FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

FEUP

Bases de Dados alternativas para *Websites*

Nuno Miguel Queirós Arantes dos Santos

Relatório do Estado da Arte

Mestrado Integrado em Engenharia Informática e Computação

Orientador: Maria Teresa Galvão Dias (Dra.)

Responsável na empresa: Rita Monteiro (Dra.)

Janeiro de 2011

Bases de Dados alternativas para *Websites*

Nuno Miguel Queirós Arantes dos Santos

Relatório do Estado da Arte
Mestrado Integrado em Engenharia Informática e Computação

Aprovado em provas públicas pelo júri:

Presidente: José Manuel Magalhães Cruz (Dr.)

Vogal: José Manuel Torres (Dr.)
Vogal: Maria Teresa Galvão Dias (Dra.)

9 de Fevereiro de 2011

Resumo

Na última década imperou o uso de bases de dados relacionais. No entanto, nos últimos anos, com o surgimento de novas tecnologias, foram criadas ou renovadas ideias que visam uma maior rapidez e eficiência do sistema. É neste sentido que a ideia das bases de dados não-relacionais foi reutilizada, tendo surgindo assim novas plataformas. Este tipo de plataformas foi tendo mais visibilidade pela sua maior rapidez e implementação em serviços como o Facebook e o Twitter.

Com o ressurgimento da discussão das vantagens sobre o uso das bases de dados não-relacionais, é importante perceber se estas são uma alternativa válida e eficiente às bases de dados relacionais. Sendo um novo movimento, ainda não existe um estudo sustentado sobre as vantagens e desvantagens da utilização destas, havendo apenas alguns estudos sobre a sua implementação e os resultados obtidos, mas nada que valide o que se apregoa sobre este tipo de bases de dados.

A dissertação será assim direccionada para a verificação dos ganhos da utilização de um sistema de gestão de bases de dados não-relacional para as plataformas estudadas, e não será um estudo exaustivo sobre as diferenças entre relacional e não-relacional.

Inicialmente tentou-se perceber qual será o melhor sistema de gestão de bases de dados existente para as plataformas Casa Sapo e Imoguia, bem como para a respectiva implementação. Para assegurar uma comparação justa foi necessário migrar a informação existente para a nova base de dados, de forma a assegurar que se trabalha com o mesmo volume de dados.

Após a implementação, foram efectuados testes comparativos com o intuito de verificar qual dos sistemas responde melhor às plataformas estudadas.

Abstract

In the last decade has reigned the use of relational databases. However, in recent years, with the emergence of new technologies, have been created or renewed ideas to achieve greater speed and efficiency. This is why the idea of non-relational databases was reused, resulting in new platforms. Such platforms have been increased visibility for its greater speed and implementation in services such as Facebook and Twitter.

With the resurgence of discussion of the advantages over the use of non-relational databases, it is important to realize that these are a viable alternative and efficient for relational databases. Being a new movement, there is no sustained study on the advantages and disadvantages of using these, there are a few studies on its implementation and results, but nothing that validates what has been said on such databases.

This dissertation will be well targeted to verify the gains of using a non-relational management system databases for the studied platforms, and will not be an exhaustive study of the differences between relational and non-relational.

Initially we tried to understand what is the best management system database to existing platforms Imoguia and Casa Sapo, as well as for its implementation. To ensure a fair comparison it was necessary to migrate existing data to the new database, to ensure that they're working with the same volume of data.

After implementation, comparative tests were carried out in order to determine which system has the best performance to the platforms studied.

Agradecimentos

Agradecimentos à Janela Digital, em especial ao Ismael Paulino, Rita Monteiro e Eduardo Amaral, e à Professora Maria Teresa Galvão Dias, pelo apoio e orientação na Dissertação.

Ao Professor João António Correia Lopes pelo fornecimento da chave académica para a utilização da ferramenta OxygenXML.

Ao NIFEUP pela muita coisa estranha que se passou nos anos de faculdades.

Aos "Duques" por me aturarem e por tudo e mais alguma coisa. Em especial, ao Hugo Peixoto, pelos constantes avisos de "Produção" e "Programação".

À Sara Monteiro pela revisão, apesar de não perceber nada disto.

No fundo, uma agradecimento forte aos "meus".

Nuno Santos

"If you steal from one another, it's plagiarism;
if you steal from many it's research."

Wilson Mizner

Conteúdo

Lista de Figuras

Abreviaturas e Símbolos

ACID	*Atomicity, Consistency, Isolation, Durability*
API	*Application Programming Interface*
BD	Base de Dados
BSON	*Binary JavaScript Object Notation*
CPU	Unidade Central de Processamento
GB	*Gigabyte*
HTTP	*Hypertext Transfer Protocol*
JSON	*JavaScript Object Notation*
KB	*Kilobyte*
MB	*Megabyte*
SGBD	Sistema de Gestão de Bases de Dados
SGBDR	Sistema de Gestão de Bases de Dados Relacional
SQL	*Structured Query Language*
XML	*Extensible Markup Language*
XSLT	*Extensible Stylesheet Language Transformations*

Capítulo 1

Introdução

Este documento descreve o trabalho desenvolvido no âmbito do projecto de dissertação "Bases de Dados alternativas para *Websites*".

Inicialmente pretende-se enunciar as características das plataformas estudadas, com o intuito de se obter uma noção do contexto onde a dissertação se insere. Posteriormente, descrever-se-ão as diferenças entre os sistemas de gestão de bases de dados (SGBD) relacionais e não-relacionais, a base do projecto.

O documento explicita ainda o caminho percorrido na criação de um SGBD não-relacional para comparar com a actual e indica as conclusões a que se chegou.

Este primeiro capítulo expõe o contexto, a motivação e os objectivos do projecto de dissertação.

1.1 Contexto / Enquadramento

Com o surgimento de novas plataformas capazes de competir com os SGBD's existentes, a discussão sobre as vantagens da utilização das bases de dados não-relacionais comparativamente às bases de dados relacionais tem vindo a crescer nos últimos anos.

Apesar de as bases de dados não-relacionais serem apresentadas como alternativas eficientes às bases de dados relacionais, não há estudos nem implementações suficientes para validar este conceito. Existem cada vez mais casos de empresas e serviços que alteram o seu SGBD relacional para um SGBD não relacional, disponibilizando alguns resultados sobre a alteração, mas não existe um estudo aprofundado sobre essas alterações e suas razões.

Assim sendo, foi possível desenvolver um estudo sobre os SGBD's em conjunto com a Janela Digital, empresa responsável pelas plataformas Imoguia e Portal Casa Sapo, que utilizarei como caso de estudo. Como consequência, este trabalho procura indicar

qual deve ser o procedimento adequado e comparar as alternativas existentes, propondo melhorias para as plataformas em questão, sendo que o estudo das SGBD's não será exaustivo.

1.2 Projecto

A contínua expansão dos serviços disponibilizados pelo Imoguia e pelo Portal Casa Sapo implicou um aumento de utilizadores e correspondente volume de dados transaccionados. Em conjunto com a filosofia e objectivos da Janela Digital, - inovar e progredir fornecendo um melhor serviço aos seus clientes - a empresa pretende renovar a sua base de dados, criada em 1999, sendo que esta já foi sofrendo pequenas alterações ao longo dos anos para corresponder às novas necessidades do mercado imobiliário. Com este projecto pretende-se não só verificar se vale a pena optimizar a actual base de dados (BD) mas também averiguar se o surgimento de novas tecnologias na área dos SGBD permite um melhoramento dos serviços.

O projecto coloca em oposição dois estilos de SGBD - o relacional e o não-relacional - permitindo um estudo e a comparação das principais características tendo em conta as especificidades do mercado imobiliário e, mais concretamente, da plataforma Imoguia e do Portal Casa Sapo.

Contrapondo estes dois tipos de SGBD conseguiremos verificar se a actual forma de gestão da BD é a mais vantajosa ou se, por outro lado, a mudança de paradigma e alteração de SGBD permitem melhorar o serviço disponibilizado. Para tal, são estudadas ao pormenor algumas SGBD não-relacionais para seleccionar aquela que permite melhoramentos mais significativos. Posteriormente, na sua implementação, verificaremos se a nova forma de gestão será mais vantajosa para os objectivos do Imoguia e o Portal Casa Sapo.

Uma vez que o projecto se relaciona com *Websites*, as características inerentes ao Portal Casa Sapo têm mais peso nas decisões tomadas, visto que o Imoguia é apenas uma plataforma e não um *website*. Apesar disso, não se pode excluir das decisões as características do Imoguia pois ambos - portal e plataforma - se encontram ligados através da mesma base de dados.

O projecto pretende, além de indicar melhoramentos nas plataformas indicadas, contrapor os diferentes tipos de SGBD, - relacional e não-relacional - indicando se as vantagens anunciadas fazem sentido.

1.3 Motivação e Objectivos

Nos últimos anos tem havido uma crescente discussão sobre a utilização de SGBD não-relacionais comparativamente a SGBD relacionais. Com este ressurgimento apare-

cem empresas e serviços que alteram o seu SGBD relacional para um SGBD não relacional, disponibilizando alguns resultados sobre essa alteração, muitos dos quais apresentando resultados bastante positivos, especialmente pelo desempenho geral do SGBD não-relacional.

O objectivo e a principal motivação deste projecto prendem-se com a inexistência de um grande estudo sobre as diferentes abordagens dos SGBD, havendo apenas resultados e algumas considerações pelas pessoas responsáveis aquando da tomada de decisão na sua substituição, comparando directamente as características, vantagens e desvantagens dos SGBD utilizados.

Este projecto pretende dar a conhecer as diferentes abordagens dentro dos SGBD e, ao mesmo tempo, distingui-las, tendo presente a variedade dos diversos SGBD.

Expondo as diferentes abordagens e características existentes, poderemos escolher a que melhor satisfaz os objectivos delineados pela Janela Digital, assim como o seu desempenho, fiabilidade e escalabilidade. Por fim, com recurso a alguns testes, averiguaremos se as características apregoadas por cada SGBD são de facto condizentes com a teoria e com os que a apregoam através de um estudo sustentado.

Capítulo 2

Estado da Arte

Sendo o objectivo da dissertação verificar a possibilidade de substituição do SGDB existente - *Microsoft SQL Server* - por um SGBD não-relacional, o presente capítulo descreve as características de ambos para fundamentar uma eventual decisão de troca de tecnologia.

Um SGBD tem como principal função gerir todo o processo de manutenção de informação do(s) sistema(s) existente(s). No entanto, as funcionalidades e o tipo de tratamento da informação dependem do SGBD utilizado, como veremos de seguida.

2.1 Sistema de Gestão de Bases de Dados Relacional

Um SGDB relacional como o *Microsoft SQL Server* ou o *MySQL*, actualmente utilizados na Janela Digital, têm como principal diferença relativamente aos restantes tipos de SGBD o facto de estruturarem os seus dados em tabelas e de se basearem em álgebra relacional, o que facilita a sua compreensão. Além disso, um SGDB relacional tem as seguintes funções:

- Manipulação de dados;
- Validação e optimização;
- Integridade e segurança dos dados;
- Concorrência e recuperação de dados;
- Criação e manutenção dos metadados;
- Desempenho.

Como podemos verificar, estes efectuam múltiplas e variadas funções, facilitando a sua utilização e manutenção por parte do utilizador, uma vez que é tudo tratado pelo SGDB[1].

Muitas destas funções são mantidas pelo conjunto de propriedades destinadas a assegurar a confiança nas transacções, conhecidas como ACID (Atomicidade, Consistência, Isolamento, Durabilidade).

- Atomicidade - cada transacção é atómica, se uma parte da transacção falha toda a transacção falha;
- Consistência - as transacções respeitam as regras impostas pelos metadados;
- Isolamento - uma informação só pode ser alterada ou acedida se não houver mais nenhuma transacção a utilizar a mesma informação;
- Durabilidade - o efeito de uma transacção com sucesso é permanente;

As propriedades enunciadas permitem garantir a fiabilidade e consistência de uma BD. Nestes SGBD o isolamento funciona através do bloqueio da informação a ser processada. A transacção marca a informação a ser acedida e o SGDB não permite que outras transacções acedam à mesma informação até que esta termine. Este funcionamento pode levar à sobrecarga do sistema e ao adiamento de outras transacções, (se uma transacção for demasiado complexa) levando a uma diminuição do desempenho do sistema.

Uma das transacções mais utilizadas mas, ao mesmo tempo, dispendiosa em termos de processamento, é a operação JOIN. Devido à sua função de junção de tabelas, é utilizada para relacionar a informação contida em várias tabelas. A operação JOIN é caracterizada por ser um cruzamento de todos os registos das tabelas seleccionadas, seguido de selecções e projecções. O resultado deste cruzamento de informação é normalmente muito maior do que o resultado final do JOIN e por essa razão não se pode desprezar a carga envolvida com a utilização desta operação.

Com esta estrutura compreendem-se as dificuldades dos SGBDR em manipular com elevada eficiência enormes quantidades de dados. O facto de a operação efectuar o cruzamento de todos os dados das tabelas seleccionadas, juntando ao facto de estas tabelas conterem um elevado número de registos é uma preocupação, pois aumenta exponencialmente o número de informação cruzada e trabalhada[1, 2, 3, 4].

A sua estrutura faz com seja necessário normalizar os dados de modo a evitar redundâncias e incongruências. Assim sendo, é necessário efectuar um estudo e uma preparação prévia para optimizar a BD e obter o melhor desempenho possível. Por outro lado, esta estrutura rígida faz com qualquer alteração seja de elevada dificuldade para a manutenção do mesmo desempenho. Muitas vezes será necessário reestruturar a BD novamente com os custos e tempo inerentes à mesma.

O facto de as BD relacionais serem a referência na área e utilizadas nos mais diversos ambientes faz também com que sejam muito testadas e com que tenham um maior suporte através da comunidade existente, bem como da empresa responsável.

As BD relacionais não foram desenhadas para serem distribuídas, sendo esta a chave para a escalabilidade, uma vez que os dispositivos de armazenamento se encontram incorporados num CPU comum e não distribuídos pelos nós do sistema[5]. Devido a esta arquitectura, a escalabilidade é um problema para sistemas em constante crescimento, pois a alteração centra-se na adição de recursos (CPU ou memória) a um nó do sistema. Isto advém do facto de haver vários nós com funções completamente distintas. Separadamente, existem os nós de escrita - reconhecidos por serem os pontos de ruptura do sistema - e os nós de leitura - associados ao espaço físico utilizado[2, 6, 7].

2.2 Bases de Dados Não-Relacional

Estes SGBD são conhecidos por NoSQL, que significa "*Not Only SQL*", salientando o facto de não conterem um esquema fixo, principal característica das BD relacionais. Estas não se relacionam por ligações nas tabelas, (porque o esquema deixa de ser em tabelas) mas continuam a existir relações entre as entidades, embora não explicitas e truncadas como nos SGBDR. Estas relações são constituídas de outra forma e não explicitamente na base de dados como nas relacionais.

Nos últimos dois anos, - 2009 e 2010 - com o ressurgimento das BD não-relacionais em grande escala, aumentou a discussão sobre as vantagens e desvantagens destas, em contraposição com as BD relacionais.

Antes de qualquer tipo de comparação, convém referir que os SGBD não-relacionais contemplam, independentemente da plataforma utilizada, as seguintes vantagens:

- Alta escalabilidade;
- Alta disponibilidade;
- Alto desempenho (mesmo com enormes quantidades de dados);
- Flexibilidade;
- Software livre (sem custos).

Convém salientar o facto de se indicar o software livre e sem custos como uma vantagem dos SGDB não-relacionais, mas a característica não é exclusiva destas. Existem vários SGBDR que são reconhecidas pela sua qualidade e que também não contemplam custos para o utilizador, como o PostgreSQL e o MySQL. Por não se colocar essa questão, não haverá referências ao custo da sua utilização, contrariamente às SGBDR.

O principal fundamento para estas vantagens - independentemente do SGDB - é o facto da lógica de validação, do controlo de acessos, do mapeamento de dados indexados, do correlacionamento dos dados, da resolução de conflitos e dos procedimentos desencadeados por *triggers* passarem da camada de base de dados para a camada da aplicação. Desta forma, a camada da BD é deixada unicamente com preocupações de desempenho e escalabilidade[8].

Por outro lado, o facto de todos os controlos serem feitos na camada de aplicação e não pelo SGDB, coloca em necessidade o seguinte:

- Atenção especial para a inconsistência nos dados
- Gestão de conflitos

O facto da validação e resolução de conflitos passar para a camada da aplicação leva a uma preocupação extra essencialmente por parte de quem desenvolve a aplicação. Com a implementação de uma SGBD não-relacional é necessário garantir que estas características são asseguradas ou então existe o risco de a BD ser inconsistente e incongruente, não sendo a mesma fiável.

Existem vários tipos de modelos de armazenamento de dados utilizados pelas BD não-relacionais: o armazenamento por valores das chaves, por documentos e por colunas.

Enquanto que o armazenamento por valores das chaves surge como um modelo extremamente rápido e eficiente pela sua simplicidade, o armazenamento de colunas providencia um modelo flexível e extremamente estruturável. Por sua vez, o armazenamento por documentos combina a simplicidade das BD não-relacionais com uma maior estruturação[9].

Outra das diferentes abordagens aos SGBD não-relacionais tem a ver com o local onde a informação é armazenada e disponibilizada. Existem algumas que guardam toda a informação em memória e, como tal, é extremamente rápida nas respostas mas tem o problema de a informação não ser durável. Por outro lado, existem aquelas que guardam toda a sua informação em disco, utilizando o algoritmo de árvores B+[1] para uma indexação eficiente dos registos.

Existem ainda os que são configuráveis, permitindo definir o tamanho da Memtable, memória volátil, e providenciando assim um controlo absoluto sobre o mesmo.

Sendo o objectivo do projecto a passagem de um SGBD relacional para não-relacional, é necessário descrever os SGBD estudados. O critério de escolha das 2 primeiras SGBD - MongoDB e CouchDB - surge a pedido da Janela Digital. O estudo do Cassandra foi uma escolha pessoal sustentada pela importância que esta teve nos últimos tempos, tanto pela demonstração de eficiência na manipulação de elevadas quantidades de informação, - demonstrado pela sua utilização em serviços como o Twitter e Facebook - como pela

[1] http://www.mec.ac.in/resources/notes/notes/ds/bplus.htm

diferente forma de manipulação dos dados. Não foi seleccionada nenhuma SGDB com o modelo de dados com armazenamento pelo valor da chave (*key-value store*) pois as mesmas não permitem a estrutura complexa necessária para o projecto em questão.

