em termos de desempenho numa solução baseada no modelo relacional do que umas solução baseada no modelo XML.

Com estes testes, pode-se concluir que o modelo relacional e XML não podem competir em termos de desempenho pura e simples. Se este fosse o único argumento a considerar na escolha de uma BD é evidente que uma BD de XML nunca poderia ser seleccionada e isto não é o que acontece na realidade, como é referido num um estudo citado por [Cox 2006], onde cerca de 29% das 500 empresas inquiridas usam repositórios e BDs XML. O estudo revela também um uso alargado de outras tecnologias relacionadas com o XML. As BDs de XML são cada vez mais usadas em aplicações reais em áreas onde as BDs relacionais não são as mais indicadas, como seja [Bourret 2005a]:

- Gestão de informação orientada a documentos;
- Integração de informação (integração entre aplicações, com informação obtida de diversas fontes);
- Gestão de informação não estruturada;
- Evolução do *schema.*

Além disto, o XML é ainda utilizado como base de tecnologias muito utilizadas hoje em dia, como *Web Services* e SOAP – *Simple Object Access Protocol*.

O XML é um formato de texto, contra o qual normalmente são apontados dois aspectos negativos [Megginson 2005]:

- Os documentos de XML são normalmente bastante maiores que documentos equivalentes noutros formatos, necessitando assim mais memória, espaço de armazenamento e largura de banda para serem transmitidos;
- Os ficheiros XML podem necessitar operações de *parsing* e transformações que podem consumir poder de processamento e tempo consideráveis.

Com efeito, a manipulação de XML utilizando uma árvore DOM ou aplicando transformações XSLT podem ser operações que consomem bastante tempo e memória.

Uma abordagem recente que pretende minorar estes problemas apontados ao XML é o

XML binário, isto é, o tratamento do XML como um formato binário. Esta abordagem ao XML está a ganhar relevância, pelo que justificou a criação de um grupo de trabalho[30] para o seu estudo por parte do W3C (*World Wide Web Consortium*). O XML normal é maior e acima de tudo menos eficiente para processar do que aquilo que a versão binária seria, o que causa um impacto negativo no desempenho de BDs e outros sistemas que manipulam XML [Geer 2005].

O XML pode ser considerado ainda uma tecnologia recente (o primeiro padrão foi aprovado em 1998) comparativamente com o modelo relacional, que já conta com cerca de 30 anos, pelo que apresenta um grande potencial de crescimento. Por esta razão, os fabricantes, que nos últimos anos têm introduzido nos seus produtos suporte para o XML ou criado produtos com o objectivo específico de o armazenar e manipular, são obrigados a introduzir nos seu produtos melhorias sucessivas com o objectivo de responderem mais eficazmente às necessidades crescentes de armazenamento e manipulação de XML.

Atendendo à maturidade do modelo relacional por um lado, e ao facto do XML ser uma tecnologia que apesar de sofrer constantes melhorias e ainda poder evoluir em termos de desempenho, podendo dar resposta a situações para as quais o modelo relacional parece não ser o mais adequado, por outro, uma solução que adopte ambos os modelos poderá ter sem dúvida um grande potencial. Tal solução aproveita o melhor de cada um deles: a rapidez associada ao modelo relacional (como demonstrado nos testes efectuados) e a capacidade de manipulação de documentos e lidar com *schemas* complexos ou que mudam frequentemente. Neste caso, parte da informação estaria armazenada em tabelas relacionais e outra parte em documentos XML, por exemplo a informação não estruturada cujo *schema* é pouco rígido e que poderá mudar mais frequentemente.

Esta solução é facilmente implementada por BDs relacionais actuais que fornecem tipos de dados XML nativos com suporte a várias tecnologias associadas ao XML (conforme o que é descrito no Capítulo 3) nomeadamente a validação de documentos (XML *Schemas* e DTD) e linguagem de *query* (como o XPath e XQuery).

Desta forma, o uso dos modelos relacional ou XML não pode ser visto numa perspectiva competitiva, mas sim complementar.

[30] *XML Binary Characterization Working Group* - http://www.w3.org/XML/Binary/

Capítulo 8

8 Conclusão

Ao terminar um projecto torna-se necessário avaliar o trabalho realizado, realçando os resultados obtidos e com uma postura crítica, reconhecer as limitações sentidas. Desta forma é possível perceber as contribuições trazidas pelo trabalho realizado e apontar caminhos a seguir em trabalhos futuros.

8.1 Síntese e discussão do trabalho realizado

Este trabalho teve início numa revisão de literatura que permitiu clarificar dois conceitos fundamentais subjacentes a este trabalho, serviços de informação e repositório, bem como esclarecer a relação entre ambos.

Os serviços de informação podem ser vistos segundo a definição de [Heijden 2002], isto é, como o fornecimento de dados sobre um determinado tema ou conjunto de temas relacionados por parte de alguém (ou entidade) com vista à sua utilização por parte de outros (ou entidades) recorrendo a dispositivos tecnológicos. Pode-se considerar que a este fornecimento de dados está sempre inerente a necessidade de armazenamento de informação, isto é, a utilização de um repositório. Este pode ser visto como uma parte integrante da aplicação que constitui o serviço de informação, responsável pelo

fornecimento de informação necessária ao seu funcionamento.

Este armazenamento de informação poderia ser feito de várias formas, no entanto o recurso a BDs é bastante comum, senão mesmo o mais comum. A questão que se coloca a este nível do armazenamento de informação é o modelo a utilizar.

No que diz respeito às BDs, o modelo relacional é sem dúvida o mais utilizado. No entanto, o XML, uma forma de representação de informação emergente e que tem cada vez mais aceitação e utilização, está a ganhar uma posição no mundo das BDs e consolida-la consideravelmente. É interessante notar que ao longo dos últimos anos todos os grandes fabricantes de BDs (como Oracle, IBM ou Microsoft) têm incluído nos seus produtos suporte para XML, o qual vai sendo sempre melhorado nas sucessivas versões dos seus produtos.

Esta entrada do XML no mundo das BDs cria novos desafios relativos ao seu armazenamento e manipulação. Isto leva ao surgimento de soluções baseadas em BDs já existentes, as quais de uma forma geral podem ser denominadas por BDs com suporte para XML, e também ao surgimento de um tipo de BDs inteiramente novo, as nativas XML, concebidas de raiz com o objectivo de armazenar e manipular XML.

Estes dois cenários tecnológicos, utilização do modelo relacional e utilização de XML, justificam pela sua relevância um estudo comparativo de desempenho. Isto foi feito recorrendo a um sistema de testes cuja arquitectura foi baseada em testes de performance (*benchmarks*) já existentes. Este sistema consiste numa aplicação *web* que gere o acesso à BD. No entanto, foi também criada uma aplicação para simular a utilização da aplicação *web* por parte de um grande número de utilizadores.

Actualmente existem vários *benchmarks* para BDs tais como Wisconsin, AS^{3}AP, TPC-C ou TPC-W que podem ser utilizados com BDs relacionais. No que diz respeito a BDs de XML, podem-se distinguir entre 2 tipos de *benchmarks*: *micro-benchmarks* e aplicacionais. O primeiro permite testar partes específicas do sistema, ao passo que o segundo testa o desempenho global do sistema. Existem vários *benchmarks* deste tipo: XMach-1, XOO7, XMark e XBench. Destes distingue-se o XMach-1, baseado numa aplicação *web*, que considera não só o desempenho do SGBD, mas inclui no seu objecto de teste todo o sistema de disponibilização de informação ao utilizador como um todo, incluindo outros componentes como o servidor de aplicação (servidor *web*).

O sistema de testes foi utilizado em diferentes cenários com o objectivo de abranger um

grande número de situações que se poderiam registar em aplicações reais. Além do tipo de BD, as restantes variáveis nos testes realizados foram o número de clientes e o tamanho da BD. Com as várias combinações destas variáveis foram realizados testes que consistiram na execução de pesquisas simples na BD.

Os resultados dos testes demonstram dois níveis de desempenho claros: alto para o modelo relacional e baixo para o modelo XML. Isto não deixaria qualquer dúvida no momento de seleccionar a BD, isto é, uma BD relacional seria a escolha imediata. No entanto, esta decisão não deve ser tomada com base na medida de desempenho pura e simples. Os resultados obtidos têm que ser devidamente enquadrados.

Um primeiro aspecto importante a considerar, é o facto de não terem sido executadas nenhumas configurações especiais nas BDs (além da utilização básica de indexação) no que diz respeito ao armazenamento de XML. As BDs permitem várias abordagens e configurações para o armazenamento interno de XML, pelo que a sua afinação precisa poderia levar a desempenhos superiores. Apesar de tudo, este aspecto por si só poderia não ser suficiente para que fosse superado o desempenho da BD relacional.

Um outro aspecto importante é o facto da informação utilizada nos testes ser originalmente relacional e como tal estar perfeitamente adaptada a este modelo. Informação com outras características poderia ser mais facilmente representada utilizando XML do que com o modelo relacional, tal como é referido no Capítulo 7. Isto é, o modelo relacional poderá não ser o mais adequado para certas situações. Em [Seltzer 2005] são identificadas algumas destas situações como *Data Wharehousing*, serviços de directório, pesquisas na *web*, gestão de XML. Para estas situações onde os SGBD relacionais não são a melhor opção, poderão ser criados sistemas flexíveis que assentassem na modularidade e na facilidade de configuração [Seltzer 2005]. O autor refere ainda que os SGBD relacionais convencionais apresentam uma arquitectura monolítica cujas características são raramente utilizadas na sua totalidade pelas aplicações, pelo que sistemas modulares que possam ser facilmente configuráveis adequam-se melhor às necessidades das aplicações.

Desta forma, a informação utilizada poderá ter também influência nos resultados obtidos. Por esta razão, seria extremamente interessante um estudo mais alargado onde fosse considerado este aspecto, já que reconhecidamente o presente trabalho não permite tirar conclusões a este nível.

8.2 Contribuição

Para a realização deste trabalho tiveram de ser superados alguns obstáculos. Aquilo que terá sido o maior de todos foi a familiarização e posterior utilização de duas BDs completamente distintas, cada uma com as suas particularidades e modo de funcionamento e acesso próprias.

De qualquer das formas, apenas após ter sido ultrapassado este obstáculo de ordem prática é que foi possível implementar a ideia concebida para um teste, não propriamente a BDs, mas sim a modelos de representação de informação que poderiam ser utilizados em repositórios que suportam serviços de informação. Na literatura consultada existem vários trabalhos onde são feitas análises de desempenho, no entanto não existe nenhum com objectivos comparáveis ao presente trabalho.

Na concretização deste trabalho estiveram envolvidos conhecimentos de várias áreas, como BDs, XML e tecnologias associadas, testes de desempenho e linguagens de programação. Trata-se de um trabalho que devido às áreas que envolveu, resultou em alguma complexidade. Apesar disto, há a perfeita consciência que este trabalho está longe de ser um trabalho completo e fechado que esgote todo o conhecimento sobre o tema, nunca tendo sido este um objectivo na sua execução. Antes pelo contrário, a melhor forma de encarar este trabalho é como o ponto de partida para estudos mais abrangentes.

Desta forma, os principais contributos deste trabalho são o facto de se propor testar dois modelos distintos para o armazenamento de informação e por outro lado poder ser visto como a base para trabalhos ainda mais elaborados.

8.3 Trabalho futuro

Com a execução deste trabalho, são deixadas em aberto algumas possibilidades de trabalho futuro que poderão ter bastante interesse.

Para um trabalho mais abrangente, poderão ser considerados os seguintes aspectos:

- Execução de testes com diferentes tipos de informação. Será bastante interessante diversificar o tipo de informação utilizada. Desta forma, poderá ser

utilizada também informação que pelas suas características se enquadre facilmente no modelo XML, explorando assim situações onde o modelo relacional poderá não ser o mais indicado;

- Modificar, tornando mais complexas, as pesquisas efectuadas às BDs. No trabalho executado, uma das preocupações foi manter as pesquisas executadas nas BDs muito simples. Se por um lado isto era desejável de forma a evitar problemas devido à má optimização das pesquisas nas BDs, por outro lado a realidade é que a informação que serviu de base ao trabalho não era suficientemente completa (isto é, existiam bastantes valores nulos) para a execução de pesquisas muito complexas. Desta forma, com a utilização de informação mais completa, poderá ser possível criar pesquisas mais complexas que se aproximem mais de pesquisas reais;
- Modificar o tamanho das BDs. No trabalho executado, o valor para o número máximo de documentos utilizados foi de pouco mais de um milhão. Este valor foi considerado razoável para a execução deste trabalho, no entanto, não foi baseado em nenhum valor real. Desta forma, poderá ter interesse modificar este valor de acordo com critérios baseados em estudos sobre BDs de médias e grandes dimensões utilizadas em aplicações reais.

Além destes aspectos, e apesar de já sair da linha deste trabalho, poderá ainda existir interesse em tecer alguns comentários referentes ao armazenamento de XML, nomeadamente estabelecer uma comparação a nível do tamanho das BDs quando armazenam a mesma informação usando modelos distintos. Dependendo da forma como é feito, o armazenamento de documentos XML em BDs poderá levar ao aumento do espaço necessário quando comparado com outros modelos, pelo que será interessante obter uma ideia concreta sobre este aspecto.

Referências

ACM. (1998). "ACM Computing Classification Systems." consultado em 05/2006, de http://www.acm.org/class/.

Antony, J. (2003). Design of Experiments for Engineers and Scientists, Butterworth-Heinemann Ltd.

Arms, W. Y., C. Blanchi e E. A. Overly. (1997). "An Architecture for Information in Digital Libraries." consultado em 04/2005, de http://www.dlib.org/dlib/february97/cnri/02arms1.html.

Atkinson, M., F. Bancilhon, D. DeWitt, K. Dittrich, D. Maier e S. Zdonik (1989). The object-oriented database system manifesto. The First International Conference on Deductive and Object-Oriented Databases, Kyoto, Elsevier Science.

Bernstein, P. A. (1998). "Repositories and object oriented databases." ACM SIGMOD Record **27**(1): 88 - 96.

Bernstein, P. A. e U. Dayal (1994). An Overview of Repository Technology. VLDB'94, 20th International Conference on Very Large Data Bases, Santiago de Chile, Chile, Morgan Kaufmann.

Bernstein, P. A., B. Harry, P. Sanders, D. Shutt e J. Zander (1997). The Microsoft Repository. 23rd International Conference on Very Large Data Bases, Athens, Greece, Morgan Kaufmann Publishers Inc.

Böhme, T. e E. Rahm (2001). XMach-1: A Benchmark for XML Data Management. Datenbanksysteme in Büro, Technik und Wissenschaft (BTW), Oldenburg, Germany, Springer-Verlag.

Böhme, T. e E. Rahm (2003). "Multi-User Evaluation of XML Data Management Systems with XMach-1." Lecture Notes in Computer Science (LNCS) **2590**: 148-159.

Bommel, P. v. (2003). Information Modeling for Internet Applications, Idea Group Publishing.

Bourret, R. (2005a). Native XML Databases in the Real World. XML 2005 Conference & Exposition, Atlanta, USA.

Bourret, R. (2005b). "XML and Databases." consultado em 03/2006, de http://www.rpbourret.com/xml/XMLAndDatabases.htm.

Bourret, R. (2006). "XML Database Products." consultado em 03/2006, de http://www.rpbourret.com/xml/XMLDatabaseProds.htm.