2.2.1 MongoDB

Desenvolvido pela 10gen[2] desde Outubro de 2007, este baseia-se em guardar informação em documentos num estilo BSON (Binary JSON). O esquema permite a utilização de estruturas de dados complexas, como na figura 2.1[10] mas, como podemos observar, é semelhante a uma BD relacional, o que facilita a sua utilização e compreensão para quem está habituado a um esquema de dados relacionais.

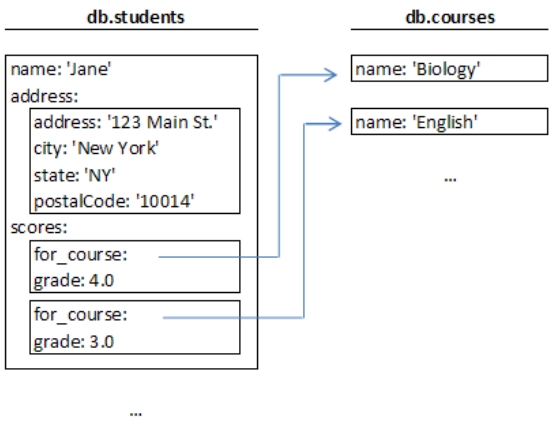

Figura 2.1: Exemplo da estrutura de um ficheiro com MongoDB

Como referido anteriormente, estas estruturas podem ser algo complexas por conseguirem situar-se entre os modelos de armazenamento de chaves - extremamente rápidos e altamente escaláveis - e os tradicionais SGBDR - que providenciam amplos tipos de consultas e profundidade nas mesmas. Podemos dizer que o modelo de dados armazenado em documentos é parecido com as BD relacionais, indo buscar as características mais vantajosas às BD não relacionais.

Além deste poderoso suporte às consultas, característica do SGBDR, a utilização de BSON habilita a indexação de qualquer tipo de dados, proporcionando o melhoramento do desempenho relativo às consultas. As consultas à BD são das operações mais utilizadas e, se as mesmas forem complexas, podem ter um elevado custo de processamento.

Não existindo indexação dos dados a manipular é sempre necessário percorrer a tabela coluna por coluna, no caso do modelo relacional. No caso do MongoDB será necessário percorrer toda a estrutura existente, sendo que quanto maior for a estrutura, maior será o tempo de processamento despendido na mesma. A indexação de uma dada informação cria um ficheiro com o seu valor da chave de procura e um apontador para o valor

[2] http://www.10gen.com

contido na entidade. Estes ficheiros de indexação geralmente são mais pequenos que a estrutura em si e, como tal, necessitam de menos tempo para a sua leitura, visto que não é obrigatório que o processo leia todo o ficheiro de indexação (caso encontre antes o pretendido)[11].

Outros dos motivos para uma rápida consulta deve-se ao facto de o MongoDB pré-alocar o espaço do ficheiro, prevenindo a fragmentação do mesmo e garantindo assim um armazenamento compacto e eficiente. Por outro lado, os ficheiros da base de dados são divididos em ficheiros mais pequenos por tamanhos pré-definidos; o primeiro contém 64MB, o segundo 128MB e assim sucessivamente até aos 2GB. Havendo necessidade de mais ficheiros para a BD, o sistema continuará a utilizar ficheiros com tamanho de 2GB. Este comportamento revela-se importante para a utilização em BD com elevados volumes de dados porque, não deixando os ficheiros fragmentados nem com demasiada informação, não perde desempenho ao ter de fazer leituras em várias zonas do disco, ao mesmo tempo que a ligação dos ficheiros se encontra bem definida.

Além de suporte para grande volumes de dados, o MongoDB providencia, através do protocolo *GridFS*, um mecanismo para armazenar grandes objectos eficientemente, visto que um objecto no formato BSON é limitado a 4MB e dividido em vários documentos. A ideia para uma eficiente ligação dos documentos é que cada um deles tenha um objecto com os seus metadados referenciados numa colecção de ficheiros, e esta contenha a lista de documentos e a ordem de cada objecto.

De salientar que o MongoDB suporta vários controladores, como o C, C++, Java, JavaScript, Perl, PHP, Phyton e Ruby. Além disso tem inúmeros controladores suportados pela comunidade.

O controlador de C# é desde 15 de Outubro suportado oficialmente, sendo que a última versão, a 0.9.0.3992, foi lançada a 6 de Dezembro. Uma vez que sua oficialização é extremamente recente e ainda não existe uma versão final, - versão 1.0 - ainda não existe uma API (Application Programming Interface) nem muita informação nos fóruns sobre a utilização correcta do controlador. De salientar que em 3 meses foram lançadas 3 versões (0.5, 0.7 e 0.9) aproximando-se uma versão definitiva, como indiciam as alterações que se encontram em curso[3].

Para finalizar, o MongoDB é suportado por uma enorme comunidade disposta a ajudar, quer seja por IRC[4] ou em fóruns. Além disso, a empresa 10gen conta com um serviço de suporte a avarias e dúvidas, providenciando também formação e apoio logístico.

Existe um contínuo desenvolvimento e uma rápida correcção de erros, como podemos ver no sítio[5][6] onde a empresa anuncia o lançamento de novas versões e correcção de erros.

[3] https://github.com/mongodb/mongo-csharp-driver
[4] irc://irc.freenode.net/#mongodb
[5] http://groups.google.com/group/mongodb-announce
[6] http://www.mongodb.org/downloads

No espaço de sete meses - Março a Setembro - foram lançadas 3 novas grandes versões, da 1.4 à 1.6.5.

A utilizar este serviço encontram-se grandes empresas e serviços como a SourceForge[7], o GitHub[8], o CollegeHumor[9] e o The New York Times[10], empresas de prestígio e que garantem a continuidade do projecto. Esta actividade é importante para a Janela Digital pois necessita de continuidade e desenvolvimento do MongoDB para assegurar qualidade dos seus produtos e para que este se mantenha ao longo dos próximos anos.

2.2.2 CouchDB

O CouchDB tem o mesmo conceito de armazenamento que o MongoDB, - o armazenamento por documentos - mas no formato JSON. Assim sendo, tem as mesmas vantagens; a estruturação complexa dos dados e a similaridade com as BD relacionais, o que torna fácil a sua compreensão.

Já em relação à sua arquitectura e desempenho, as estruturas são diferentes, como podemos ver na Figura 2.2. Esta plataforma assenta numa arquitectura *Representational State Transfer*, também conhecido por REST.

O REST assenta a sua arquitectura em 6 pontos essenciais:

- Servidor/Cliente - existe a separação entre servidores e clientes. O armazenamento de dados é da preocupação dos servidores enquanto que os clientes ocupam-se com o interface e estado dos utilizadores. Desta forma, garante-se portabilidade do código e escalabilidade;
- Sem estado - cada pedido do utilizador contém toda a informação necessária para obter a resposta do serviço pretendido sendo que o seu estado é mantido pelo próprio utilizador;
- Com *cache* - desta forma previne a utilização de informação obsoleta e inapropriada nas respostas a pedidos efectuados.
- Sistema por camadas - com a existência de servidores intermédios pode-se aumentar a escalabilidade do sistema utilizando sistemas de balanceamento de carga e providenciando *caches* conjuntas, ao mesmo tempo que pode melhorar a segurança do sistema;
- Interface uniforme - a existência de um interface uniforme entre os clientes e servidores simplifica e dissocia a arquitectura, permitindo que cada parte consiga ser desenvolvida independentemente garantindo portabilidade;

[7] http://sourceforge.net
[8] http://github.com
[9] http://www.collegehumor.com
[10] http://www.nytimes.com

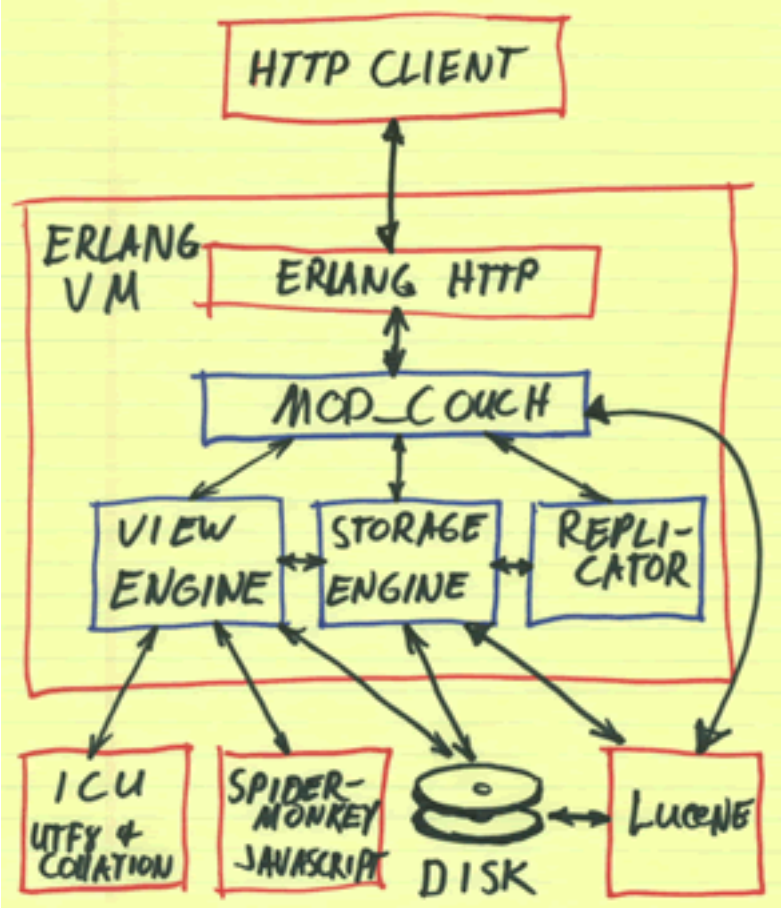

Figura 2.2: Arquitectura CouchDB

- *Code on demand* (opcional) - servidores podem costumizar a funcionalidade de um cliente transferindo código que este consegue executar.

Com excepção do *code on demand*, todas essas características são obrigatórias para um sistema ser considerado REST.

A utilização desta arquitectura possibilita o acesso à BD através de qualquer ambiente que permita pedidos HTTP. Aliás, o REST utiliza os métodos HTTP - `POST`, `GET`, `PUT` e `DELETE` - para as operações CRUD (Create, Read, Update e Delete) em todos os recursos. Isto é possível porque cada item contém um URI (Uniform Resource Identifier) que permite o acesso através de HTTP [12, 13].

A possibilidade de efectuar qualquer tipo de operação através de HTTP é uma vantagem pois este é um protocolo amplamente utilizado, compreendido e testado, além de ser interoperável, possibilitando a utilização do CouchDB em qualquer ambiente e com qualquer linguagem de programação que permita pedidos HTTP. Além do uso do protocolo, o CouchDB consegue obter vantagens com o HTTP por existirem inúmeras ferramentas de *caching*, *proxying* e para equilíbrio de carga que permitem melhorar o serviço disponibilizado.

Por outro lado, a utilização da arquitectura REST e, consequentemente, a utilização do protocolo HTTP, limitam o desempenho do sistema pois criam muitos pedidos - do utilizador para o cliente, do cliente para o servidor, do servidor para o cliente e do cliente para o utilizador final. Este sistema perde rapidez de resposta para outras arquitecturas bem mais directas e sem tantos entraves na execução de pedidos.

O CouchDB dispõe, como qualquer BD relacional, de um sistema de controlo de semântica ACID, só que implementa este sistema através de *Multi-Version Concurrency Control* (MVCC), conseguindo assim trabalhar com elevadas quantidades de pedidos concorrentes sem conflitos.

Com este sistema, o bloqueio de informação só acontece a operações do mesmo tipo. Isto significa que uma transacção de escrita só bloqueia transacções de escrita mas é possível que uma transacção de leitura obtenha a informação pretendida sem esperar que a mesma seja alterada. A situação inversa também se verifica[14, 15].

Para finalizar, o suporte é feito por terceiros - a Couchio[11] e Cloudant[12]. Já em termos de comunidade, esta encontra-se amplamente activa e sempre disposta a ajudar. A Apache, empresa responsável, apenas disponibiliza alguns livros para a sua compreensão.

Esta é uma SGBD utilizada pela BBC[13] e pela IBM[14], sinónimo de continuidade e melhoramentos no projecto. Actualmente, o CouchDB vai na versão 1.0.1, actualização de Agosto de 2010. Pelo histórico existente sai em média uma actualização de 6 em 6 meses.

2.2.3 Cassandra

Tem uma ampla e conceituada utilização em diversos serviços que envolvem manipulação de enormes quantidades de dados. O facto de ter uma abordagem e características diferentes dos dois sistemas anteriores, em particular no armazenamento por colunas, fez com que se levasse a mesma em consideração para um estudo mais aprofundado.

O armazenamento por colunas faz com que cada família de colunas - o correspondente a uma tabela no modelo relacional - esteja em ficheiros separados. As colunas relacionadas, as que se devem aceder em conjunto, mantêm-se na mesma família de colunas por uma questão de eficiência. Todos os ficheiros encontram-se representados no formato JSON.

Este tipo de armazenamento possibilita um suporte para estruturas de dados complexos, como no exemplo a seguir:

[11] http://www.couch.io
[12] https://cloudant.com
[13] http://www.bbc.co.uk
[14] http://www.ibm.com

```
{
    "mccv":{
        "Users":{
            "emailAddress":{"name":"emailAddress", "value":"foo@bar.com"},
            "webSite":{"name":"webSite", "value":"http://bar.com"}
        },
        "Stats":{
            "visits":{"name":"visits", "value":"243"}
        }
    },
    "user2":{
        "Users":{
            "emailAddress":{"name":"emailAddress", "value":"user2@bar.com"},
            "twitter":{"name":"twitter", "value":"user2"}
        }
    }
}
```

Figura 2.3: Exemplo de estrutura de uma família de colunas com Cassandra

Através da imagem 2.3 podemos verificar que a chave "`mcc`" identifica dois tipos diferentes de famílias de colunas: os "`Users`" e "`Stats`". O facto de haver dados na mesma chave de diferentes famílias de colunas é inteiramente da responsabilidade da aplicação e não significa que estas estejam relacionadas. Com este SGBD é possível e completamente válido que uma família de colunas tenha nomes distintos. Isto acontece na família de colunas "`Users`", em que o "`mccv`" e o "`user2`" têm diferentes nomes de colunas.

Para quem vem dos SGBDR estas características são novas. Como tal, o Cassandra aparenta no seu início uma baixa curva de aprendizagem, até os novos conceitos serem completamente assimilados.

Outra das principais características do Cassandra é o facto de o sistema funcionar como um serviço de rede distribuída, havendo quem considere que a plataforma não é uma BD mas sim um conjunto de nós de BD que formam um sistema de rede distribuída[16]. Considerando a plataforma como um serviço de rede distribuída, é fácil identificar algumas das vantagens inerentes a esta condição.

Uma destas vantagens é a tolerância a falhas pois neste tipo de arquitectura não existe um ponto crítico susceptível às mesmas, uma vez que todos os nós do sistema são idênticos. A informação é replicada automaticamente para todos os nós do sistema para prevenir falhas e permitir a consistência dos dados.

Outra das vantagens tem a ver com o desempenho de escrita e leitura na BD. Mais uma vez, todos os nós são tratados como iguais e cada uma das operações é direccionada para o nó mais próximo e disponível, de forma a obter a melhor resposta no mais curto espaço de tempo. Esta disposição faz com que não haja sobrecarga de um dos nós mas sim uma ampla utilização de todos os nós.

Com estas condições a escalabilidade é assegurada, bastando acrescentar um nó ao sistema sem ser preciso haver qualquer tipo de interrupção do serviço. A capacidade de

resposta do sistema depende do número de máquinas que o sistema contém, sendo que este aumenta linearmente com a adição de novas máquinas. A eficiência do sistema, além do número de máquinas, depende do sistema de *routing* existente[17].

Ao contrário de muitos SGBD esta plataforma é completamente configurável nos seus mais diversos parâmetros. Um dos que mais se destaca é o facto de ser permitido alterar a quantidade de memória volátil utilizável, sendo que por defeito utiliza 128MB[2].

Esta utilização da memória volátil pode ser importantíssima, isto porque, numa operação de escrita, a plataforma utiliza a memória volátil e posteriormente coloca a informação em disco. O processo de escrita em memória é mais rápido do que o processo de escrita em disco, poupando assim tempo precioso para outras operações. Quando o limite de memória é excedido, esta é limpa.

O Cassandra utiliza a *framework* Thrift como API para os seus clientes ou permite utilizar directamente alguns tipos de linguagem suportados. Esta *framework* suporta C++, Java, Python, PHP, Ruby, Erlang, Perl, Haskell, C#, Cocoa, Smalltalk e OCaml. No entanto, não será necessário recorrer a esta framework, uma vez que as linguagens C# e .NET poderão ser utilizadas directamente através da utilização de um dos seguintes clientes:

- Aquiles [15]
- Hector Sharp [16]
- Fluent Cassandra [17]

Para finalizar, verifica-se que esta plataforma contém uma enorme comunidade e inúmeras empresas a utilizar, como o Facebook, o Digg, a Cisco e o Twitter. Como tal, prevê-se a continuação e melhoramento do Cassandra que neste momento se encontra na versão 0.7.0, disponibilizada a 9 de Janeiro de 2011. Em média, o Cassandra tem actualizações quase todos os meses.

A empresa responsável, a *Apache*[18], coloca toda a responsabilidade do suporte técnico profissional à *Riptano*[19].

2.3 Conclusões

Verifica-se que, em termos teóricos, os SGBD não-relacionais são mais rápidos graças à sua arquitectura, que atribui menos tarefas para o SGBD, deixando a validação e concorrência de dados para a camada de aplicação. Também se verifica que, nos SGBD

[15] http://aquiles.codeplex.com
[16] http://www.hectorsharp.com
[17] http://github.com/managedfusion/fluentcassandra
[18] http://www.apache.org
[19] http://riptano.com

não-relacionais, se encontram características que potenciam a escalabilidade e o seu desempenho.

Não foram estudados SGBD com armazenamento por valores das chaves porque são extremamente simples e não resolviam eficientemente o problema relacionado com o Imoguia e o Portal Casa Sapo.