Bressan, S., M. L. Lee, Y. G. Li, Z. Lacroix e U. B. Nambiar (2003). XML Management System Benchmarks. XML Data Management: Native XML and XML-Enabled Database Systems. A. B.Chaudhri, A. Rashid and R. Zicari, Addison Wesley**:** 477-498.

Carey, M. J., D. J. DeWitt e J. F. Naughton (1993). The 007 Benchmark. ACM SIGMOD international conference on Management of data, Washington, D.C., United States, ACM Press, New York, NY, USA.

Chamberlin, D., P. Fankhauser, M. Marchiori e J. Robie. (2003). "XML Query (XQuery) Requirements." consultado em 03/2005, de http://www.w3.org/TR/xquery-requirements/.

Champion, M. (2001). "Storing XML in Databases." EAI Journal(Outubro 2001): 53-55.

Chen, P. P.-S. (1976). "The entity-relationship model - toward a unified view of data." ACM Transactions on Database Systems **1**(1): 9-36.

Chiu, E. (2002). ebXML Simplified: A Guide to the New Standard for Global E-Commerce, John Wiley & Sons Inc.

Codd, E. F. (1970). "A relational model of data for large shared data banks." Communications of the ACM **13**(6): 377 - 387.

Collin, S. M. H. (2002). Dictionary of Information Technology, Third Edition, Peter Collin Publishing.

Cox, J. (2006). "XML databases gaining acceptance." consultado em 04/2006, de http://www.networkworld.com/news/2006/032006-specialfocus-xml-database.html.

cXML. (1999). "cXML." consultado em 05/2006, de http://www.cxml.org.

DeWitt, D. J. (1993). The Wisconsin Benchmark: Past, Present, and Future. The Benchmark Handbook. J. Gray, Morgan Kaufmann Publishers, Inc.

Dobbie, G., W. Xiaoying, T. W. Ling e M. L. Lee (2001). ORA-SS: An Object-Relationship-Attribute Model for Semi-Structured Data, National University of Singapore.

DSpace. (2004). "DSpace Federation." consultado em 02/2004, de http://www.dspace.org/.

Du, W., M. Li, L. Tok e W. Ling (2001). XML Structures for Relational Data. Second International Conference on Web Information Systems Engineering (WISE'01), IEEE Computer Society.

Duta, A. C., K. Barker e R. Alhajj (2004). ConvRel: Relationship Conversion to XML Nested Structures. Symposium on Applied Computing, Nicosia, Cyprus, ACM Press.

ebXML. (2001). "ebXML Registry Services Specification v2.0." consultado em 02/2005, de http://www.ebxml.org/specs/ebrs2.pdf.

Elmasri, R. e S. B. Navathe (2003). Fundamentals of Database Systems (International Edition), Addison Wesley.

Fegaras, L. e R. Elmasri (2001). Query Engines for Web-Accessible XML Data. 27th VLDB Conference, Roma, Italy.

Fiebig, T., S. Helmer, C.-C. Kanne, G. Moerkotte, J. Neumann, R. Schiele e T. Westmann (2002). "Anatomy of a native XML base management system." The

VLDB Journal - The International Journal on Very Large Data Bases **11**(4): 292 - 314.

Florescu, D. e D. Kossmann (1999). "Storing and Querying XML Data using an RDMBS." IEEE Data Engineering Bulletin **22**: 27-34.

García, D. F. e J. García (2003). TPC-W E-Commerce Benchmark Evaluation. Computer. **36:** 42-48.

Geer, D. (2005). "Will binary XML speed network traffic?" Computer **38**(4): 16-18.

Graves, M. (2002). Designing XML Databases, Prentice Hall PTR.

Gray, J. (1993). Database and Transaction Processing Performance Handbook. The Benchmark Handbook, Morgan Kaufmann Publishers, Inc.

Heijden, H. v. d. (2002). The personal usage of online information services: theory and empirical investigation. Serie Research Memoranda, Free University Amsterdam, Faculty of Economics, Business Administration and Econometrics.

Hernandez, M. J. (2003). Database Design for Mere Mortals: A Hands-On Guide to Relational Database Design, Addison-Wesley Professional.

Jackson, M. (1999). "Thirty years (and more) of databases." Information and Software Technology **41**(14): 969-978.

JR, R. C. D., D. A. Menascé e D. Barbará (2001). Testing E-Commerce Site Scability With TPC-W. Computer Measurement Group Conference 2001 Orlando, FL.

Kahn, R. e R. Wilensky. (1995). "A Framework for Distributed Digital Object Services." consultado em 02/2005, de http://www.cnri.reston.va.us/home/cstr/arch/k-w.html.

Khan, L. e Y. Rao (2001). A performance evaluation of storing XML data in relational database management systems. WIDM 2001, Atlanta, Georgia, USA, ACM.

Khouja, M., F. Kamoun, C. M. Lladó e R. Puigjaner (2004). Experimenting With the TPC-W E-commerce Benchmark. 30ma Conferencia Latinoamericana de Informática (CLEI2004). Arequipa, Perú.

Lapis, G. (2005). XML and Relational Storage - Are they mutually exclusive? XTech 2005, Amsterdam, The Netherlands.

Li, Y. G., S. Bressan, G. Dobbie, Z. Lacroix, M. L. Lee, U. Nambiar e B. Wadhwa (2001). XOO7: Applying OO7 Benchmark to XML Query Processing Tools. Conference on Information and Knowledge Management, Atlanta, Georgia, USA, ACM Press, New York, NY, USA.

Linthicum, D. S. (2001). Enterprise Application Integration, Addison Wesley.

Maguire, C., E. J. Kazlauskas e A. D. Weir (1994). Information Services for Innovative Organizations, Academic Press.

Mason, R. L., R. F. Gunst e J. L. Hess (2003). Statistical Design and Analysis of Experiments: With Applications to Engineering and Science, John Wiley & Sons Inc.

McGee, W. C. (1977). "The information management system IMS/VS, Part I: General structure and operation." IBM Systems Journal **16**(2).

Megginson, D. (2005). XML Performance and Size. Imperfect XML: Rants, Raves, Tips, and Tricks from an Insider, Addison Wesley.

Menascé, D. A. (2002). TPC-W: A Benchmark for E-Commerce. IEEE Internet Computing. **6:** 83 - 87.

Menascé, D. A. e V. Almeida (2001). Capacity Planning for Web Performance: Metrics, Models and Methods Prentice Hall PTR.

Microsoft. (2000). "Meta Data Services." consultado em 05/2004, de http://www.microsoft.com/sql/evaluation/features/metadata.asp.

Nambiar, U., Z. Lacroix, S. Bressan, M.-L. Lee e Y. G. Li (2002a). Efficient XML Data Management: An Analysis. Third International Conference on E-Commerce and Web Technologies, Aix-en-Provence, France, Springer-Verlag.

Nambiar, U., Z. Lacroix, S. Bressan, M. L. Lee e Y. Li (2001). Benchmarking XML Management Systems: The XOO7 Way, Arizona State University - Dept. of Computer Science**:** 19.

Nambiar, U., Z. Lacroix, S. Bressan, M. L. Lee e Y. Li (2002b). "Current Approaches to XML Management." IEEE Internet Computing **6**(4): 43 - 51.

Navathe, S. B. (1992). Evolution of data modeling for databases. Communications of the ACM. **35:** 112-123.

OASIS. (2004). "ebXML." consultado em 02/2004, de http://www.ebxml.org/.

Obasanjo, D. (2001). "XML & Databases." consultado em 01/2005, de http://www.25hoursaday.com/StoringAndQueryingXML.html.

Ozu, N., J. Duckett, A. Watt, S. Mohr, K. Williams, O. G. Gudmundsson, D. Marcus, P. Kobak, E. Lenz, M. Birbeck, Z. Zaev, S. Livingstone, J. Pinnock e K. Visco (2001). Professional XML, Wrox Press Ltd.

Ramakrishnan, R. e J. Gehrke (2001). Database Management Systems, McGraw-Hill Education.

Runapongsa, K., J. M. Patel, H. V. Jagadish, Y. Chen e S. Al-Khalifa (2003). The Michigan Benchmark: Towards XML Query Performance Diagnostics. 29th VLDB Conference, Berlin, Germany.

Santos, L. (2004). Factores Determinantes do Sucesso de Serviços de Informação Online em Sistemas de Gestão de Ciência e Tecnologia, Tese de Doutoramento, Departamento de Sistemas de Informação. Guimarães, Universidade do Minho.

SAP AG. (2004). "SAP Standard Application Benchmarks." consultado em 04/2005, de http://www50.sap.com/benchmark/.

SAX. (2004). "SAX - Simple API for XML." consultado em 03/2006, de http://www.saxproject.org.

Schmidt, A. (2002). Processing XML in Database Systems, PhD, University of Amsterdam.

Schmidt, A., M. Kersten, M. Windhouwer e F. Waas (2000). "Efficient Relational Storage and Retrieval of XML Documents." Lecture Notes in Computer Science.

Schmidt, A., F. Waas, M. Kersten, M. J. Carey, I. Manolescu e R. Busse (2002). XMark: A benchmark for XML Data Management. 28th VLDB Conference, Hong Kong, China.

Schmidt, A., F. Waas, M. Kersten, D. Florescu, M. J. Carey, I. Manolescu e R. Busse (2001). "Why and how to benchmark XML databases." ACM SIGMOD Record **30**(3): 27 - 32.

Schöning, H. (2003). Tamino - Software AG's Native XML Server. XML Data Management: Native XML and XML-enabled Database Systems. A. B. Chaudhri, R. Zicari and A. Rashid, Addison Wesley**:** 21-42.

Schöning, H. e J. Wäsch (2000). "Tamino - An internet database system." ADVANCES IN DATABSE TECHNOLOGY-EDBT 2000, PROCEEDINGS **1777**: 383-387.

Seltzer, M. (2005). "Beyond relational databases." Queue **3**(3): 50-58.

Seng, J.-L., Y. Lin e J.-C. Wang (2002). A Comparative Study of Database Benchmark in Internet Commerce. 35th Annual Hawaii International Conference on System Sciences (HICSS'02), Big Island, Hawaii, IEEE Computer Society.

Seng, J.-L., Y. Lin, J. Wang e J. Yu (2003). "An analytic study of XML database techniques." Industrial Management and Data Systems **103**(2): 111-120.

Seng, J.-L., S. B. Yao e A. R. Hevner (2005). "Requirements-driven database systems benchmark method." Decision Suppot Systems **38**(4): 629 - 648.

Shanmugasundaram, J., K. Tufte, G. He, C. Zhang, D. DeWitt e J. Naughton (1999). Relational Databases for Querying XML Documents: Limitations and Opportunities. 25th VLDB Conference, Edinburgh, Scotland.

Snelson, J. (2005). All XML Databases are Equal. XTech 2005, Amsterdam, The Netherlands.

Steegmans, B., R. Bourret, O. Cline, O. Guyennet, S. Kulkarni, S. Priestley, V. Sylenko e U. Wahli (2004). XML and databases. XML for DB2 Information Integration, IBM Redbooks.

Straube, D. D. e M. T. Özsu (1990). "Queries and query processing in object-oriented database systems." ACM Transactions on Information Systems (TOIS) **8**(4): 387 - 430.

Tansley, R., M. Bass, D. Stuve, M. Branschofsky, D. Chudnov, G. McClellan e M. Smith (2003). The DSpace institutional digital repository system: current functionality. Third ACM/IEEE-CS joint conference on Digital libraries, Houston, Texas, IEEE Computer Society.

The Fedora™ Project. (2004). "The Fedora™ Project." consultado em 02/2004, de http://www.fedora.info/.

TPC (2002). Transaction Processing Performance Council - TPC Benchmark W, ver. 1.8. **2004**.

TPC (2005). Transaction Processing Performance Council - TPC Benchmark C, ver. 5.4. **2005**.

Turbyfill, C., C. Orji e D. Bitton (1989). AS3AP:a comparative relational database benchmark. 34th IEEE Computer Society International Conference, San Francisco, CA.

Vakali, A., B. Catania e A. Maddalena (2005). "XML Data Stores: Emerging Practices." Internet Computing, IEEE **9**(2): 62-69.

W3C. (1999). "XML Path Language." consultado em 02/2005, de http://www.w3.org/TR/xpath.

W3C. (2001a). "Document Object Model." consultado em 02/2005, de http://www.w3.org/DOM/.

W3C. (2001b). "XLink." consultado em 02/2005, de http://www.w3.org/TR/xlink/.

W3C. (2004a). "Extensible Markup Language (XML) 1.0 (Third Edition)." consultado em 02/2005, de http://www.w3.org/TR/REC-xml.

W3C. (2004b). "XML Schema." consultado em 02/2006, de http://www.w3.org/XML/Schema.

W3C. (2005). "XQuery 1.0: An XML Query Language." consultado em 04/2006, de http://www.w3.org/TR/xquery/.

Williams, K., P. Dengler, J. Gabriel, A. Hoskinson, M. Kay, M. Brundage, T. Maxwell, M. Ochoa, J. Papa e M. Vanmane (2000). Professional XML Databases, Wrox Press Ltd.

XML:DB. (2004). "XML:DB Initiative for XML Databases." consultado em 05/2004, de http://www.xmldb.org/.

Yao, B. B., M. T. Özsu e J. Keenleyside (2002). XBench - A Family of Benchmarks for XML DBMSs, University of Waterloo.

Yao, B. B., M. T. Özsu e N. Khandelwal (2004). XBench Benchmark and Performance Testing of XML DBMSs. 20th International Conference on Data Engineering, Boston, Massachusetts, U.S.A., IEEE Computer Society.

Zand, M., V. Collins e D. Caviness (1995). "A survey of current object-oriented databases." ACM SIGMIS Database **26**(1): 14 - 29.

Anexos

Anexo I: Pesquisas utilizadas

Neste trabalho foram utilizadas 5 pesquisas. Na Tabela 9 está descrita a correspondência (assinalando com um "X") entre as pesquisas e requisitos que satisfazem. Estes estão definidos no Capítulo 5, e são os seguintes:

- **R1:** Acesso directo aos itens por chave (pesquisa por chave);
- **R2:** Obtenção de resultados ordenados segundo determinados critérios;
- **R3:** Obtenção de indicadores com base na informação armazenada;
- **R4:** Suporte eficiente para resultados de grandes dimensões (elevado número de itens);
- **R5:** Pesquisas textuais.

Requisito / Pesquisa	R1	R2	R3	R4	R5
P1				X	X
P2	X				
P3			X	X	
P4		X		X	X
P5		X		X	X

Tabela 9: Correspondência entre as pesquisas e requisitos

De seguida são fornecidos exemplos de todas as versões dos *queries* envolvidos nas pesquisas utilizadas.