Tendo em atenção o facto de as plataformas serem feitas em C# e .NET e terem como preocupação a escalabilidade, fiabilidade e continuação do projecto, todos os SGBD estudados - MongoDB, CouchDB e Cassandra - podem ser utilizados.

Como a principal preocupação da Janela Digital é garantir um óptimo desempenho dos seus serviços, verificamos que o CouchDB não suprime essa necessidade devido às suas fracas respostas comparativamente com os outros SGBD estudados. Tal sucede devido à quantidade de pedidos efectuados e consequente demora nas respostas. Tendo o MongoDB melhores respostas e com características semelhantes, o CouchDB é descartado por não satisfazer o principal requisito, o desempenho.

Como tal, a decisão prende-se entre o Cassandra e o MongoDB. O Cassandra apresenta vantagens em termos de escalabilidade, desempenho e flexibilidade, pelo facto de ser um serviço de rede distribuída e por potenciar uma estrutura de dados complexa.

Por outro lado, o MongoDB tem como vantagem o facto de ser parecido com os SGBDR, com as vantagens de utilização de um SGBD não-relacional e o facto de o armazenamento e estrutura ser mais condizente com a complexidade e necessidades do Imoguia e do Portal Casa Sapo. Como tal, a implementação para comparação com o actual SGBD será feita com o MongoDB.

Capítulo 3

Implementação

Depois do estudo dos SGBDs existentes, este capítulo apresenta os detalhes da implementação do projecto. Esta encontra-se separada em quatro partes distintas.

Primeiro é apresentado o estudo das plataformas e sua estrutura actual, de forma a perceber o seu funcionamento e as suas características. Posteriormente são indicadas as decisões relativamente à constituição da estrutura da base de dados para o MongoDB. Após a constituição da nova estrutura obtêm-se os dados suficientes para a migração de informação e é explicado todo o processo de transformação da mesma. O capítulo finaliza com a apresentação e conclusão dos resultados obtidos com os testes efectuados.

3.1 Contexto actual

Para um estudo completo antes da implementação do novo sistema de gestão de bases de dados, é necessário efectuar um estudo aprofundado das plataformas e da estrutura existente para uma melhor percepção da sua utilização.

O estudo das ferramentas Imoguia e do Portal Casa Sapo permite visualizar as tarefas e acções mais utilizadas, permitindo assim perceber quais serão as informações com maior carga de utilização. Com este estudo obtém-se uma percepção das operações mais utilizadas, dando assim lugar, à escolha dos testes a efectuar mais convenientes.

3.1.1 Plataformas

A aplicação Imoguia e o Portal Casa Sapo são duas importantes ferramentas na pesquisa e divulgação de imóveis.

O Imoguia é a plataforma que a Janela Digital disponibiliza para os seus agentes imobiliários. Esta plataforma permite aos profissionais do sector imobiliário centralizar a

gestão dos imóveis, dos proprietários e clientes. Tendo toda esta informação focalizada numa única plataforma existe um maior dinamismo e eficácia no trabalho efectuado.

A plataforma, cuja janela principal é apresentada na Figura 3.1, tem como principais objectivos:

- a gestão do dia-a-dia;
- o marketing da empresa imobiliária;
- publicação automática de produtos e serviços na Internet.

Figura 3.1: Software Imoguia

A gestão da carteira de imobiliário e de contactos é bastante crítica para a actividade, por isso, é importante assegurar a fiabilidade destas informações. Esta aplicação pode funcionar num modo *offline* e, como tal, é necessário garantir que as todas as alterações passem correctamente para o servidor, que se encontra *online*. Além da parte de gestão, este contém vários links externos para estatísticas da empresa, como contagem e percentagens de imóveis de certas regiões.

Além da gestão dos seus contactos, a plataforma também possibilita a gestão da agenda e reuniões, a actualização automática da página *web* e a promoção em vários portais, - nacionais como internacionais - sendo a única que permite a exportação da carteira de imóveis para o Portal Casa Sapo[1], o portal imobiliário líder em Portugal.

A agenda e a facturação não fazem parte da estrutura estudada pois encontram-se num sistema diferente, apesar de se encontrarem dentro da mesma aplicação.

Já o Portal Casa Sapo pode ser acedido por qualquer pessoa, sendo direccionado exclusivamente para a divulgação e pesquisa. Como tal é extremamente importante a eficiência desta plataforma na suas pesquisas.

[1] http://casa.sapo.pt

Figura 3.2: Página inicial do Portal CasaSapo.pt

Como podemos verificar na Figura 3.2, utiliza-se o portal português CasaSapo.pt. Na parte superior encontram-se várias secções separadas pelo tipo de imóvel e de negócio, como os imóveis, arrendamento e empreendimentos. Cada separador existente direcciona para uma outra página onde na sua maioria existe numa posição privilegiada - no topo da página - a pesquisa avançada.

O que chama mais a atenção das pessoas é a existência do mapa de Portugal, que se direcciona a pesquisa pela localização do imóvel. É um aspecto importante a salientar visto que as pessoas, na maioria dos casos, sabem qual a zona para a pesquisa de imóveis.

Com a utilização do Portal verifica-se que é recorrente a utilização de imagens, por isso, é importante a sua indexação visto que um elevado número de imagens aumenta a carga de utilização da BD.

Para conseguir manter todas estas informações disponíveis para as empresas imobiliárias, a Janela Digital precisa de manter a sua extensa base de dados com uma resposta rápida e sempre disponível. Actualmente a empresa utiliza o *Microsoft SQL Server 2005*

versão 64 *bits* para processar e gerir toda a informação, recorrendo à *Framework 2.0* em *C#* e *.NET*. Na versão mais básica do Imoguia é utilizado o *MySQL* como elo de ligação à BD principal.

Para se ter uma ideia da complexidade da base de dados é apresentado o diagrama representativo da base de dados existente, na Figura 3.3, e também a estrutura utilizada pelo Portal Casa Sapo na Figura 3.4.

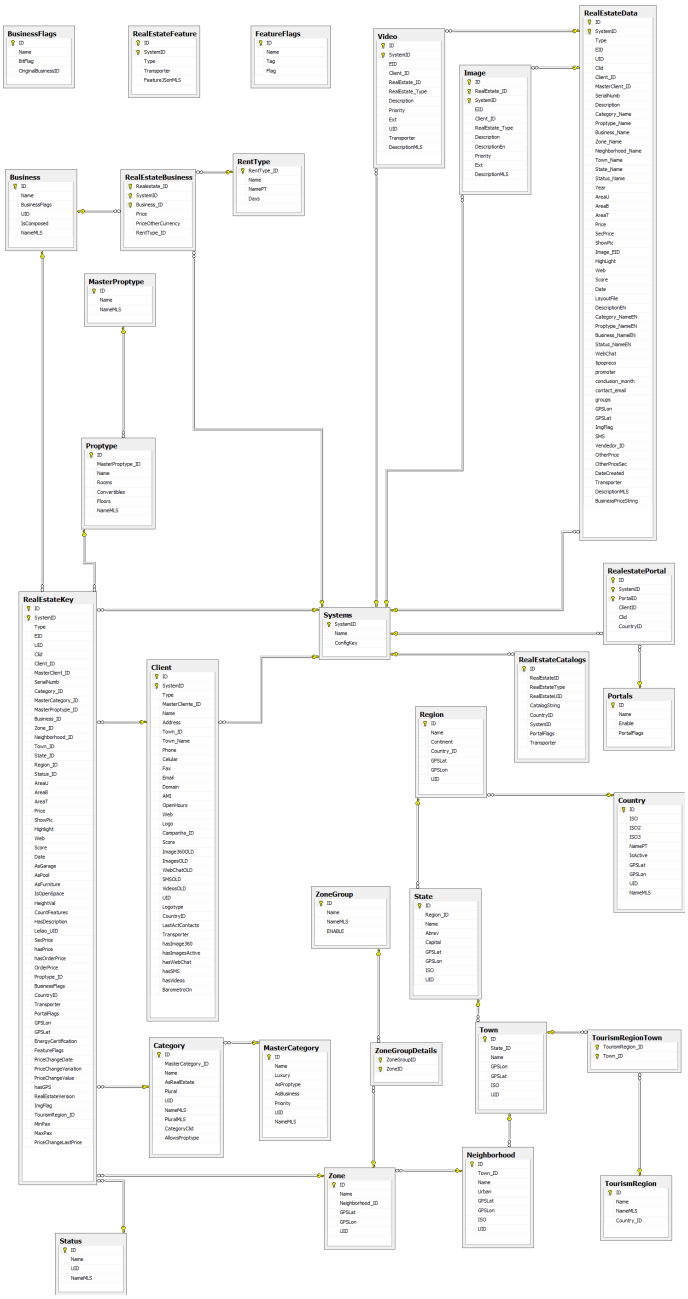

Figura 3.3: Estrutura da base de dados

Esta base de dados foi concebida há 10 anos, em 1999, - aquando da criação do produto - mas foi sendo alterada ao longo dos anos para suprir necessidades que não foram

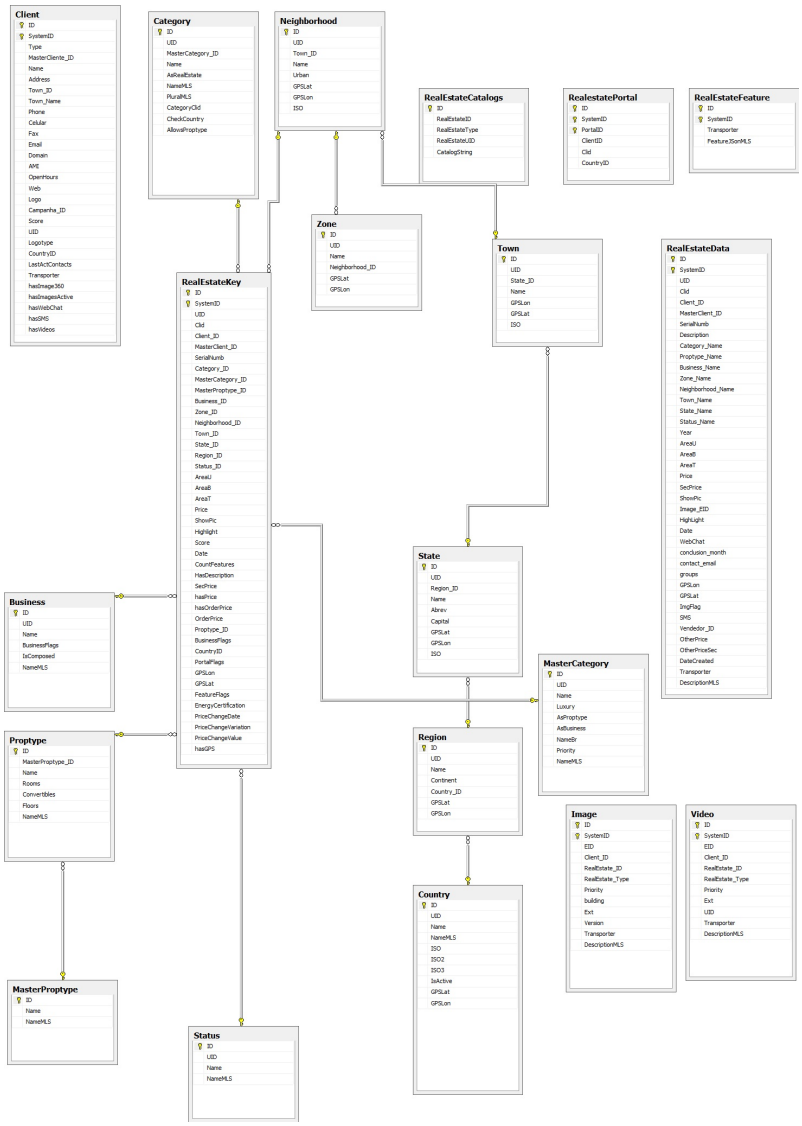

Figura 3.4: Estrutura da base de dados do Portal Casa Sapo

inicialmente idealizadas. Estas pequenas alterações levaram a algumas incongruências e redundâncias, o que equivale a dizer que a BD não se encontra optimizada nem a funcionar com o desempenho pretendido.

Apesar de se trabalhar com a BD completa, Figura 3.3, a plataforma Portal Casa Sapo será um peso superior nas decisões tomadas, visto que é o *website* e tem uma maior utilização do que o Imoguia.

Como se pode ver na Figura 3.4, a BD não é muito complexa, podendo ser considerada uma BD de média dimensão. Tem uma enorme carga de utilização devido ao elevado número de utilizadores existentes bem como uma grande quantidade de imagens de qualidade existentes por cada imóvel, como podemos ver na Figura 3.5:

Para conseguir dar uma resposta eficiente, a Janela Digital tem as entidades das principais pesquisas guardadas em *cache*. Sendo que a procura por uma outra entidade leva à

Table Name	# Records	Reserved (KB)	Data (KB)	Indexes (KB)	Unused (KB)
dbo.Business	16	16	8	8	0
dbo.Category	207	144	120	16	8
dbo.Client	10.485	3.248	2.784	336	128
dbo.Country	23	16	8	8	0
dbo.Image	5.022.561	2.394.744	861.816	1.531.056	1.872
dbo.MasterCategory	10	16	8	8	0
dbo.MasterProptype	7	16	8	8	0
dbo.Neighborhood	64.432	6.928	5.856	928	144
dbo.Proptype	64	80	24	16	40
dbo.RealEstateCatalogs	424.217	738.744	684.264	49.232	36.792
dbo.RealEstateData	424.217	1.063.072	1.018.528	42.200	2.344
dbo.RealEstateFeature	424.217	3.065.760	3.013.640	3.568	48.552
dbo.RealEstateKey	424.217	402.520	138.160	263.568	792
dbo.RealestatePortal	818.890	39.696	39.480	160	56
dbo.Region	17	16	8	8	0
dbo.State	113	32	16	16	0
dbo.Status	9	16	8	8	0
dbo.Town	6.262	528	464	16	48
dbo.Video	8.724	2.696	1.464	1.176	56
dbo.Zone	36.081	5.520	3.032	2.320	168

Figura 3.5: Carga DB

necessidade de uma pesquisa profunda na base de dados, com elevados custos de desempenho pois necessita de percorrer a BD.

3.1.2 Estudo da BD

Após o estudo das plataformas entende-se a utilização dada às mesmas, ao mesmo tempo, que se obtém uma noção mais abrangente destas. Depois deste efectuado este estudo, analisou-se a estrutura para se obter uma percepção mais aprofundada do projecto existente, de forma a compreender detalhadamente todos os seus pormenores.

A primeira coisa que se verifica é que a base de dados é algo complexa, como podemos ver na Figura 3.3. Contém 30 tabelas, muitas relações, muitos atributos em várias tabelas e existem várias tabelas "mãe". A base de dados pode apresentar problemas de eficiência em pesquisas profundas se esta não estiver optimizada.

Num primeiro olhar a tabela `Systems` é a tabela principal pois encontra-se ligada a grande parte das outras tabelas. Isto acontece pois a Janela Digital dispõe de vários serviços ligados entre si e é necessário fazer essa distinção. Mas as tabelas críticas deste sistema são as `RealEstateKey` e a `RealEstateData`.

Estas tabelas encontram-se divididas apesar de terem os mesmos registos devido à complexidade das pesquisas. O acréscimo da complexidade leva a uma diminuição da eficiência do sistema. A tabela `RealEstateKey` encontra-se optimizada para as pesquisas, por isso contém os identificadores de outras tabelas, enquanto que a tabela `RealEstateData` encontra-se optimizada para mostrar os detalhes de cada imóvel, contendo os nomes dos campos e não os seus identificadores.

Juntamente com as tabelas `RealEstateKey` e `RealEstateData`, as tabelas `RealEstateFeature` e `RealEstateCatalogs` também contêm o identificador em comum, ou seja, existe um

registo por imóvel em cada uma destas quatro tabelas. Visto que são muitas tabelas diferentes, e como o imóvel é a essência desta estrutura, é necessário garantir consistência dos dados.

A localização de um imóvel encontra-se extremamente estruturada, contendo oito tipos de locais distintos, - `ZoneGroup`, `Zone`, `Neighborhood`, `Town`, `TourismRegion`, `State`, `Region` e `Country` - e apesar de ser bastante perceptível estruturalmente, pode ser bastante complexo pesquisar um imóvel e obter a sua localização completa. Felizmente as tabelas referentes a imóveis encontram-se optimizadas e têm o identificador ou o nome de cada tipo de local.

Como sabemos, a BD foi sofrendo alterações ao longo do tempo e isso nota-se em alguns campos que já não são utilizados, como os atributos com a terminologia *OLD*.

Relativamente à indexação nota-se uma preocupação com o tipo de cliente - respeitante à diferenciação do tipo de clientes no Imoguia - e na categoria e tipo de negócio dos imóveis na RealEstateKey - a tabela utilizada para pesquisa de imóveis. Como seria de esperar, todas as chaves primárias encontram-se indexadas. Assim sendo, os identificadores, parte importante de pesquisa, encontram-se indexados, de modo a permitir uma pesquisa mais rápida.

Em termos de consumo de recursos, a BD actual, ocupa em disco cerca de 22,5GB, Figura 3.6. A maioria do espaço ocupado advém do registo de acções, pois o armazenamento dos dados em si é aproximadamente 1,25GB, como podemos ver na Figura 3.7.

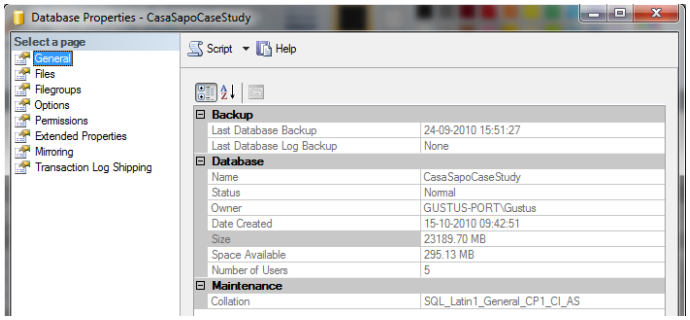

Figura 3.6: Espaço em disco SQL - Total

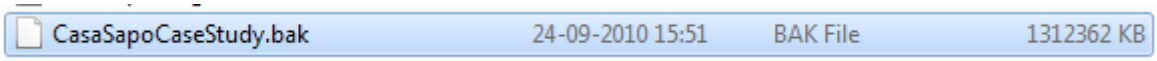

Figura 3.7: Espaço em disco SQL - Dados

Já em termos de utilização de memória e CPU, verifica-se que tem vários serviços inerentes à sua utilização. Como se pode comprovar na Figura 3.8, as imagens representam os recursos utilizados durante um dos testes. Não sendo muito dispendiosa a sua utilização, ainda consome alguns recursos.