I.1 Pesquisa 1

I.1.1 Relacional

(sem indexação)

```
select * from(
select NME_RH, NRO_ID_CNPQ from EN_Recurso_Humano
      where (
      NME_RH like 'Mendonça %' OR
      NME_RH like '% Mendonça %' OR
      NME_RH like '% Mendonça')
) where rownum<=1000
```

(com indexação)

```
select * from(
select NME_RH, NRO_ID_CNPQ from EN_Recurso_Humano
      where contains (NME_RH,'Mendonça')>0
) where rownum<=1000
```

I.1.2 Tamino - XQuery

```
declare namespace ft="http://www.w3.org/2002/04/xquery-operators-text"

for $a in input()/EN_RECURSO_HUMANO
where ft:text-contains($a/@NME_RH, "José")
return
<cv>
      {$a/@NME_RH}
      {$a/@NRO_ID_CNPQ}
</cv>
```

I.1.3 Oracle XML Type Table

(sem indexação)

```
select xtab.NME_RH, xtab.NRO_ID_CNPQ
from cvxml, XMLTable('
for $a in /EN_RECURSO_HUMANO
where contains($a/@NME_RH, "José")
return
<cv NME_RH="{$a/@NME_RH}" NRO_ID_CNPQ="{$a/@NRO_ID_CNPQ}">
</cv>
' PASSING OBJECT_VALUE
COLUMNS NME_RH varchar2(255) PATH '/cv/@NME_RH',
      NRO_ID_CNPQ varchar2(16) PATH '/cv/@NRO_ID_CNPQ' ) xtab
where rownum<=1000
```

(com indexação)

```
SELECT extractValue(OBJECT_VALUE, '/EN_RECURSO_HUMANO/@NME_RH')
"NME_RH",
extractValue(OBJECT_VALUE, '/EN_RECURSO_HUMANO/@NRO_ID_CNPQ')
"NRO_ID_CNPQ"
FROM cvxml where
CONTAINS(OBJECT_VALUE, 'Pedro INPATH (/EN_RECURSO_HUMANO/@NME_RH)') >
0
and rownum<1000
```

I.2 Pesquisa 2

I.2.1 Relacional

```
SELECT * FROM EN_RECURSO_HUMANO WHERE NRO_ID_CNPQ='9999950000203248'
SELECT * FROM EN_FORMACAO WHERE NRO_ID_CNPQ='9999950000203248'
SELECT * FROM EN_CURSO_OUTRO WHERE NRO_ID_CNPQ='9999950000203248'
SELECT * FROM EN_INSTITUICAO_OUTRA WHERE
NRO_ID_CNPQ='9999950000203248'
SELECT * FROM EN_ATIVIDADE_PROFISSIONAL WHERE
NRO_ID_CNPQ='9999950000203248'
SELECT * FROM EN_ENDERECO_RH WHERE NRO_ID_CNPQ='9999950000203248'
SELECT * FROM RE_RH_IDIOMA WHERE NRO_ID_CNPQ='9999950000203248'
SELECT * FROM EN_PRODUCAO_CIENTIF_TECNOL WHERE
NRO_ID_CNPQ='9999950000203248'
```

I.2.2 Tamino - Xquery

```
for $a in input()/EN_RECURSO_HUMANO
      where $a/@NRO_ID_CNPQ="9999950000203248"
return
$a
```

I.2.3 Oracle XML Type Table

```
select xtab.NME_RH, xtab.NRO_ID_CNPQ
from cvxml, XMLTable('
for $a in /EN_RECURSO_HUMANO
      where $a/@NRO_ID_CNPQ="0037482275072878"
return
$a
' PASSING OBJECT_VALUE
COLUMNS NME_RH varchar2(255) PATH '@NME_RH',
      NRO_ID_CNPQ varchar2(16) PATH '@NRO_ID_CNPQ' ) xtab
```

I.3 Pesquisa 3

I.3.1 Relacional

```
select NME_RH, EN_Recurso_Humano.NRO_ID_CNPQ
(
select COUNT(EN_PRODUCAO_CIENTIF_TECNOL.NRO_ID_CNPQ)
      FROM EN_PRODUCAO_CIENTIF_TECNOL WHERE
EN_PRODUCAO_CIENTIF_TECNOL.NRO_ID_CNPQ=EN_Recurso_Humano.NRO_ID_CNPQ
) AS total_prod_cientif,
(
select MAX(ANO_PRODUCAO)
      FROM EN_PRODUCAO_CIENTIF_TECNOL WHERE
EN_PRODUCAO_CIENTIF_TECNOL.NRO_ID_CNPQ=EN_Recurso_Humano.NRO_ID_CNPQ
) AS ultimo_ano_prod,
(
select COUNT(EN_Formacao.NRO_ID_CNPQ)
      FROM EN_Formacao WHERE
EN_Formacao.NRO_ID_CNPQ=EN_Recurso_Humano.NRO_ID_CNPQ
) AS total_formacao,
(
```

```
select MAX(ANO_TER_FORM)
       FROM EN_Formacao WHERE
EN_Formacao.NRO_ID_CNPQ=EN_Recurso_Humano.NRO_ID_CNPQ
)
as ultimo_ano_form
from EN_Recurso_Humano
where EN_Recurso_Humano.NRO_ID_CNPQ='9999950000203248'
```

I.3.2 Tamino - XQuery

```
for $a in input()/EN_RECURSO_HUMANO
      let $b:=$a/EN_PRODUCAO_CIENTIF_TECNOL
      let $c:=$a/EN_FORMACAO
where $a/@NRO_ID_CNPQ='9999950000203248'
return
<cv total_prod_cientif=5 ultimo_ano_prod={max($b/@ANO_PRODUCAO)}
total_formacao={count($c)} ultimo_ano_form={max($c/@ANO_TER_FORM)}>
      {$a/@NME_RH}
      {$a/@NRO_ID_CNPQ}
</cv>
```

I.3.3 Oracle XML Type Table

```
select xtab.NME_RH, xtab.NRO_ID_CNPQ, xtab.total_formacao,
xtab.total_prod_cientif, xtab.ultimo_ano_form, xtab.ultimo_ano_prod
from cvxml, XMLTable('
for $a in /EN_RECURSO_HUMANO
      let $b:=$a/EN_PRODUCAO_CIENTIF_TECNOL
      let $c:=$a/EN_FORMACAO
where $a/@NRO_ID_CNPQ="0037482275072878"
return
<cv NME_RH="{$a/@NME_RH}" NRO_ID_CNPQ="{$a/@NRO_ID_CNPQ}"
total_prod_cientif="{count($b)}
" ultimo_ano_prod="{max($b/@ANO_PRODUCAO)}"
total_formacao="{count($c)}"
ultimo_ano_form="{max($c/@ANO_TER_FORM)}">
</cv>
' PASSING OBJECT_VALUE
COLUMNS NME_RH varchar2(255) PATH '/cv/@NME_RH',
           NRO_ID_CNPQ varchar2(16) PATH '/cv/@NRO_ID_CNPQ',
```

```
            total_formacao varchar(10) PATH '/cv/@total_formacao',
            total_prod_cientif varchar(10) PATH
'/cv/@total_prod_cientif',
            ultimo_ano_form varchar(10) PATH '/cv/@ultimo_ano_form',
            ultimo_ano_prod varchar(10) PATH '/cv/@ultimo_ano_prod' )
xtab
```

I.4 Pesquisa 4

I.4.1 Relacional

(sem indexação)

```
select * from(
select NME_RH, NRO_ID_CNPQ from EN_Recurso_Humano
      where (
      NME_RH like 'Mendonça %' OR
      NME_RH like '% Mendonça %' OR
      NME_RH like '% Mendonça')
ORDER BY COD_ORIGEM_CURRICULO
) where rownum<=1000
```

(com indexação)

```
select * from(
select NME_RH, NRO_ID_CNPQ from EN_Recurso_Humano
      where contains (NME_RH,'Mendonça')>0
ORDER BY COD_ORIGEM_CURRICULO
) where rownum<=1000
```

I.4.2 Tamino - XQuery

```
declare namespace ft="http://www.w3.org/2002/04/xquery-operators-text"
declare namespace xs = "http://www.w3.org/2001/XMLSchema"

for $a in input()/EN_RECURSO_HUMANO
let $b := $a/@COD_ORIGEM_CURRICULO
where ft:text-contains($a/@NME_RH, "Mendonça")
return
```

```
<cv COD_ORIGEM_CURRICULO={$b}>
      {$a/@NRO_ID_CNPQ}
      {$a/@NME_RH}
</cv>
sort by (@COD_ORIGEM_CURRICULO)
```

I.4.3 Oracle XML Type Table

(sem indexação)

```
select * from (
select /*+parallel (cvxml, 6)*/ xtab.NME_RH, xtab.NRO_ID_CNPQ,
xtab.COD_ORIGEM_CURRICULO
from cvxml, XMLTable('
for $a in /EN_RECURSO_HUMANO
where contains($a/@NME_RH, "José")
return
<cv COD_ORIGEM_CURRICULO="{$a/@COD_ORIGEM_CURRICULO}"
NME_RH="{$a/@NME_RH}" NRO_ID_CNPQ="{$a/@NRO_ID_CNPQ}">
</cv>
' PASSING OBJECT_VALUE
COLUMNS
            COD_ORIGEM_CURRICULO number PATH '@COD_ORIGEM_CURRICULO',
            NME_RH varchar2(255) PATH '@NME_RH',
            NRO_ID_CNPQ varchar2(16) PATH '@NRO_ID_CNPQ' ) xtab
order by COD_ORIGEM_CURRICULO
)
where rownum<=1000
```

(com indexação)

```
select * from(
SELECT
extractValue(OBJECT_VALUE, '/EN_RECURSO_HUMANO/@COD_ORIGEM_CURRICULO')
"COD_ORIGEM_CURRICULO",
extractValue(OBJECT_VALUE, '/EN_RECURSO_HUMANO/@NME_RH') "NME_RH",
extractValue(OBJECT_VALUE, '/EN_RECURSO_HUMANO/@NRO_ID_CNPQ')
"NRO_ID_CNPQ"
FROM cvxml where
CONTAINS(OBJECT_VALUE, 'Pedro INPATH (/EN_RECURSO_HUMANO/@NME_RH)') >
0
order by COD_ORIGEM_CURRICULO
```

```
)
where rownum<1000
```

I.5 Pesquisa 5

I.5.1 Relacional

(sem indexação)

```
select  * from(
select NME_RH, NRO_ID_CNPQ from EN_Recurso_Humano
      where (
      NME_RH like 'Mendonça %' OR
      NME_RH like '% Mendonça %' OR
      NME_RH like '% Mendonça')
ORDER BY NME_RH
) where rownum<=1000
```

(com indexação)

```
select * from(
select NME_RH, NRO_ID_CNPQ from EN_Recurso_Humano
      where contains (NME_RH,'Mendonça')>0
ORDER BY NME_RH
) where rownum<=1000
```

I.5.2 Tamino - XQuery

```
declare namespace ft="http://www.w3.org/2002/04/xquery-operators-text"

for $a in input()/EN_RECURSO_HUMANO
where ft:text-contains($a/@NME_RH, "Mendonça")
return
<cv>
      {$a/@NME_RH}
      {$a/@NRO_ID_CNPQ}
</cv>
sort by (@NME_RH)
```

I.5.3 Oracle XML Type Table

(sem indexação)

```
select * from(
select /*+parallel (cvxml, 6)*/ xtab.NME_RH, xtab.NRO_ID_CNPQ
from cvxml, XMLTable('
for $a in /EN_RECURSO_HUMANO
where contains($a/@NME_RH, "José")
return
<cv NME_RH="{$a/@NME_RH}" NRO_ID_CNPQ="{$a/@NRO_ID_CNPQ}">
</cv>
' PASSING OBJECT_VALUE
COLUMNS NME_RH varchar2(255) PATH '@NME_RH',
            NRO_ID_CNPQ varchar2(16) PATH '@NRO_ID_CNPQ' ) xtab
order by nme_rh
)
where rownum<=1000
```

(com indexação)

```
select * from(
SELECT extractValue(OBJECT_VALUE, '/EN_RECURSO_HUMANO/@NME_RH')
"NME_RH",
extractValue(OBJECT_VALUE, '/EN_RECURSO_HUMANO/@NRO_ID_CNPQ')
"NRO_ID_CNPQ"
FROM cvxml where
CONTAINS(OBJECT_VALUE, 'Pedro INPATH (/EN_RECURSO_HUMANO/@NME_RH)') >
0
order by NME_RH
)
where rownum<1000
```

Anexo II: Diagrama ER

A BD relacional inicial contém um grande número de tabelas. Neste trabalho foi seleccionada uma parte das tabelas, mas que mantêm a estrutura básica dos CVs. A figura seguinte (Figura 25) representa o diagrama ER parcial da BD utilizada.

Deste diagrama fazem parte, por uma questão de simplicidade, apenas as entidades, relações e atributos principais, que permitem perceber o conceito de CV utilizado neste trabalho.

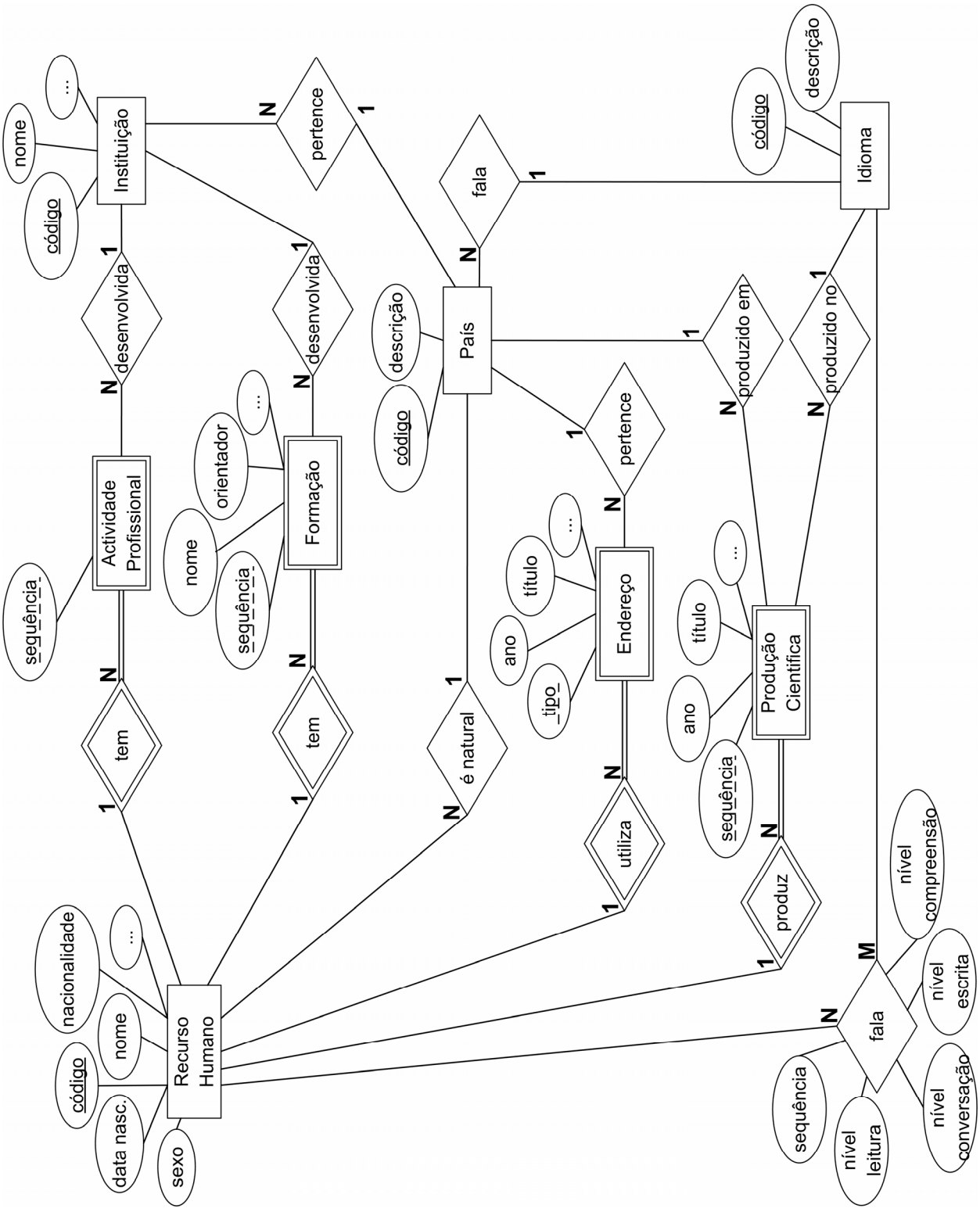

Figura 25: Diagrama ER da BD utilizada.