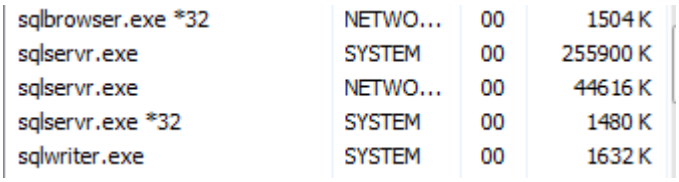

Figura 3.8: Utilização memória e CPU - SQL

3.2 Construção da estrutura para noSQL

Com a revisão dos requisitos e da estrutura actual, é possível conceptualizar a nova base de dados para o SGBD não-relacional. De salientar que a concepção da nova estrutura foi dificultada pois o seu funcionamento e modelamento dos SGBD é completamente diferente do que se estuda nos dias de hoje.

Já é complicado compreender um modelo de concepção completamente distinto mas torna-se mais difícil quando se faz uma migração do sistema. Visto que constantemente somos confrontados com a concepção do modelo relacional dificulta a percepção e respectiva implementação do modelo não-relacional. Seria bastante mais fácil implementar uma nova aplicação sem antecedentes relacionais.

Foi decidido não implementar uma estrutura ideal para protecção da consistência da informação na sua migração. Assim sendo foi necessário recorrer a um sistema híbrido para suportar e contemplar a consistência da informação existente com o actual sistema. Para tal, é necessário preservar os identificadores e trabalhar através deles.

Numa estrutura não-relacional ideal utilizaria-se o atributo *_id*, para qualquer tipo de ligação mas o mesmo só é criado após a inserção da informação, ou então o valor do atributo. Por isso, a estrutura proposta é uma mistura entre BD relacional e não-relacional.

Sendo que a tabela `RealEstateData` e a `RealEstateKey` são das tabelas mais importantes foi necessário assegurar que as mesmas mantém uma performance elevada.

Por isso, houve a dúvida de aglomeração das tabelas `RealEstateData` e `RealEstateKey` com todas as informações sobre os imóveis numa única entidade ou a manutenção de duas entidades separadas optimizadas - uma para pesquisa outra para mostrar informação sobre o imóvel. Decidiu-se manter duas entidades separadas pois uma entidade conjunta teria muitos campos, e como é recomendado que cada entidade não seja muito grande optou-se por manter esta separação.

De salientar que as entidades `Video` e `Image`, especialmente esta última, são bastantes importantes para a eficiência do Portal Casa Sapo pois contêm um elevado número de registos e são constantemente utilizados.

Apesar de a estrutura ter sido adaptada às necessidades da plataforma, as entidades principais continuam as mesmas, e como tal, os índices permanecem quase os mesmos. Numa BD não-relacional não existe o conceito de chave primária, estas na BD actual

praticamente só cobrem os identificadores de cada uma. Ao inserir dados no MongoDB os identificadores únicos `_id` são automaticamente indexados.

Com estas características e necessidades para assegurar o bom funcionamento da aplicação chegou-se à estrutura descrita nas secções seguintes.

3.2.1 Entidade Business

Esta entidade, representada na Tabela 3.1, indica os tipos de negócio a que um imóvel se propõe. É necessária uma entidade própria para preservação dos tipos de negócio.

Esta entidade mantém todas as características da tabela homónima. Perdeu apenas o UID.

Além da indexação pelo identificador único - `_id` - esta, equacionou-se a indexação do identificador - `ID` - mas visto que esta entidade não terá muitos registos não será necessária a sua indexação.

Business
ID
Name
BusinessFlags
IsComposed
NameMLS

Tabela 3.1: Entidade Business

3.2.2 Entidade BusinessFlags

Esta entidade, representada na Tabela 3.2, serve para identificar os vários tipos de negócio que se pode fazer com o imóvel em questão.

Foi retirado o ID pois não havia referência nenhuma deste a outras tabelas. O campo utilizado noutras tabelas é o campo *BitFlag* em que refere quais os tipos de negócio possíveis. Esta entidade mantém a mesma estrutura.

Além do identificador único, esta entidade não contém mais índices.

BusinessFlags
Name
BitFlag
OriginalBusinessID

Tabela 3.2: Entidade BusinessFlags

3.2.3 Entidade Category

Esta entidade, representada na Tabela 3.3, serve para identificar a categoria de um imóvel. É a tabela *Category* integrada com as informações da tabela *MasterCategory*. Em NoSQL não é neecssária tanta divisão na Categoria de um imóvel.

Foi retirado o UID de ambos e, como se uniu as tabelas, não aparece o identificador do *MasterCategory* - o elemento de junção. Com esta junção de tabelas foi necessário alterar o nome do campo *Name* proveniente da *MasterCategory* para *Type*, visto que a tabela *Category* já contém um campo com esse nome.

É criado um índice para o identificador, além do índice no identificador único.

Category
ID
Name
AsRealEstate
Plural
NameMLS
PluralMLS
CategoryClid
AllowsPropType
Type
Luxury
AsProptype
AsBusiness
Priority
TypeMLS

Tabela 3.3: Entidade Category

3.2.4 Entidade Client

Esta entidade, representada na Tabela 3.4, contém os dados dos clientes. Foram mantidos todos os campos, excepto o UID e os campos terminados em *OLD*, pois estes como o próprio nome indica já não são utilizados. Além disso, foi acrescentado um campo *RealEstates* que contém uma listagem de todos os imóveis associados ao cliente. É preciso ter cuidado com o campo `Logo` pois é uma imagem e no processo de migração de dados não foi possível mantê-la, mas é possível guardar através da utilização do protocolo GridFS.

No que respeita a índices nesta entidade são criados mais dois, um para o `ID` e outro para o `Type` devido à sua constante utilização na plataforma Imoguia.

3.2.5 Entidade FeatureFlags

Esta entidade, representada na Tabela 3.5, indica o tipo de características um imóvel pode conter. Através do campo `Flag` consegue-se verificar se um imóvel contém essa

Client
ID
SystemID
Type
MasterCliente_ID
Name
Address
TownID
TownName
Phone
Celular
Fax
Email
AMI
OpenHours
Web
Campanha_ID
Score
Logo
Logotype
CountryID
LastActContacts
Transporter
hasImage360
hasImagesActive
hasWebChat
hasSMS
hasVideos
BarometroOn
RealEstates []

Tabela 3.4: Entidade Client

característica. Por essa razão e pelo facto de o identificador não ser utilizado em mais lado nenhum, foi suprimido o `ID` para esta estrutura.

Além do índice obrigatório, o identificador único, esta entidade não necessita de mais nenhum.

FeatureFlags
Name
Tag
Flag

Tabela 3.5: Entidade FeatureFlags

3.2.6 Entidade Image

Como o próprio nome indica, esta entidade, representada na Tabela 3.6, contém as informações relativas às imagens dos imóveis. Foi retirado o UID.

No processo de migração utilizado não foi possível migrar os campos `Description`, `DescriptionEn` e `DescriptionMLS` pois contêm caracteres, o carácter de fim de linha e tabulação, que tornariam a migração dos dados inconsistente.

Além do obrigatório, nesta entidade é necessário criar o índice no campo `ID`.

Image
ID
RealEstate_ID
System_ID
EID
Client_ID
RealEstateType
Description
DescriptionEn
Priority
Ext
DescriptionMLS

Tabela 3.6: Entidade Image

3.2.7 Entidade Localization

Esta entidade contém todos os tipos de localização registados. Na actual BD a localização encontrava-se muito "separada" e estruturada, com a implementação de um SGBD não-relacional, isto deixa de fazer sentido, e como tal, implementou-se um sistema de vectores para a identificação das regiões onde se insere, como podemos ver na Figura . Finaliza com o nome do País e inicia com o seu superior regional.

```
> db.Localization.findOne({"Type" : "ZoneGroup"})
{
        "_id" : ObjectId("4d289de0f62347151445d587"),
        "ID" : "1",
        "Name" : "Testes",
        "Type" : "ZoneGroup",
        "NameMLS" : "[CA]TestesCA[/CA][BR]TestesBR[/BR][EN]TestesEN[/EN][PT]Test
es[/PT][ES]TestesES[/ES]",
        "Enable" : "False",
        "BelongsTo" : [
                "Salgueirinha",
                "A dos Francos",
                "Caldas da Rainha",
                "Distrito de Leiria",
                "Centro",
                "Portugal"
        ]
}
>
```

Figura 3.9: Exemplo da entidade Localization

Manteve-se os identificadores existentes na anterior BD para conseguir correlacionar com as restantes entidades. As outras entidades identificam univocamente através do seu identificador e do tipo de localização - `Type`.

Como cada tipo tem os seus atributos e, como o MongoDB permite, cada tipo terá os seus atributos, como se encontra representado na Tabela 3.7.

Aqui criou-se um índice composto pelos campos *Type* e *ID*, pois a identificação numa pesquisa é sempre pelo tipo de localização e o seu identificador. Existem identificadores iguais mas como o tipo de local é diferente refere-se a locais distintos.

Localization

Column	ZoneGroup	Zone	Neighborhood	Town	TourismRegion	State	Region	Country
ID	x	x	x	x	x	x	x	x
Name	x	x	x	x	x	x	x	x
Type	x	x	x	x	x	x	x	x
BelongsTo[]	x	x	x	x	x	x	x	
Enable	x							
NameMLS	x				x			x
GPSLon		x	x	x		x	x	
GPSLat		x	x	x		x	x	
Urban			x					
ISO			x	x		x		x
Abrev						x		
Capital						x		
Continent							x	
ISO2								x
ISO3								x
IsActive								x

Tabela 3.7: Entidade Localization

3.2.8 Entidade Portals

Esta entidade, representada na Tabela 3.8, contém os portais existentes e aos quais se pode associar um imóvel.

Além do habitual índice, o identificador único, é necessário criar um no `ID`.

Portals
ID
Name
Enable
PortalFlags

Tabela 3.8: Entidade Portals

3.2.9 Entidade RealEstate

Esta entidade, representada na Tabela 3.9, corresponde ao `RealEstateData` e foi acrescentado um vector com as imagens e outro com os vídeos que são correspondentes ao imóvel em questão, para rápida pesquisa dos mesmos.

Foi criado um índice com o `RealEstateID`.

RealEstate	
RealEstateID	Score
SystemID	Date
Type	LayoutFile
EID	Category_NameEN
Clid	Proptype_NameEN
Client_ID	Business_NameEN
MasterClient_ID	Status_NameEN
SerialNumb	WebChat
Category_Name	tipopreco
Proptype_Name	promoter
Business_Name	conclusion_month
Zone_Name	contact_email
Neighborhood_Name	groups
Town_Name	GPSLon
State_Name	GPSLat
Status_Name	ImgFlag
Year	SMS
AreaU	Vendedor_ID
AreaB	OtherPrice
AreaT	OtherPriceSec
Price	DateCreated
SecPrice	Transporter
ShowPic	BusinessPriceString
Image_EID	Images []
HighLight	Videos []
Web	

Tabela 3.9: Entidade RealEstate

3.2.10 Entidade RealEstateBusiness

Esta entidade, representada na Tabela 3.10, contém as informações do seu homónimo mais as informações relativas ao `RentType` se tal se justificar (o negócio for um aluguer ou derivado).

Foi criado um índice para o `RealEstateID`.

RealEstateBusiness
RealEstateID
Business_ID
SystemID
Price
PriceOtherCurrency
RenttTypeID
RentTypeNameMLS
RentTypeName
Days

Tabela 3.10: Entidade RealEstateBusiness

3.2.11 Entidade RealEstateCatalogs

Esta entidade, representada na Tabela 3.11, contém as descrições para os catálogos de cada imóvel. A única diferença para a tabela existente foi a perda do `ID`, pois o mesmo não é referenciado em nenhuma entidade.

Foi criado um índice para o `RealEstateID`.

RealEstateCatalogs
RealEstateID
RealEstateType
CountryID
SystemID
PortalFlags
Transporter

Tabela 3.11: Entidade RealEstateCatalogs

3.2.12 Entidade RealEstateFeature

Esta entidade, representada na Tabela 3.12, funde os dados das tabelas `RealEstateKey`, `RealEstateFeature`, `Status`, `PropType` e `MasterProptype`, contendo as características dos imóveis, estas serve para pesquisa.

Durante o processo de migração os atributos `StatusMLS`, `PropTypeMLS`, `MasterPropTypeMLS` não foram passados pois continham caracteres - mudança de linha e tabulações - que prejudicavam a consistência da informação.

Foram criados os seguintes índices: `RealEstateID`, `Category_ID`, `Business_ID`, `Status e PropTypeName`.

3.2.13 Entidade RealEstatePortal

Esta entidade, representada na Tabela 3.13, indica quais os portais a que cada imóvel se encontra associado.

RealEstateFeature	
RealEstateID	Leilao_UID
SystemID	SecPrice
FeatureType	hasPrice
Transporter	hasOrderPrice
EID	OrderPrice
Clid	BusinessFlags
Client_ID	CountryID
MasterClient_ID	PortalFlags
SerialNumb	GPSLon
Category_ID	GPSLat
Business_ID	EnergyCertification
Zone_ID	FeatureFlags
Neighborhood_ID	PriceChangeDate
Town_ID	PriceChangeVariation
State_ID	PriceChangeValue
Region_ID	hasGPS
Status_ID	RealEstateVersion
AreaU	ImgFlag
AreaB	TourismRegion_ID
AreaT	MinPax
Price	MaxPax
ShowPic	PriceChangeLastPrice
Highlight	Status
Web	StatusMLS
Score	PropTypeName
Date	PropTypeMLS
AsGarage	MasterPropType
AsPool	MasterPropTypeMLS
AsFurniture	Rooms
IsOpenSpace	Convertibles
HeightVal	Floors
CountFeatures	HasDescription

Tabela 3.12: Entidade RealEstateFeature

No processo de migração de dados o atributo `CatalogString` foi retirado pois contém quebras de linhas e tabulações, prejudiciais ao processo.

Foi criado um índice para o `RealEstateID`.

3.2.14 Entidade System

Esta entidade, representada na Tabela 3.14, indica os sistemas existentes.

Além do índice no identificador único, não contém nenhum índices.

RealEstatePortal
RealEstateID
SystemID
PortalID
ClientID
Clid
CountryID
CatalogString

Tabela 3.13: Entidade RealEstatePortal

System
SystemID
Name
ConfigKey

Tabela 3.14: Entidade System

3.2.15 Entidade Video

Como a próprio nome indica, esta entidade, representada na Tabela 3.15, contém as informações dos vídeos relacionados com os imóveis.

Foi criado um índice no atributo ID.

Video
ID
SystemID
EID
Client_ID
RealEstate_ID
RealEstate_Type
Description
Priority
Ext
Transporter
DescriptionMLS

Tabela 3.15: Entidade Video

Esta estrutura foi a considerada ideal dentro das condições anteriormente expostas com um suporte não-relacional das plataformas. As alterações efectuadas visam melhorar a eficiência com a preocupação de manter a consistência dos dados.

Os índices foram escolhidos através do estudo e utilização das plataformas e com as necessidades existentes com a estrutura não-relacional.

Em termos de utilização de memória ao inserir estes dados verifica-se que o espaço despendido em disco é cerca de 400MB, como se pode ver pelo campo *FileSize*, representado na Figura 3.10 (todos os campos relativos a dimensões são apresentados em bytes). Podemos constatar que apenas 270MB é que estão a ser utilizados, através do campo *StorageSize*.

Embora não se tenha conseguido migrar todos os atributos, toda a informação perdida não faz uma diferença significativa no espaço utilizado pela BD.

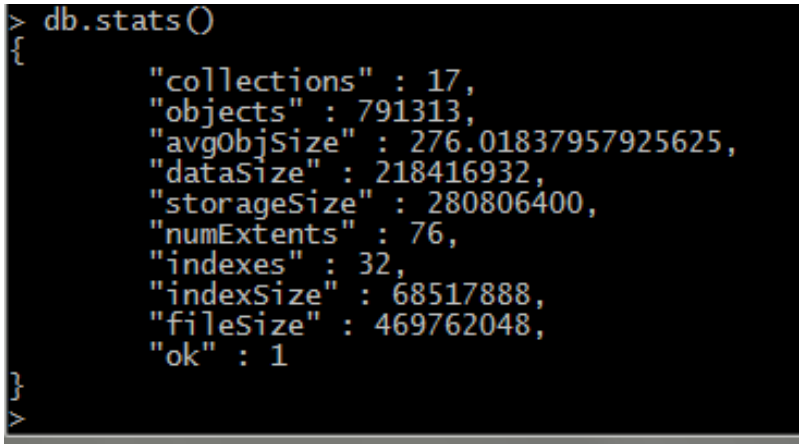

Figura 3.10: Espaço em disco NoSQL

Em termos de consumo de CPU e de memória, são consumidos poucos recursos, como podemos verificar na Figura (estas medidas foram efectuadas no decorrer de uma bateria de testes).

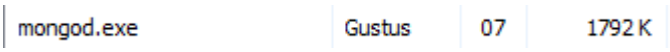

Figura 3.11: Consumo de memória e CPU - MongoDB

3.3 Migração de dados

Para uma correcta comparação dos SGBD's é peremptório utilizar a mesma informação. Para tal, é necessário transformar a informação existente no SGBD de forma a ser compatível com a nova estrutura e o novo SGBD.

Todo este processo de transformação foi tratado como se fosse um único processo, pois todos os passos e decisões interligam-se. Este processo, apesar de único, tem três fases distintas: o processo de exportação da informação na base de dados existente para ficheiros de texto; a passagem desta informação para XML e a consequente transformação para introdução na nova BD.

3.3.1 Exportação para ficheiro de texto

Primeiro os dados são exportados para um ficheiro de texto através da opção existente no *Microsoft SQL Server Management Studio* de exportar dados.