Anexo III: XML *Schema*

Este Anexo descreve o XML *Schema* utilizado nas BDs de XML. Este *Schema*, que foi criado com base no *schema* relacional simplificado, permite validar os documentos que representam CV utilizados neste trabalho, nas BDs que manipulam XML.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xdb="http://xmlns.oracle.com/xdb" version="0">
  <xs:element name="EN_RECURSO_HUMANO" xdb:defaultTable="EN_RECURSO_HUMANO">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="EN_FORMACAO" minOccurs="0" maxOccurs="unbounded">
          <xs:complexType>
            <xs:attribute name="SEQ_FORM" type="xs:decimal"/>
            <xs:attribute name="COD_INST" type="xs:string"/>
            <xs:attribute name="COD_INST_FINANC" type="xs:string"/>
            <xs:attribute name="COD_INST_OUTRA" type="xs:string"/>
            <xs:attribute name="COD_INST_OUTRA_FINANC" type="xs:string"/>
            <xs:attribute name="COD_CURSO" type="xs:string"/>
            <xs:attribute name="COD_CURSO_OUTRO" type="xs:string"/>
            <xs:attribute name="COD_NIVEL_FORM" type="xs:string"/>
            <xs:attribute name="TXT_TIT_FORM" type="xs:string"/>
            <xs:attribute name="NME_ORIENT_FORM" type="xs:string"/>
            <xs:attribute name="ANO_INI_FORM" type="xs:string"/>
            <xs:attribute name="ANO_TER_FORM" type="xs:string"/>
            <xs:attribute name="TOT_CARGA_HORARIA" type="xs:decimal"/>
            <xs:attribute name="STA_BOLSISTA" type="xs:string"/>
            <xs:attribute name="ANO_OBTEN_FORM" type="xs:string"/>
            <xs:attribute name="STA_FORMACAO_CONCLUIDA" type="xs:string"/>
            <xs:attribute name="TXT_CIDADE_FORM" type="xs:string"/>
          </xs:complexType>
        </xs:element>
        <xs:element name="EN_ATIVIDADE_PROFISSIONAL" minOccurs="0"
maxOccurs="unbounded">
          <xs:complexType>
            <xs:attribute name="SEQ_ATIVIDADE" type="xs:decimal"/>
            <xs:attribute name="COD_INST" type="xs:string"/>
            <xs:attribute name="COD_INST_OUTRA" type="xs:string"/>
          </xs:complexType>
        </xs:element>
        <xs:element name="EN_ENDERECO_RH" minOccurs="0" maxOccurs="unbounded">
          <xs:complexType>
            <xs:attribute name="TPO_ENDER" type="xs:string"/>
```

```
      <xs:attribute name="COD_MUNICIPIO_END" type="xs:string"/>
      <xs:attribute name="COD_INST_OUTRA" type="xs:string"/>
      <xs:attribute name="SGL_PAIS_END" type="xs:string"/>
      <xs:attribute name="COD_INST" type="xs:string"/>
      <xs:attribute name="SGL_UF_END" type="xs:string"/>
      <xs:attribute name="TXT_LOGR_END" type="xs:string"/>
      <xs:attribute name="TXT_BAIRRO_END" type="xs:string"/>
      <xs:attribute name="TXT_CIDADE_END" type="xs:string"/>
      <xs:attribute name="NRO_CAIXA_POSTAL" type="xs:string"/>
      <xs:attribute name="COD_ZIP_END" type="xs:string"/>
      <xs:attribute name="COD_DDD" type="xs:string"/>
      <xs:attribute name="NRO_FONE" type="xs:string"/>
      <xs:attribute name="NRO_RAMAL" type="xs:string"/>
      <xs:attribute name="NRO_FAX" type="xs:string"/>
      <xs:attribute name="TXT_EMAIL" type="xs:string"/>
      <xs:attribute name="URL_HOME_PAGE" type="xs:string"/>
      <xs:attribute name="STA_ENDERECO_ATUALIZADO" type="xs:string"/>
      <xs:attribute name="SGL_PROVINCIA_END" type="xs:string"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="EN_INSTITUICAO_OUTRA" minOccurs="0" maxOccurs="unbounded">
    <xs:complexType>
      <xs:attribute name="COD_INST_OUTRA" type="xs:string"/>
      <xs:attribute name="SGL_UF_END_INST" type="xs:string"/>
      <xs:attribute name="SGL_PAIS" type="xs:string"/>
      <xs:attribute name="NME_INST" type="xs:string"/>
      <xs:attribute name="SGL_INST" type="xs:string"/>
      <xs:attribute name="COD_INST" type="xs:string"/>
      <xs:attribute name="STA_AGENCIA_FOMENTO" type="xs:string"/>
      <xs:attribute name="STA_INSTITUICAO_ENSINO" type="xs:string"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="EN_CURSO_OUTRO" minOccurs="0" maxOccurs="unbounded">
    <xs:complexType>
      <xs:attribute name="COD_CURSO_OUTRO" type="xs:string"/>
      <xs:attribute name="COD_INST_OUTRA" type="xs:string"/>
      <xs:attribute name="COD_INST" type="xs:string"/>
      <xs:attribute name="COD_AREA_CONHEC_OUTRA" type="xs:string"/>
      <xs:attribute name="DSC_CURSO_OUTRO" type="xs:string"/>
      <xs:attribute name="COD_NIVEL_CURSO" type="xs:string"/>
      <xs:attribute name="COD_AREA_CONHEC" type="xs:string"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="RE_RH_IDIOMA" minOccurs="0" maxOccurs="unbounded">
    <xs:complexType>
      <xs:attribute name="SEQ_IDIOMA" type="xs:decimal"/>
      <xs:attribute name="COD_IDIOMA" type="xs:string"/>
      <xs:attribute name="DSC_IDIOMA" type="xs:string"/>
      <xs:attribute name="NVL_LEITURA" type="xs:string"/>
      <xs:attribute name="NVL_CONVERSACAO" type="xs:string"/>
      <xs:attribute name="NVL_ESCRITA" type="xs:string"/>
```