Como podemos ver na Figura 3.12, a primeira opção serve para seleccionar a fonte da exportação, a base de dados, sendo que de seguida é pedido para escolher o destino da informação a ser exportada. O facto da escolha da exportação da informação para um ficheiro de texto, faz com que se escolha a opção "*Flat File Destination*".

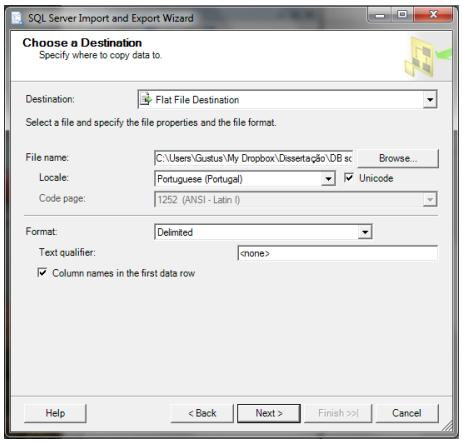

Figura 3.12: Transformação de dados para um ficheiro de texto - parte 1

A escolha da paginação *Unicode* é efectuada pois a maioria das tabelas necessitam deste tipo de paginação, visto que os atributos do tipo `DT_NTEXT` têm que ser exportados no formato *Unicode*, enquanto que os outros formatos podem ser exportados com esta paginação sem perder qualquer tipo de informação. Há uma excepção, os atributos do tipo `DT_TEXT`, que não podem ser exportados neste formato[18].

Em relação ao formato das colunas, foi escolhida a opção `Delimited` pois o comprimento dos atributos são bastante variáveis, assim sendo a única preocupação na sua leitura será o carácter de separação das colunas.

A opção de ter a primeira linha com o nome das colunas serve para identificação futura das entidades na criação do XML, facilitando a migração.

A opção seguinte, como se pode verificar na Figura 3.13, é colocar a consulta pretendida seleccionando a opção mais conveniente. As consultas que só envolvem uma tabela podem ser efectuadas através da 1ª opção, "*Copy data from one more tables or views*", que apresenta uma lista das tabelas existentes. Para consultas mais complexas é necessário utilizar a 2ª opção, "*Write a query to specify the data to transfer*". As consultas efectuadas são apresentadas no Anexo A.

A exportação da informação corresponde na íntegra à estrutura existente na BD não-relacional. Assim sendo, cada ficheiro exportado contém as informações relativas a uma única entidade, excepto a entidade `Localization` e os vídeos da entidade `RealEstate`.

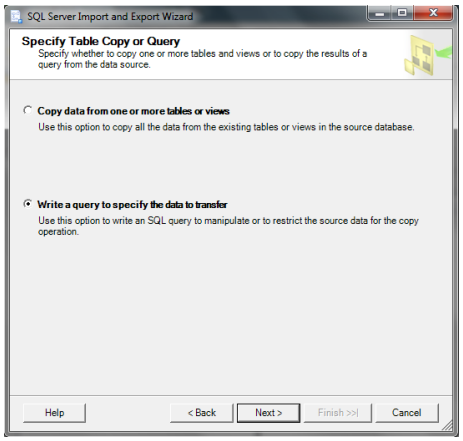

Figura 3.13: Transformação de dados para um ficheiro de texto - parte 2

No caso da entidade `Localization` foi necessário exportar os dados separadamente por zonas. A informação foi sendo exportada do local mais específico - `ZoneGroup` - para o mais abrangente - `Country`. Esta separação serve para identificar o tipo de localização e conseguir diferenciá-los.

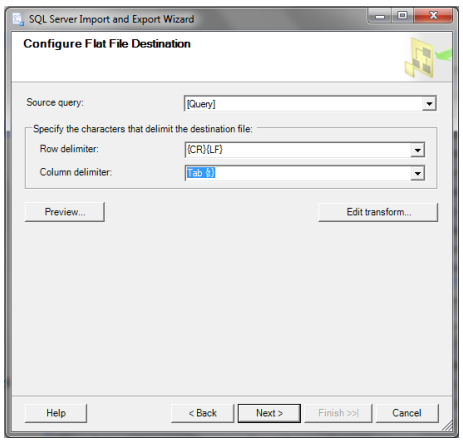

Figura 3.14: Transformação de dados para um ficheiro de texto - parte 3

Neste segmento, apresentado na Figura 3.14, escolheu-se como separadores o `{CR}{LF}`, - o fim e quebra de linha - para separar os registos existentes e o carácter `Tab {t}` para distinguir cada coluna. A selecção deste carácter coaduna-se com o facto de quase nenhum campo utilizar este carácter na sua composição e ser um dos caracteres possíveis para a importação para XML, como veremos posteriormente.

No caso de se alterar os campos de saída da consulta ou no caso de se ter que retirar alguns dos campos, utiliza-se a opção "*Edit transform*".

A opção seguinte, apresentada na Figura 3.15, serve apenas para confirmar que a mesma é executada sem ser necessário recorrer à encriptação da informação, visto que vamos utilizá-la com outro programa que ao encripta-la não tinha hipótese de a ler.

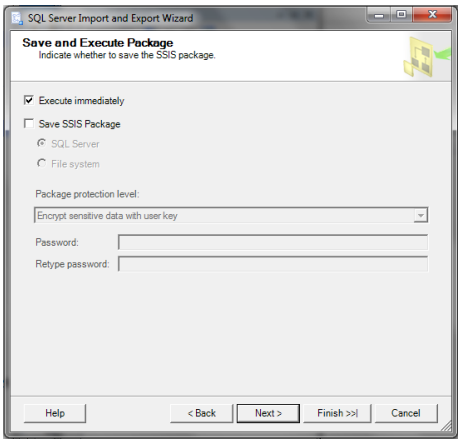

Figura 3.15: Transformação de dados para um ficheiro de texto - parte 4

A exportação de informação decorreu com as características descritas acima mas houve algumas excepções:

- Na entidade `Client`, o atributo `Logo` não foi exportado visto que o mesmo é do tipo `Image` e como tal não dá para colocá-lo num ficheiro de texto.
- A entidade `RealEstate` foi exportada com a paginação *ANSI - Latin I (1252)* pois o atributo `BusinessPriceString` é do tipo *DT_TEXT* e como tal não pode ser exportado para através do código de paginação *Unicode*. Assim sendo escolheu-se não exportar o atributo `DescriptionMLS` devido à alteração da paginação e o mesmo contém caracteres que tornavam a informação exportada inconsistente.
- Na entidade `RealEstateFeature` não foi exportado o atributo `FeatureJSonMLS` pois contém quebras de linhas e tabulações dando registos inconsistentes.
- Na entidade `Imagem` os atributos `Description`, `DescriptionEn` e `DescriptionMLS` não foram exportados devido a conterem caracteres, como os descritos no ponto anterior, que não permitiam a migração dos dados e assegurar consistência.
- Na entidade `RealEstateCatalogs` o atributo `CatalogString` não foi exportado pois contém quebras de linhas e como tal iria dar alguns problemas na consistência dos dados.

3.3.2 Transformação para XML

A escolha do XML deve-se à sua facilidade de manipulação e transformação. Os ficheiros de texto que contêm os dados são transformados em ficheiros *XML* para uma fácil manipulação, através da opção existente no *OxygenXML*, como podemos ver na Figura 3.16. Neste caso, com enormes quantidades de texto a processar, existe um ficheiro

com 800MB, é difícil pois a transformação consome demasiada memória e nesses casos, com o computador de trabalho, será necessário arranjar uma solução alternativa.

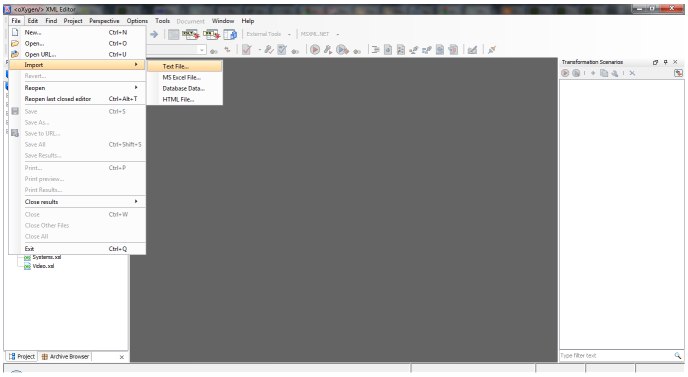

Figura 3.16: Importação de ficheiro de texto para XML

Na Figura 3.17, podemos verificar a razão da exportação para o ficheiro de texto com os nomes das colunas na 1ª linha, assim o *OxygenXML* edita o nome dos elementos correctamente para uma fácil identificação. Verifica-se também a escolha do carácter delimitador de coluna ser a tabulação, pois na selecção de caracteres disponível é bem mais reduzida do que na exportação.

É seleccionada a opção de guardar o resultado num ficheiro XML com o nome da entidade.

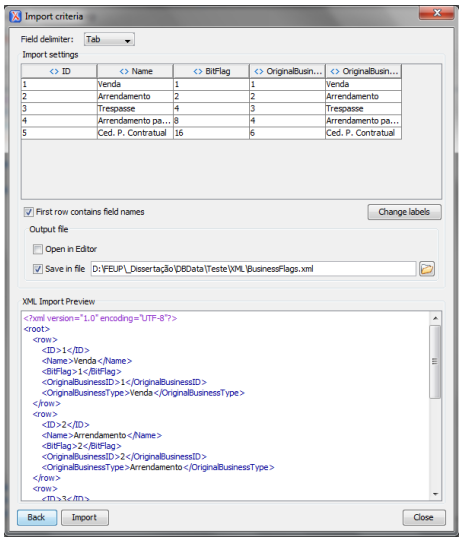

Figura 3.17: Importação de um ficheiro de texto para o formato XML

De salientar que a criação da entidade `Localization` encontra-se dividida em várias partes, pois o conceito implementado é diferente da actual concepção e, como tal, foi necessário alterar um dos parâmetros para diferenciar os diferentes tipos de localização (*zone*, *zonegroup*, *neighborhood*, *town*, *tourismregion*, *region*, *state* e *country*).

Para que se diferenciarem foi necessário alterar o nome da entidade para o tipo de localização, como podemos ver na Figura 3.18, finalizada a transformação de todos os tipos uniu-se os ficheiros XML obtidos.

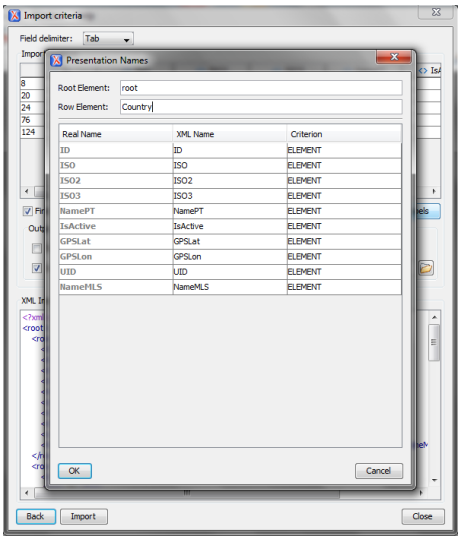

Figura 3.18: Importação da Entidade Localization - Country

3.3.3 Formato final e introdução dos dados

Com a importação dos dados para XML torna-se mais fácil a transformação da informação para o formato final antes da inserção. Esta transformação não foi efectuada inicialmente pela forma como a informação estava da 1ª vez, visto que a mesma se encontrava misturada, não permitindo a sua correcta introdução.

Pressupostos para a transformação:

- Todos os componentes não contêm espaço entre si
- Cada registo é separado pelo carácter |
- Cada atributo contém 2 valores, o seu nome e o seu valor, separados pelo carácter :

Obtidos os ficheiros XML, foram criados para cada entidade da nova estrutura um ficheiro de transformação XSLT (*Extensible Stylesheet Language Transformations*) onde é estruturado o ficheiro para o formato final antes da introdução na base de dados. Podemos ver a estrutura de cada entidade no Anexo B.

Neste processo de transformação foi feita a leitura registo a registo visto que os mesmos encontravam-se completos. Enquanto que nas entidades `Client` e `RealEstate` com a existência de uma listagem de imóveis e imagens, respectivamente, por cada registo existiam várias linhas com os diferentes componentes da lista. Assim sendo, foi necessário verificar cada registo e juntar esses elementos num único registo.

Como a entidade `RealEstate` contém muita informação não foi possível, com o posto de trabalho existente, transformar tudo num único ficheiro. Foram transformados os 5 ficheiros XML correspondentes e posteriormente fundido num só. Em cada um dos 5 ficheiros foram colocados aproximadamente 100.000 registos, com a preocupação de não separar os elementos com o mesmo identificador, pois se tal acontecesse haveria incongruências, pois apareceria mais do que um elemento com o mesmo identificador.

Além disto, a entidade `RealEstate`, contém 2 listas distintas - imagens e vídeos -, e não era possível, tratar ambas as listas ao mesmo tempo. Como tal, a listagem de imagens é tratada com esta entidade enquanto que a listagem de vídeos é transformada e tratada à parte e só é introduzida posteriormente na BD.

Já no caso da entidade `Localization`, como foi separada nos vários tipos de localização, o ficheiro de transformação efectua as transformações de todos eles. O ficheiro XML foi fundido num único ficheiro, com a preocupação de retirar o texto (<*?xml version="1.0" encoding="UTF-8"?*>) que é acrescentado automaticamente ao transformar um ficheiro, sendo que a transformação resulta num ficheiro só.

Estes ficheiros de texto depois são processados em C#, foi escolhida esta opção pois a aplicação encontra-se desenvolvida em C# e .NET. Neste processo é necessário retirar <*?xml version="1.0" encoding="UTF-8"?*> do início de cada ficheiro.

Verificou-se que o formato do MongoDB é *ASCII-16* por isso é necessário ter atenção ao introduzir caracteres especiais.

3.4 Testes

Com a percepção do funcionamento e das partes críticas da aplicação, foram efectuados testes a ambas as SGBD para uma comparação mais clara. Cada teste da BD não-relacional tem o correspondente teste na BD relacional. Os testes foram efectuados em C# e a ferramenta utilizada foi o Microsoft Visual Studio para os testes unitários.

Foi utilizada a versão 1.6.5 do MongoDB e a nova versão do controlador oficial de C#, a 0.9.0.3992. Para o SQL foi sutilizado o Microsoft SQL Server 2005, o mesmo utilizado com as plataformas.

De modo a garantir que todos os testes são realizados nas mesmas condições e que não existem dados ou resultados de consultas em memória, o SGBD é reiniciado após cada carga de testes efectuada.

Nos testes, foi utilizada uma ligação local e um só computador. Não houve condições para testar com várias máquinas para potenciar os testes e verificar se o MongoDB mantém a sua performance com muitas máquinas.

Sistema:

- Pentium(R) Dual-Core CPU T4300 @ 2.10GHz

- RAM : 4 GB
- Sistema 64-bit

Os testes consistem em verificar as operações mais utilizadas na aplicação.

Em termos de nomenclatura dos testes, utilizou-se um prefixo (*Non*) para distinguir os testes relativos ao MongoDB dos efectuados ao SQL.

3.4.1 Testes unitários

Inserção de 100.000 registos, com os resultados obtidos nas Tabelas 3.16 e 3.17

Nota-se uma enorme discrepância de resultados na introdução de uma larga quantidade de informação no sistema. Enquanto que no MongoDB a inserção de 100.000 registos dura em média 9 segundos, no SQLServer este mesmo procedimento dura em média 2 minutos e 4 segundos.

Podemos verificar que na inserção de grandes quantidades de informação o MongoDB ganha uma larga vantagem. Isto acontece visto que é possível introduzir vários registos numa única chamada à base de dados enquanto que no SQL tal não é possível.

Teste	Tempo (s)
1	6.3310
2	9.1623
3	7.8209
4	9.5777
5	10.0931
6	11.1592
7	8.3367
8	11.0041
9	8.1603
10	8.3300
Média	8.99753

Tabela 3.16: NonTestInsert

Teste	Tempo (s)
1	147.8815
2	131.2939
3	140.6439
4	121.2959
5	137.6228
6	94.8146
7	113.2489
8	84.6143
9	104.3903
10	162.7003
Média	123.85064

Tabela 3.17: TestInsert

Pesquisa e actualização desses registos, Tabela 3.19 e 3.18.

Neste teste os resultados foram bastante semelhantes sendo que ambos têm uma média de 0,7 segundos. Podemos então verificar que na pesquisa e actualização de um registo ambos têm um desempenho semelhante.

Teste	Tempo (s)
1	0.7101
2	0.8289
3	0.7492
4	0.6921
5	0.7388
6	0.7156
7	0.7726
8	0.6772
9	0.6669
10	0.7524
Média	0.73059

Tabela 3.18: NonTestUpdate

Teste	Tempo (s)
1	0.8184
2	0.7931
3	0.7367
4	1.0576
5	0.7421
6	0.5564
7	0.7225
8	0.5560
9	0.5682
10	0.6221
Média	0.71731

Tabela 3.19: TestUpdate

Apagar 100.000 registos - 3.20 e 3.21.

Verifica-se que é bastante mais rápido apagar uma tabela com o MongoDB do que com o SQL. Apesar de a diferença ser significativa, como este tipo de operação não é efectuada muitas vezes não é muito representativa.

De salientar que no MongoDB a tabela é mesmo apagada enquanto que no SQL são apagados todos os registos. Isto tem a ver com o facto de ser necessário no SQL ter a tabela para a sua inserção de dados enquanto que no MongoDB não é necessário a entidade existir para a sua criação.

Teste	Tempo (s)
1	0.0770
2	0.0763
3	0.0589
4	0.0596
5	0.0399
6	0.0333
7	0.0325
8	0.0323
9	0.0273
10	0.0576
Média	0.04947

Tabela 3.20: NonTestDrop

Teste	Tempo (s)
1	0.6703
2	0.6471
3	0.7481
4	0.8056
5	0.6443
6	0.3884
7	0.4493
8	0.4339
9	0.4273
10	0.4271
Média	0.56414

Tabela 3.21: TestDrop

Listagem de todos os negócios efectuados, Tabela 3.22 e 3.23.

Verifica-se que a listagem dos negócios no MongoDB é mais rápida mas constata-se que a diferença não é significativa. Pelos testes efectuados verifica-se que o último teste do

SQL é bastante anormal, visto que com os outros testes não houve muito desvio no tempo de resposta, e como tal, é uma das principais razões para piorar a média comparativamente ao MongoDB.

Teste	Tempo (s)
1	1.0971
2	1.1434
3	1.5580
4	1.1308
5	1.2709
6	1.1765
7	0.9578
8	1.0593
9	0.8654
10	0.9660
Média	1.12252

Tabela 3.22: NonBusinessTest1

Teste	Tempo (s)
1	2.6952
2	1.5029
3	2.1064
4	1.6146
5	2.0743
6	1.1508
7	2.3045
8	1.1722
9	1.1161
10	3.5697
Média	1.93067

Tabela 3.23: BusinessTest1

Listagem de todos os clientes, Tabela 3.24 e 3.25

Neste teste verifica-se que o SQL obtém uma resposta mais rápida, esta diferença ainda é significativa.

Teste	Tempo (s)
1	0.2198
2	0.4422
3	0.3537
4	0.2548
5	0.2604
6	0.4490
7	0.3873
8	0.3127
9	0.2715
10	0.3332
Média	0.32846

Tabela 3.24: NonTestClient1

Teste	Tempo (s)
1	0.2181
2	0.2181
3	0.1409
4	0.1681
5	0.1394
6	0.1352
7	0.1757
8	0.1611
9	0.1643
10	0.1597
Média	0.16806

Tabela 3.25: TestClient1

Pesquisa e listagem detalhada de um cliente, Tabela 3.26 e 3.27.

Aqui o MongoDB obtém tempos de resposta mais rápidos, apesar de ter dois testes anormais - Teste 2 e 6. Não são tempos consideráveis mas a diferença é bastante significativa.

Teste	Tempo (s)
1	0.0093
2	0.0361
3	0.0089
4	0.0054
5	0.0053
6	0.0256
7	0.0071
8	0.0057
9	0.0113
10	0.0057
Média	0.01204

Tabela 3.26: NonTestClient2

Teste	Tempo (s)
1	0.0805
2	0.0671
3	0.0527
4	0.0707
5	0.0549
6	0.0527
7	0.0540
8	0.0831
9	0.0605
10	0.0967
Média	0.06729

Tabela 3.27: TestClient2

Pesquisa rápida por um cliente, Tabela 3.28 e 3.29.

Este teste lista 1184 registos, sendo que o SQL, em média, é mais rápido na sua disposição. Mas verifica-se que o teste número 1 do MongoDB é bastante anormal, sem aparente razão, e que sem contar com este teste ambos os testes ficam semelhantes na eficiência de resposta.

Teste	Tempo (s)
1	0.4445
2	0.0404
3	0.0795
4	0.0434
5	0.0396
6	0.0624
7	0.0646
8	0.0434
9	0.1446
10	0.0390
Média	0.10014

Tabela 3.28: NonTestClient3

Teste	Tempo (s)
1	0.0230
2	0.0289
3	0.0515
4	0.0272
5	0.0265
6	0.0381
7	0.0188
8	0.0255
9	0.0324
10	0.0734
Média	0.03453

Tabela 3.29: TestClient3

Listagem de todos os imóveis, Tabela 3.30 e 3.31.

Com cerca de 300.000 registos verifica-se que a diferença de performance é enorme. Sendo que o MongoDB, em média, responde em 6 segundos, o SQL responde ao mesmo pedido com uma média de 30 segundos.

Teste	Tempo (s)
1	3.7726
2	7.2425
3	7.2574
4	5.5499
5	5.9402
6	8.4587
7	6.2307
8	6.2785
9	6.5990
10	6.5502
Média	6.38797

Tabela 3.30: NonTestRealEstate1

Teste	Tempo (s)
1	34.1952
2	29.0337
3	30.7104
4	35.5785
5	31.2297
6	26.8766
7	28.7820
8	28.4881
9	29.3887
10	31.3676
Média	30.56505

Tabela 3.31: TestRealEstate1

Pesquisa e listagem detalhada de um imóvel, Tabela 3.32 e 3.33.

Verifica-se que ambos são semelhantes neste procedimento mas verifica-se a existência de três testes muito díspares no MongoDB - testes 1, 2 e 6 - enquanto que no SQL apenas encontramos um - o teste 1.

Teste	Tempo (s)
1	0.0323
2	0.0428
3	0.0084
4	0.0024
5	0.0023
6	0.0219
7	0.0090
8	0.0042
9	0.0065
10	0.0036
Média	0.01334

Tabela 3.32: NonTestRealEstate2

Teste	Tempo (s)
1	0.0320
2	0.0177
3	0.0193
4	0.0147
5	0.0074
6	0.0060
7	0.0053
8	0.0094
9	0.0063
10	0.0052
Média	0.01233

Tabela 3.33: TestRealEstate2

Pesquisa rápida por um imóvel, Tabela 3.34 e 3.35.

Aqui verifica-se que o SQL é mais rápido que o MongoDB, com uma diferença significativa.

Teste	Tempo (s)
1	0.7300
2	1.1212
3	1.1978
4	1.0776
5	1.8455
6	1.3893
7	1.2970
8	1.2041
9	1.1511
10	1.1934
Média	1.2207

Tabela 3.34: NonTestRealEstate3

Teste	Tempo (s)
1	0.9376
2	0.5973
3	0.6441
4	0.6645
5	0.6662
6	0.3789
7	0.4271
8	0.4056
9	0.4004
10	0.3758
Média	0.54933

Tabela 3.35: TestRealEstate3

Pesquisa avançada por um imóvel, Tabela 3.36 e 3.37.

O SQL obtém uma melhor performance com uma diferença pequena comparativamente ao MongoDB.

Teste	Tempo (s)
1	0.1312
2	0.2214
3	0.2516
4	0.2594
5	0.2218
6	0.2891
7	0.2453
8	0.2243
9	0.3179
10	0.2355
Média	0.23975

Tabela 3.36: NonTestRealEstate4

Teste	Tempo (s)
1	0.1843
2	0.2070
3	0.1447
4	0.1468
5	0.1589
6	0.0924
7	0.1342
8	0.0910
9	0.1090
10	0.1077
Média	0.1376

Tabela 3.37: TestRealEstate4

Contagem de registos após pesquisa avançada, Tabela 3.38 e 3.39.

A diferença não é muita, são bastante semelhantes apesar de o SQL ser mais rápido.

Teste	Tempo (s)
1	0.1268
2	0.1135
3	0.1464
4	0.1458
5	0.1345
6	0.1463
7	0.3689
8	0.1139
9	0.4986
10	0.1180
Média	0.19127

Tabela 3.38: NonTestRealEstate5

Teste	Tempo (s)
1	0.1291
2	0.1546
3	0.0775
4	0.1290
5	0.1890
6	0.0535
7	0.0829
8	0.0553
9	0.0681
10	0.0926
Média	0.10316

Tabela 3.39: TestRealEstate5

Após os testes, verifica-se que não existe um sistema que domine os testes, apesar do SQL obter melhores resultados em mais testes.

Verifica-se que nos testes que envolve grandes quantidades de informação - o teste de inserção e apagar de 100.000 registos e as listagens dos negócios e dos imóveis - o MongoDB obtém uma enorme vantagem, com performances muito mais satisfatórias que o SQL.

Já em relações a listagem de um registo ambos os sistemas apresentam uma resposta semelhante. Na listagem de um cliente, o MongoDB, apresenta um resposta mais rápida enquanto que na listagem de imóvel, apesar de uma performance ligeiramente inferior obteve um bom resultado.

Nos testes de pesquisa, seja a pesquisa por um pedaço de texto em qualquer atributo da entidade ou então uma pesquisa avançada, o SQL obtém uma melhor performance com uma pequena diferença.

Como as plataformas escolhidas têm como principais funções críticas, a pesquisa e a listagem de imóveis e clientes, os testes efectuados acabam por não dissipar as dúvidas sobre qual o melhor sistema. Ambos têm performances semelhantes sendo que as diferenças mais significativas são favoráveis ao MongoDB.

Capítulo 4

Conclusões e Trabalho Futuro

4.1 Conclusões

Pelas características apresentadas pelos sistemas de gestão de bases de dados não-relacionais, verificamos que estes são bastante flexíveis e que a maior parte não asseguram os pedidos efectuados à base de dados. Aliás, os sistemas de gestão de bases de dados não-relacionais só têm preocupações ao nível do desempenho e da escalabilidade.

A parte de manipulação e de validação dos dados, de resolução de conflitos e de controlo de acesso - a parte integrante de qualquer sistema de gestão de bases de dados relacional através do conjunto de propriedades ACID - passam a pertencer à camada da aplicação. Assim, quem implementa a aplicação tem mais responsabilidade no sentido de evitar inconsistências na informação.

Estas características intrínsecas aos SGDB não-relacionais levam a que não seja aconselhável utilizar este tipo de plataformas em sistemas críticos ou extremamente complexos. O sistema é recomendado para aplicações com uma complexidade pequena ou média, onde o que importa é a rapidez e eficiência.

Já em termos de quantidade de dados, os sistema de gestão de bases de dados não-relacionais manipulam mais eficientemente enormes quantidades de dados pois não é a plataforma que trata da relação, mas sim a camada de aplicação. Os sistemas relacionais têm um grande problema no que toca à manipulação de quantidades significativas de dados, pois a operação JOIN - uma das operações mais utilizadas - é extremamente dispendiosa, uma vez que cruza toda a informação das tabelas antes de seleccionar e projectar os dados pretendidos. Com os sistemas não-relacionais esse cruzamento de dados é feito pela aplicação, diminuindo a carga de utilização da base de dados, libertando-a assim para outros pedidos.

Verifica-se que existem 3 grandes tipos de sistemas de gestão de bases de dados separados pelo modelo de armazenamento dos dados. O armazenamento por valores de chave (*key-value*), - extremamente simplista e rápido e optimizado para uma utilização previsível e de baixa latência - o armazenamento de colunas, - extremamente flexível e estruturável - e o armazenamento por documentos - que combina as características das bases de dados não-relacionais com a estrutura das bases de dados relacionais.

Para o caso de estudo foram analisados 3 SGBD's: o MongoDB e o CouchDB - do tipo de armazenamento por documentos - e o Cassandra - do tipo de armazenamento de colunas. Não foi tomado em conta nenhum sistema de armazenamento por valor de chave devido à sua simplicidade e porque, por esta razão, não teria as características necessárias para o caso de estudo. Ao estudar os outros tipos podem-se contrapor algumas das suas características e verificar o seu funcionamento.

O estudo dos SGDB não-relacionais tem como principais diferenciadores a escalabilidade, fiabilidade e desempenho. Assim sendo, todos eles suprimem a necessidade de escalabilidade e fiabilidade. Já em termos de desempenho o CouchDB deixa algo a desejar, visto que o sistema assenta numa arquitectura REST (Representational State Tansfer) que se baseia em muitos pedidos entre os vários nós, fazendo com que o seu desempenho não seja o melhor comparativamente com o MongoDB.

A decisão pende assim para o MongoDB e para o Cassandra. O Cassandra funciona como um sistema de rede distribuída cujo tipo de armazenamento potencia uma estrutura de dados complexa e é por isso optimizado para trabalhar com enormes quantidades de dados. Por outro lado, o MongoDB tem como vantagem o facto de ser parecido com os SGBDR e pelo tipo de armazenamento ser mais condizente com a complexidade e necessidades das plataformas Imoguia e Portal Casa Sapo, recaindo assim a escolha deste SGBD para implementação.

Após a selecção do MongoBD foi efectuado um estudo às características e estrutura das plataformas existentes para compreender a sua utilização e as suas necessidades na criação da nova estrutura.

Verificou-se que o Imoguia é uma ferramenta de gestão para agentes imobiliários que faz a gestão da carteira de imóveis, contactos e gestão de agenda. Já o Portal Casa Sapo permite a qualquer pessoa ter um espaço para a divulgação de imóveis, além da sua utilização por parte dos profissionais.

Sendo estas duas plataformas direccionadas para a pesquisa e divulgação de imóveis, o seu ponto crítico é a pesquisa dos mesmos, seja numa pesquisa rápida ou avançada, por categoria ou localização. Além disso, no Imoguia denota-se uma grande preocupação com a entidade clientes, uma vez que estes se encontram divididos por vários tipos. Em termos de carga da base de dados confirma-se uma grande utilização das tabelas relacionadas com o imóvel e verifica-se que as imagens são o factor que maior carga traz à base de dados, visto que cada imóvel pode conter várias imagens.

Para uma comparação justa foi necessário migrar os dados existentes para a estrutura não- relacional. Como a estrutura é diferente, foi necessário recorrer a um método para transformar, exportar para XML e introduzir os dados na nova estrutura. Verificou-se que não foi o melhor método, visto que não foi possível colocar todos os atributos na nova base de dados. Devia-se por isso ter utilizado uma ligação "directa", utilizando C# e .NET - linguagem em que as plataformas foram implementadas - para a migração dos dados, tornando-a mais fácil e eficiente.

Devido a esta migração, o modelo da base de dados não-relacional não aproveita todas as suas potencialidades, uma vez que é preciso um sistema híbrido para conseguir manter as relações existentes na nova BD. Assim sendo, devia ter-se migrado a informação para o modelo óptimo.

Foram efectuados testes unitários às utilizações mais pertinentes da plataforma, tais como: listagens de imóveis, clientes e negócios; pesquisa rápida ou avançada; contagem para estatísticas e testes de CRUD (`Create`, `Remove`, `Update` e `Delete`).

Com os testes efectuados verificou-se que o MongoDB é mais eficiente a tratar grandes quantidades de informação. Nos restantes testes unitários verificou-se uma performance bastante semelhante ao SQL, algumas vezes até superior.

Visto que os testes foram efectuados apenas com um único computador, não foi possível verificar um dos objectivos - a escalabilidade e o desempenho com vários servidores - para verificar se com o MongoDB o desempenho se mantém constante ou melhora. É conhecido o problema de afunilamento com o SQL - só existe um nó de escrita - e a consequente perda de desempenho com o aumento de utilizadores.

Além das conclusões acima referidas, foram verificados os recursos físicos utilizados por ambos os sistemas. Constata-se então que o MongoDB utiliza muito menos recursos no armazenamento físico - 400MB contra os 1,25GB do ocupado pelo SQL - e no recurso de memória - em que o servidor do MongoDB utiliza 1792KB contra os 300MB, com vários serviços activos.

Pegando no objectivo do caso de estudo, - o de verificar se vale a pena trocar o SGDB existente por um sistema não-relacional - além do descrito acima é preciso verificar algumas coisas específicas para a plataforma em si.

O suporte do controlador para C# é muito recente, sendo que ainda não existe um versão final (ainda se encontra na versão 0.9.0.3992). Não existe até ao momento uma API, sendo que todas as referências em fóruns são sobre os controladores suportados pela comunidade, o que pode tornar a sua utilização confusa. Outro dos problemas é o facto da codificação do MongoDB ser em *ASCII-16*, o que faz com que não sejam aceites caracteres especiais, sendo isso um problema específico para a língua portuguesa.

Em suma, a alteração de Microsoft Server 2005 para o MongoDB pode ser bastante benéfica, apesar de que será necessária a realização de alguns testes com vários servidores para ter a certeza de que se obtém um desempenho melhor. Apesar de se considerar

vantajosa a alteração, aconselha-se a espera por um controlador final (versão 1.0) ou então a contratação da empresa, uma vez que actualmente não existe um local como referência para apoio imediato na implementação deste controlador.

4.2 Satisfação dos Objectivos

A conclusão dos objectivos foi satisfatória, não sendo superior por não ter sido possível implementar um protótipo funcional nem obter resultados significativos com os testes de carga. Apesar disso, considero que o estudo realizado foi aprofundado, sustentando as diferenças entre os sistemas de gestão de bases de dados.

Houve ainda alguns aspectos que poderiam ter corrido melhor. Após término da dissertação, e com uma maior consciência e sabedoria sobre o assunto, considero que, relativamente à estrutura, melhoramentos poderiam ter sido feitos. Assim, poderia ter sido feita a junção das duas entidades para o imóvel, pois a separação destas não faz muito sentido numa base de dados não-relacional.

Outra das coisas que teria feito de modo distinto prende-se com a implementação da estrutura, sendo que, havendo a possibilidade de reformular o estudo, teria implementado uma estrutura completamente coerente com as ideias das bases de dados não-relacionais.

Infelizmente não houve a possibilidade de estagiar na empresa, sediada em Óbidos. Estou certo de que o contacto directo e acompanhamento de pessoas ligadas às plataformas teria sido um factor benéfico para este projecto.

4.3 Trabalho Futuro

Tendo este projecto como base é possível um estudo exaustivo sobre as diferenças das bases de dados relacionais e das não-relacionais pois consegue-se assim verificar com mais precisão quais são as ideais para cada tipo de problema.

Em relação ao caso de estudo, como escrito anteriormente, é necessário melhorar a estrutura da base de dados não-relacional e a migração dos dados.

Referências

[1] Chris J. Date. *An Introduction to Database Systems*. Pearson Addison-Wesley, Boston, MA, 8. edição, 2004.

[2] Vineet Gupta. Nosql databases - part 1 - landscape, Janeiro 2010. http://www.vineetgupta.com/2010/01/nosql-databases-part-1-landscape.html.

[3] Jim Gray. The transaction concept: Virtues and limitations. *Proceedings of the 7th International Conference on Very Large Databases*, páginas 144–154, 1981/06.

[4] Raghu Ramakrishnan e Johannes Gehrke. *Database Management Systems*. McGraw-Hill, Inc., New York, NY, USA, 2003.

[5] M. Tamer Özsu e Patrick Valduriez. *Principles of distributed database systems (2nd ed.)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1999.

[6] Burleson Consulting. Vertical vs. horizontal scalability for oracle, Junho 2010. http://www.dba-oracle.com/t_Vertical_vs_Horizontal_scalability_Oracle.htm.

[7] TechTarget. What is vertical scalability?, Junho 2010. http://searchcio.techtarget.com/sDefinition/0,,sid182_gci928995,00.html.

[8] Kris Zyp. Nosql architecture, Maio 2010. http://www.sitepen.com/blog/2010/05/11/nosql-architecture.

[9] Neal Leavitt. Will nosql databases live up to their promise? *Computer*, 43(2):12–14, 2010.

[10] Inc. 10gen. Mongodb, Junho 2010. http://www.mongodb.org.

[11] IBM. Index management, Junho 2010. http://publib.boulder.ibm.com/infocenter/db2luw/v8/index.jsp?topic=/com.ibm.db2.udb.doc/admin/c0005052.htm.