```
            <xs:attribute name="NVL_COMPREENSAO" type="xs:string"/>
          </xs:complexType>
        </xs:element>
        <xs:element name="EN_PRODUCAO_CIENTIF_TECNOL" minOccurs="0"
maxOccurs="unbounded">
          <xs:complexType>
            <xs:attribute name="SEQ_PRODUCAO" type="xs:decimal"/>
            <xs:attribute name="COD_SUBTIPO_PRODUCAO" type="xs:string"/>
            <xs:attribute name="SGL_PAIS" type="xs:string"/>
            <xs:attribute name="COD_IDIOMA" type="xs:string"/>
            <xs:attribute name="COD_TIPO_PRODUCAO" type="xs:string"/>
            <xs:attribute name="ANO_PRODUCAO" type="xs:string"/>
            <xs:attribute name="MES_PRODUCAO" type="xs:string"/>
            <xs:attribute name="TXT_TITULO_PRODUCAO" type="xs:string"/>
            <xs:attribute name="TXT_TPO_COMPLEMENTAR" type="xs:string"/>
            <xs:attribute name="SEQ_ORDEM_AUTORIA" type="xs:decimal"/>
            <xs:attribute name="STA_RELEVANCIA" type="xs:string"/>
            <xs:attribute name="TPO_MEIO" type="xs:string"/>
            <xs:attribute name="URL_PRODUCAO" type="xs:string"/>
            <xs:attribute name="TPO_ORIGEM" type="xs:string"/>
            <xs:attribute name="COD_IDENT_PRODUCAO" type="xs:string"/>
            <xs:attribute name="TXT_OBSERVACAO" type="xs:string"/>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="NRO_ID_CNPQ" type="xs:string"/>
      <xs:attribute name="NME_RH_FILTRO" type="xs:string"/>
      <xs:attribute name="COD_SIST_PROCED" type="xs:string"/>
      <xs:attribute name="SGL_UF_CART_IDENT" type="xs:string"/>
      <xs:attribute name="SGL_PAIS_NASC" type="xs:string"/>
      <xs:attribute name="COD_NIVEL_FORM" type="xs:string"/>
      <xs:attribute name="COD_RH" type="xs:decimal"/>
      <xs:attribute name="CPF_RH" type="xs:string"/>
      <xs:attribute name="NME_RH" type="xs:string"/>
      <xs:attribute name="TPO_NACIONALIDADE" type="xs:string"/>
      <xs:attribute name="NRO_PASSAPORTE" type="xs:string"/>
      <xs:attribute name="DTA_ATUALIZACAO_CURRICULO" type="xs:date"/>
      <xs:attribute name="COD_SEXO" type="xs:string"/>
      <xs:attribute name="DTA_NASC" type="xs:date"/>
      <xs:attribute name="NME_CITACAO_BIBLIOG" type="xs:string"/>
      <xs:attribute name="DTA_VALID_VISTO" type="xs:date"/>
      <xs:attribute name="TPO_ENDER_CORR" type="xs:string"/>
      <xs:attribute name="NRO_CART_IDENT" type="xs:string"/>
      <xs:attribute name="ORG_CART_IDENT" type="xs:string"/>
      <xs:attribute name="DTA_CART_IDENT" type="xs:date"/>
      <xs:attribute name="TXT_LOCAL_NASC_RH" type="xs:string"/>
      <xs:attribute name="COD_ESTADO_CIVIL" type="xs:string"/>
      <xs:attribute name="TXT_SENHA_LOCAL" type="xs:string"/>
      <xs:attribute name="TXT_SENHA_CNPQ" type="xs:string"/>
      <xs:attribute name="NME_PAI" type="xs:string"/>
      <xs:attribute name="NME_MAE" type="xs:string"/>
```

```
      <xs:attribute name="STA_PERMISSAO_APRES" type="xs:string"/>
      <xs:attribute name="SGL_UF_NASC" type="xs:string"/>
      <xs:attribute name="TXT_SITUACAO_LOCAL" type="xs:string"/>
      <xs:attribute name="TXT_SITUACAO_CNPQ" type="xs:string"/>
      <xs:attribute name="COD_PESQ_MCT" type="xs:string"/>
      <xs:attribute name="NME_CITACAO_BIBLIOG_FILTRO" type="xs:string"/>
      <xs:attribute name="NRO_ID_INST" type="xs:string"/>
      <xs:attribute name="COD_ORIGEM_CURRICULO" type="xs:decimal"/>
    </xs:complexType>
    <xs:key name="COD_INST_OUTRA">
      <xs:selector xpath="EN_INSTITUICAO_OUTRA"/>
      <xs:field xpath="@COD_INST_OUTRA"/>
    </xs:key>
    <xs:keyref name="KEYREF_COD_INST_OUTRA" refer="COD_INST_OUTRA">
      <xs:selector
xpath="EN_FORMACAO|EN_ATIVIDADE_PROFISSIONAL|EN_ENDERECO_RH|EN_CURSO_OUTRO"/>
      <xs:field xpath="@COD_INST_OUTRA"/>
    </xs:keyref>
    <xs:keyref name="KEYREF_COD_INST_OUTRA_FINANC" refer="COD_INST_OUTRA">
      <xs:selector xpath="EN_FORMACAO"/>
      <xs:field xpath="@COD_INST_OUTRA_FINANC"/>
    </xs:keyref>
    <xs:key name="COD_CURSO_OUTRO">
      <xs:selector xpath="EN_CURSO_OUTRO"/>
      <xs:field xpath="@COD_CURSO_OUTRO"/>
    </xs:key>
    <xs:keyref name="KEYREF_COD_CURSO_OUTRO" refer="COD_CURSO_OUTRO">
      <xs:selector xpath="EN_FORMACAO"/>
      <xs:field xpath="@COD_CURSO_OUTRO"/>
    </xs:keyref>
  </xs:element>
</xs:schema>
```

Anexo IV: Valores obtidos

Este Anexo descreve todas as medições efectuadas (Tabela 10), isto é, contém os números de pesquisas contabilizadas com todas as combinações dos factores considerados: tamanho da BD, número de clientes e tipo de BD. Para cada combinação foram feitas 7 medições.

Tam. BD	Clientes	Sup. XML	Nativa XML	Relacional	Tam. BD	Clientes	Sup. XML	Nativa XML	Relacional	Tam. BD	Clientes	Sup. XML	Nativa XML	Relacional
115k	25	215	250	320	517k	25	45	152	338	1000k	25	17	21	16
115k	25	282	255	369	517k	25	63	179	345	1000k	25	13	35	27
115k	25	279	267	373	517k	25	77	185	353	1000k	25	24	29	15
115k	25	283	260	343	517k	25	63	230	350	1000k	25	32	22	18
115k	25	262	263	372	517k	25	69	189	360	1000k	25	22	18	29
115k	25	249	218	367	517k	25	71	194	360	1000k	25	25	27	20
115k	25	259	245	361	517k	25	71	195	354	1000k	25	21	27	16
115k	50	389	268	651	517k	50	90	174	679	1000k	50	21	28	36
115k	50	341	301	719	517k	50	136	170	541	1000k	50	30	20	35
115k	50	379	286	712	517k	50	135	216	663	1000k	50	37	22	44
115k	50	406	303	728	517k	50	143	192	685	1000k	50	20	23	26
115k	50	407	294	713	517k	50	108	138	694	1000k	50	24	27	36
115k	50	395	307	723	517k	50	111	209	661	1000k	50	26	25	34
115k	50	422	278	715	517k	50	145	211	698	1000k	50	30	30	34
115k	75	401	283	1029	517k	75	134	149	986	1000k	75	20	21	42
115k	75	397	280	1042	517k	75	193	158	935	1000k	75	22	24	48
115k	75	402	276	995	517k	75	211	136	958	1000k	75	44	25	42
115k	75	381	287	1006	517k	75	140	153	971	1000k	75	29	19	41
115k	75	389	287	1029	517k	75	170	161	894	1000k	75	37	32	58
115k	75	394	285	975	517k	75	137	179	957	1000k	75	21	19	44
115k	75	395	282	1022	517k	75	144	193	911	1000k	75	35	32	48
115k	100	392	283	1265	517k	100	166	150	1171	1000k	100	24	22	55
115k	100	391	276	1275	517k	100	164	157	1145	1000k	100	39	27	37
115k	100	366	288	1165	517k	100	184	166	1084	1000k	100	40	25	45
115k	100	366	291	1201	517k	100	201	160	1133	1000k	100	37	20	36
115k	100	416	287	1216	517k	100	162	180	1119	1000k	100	38	17	37
115k	100	405	283	1264	517k	100	164	167	1076	1000k	100	38	19	42
115k	100	416	289	1196	517k	100	208	172	1120	1000k	100	27	26	41
115k	125	387	263	1259	517k	125	134	156	1133	1000k	125	33	27	45
115k	125	427	273	1282	517k	125	244	148	1070	1000k	125	35	34	38
115k	125	432	288	1348	517k	125	184	174	1148	1000k	125	19	29	51
115k	125	406	281	1318	517k	125	163	150	1044	1000k	125	44	20	36
115k	125	378	283	1319	517k	125	196	162	1133	1000k	125	31	35	38
115k	125	420	252	1254	517k	125	152	171	1120	1000k	125	15	18	64
115k	125	457	287	1370	517k	125	190	153	1087	1000k	125	27	19	39

Tabela 10: Valores obtidos