[12] Roy T. Fielding e Richard N. Taylor. Principled design of the modern web architecture. *ACM Trans. Internet Technol.*, 2(2):115–150, May 2002.

[13] Cesare Pautasso, Olaf Zimmermann, e Frank Leymann. Restful web services vs. big web services: Making the right architectural decision. Em *17th International World Wide Web Conference (WWW2008)*, páginas 805–814, Beijing, China, April 2008.

[14] Joe Lennon. Exploring couchdb: A document-oriented database for web applications, Março 2009. http://www.ibm.com/developerworks/opensource/library/os-couchdb/index.html.

[15] J. Chris Anderson, Jan Lehnardt, e Noah Slater. *CouchDB: The Definitive Guide Time to Relax*. O'Reilly Media, Inc., 2010.

[16] JavaEye.com. Nosql discussion of the database - why should non-relational database?, Novembro 2009. http://www.javaeye.com/topic/524977, ver em Inglês através do Google Translator http://translate.google.com/translate?js=y&prev=_t&hl=pt-PT&ie=UTF-8&layout=1&eotf=1&u=http%3A%2F%2Fwww.javaeye.com%2Ftopic%2F524977&sl=auto&tl=en.

[17] Eric Evans. The cassandra distributed database, Fevereiro 2010. http://www.slideshare.net/jericevans/the-cassandra-distributed-database?from=ss_embed.

[18] Microsoft Corporation. Msdn | desenvolvimento na plataforma .net da microsoft, assinaturas, webcasts, vídeos, e mais, Novembro 2010. http://msdn.microsoft.com.

Anexo A

SQL Queries

A.1 Business

```
SELECT *
FROM Business
```

A.2 BusinessFlags

```
SELECT BusinessFlags.ID, BusinessFlags.Name,
BusinessFlags.BitFlag, BusinessFlags.OriginalBusinessID,
Business.Name AS OriginalBusinessType
FROM      BusinessFlags INNER JOIN
Business ON BusinessFlags.OriginalBusinessID = Business.ID
```

A.3 Category

```
SELECT Category.ID, Category.MasterCategory_ID,
Category.Name, Category.AsRealEstate, Category.Plural,
Category.UID, Category.NameMLS, Category.PluralMLS,
          Category.CategoryClid, Category.AllowsProptype,
          MasterCategory.ID AS MasterCategoryID,
          MasterCategory.Name AS MasterCategoryName,
          MasterCategory.Luxury, MasterCategory.AsProptype,
          MasterCategory.AsBusiness, MasterCategory.Priority,
          MasterCategory.UID AS MasterCategoryUID,
          MasterCategory.NameMLS AS MasterCategoryNameMLS
FROM      Category INNER JOIN
          MasterCategory ON
          Category.MasterCategory_ID = MasterCategory.ID
```

A.4 Client

```
SELECT Client.ID, Client.SystemID, Client.Type,
Client.MasterCliente_ID, Client.Name,
```

```
Client.Address, Client.Town_ID, Client.Town_Name,
Client.Phone, Client.Celular, Client.Fax,
Client.Email, Client.Domain, Client.AMI,
Client.OpenHours, Client.Web, Client.Logo,
Client.Campanha_ID, Client.Score, Client.Image360OLD,
Client.ImagesOLD, Client.WebChatOLD, Client.SMSOLD,
Client.VideosOLD, Client.UID, Client.Logotype,
Client.CountryID, Client.LastActContacts, Client.Transporter,
Client.hasImage360, Client.hasImagesActive, Client.hasWebChat,
Client.hasSMS, Client.hasVideos, Client.BarometroOn,
RealEstateKey.ID AS RealEstateID
FROM Client left JOIN
          RealEstateKey ON RealEstateKey.Client_ID = Client.ID
ORDER BY  Client.ID
```

A.5 FeatureFlags

```
SELECT *
FROM FeatureFlags
```

A.6 Imagem

```
SELECT *
FROM Image
```

A.7 Localization

```
SELECT ID, ISO, ISO2, ISO3, NamePT, IsActive, GPSLat, GPSLon, NameMLS
FROM Country
```

```
SELECT Neighborhood.ID, Neighborhood.Name, Neighborhood.Urban,
Neighborhood.GPSLat, Neighborhood.GPSLon, Neighborhood.ISO,
Town.Name AS Town, State.Name AS State, Region.Name AS Region,
Country.NamePT AS Country
FROM Neighborhood INNER JOIN
Town ON Neighborhood.Town_ID = Town.ID INNER JOIN
State ON Town.State_ID = State.ID INNER JOIN
Region ON State.Region_ID = Region.ID INNER JOIN
Country ON Region.Country_ID = Country.ID
```

```
SELECT Region.ID, Region.Name, Region.Continent, Region.GPSLat,
Region.GPSLon, Country.NamePT AS Country
FROM Region INNER JOIN
Country ON Region.Country_ID = Country.ID
```

```
SELECT TourismRegion.ID, TourismRegion.Name, TourismRegion.NameMLS,
Town.Name AS Town, State.Name AS State, Region.Name AS Region,
Country.NamePT AS Country
FROM TourismRegion INNER JOIN
TourismRegionTown ON
TourismRegion.ID = TourismRegionTown.TourismRegion_ID INNER JOIN
Town ON TourismRegionTown.Town_ID = Town.ID INNER JOIN
State ON Town.State_ID = State.ID INNER JOIN
Region ON State.Region_ID = Region.ID INNER JOIN
Country ON Region.Country_ID = Country.ID

SELECT Town.ID,Town.Name, Town.GPSLon, Town.GPSLat, Town.ISO,
State.Name AS State, Region.Name AS Region,
Country.NamePT AS Country
FROM Town INNER JOIN
State ON Town.State_ID = State.ID INNER JOIN
Region ON State.Region_ID = Region.ID INNER JOIN
Country ON Region.Country_ID = Country.ID

SELECT Zone.ID, Zone.Name, Zone.GPSLat, Zone.GPSLon,
Neighborhood.Name AS Neighborhood, Town.Name AS Town,
State.Name AS State, Region.Name AS Region,
Country.NamePT AS Country
FROM Zone INNER JOIN
Neighborhood ON Zone.Neighborhood_ID = Neighborhood.ID INNER JOIN
Town ON Neighborhood.Town_ID = Town.ID INNER JOIN
State ON Town.State_ID = State.ID INNER JOIN
Region ON State.Region_ID = Region.ID INNER JOIN
Country ON Region.Country_ID = Country.ID

SELECT ZoneGroup.ID, ZoneGroup.Name, ZoneGroup.NameMLS,
ZoneGroup.ENABLE, Zone.Name AS Zone,
Neighborhood.Name AS Neighborhood, Town.Name AS Town,
State.Name AS State, Region.Name AS Region,
Country.NamePT AS Country
FROM ZoneGroup INNER JOIN
ZoneGroupDetails ON
ZoneGroup.ID = ZoneGroupDetails.ZoneGroupID INNER JOIN
          Zone ON Zone.ID = ZoneGroupDetails.ZoneID INNER JOIN
          Neighborhood ON Zone.Neighborhood_ID = Neighborhood.ID
          INNER JOIN
          Town ON Neighborhood.Town_ID = Town.ID INNER JOIN
          State ON Town.State_ID = State.ID INNER JOIN
          Region ON State.Region_ID = Region.ID INNER JOIN
          Country ON Region.Country_ID = Country.ID

SELECT State.ID, State.Name, State.Abrev, State.Capital,
State.GPSLat, State.GPSLon, State.ISO,
```

```
Region.Name AS Region, Country.NamePT AS Country
FROM State INNER JOIN
Region ON State.Region_ID = Region.ID INNER JOIN
          Country ON Region.Country_ID = Country.ID
```

A.8 Portals

```
SELECT  *
FROM Portals
```

A.9 RealEstate

```
SELECT RealEstateData.ID, RealEstateData.SystemID, RealEstateData.Type,
RealEstateData.EID, RealEstateData.UID, RealEstateData.Clid,
RealEstateData.Client_ID, RealEstateData.MasterClient_ID,
RealEstateData.SerialNumb, RealEstateData.Category_Name,
RealEstateData.Proptype_Name, RealEstateData.Business_Name,
RealEstateData.Zone_Name, RealEstateData.Neighborhood_Name,
RealEstateData.Town_Name, RealEstateData.State_Name,
          RealEstateData.Status_Name, RealEstateData.Year,
          RealEstateData.AreaU, RealEstateData.AreaB,
          RealEstateData.AreaT, RealEstateData.Price,
          RealEstateData.SecPrice, RealEstateData.ShowPic,
          RealEstateData.Image_EID, RealEstateData.HighLight,
          RealEstateData.Web, RealEstateData.Score, RealEstateData.Date,
          RealEstateData.LayoutFile, RealEstateData.Category_NameEN,
          RealEstateData.Proptype_NameEN, RealEstateData.Business_NameEN,
          RealEstateData.Status_NameEN, RealEstateData.WebChat,
          RealEstateData.tipopreco, RealEstateData.promoter,
          RealEstateData.conclusion_month, RealEstateData.contact_email,
          RealEstateData.groups, RealEstateData.GPSLon,
          RealEstateData.GPSLat, RealEstateData.ImgFlag,
          RealEstateData.SMS, RealEstateData.Vendedor_ID,
          RealEstateData.OtherPrice, RealEstateData.OtherPriceSec,
          RealEstateData.DateCreated, RealEstateData.Transporter,
          RealEstateData.BusinessPriceString, Image.ID AS ImageID
FROM RealEstateData LEFT OUTER JOIN
          Image ON RealEstateData.ID = Image.RealEstate_ID
```

A.10 RealEstateBusiness

```
SELECT RealEstateBusiness.Realestate_ID, RealEstateBusiness.SystemID,
RealEstateBusiness.Business_ID, RealEstateBusiness.Price,
RealEstateBusiness.PriceOtherCurrency,
RealEstateBusiness.RentType_ID, RentType.RentType_ID AS Expr1,
```

```
RentType.Name, RentType.NamePT, RentType.Days
FROM RealEstateBusiness LEFT OUTER JOIN
RentType ON RealEstateBusiness.RentType_ID = RentType.RentType_ID
```

A.11 RealEstateCatalogs

```
SELECT *
FROM RealEstateCatalogs
```

A.12 RealEstateFeature

```
SELECT RealEstateFeature.ID, RealEstateFeature.SystemID,
RealEstateFeature.Type AS FeatureType,
RealEstateFeature.Transporter, RealEstateKey.EID,
RealEstateKey.UID, RealEstateKey.Clid,
RealEstateKey.Client_ID, RealEstateKey.MasterClient_ID,
RealEstateKey.SerialNumb, RealEstateKey.Category_ID,
RealEstateKey.Business_ID, RealEstateKey.Zone_ID,
RealEstateKey.Neighborhood_ID, RealEstateKey.Town_ID,
RealEstateKey.State_ID, RealEstateKey.Region_ID,
RealEstateKey.Status_ID, RealEstateKey.AreaU, RealEstateKey.AreaB,
RealEstateKey.AreaT, RealEstateKey.Price, RealEstateKey.ShowPic,
RealEstateKey.Highlight, RealEstateKey.Web, RealEstateKey.Score,
RealEstateKey.Date, RealEstateKey.AsGarage, RealEstateKey.AsPool,
RealEstateKey.AsFurniture, RealEstateKey.IsOpenSpace,
          RealEstateKey.HeightVal, RealEstateKey.CountFeatures,
          RealEstateKey.HasDescription, RealEstateKey.Leilao_UID,
          RealEstateKey.SecPrice, RealEstateKey.hasPrice,
          RealEstateKey.hasOrderPrice, RealEstateKey.OrderPrice,
          RealEstateKey.Proptype_ID, RealEstateKey.BusinessFlags,
          RealEstateKey.CountryID, RealEstateKey.PortalFlags,
          RealEstateKey.GPSLon, RealEstateKey.GPSLat,
          RealEstateKey.EnergyCertification, RealEstateKey.FeatureFlags,
          RealEstateKey.PriceChangeDate,
          RealEstateKey.PriceChangeVariation,
          RealEstateKey.PriceChangeValue, RealEstateKey.hasGPS,
          RealEstateKey.RealEstateVersion, RealEstateKey.ImgFlag,
          RealEstateKey.TourismRegion_ID, RealEstateKey.MinPax,
          RealEstateKey.MaxPax, RealEstateKey.PriceChangeLastPrice,
          Status.Name AS status, Status.NameMLS AS StatusMLS,
          Proptype.Name AS PropTypeName, Proptype.Rooms,
          Proptype.Convertibles,
          Proptype.Floors, Proptype.NameMLS AS PropTypeMLS,
          MasterProptype.Name AS Type, MasterProptype.NameMLS AS TypeMLS
FROM RealEstateFeature INNER JOIN
RealEstateKey ON RealEstateFeature.ID = RealEstateKey.ID LEFT OUTER JOIN
```

```
Status ON Status.ID = RealEstateKey.Status_ID LEFT OUTER JOIN
Proptype ON Proptype.ID = RealEstateKey.Proptype_ID LEFT OUTER JOIN
MasterProptype ON RealEstateKey.MasterProptype_ID = MasterProptype.ID
```

A.13 RealEstatePortal

```
SELECT *
FROM RealEstatePortal
```

A.14 System

```
SELECT *
FROM System
```

A.15 Video

```
SELECT *
FROM Video
```

A.16 Videos

```
SELECT ID, RealEstate_ID
FROM Video
ORDER BY  RealEstate_ID
```

Anexo B

Ficheiros de transformação XLST

B.1 Business

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:transform
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    version="2.0">

    <xsl:template match="root">

        <xsl:for-each select="row">

            <xsl:text>ID,</xsl:text>
            <xsl:value-of select="ID" />
            <xsl:text>:</xsl:text>

            <xsl:text>Name,</xsl:text>
            <xsl:value-of select="Name" />
            <xsl:text>:</xsl:text>

            <xsl:text>BusinessFlags,</xsl:text>
            <xsl:value-of select="BusinessFlags" />
            <xsl:text>:</xsl:text>

            <xsl:text>IsComposed,</xsl:text>
            <xsl:value-of select="IsComposed" />
            <xsl:text>:</xsl:text>

            <xsl:text>NameMLS,</xsl:text>
            <xsl:value-of select="NameMLS" />
            <xsl:text>|</xsl:text>

        </xsl:for-each>
    </xsl:template>
</xsl:transform>
```

B.2 BusinessFlags

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:transform
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    version="2.0">

    <xsl:template match="root">

        <xsl:for-each select="row">

            <xsl:text>Name,</xsl:text>
            <xsl:value-of select="Name" />
            <xsl:text>:</xsl:text>

            <xsl:text>BitFlag,</xsl:text>
            <xsl:value-of select="BitFlag" />
            <xsl:text>:</xsl:text>

            <xsl:text>OriginalBusinessID,</xsl:text>
            <xsl:value-of select="OriginalBusinessID" />
            <xsl:text>|</xsl:text>

        </xsl:for-each>
    </xsl:template>
</xsl:transform>
```

B.3 Category

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:transform
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    version="2.0">

    <xsl:template match="root">

        <xsl:for-each select="row">

                <xsl:text>ID,</xsl:text>
                <xsl:value-of select="ID" />
                <xsl:text>:</xsl:text>

                <xsl:text>Name,</xsl:text>
                <xsl:value-of select="Name" />
                <xsl:text>:</xsl:text>

                <xsl:text>AsRealEstate,</xsl:text>
                <xsl:value-of select="AsRealEstate" />
                <xsl:text>:</xsl:text>

                <xsl:text>Plural,</xsl:text>
                <xsl:value-of select="Plural" />
                <xsl:text>:</xsl:text>

                <xsl:text>NameMLS,</xsl:text>
                <xsl:value-of select="NameMLS" />
                <xsl:text>:</xsl:text>

                <xsl:text>PluralMLS,</xsl:text>
                <xsl:value-of select="PluralMLS" />
                <xsl:text>:</xsl:text>

                <xsl:text>CategoryClid,</xsl:text>
                <xsl:value-of select="CategoryClid" />
                <xsl:text>:</xsl:text>

                <xsl:text>AllowsPropType,</xsl:text>
                <xsl:value-of select="AllowsPropType" />
                <xsl:text>:</xsl:text>

                <xsl:text>Type,</xsl:text>
                <xsl:value-of select="MasterCategoryName" />
                <xsl:text>:</xsl:text>
```

```
                <xsl:text>Luxury,</xsl:text>
                <xsl:value-of select="Luxury" />
                <xsl:text>:</xsl:text>

                <xsl:text>AsProptype,</xsl:text>
                <xsl:value-of select="AsProptype" />
                <xsl:text>:</xsl:text>

                <xsl:text>AsBusiness,</xsl:text>
                <xsl:value-of select="AsBusiness" />
                <xsl:text>:</xsl:text>

                <xsl:text>Priority,</xsl:text>
                <xsl:value-of select="Priority" />
                <xsl:text>:</xsl:text>

                <xsl:text>TypeMLS,</xsl:text>
                <xsl:value-of select="MasterCategoryNameMLS" />
                <xsl:text>|</xsl:text>

        </xsl:for-each>
    </xsl:template>
</xsl:transform>
```

B.4 Client

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:transform
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    version="2.0">

    <xsl:template match="root">
        <xsl:for-each select="row">
            <xsl:choose>
                <xsl:when test="position()=1">

                    <xsl:text>ID,</xsl:text>
                    <xsl:value-of select="ID" />
                    <xsl:text>:</xsl:text>

                    <xsl:text>SystemID,</xsl:text>
                    <xsl:value-of select="SystemID" />
                    <xsl:text>:</xsl:text>

                    <xsl:text>Type,</xsl:text>
                    <xsl:value-of select="Type" />
                    <xsl:text>:</xsl:text>

                    <xsl:text>MasterCliente_ID,</xsl:text>
                    <xsl:value-of select="MasterCliente_ID" />
                    <xsl:text>:</xsl:text>

                    <xsl:text>Name,</xsl:text>
                    <xsl:value-of select="Name" />
                    <xsl:text>:</xsl:text>

                    <xsl:text>Address,</xsl:text>
                    <xsl:value-of select="Address" />
                    <xsl:text>:</xsl:text>

                    <xsl:text>TownID,</xsl:text>
                    <xsl:value-of select="Town_ID" />
                    <xsl:text>:</xsl:text>

                    <xsl:text>Town_Name,</xsl:text>
                    <xsl:value-of select="Town_Name" />
                    <xsl:text>:</xsl:text>

                    <xsl:text>Phone,</xsl:text>
                    <xsl:value-of select="Phone" />
                    <xsl:text>:</xsl:text>
```

```
<xsl:text>Celular,</xsl:text>
<xsl:value-of select="Celular" />
<xsl:text>:</xsl:text>

<xsl:text>Fax,</xsl:text>
<xsl:value-of select="Fax" />
<xsl:text>:</xsl:text>

<xsl:text>Email,</xsl:text>
<xsl:value-of select="Email" />
<xsl:text>:</xsl:text>

<xsl:text>AMI,</xsl:text>
<xsl:value-of select="AMI" />
<xsl:text>:</xsl:text>

<xsl:text>OpenHours,</xsl:text>
<xsl:value-of select="OpenHours" />
<xsl:text>:</xsl:text>

<xsl:text>Web,</xsl:text>
<xsl:value-of select="Web" />
<xsl:text>:</xsl:text>

<xsl:text>Campanha_ID,</xsl:text>
<xsl:value-of select="Campanha_ID" />
<xsl:text>:</xsl:text>

<xsl:text>Score,</xsl:text>
<xsl:value-of select="Score" />
<xsl:text>:</xsl:text>

<xsl:text>Logo,</xsl:text>
<xsl:value-of select="Logo" />  <!-- Não existe por é uma
<xsl:text>:</xsl:text>

<xsl:text>Logotype,</xsl:text>
<xsl:value-of select="Logotype" />
<xsl:text>:</xsl:text>

<xsl:text>CountryID,</xsl:text>
<xsl:value-of select="CountryID" />
<xsl:text>:</xsl:text>

<xsl:text>LastActContacts,</xsl:text>
<xsl:value-of select="LastActContacts" />
```

```
        <xsl:text>:</xsl:text>

        <xsl:text>Transporter,</xsl:text>
        <xsl:value-of select="Transporter" />
        <xsl:text>:</xsl:text>

        <xsl:text>hasImage360,</xsl:text>
        <xsl:value-of select="hasImage360" />
        <xsl:text>:</xsl:text>

        <xsl:text>hasImagesActive,</xsl:text>
        <xsl:value-of select="hasImageActive" />
        <xsl:text>:</xsl:text>

        <xsl:text>hasWebChat,</xsl:text>
        <xsl:value-of select="hasWebChat" />
        <xsl:text>:</xsl:text>

        <xsl:text>hasSMS,</xsl:text>
        <xsl:value-of select="SMS" />
        <xsl:text>:</xsl:text>

        <xsl:text>hasVideos,</xsl:text>
        <xsl:value-of select="hasVideos" />
        <xsl:text>:</xsl:text>

        <xsl:text>BarometroOn,</xsl:text>
        <xsl:value-of select="BarometroOn" />
        <xsl:text>:</xsl:text>

        <xsl:text>RealEstates,</xsl:text>
        <xsl:value-of select="RealEstateID" />

    </xsl:when>
    <xsl:when test="position()=last()">
        <xsl:choose>

        <xsl:when test="preceding::ID[1]!=ID">
            <xsl:text>|</xsl:text>

            <xsl:text>ID,</xsl:text>
            <xsl:value-of select="ID" />
            <xsl:text>:</xsl:text>

            <xsl:text>SystemID,</xsl:text>
            <xsl:value-of select="SystemID" />
            <xsl:text>:</xsl:text>
```

```
<xsl:text>Type,</xsl:text>
<xsl:value-of select="Type" />
<xsl:text>:</xsl:text>

<xsl:text>MasterCliente_ID,</xsl:text>
<xsl:value-of select="MasterCliente_ID" />
<xsl:text>:</xsl:text>

<xsl:text>Name,</xsl:text>
<xsl:value-of select="Name" />
<xsl:text>:</xsl:text>

<xsl:text>Address,</xsl:text>
<xsl:value-of select="Address" />
<xsl:text>:</xsl:text>

<xsl:text>TownID,</xsl:text>
<xsl:value-of select="Town_ID" />
<xsl:text>:</xsl:text>

<xsl:text>Town_Name,</xsl:text>
<xsl:value-of select="Town_Name" />
<xsl:text>:</xsl:text>

<xsl:text>Phone,</xsl:text>
<xsl:value-of select="Phone" />
<xsl:text>:</xsl:text>

<xsl:text>Celular,</xsl:text>
<xsl:value-of select="Celular" />
<xsl:text>:</xsl:text>

<xsl:text>Fax,</xsl:text>
<xsl:value-of select="Fax" />
<xsl:text>:</xsl:text>

<xsl:text>Email,</xsl:text>
<xsl:value-of select="Email" />
<xsl:text>:</xsl:text>

<xsl:text>AMI,</xsl:text>
<xsl:value-of select="AMI" />
<xsl:text>:</xsl:text>

<xsl:text>OpenHours,</xsl:text>
<xsl:value-of select="OpenHours" />
```

```
<xsl:text>:</xsl:text>

<xsl:text>Web,</xsl:text>
<xsl:value-of select="Web" />
<xsl:text>:</xsl:text>

<xsl:text>Campanha_ID,</xsl:text>
<xsl:value-of select="Campanha_ID" />
<xsl:text>:</xsl:text>

<xsl:text>Score,</xsl:text>
<xsl:value-of select="Score" />
<xsl:text>:</xsl:text>

<xsl:text>Logo,</xsl:text>
<xsl:value-of select="Logo" />  <!-- Não existe p
<xsl:text>:</xsl:text>

<xsl:text>Logotype,</xsl:text>
<xsl:value-of select="Logotype" />
<xsl:text>:</xsl:text>

<xsl:text>CountryID,</xsl:text>
<xsl:value-of select="CountryID" />
<xsl:text>:</xsl:text>

<xsl:text>LastActContacts,</xsl:text>
<xsl:value-of select="LastActContacts" />
<xsl:text>:</xsl:text>

<xsl:text>Transporter,</xsl:text>
<xsl:value-of select="Transporter" />
<xsl:text>:</xsl:text>

<xsl:text>hasImage360,</xsl:text>
<xsl:value-of select="hasImage360" />
<xsl:text>:</xsl:text>

<xsl:text>hasImagesActive,</xsl:text>
<xsl:value-of select="hasImageActive" />
<xsl:text>:</xsl:text>

<xsl:text>hasWebChat,</xsl:text>
<xsl:value-of select="hasWebChat" />
<xsl:text>:</xsl:text>

<xsl:text>hasSMS,</xsl:text>
```

```
            <xsl:value-of select="SMS" />
            <xsl:text>:</xsl:text>

            <xsl:text>hasVideos,</xsl:text>
            <xsl:value-of select="hasVideos" />
            <xsl:text>:</xsl:text>

            <xsl:text>BarometroOn,</xsl:text>
            <xsl:value-of select="BarometroOn" />
            <xsl:text>:</xsl:text>

            <xsl:text>RealEstates,</xsl:text>
            <xsl:value-of select="RealEstateID" />

        </xsl:when>
        <xsl:otherwise>
            <xsl:text>,</xsl:text>
            <xsl:value-of select="RealEstateID" />
        </xsl:otherwise>
        </xsl:choose>
        <xsl:text>|</xsl:text>
    </xsl:when>

    <xsl:when test="preceding::ID[1]!=ID">
        <xsl:text>|</xsl:text>

        <xsl:text>ID,</xsl:text>
        <xsl:value-of select="ID" />
        <xsl:text>:</xsl:text>

        <xsl:text>SystemID,</xsl:text>
        <xsl:value-of select="SystemID" />
        <xsl:text>:</xsl:text>

        <xsl:text>Type,</xsl:text>
        <xsl:value-of select="Type" />
        <xsl:text>:</xsl:text>

        <xsl:text>MasterCliente_ID,</xsl:text>
        <xsl:value-of select="MasterCliente_ID" />
        <xsl:text>:</xsl:text>

        <xsl:text>Name,</xsl:text>
        <xsl:value-of select="Name" />
        <xsl:text>:</xsl:text>

        <xsl:text>Address,</xsl:text>
```

```
<xsl:value-of select="Address" />
<xsl:text>:</xsl:text>

<xsl:text>TownID,</xsl:text>
<xsl:value-of select="Town_ID" />
<xsl:text>:</xsl:text>

<xsl:text>Town_Name,</xsl:text>
<xsl:value-of select="Town_Name" />
<xsl:text>:</xsl:text>

<xsl:text>Phone,</xsl:text>
<xsl:value-of select="Phone" />
<xsl:text>:</xsl:text>

<xsl:text>Celular,</xsl:text>
<xsl:value-of select="Celular" />
<xsl:text>:</xsl:text>

<xsl:text>Fax,</xsl:text>
<xsl:value-of select="Fax" />
<xsl:text>:</xsl:text>

<xsl:text>Email,</xsl:text>
<xsl:value-of select="Email" />
<xsl:text>:</xsl:text>

<xsl:text>AMI,</xsl:text>
<xsl:value-of select="AMI" />
<xsl:text>:</xsl:text>

<xsl:text>OpenHours,</xsl:text>
<xsl:value-of select="OpenHours" />
<xsl:text>:</xsl:text>

<xsl:text>Web,</xsl:text>
<xsl:value-of select="Web" />
<xsl:text>:</xsl:text>

<xsl:text>Campanha_ID,</xsl:text>
<xsl:value-of select="Campanha_ID" />
<xsl:text>:</xsl:text>

<xsl:text>Score,</xsl:text>
<xsl:value-of select="Score" />
<xsl:text>:</xsl:text>
```

```
<xsl:text>Logo,</xsl:text>
<xsl:value-of select="Logo" />  <!-- Não existe por é uma
<xsl:text>:</xsl:text>

<xsl:text>Logotype,</xsl:text>
<xsl:value-of select="Logotype" />
<xsl:text>:</xsl:text>

<xsl:text>CountryID,</xsl:text>
<xsl:value-of select="CountryID" />
<xsl:text>:</xsl:text>

<xsl:text>LastActContacts,</xsl:text>
<xsl:value-of select="LastActContacts" />
<xsl:text>:</xsl:text>

<xsl:text>Transporter,</xsl:text>
<xsl:value-of select="Transporter" />
<xsl:text>:</xsl:text>

<xsl:text>hasImage360,</xsl:text>
<xsl:value-of select="hasImage360" />
<xsl:text>:</xsl:text>

<xsl:text>hasImagesActive,</xsl:text>
<xsl:value-of select="hasImageActive" />
<xsl:text>:</xsl:text>

<xsl:text>hasWebChat,</xsl:text>
<xsl:value-of select="hasWebChat" />
<xsl:text>:</xsl:text>

<xsl:text>hasSMS,</xsl:text>
<xsl:value-of select="SMS" />
<xsl:text>:</xsl:text>

<xsl:text>hasVideos,</xsl:text>
<xsl:value-of select="hasVideos" />
<xsl:text>:</xsl:text>

<xsl:text>BarometroOn,</xsl:text>
<xsl:value-of select="BarometroOn" />
<xsl:text>:</xsl:text>

<xsl:text>RealEstates,</xsl:text>
<xsl:value-of select="RealEstateID" />
```

```
                </xsl:when>
                <xsl:otherwise>
                    <xsl:text>,</xsl:text>
                    <xsl:value-of select="RealEstateID" />
                </xsl:otherwise>
            </xsl:choose>

        </xsl:for-each>
    </xsl:template>
</xsl:transform>
```

B.5 FeatureFlags

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:transform
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    version="2.0">

    <xsl:template match="root">

        <xsl:for-each select="row">

            <xsl:text>Name,</xsl:text>
            <xsl:value-of select="Name" />
            <xsl:text>:</xsl:text>

            <xsl:text>Tag,</xsl:text>
            <xsl:value-of select="Tag" />
            <xsl:text>:</xsl:text>

            <xsl:text>Flag,</xsl:text>
            <xsl:value-of select="Flag" />
            <xsl:text>|</xsl:text>

        </xsl:for-each>
    </xsl:template>
</xsl:transform>
```

B.6 Imagem

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:transform
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    version="2.0">

    <xsl:template match="root">

        <xsl:for-each select="row">

            <xsl:text>ID,</xsl:text>
            <xsl:value-of select="ID" />
            <xsl:text>:</xsl:text>

            <xsl:text>RealEstate_ID,</xsl:text>
            <xsl:value-of select="RealEstate_ID" />
            <xsl:text>:</xsl:text>

            <xsl:text>System_ID,</xsl:text>
            <xsl:value-of select="SystemID" />
            <xsl:text>:</xsl:text>

            <xsl:text>EID,</xsl:text>
            <xsl:value-of select="EID" />
            <xsl:text>:</xsl:text>

            <xsl:text>Client_ID,</xsl:text>
            <xsl:value-of select="Client_ID" />
            <xsl:text>:</xsl:text>

            <xsl:text>RealEstateType,</xsl:text>
            <xsl:value-of select="RealEstate_Type" />
            <xsl:text>:</xsl:text>

            <xsl:text>Description,</xsl:text>
            <xsl:value-of select="Description" />
            <xsl:text>:</xsl:text>

            <xsl:text>DescriptionEn,</xsl:text>
            <xsl:value-of select="DescriptionEn" />
            <xsl:text>:</xsl:text>

            <xsl:text>Priority,</xsl:text>
            <xsl:value-of select="Priority" />
            <xsl:text>:</xsl:text>
```

```
            <xsl:text>Ext,</xsl:text>
            <xsl:value-of select="Ext" />
            <xsl:text>:</xsl:text>

            <xsl:text>DescriptionMLS,</xsl:text>
            <xsl:value-of select="DescriptionMLS" />
            <xsl:text>|</xsl:text>

        </xsl:for-each>
    </xsl:template>
</xsl:transform>
```

B.7 Localization

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:transform
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    version="2.0">

    <xsl:template match="root">

        <xsl:for-each select="ZoneGroup">

            <xsl:text>ID,</xsl:text>
            <xsl:value-of select="ID" />
            <xsl:text>:</xsl:text>

            <xsl:text>Name,</xsl:text>
            <xsl:value-of select="Name" />
            <xsl:text>:</xsl:text>

            <xsl:text>Type,ZoneGroup:</xsl:text>

            <xsl:text>NameMLS,</xsl:text>
            <xsl:value-of select="NameMLS" />
            <xsl:text>:</xsl:text>

            <xsl:text>Enable,</xsl:text>
            <xsl:value-of select="ENABLE" />
            <xsl:text>:</xsl:text>

            <xsl:text>BelongsTo,</xsl:text>
            <xsl:value-of select="Zone" /><xsl:text>,</xsl:text>
            <xsl:value-of select="Neighborhood" /><xsl:text>,</xsl:text>
            <xsl:value-of select="Town" /><xsl:text>,</xsl:text>
            <xsl:value-of select="State" /><xsl:text>,</xsl:text>
            <xsl:value-of select="Region" /><xsl:text>,</xsl:text>
            <xsl:value-of select="Country" /><xsl:text>|</xsl:text>

        </xsl:for-each>

        <xsl:for-each select="Zone">

            <xsl:text>ID,</xsl:text>
            <xsl:value-of select="ID" />
            <xsl:text>:</xsl:text>

            <xsl:text>Name,</xsl:text>
            <xsl:value-of select="Name" />
```

```
        <xsl:text>:</xsl:text>

        <xsl:text>GPSLon,</xsl:text>
        <xsl:value-of select="GPSLon" />
        <xsl:text>:</xsl:text>

        <xsl:text>GPSLat,</xsl:text>
        <xsl:value-of select="GPSLat" />
        <xsl:text>:</xsl:text>

        <xsl:text>Type,Zone:</xsl:text>

        <xsl:text>BelongsTo,</xsl:text>
        <xsl:value-of select="Neighborhood" /><xsl:text>,</xsl:text>
        <xsl:value-of select="Town" /><xsl:text>,</xsl:text>
        <xsl:value-of select="State" /><xsl:text>,</xsl:text>
        <xsl:value-of select="Region" /><xsl:text>,</xsl:text>
        <xsl:value-of select="Country" /><xsl:text>|</xsl:text>

    </xsl:for-each>

    <xsl:for-each select="Neighborhood">

        <xsl:text>ID,</xsl:text>
        <xsl:value-of select="ID" />
        <xsl:text>:</xsl:text>

        <xsl:text>Name,</xsl:text>
        <xsl:value-of select="Name" />
        <xsl:text>:</xsl:text>

        <xsl:text>GPSLon,</xsl:text>
        <xsl:value-of select="GPSLon" />
        <xsl:text>:</xsl:text>

        <xsl:text>GPSLat,</xsl:text>
        <xsl:value-of select="GPSLat" />
        <xsl:text>:</xsl:text>

        <xsl:text>Type,Neighborhood:</xsl:text>

        <xsl:text>Urban,</xsl:text>
        <xsl:value-of select="Urban" />
        <xsl:text>:</xsl:text>

        <xsl:text>ISO,</xsl:text>
        <xsl:value-of select="ISO" />
```

```
        <xsl:text>:</xsl:text>

        <xsl:text>BelongsTo,</xsl:text>
        <xsl:value-of select="Town" /><xsl:text>,</xsl:text>
        <xsl:value-of select="State" /><xsl:text>,</xsl:text>
        <xsl:value-of select="Region" /><xsl:text>,</xsl:text>
        <xsl:value-of select="Country" /><xsl:text>|</xsl:text>

    </xsl:for-each>

    <xsl:for-each select="Town">

        <xsl:text>ID,</xsl:text>
        <xsl:value-of select="ID" />
        <xsl:text>:</xsl:text>

        <xsl:text>Name,</xsl:text>
        <xsl:value-of select="Name" />
        <xsl:text>:</xsl:text>

        <xsl:text>GPSLon,</xsl:text>
        <xsl:value-of select="GPSLon" />
        <xsl:text>:</xsl:text>

        <xsl:text>GPSLat,</xsl:text>
        <xsl:value-of select="GPSLat" />
        <xsl:text>:</xsl:text>

        <xsl:text>Type,Town:</xsl:text>

        <xsl:text>ISO,</xsl:text>
        <xsl:value-of select="ISO" />
        <xsl:text>:</xsl:text>

        <xsl:text>BelongsTo,</xsl:text>
        <xsl:value-of select="State" /><xsl:text>,</xsl:text>
        <xsl:value-of select="Region" /><xsl:text>,</xsl:text>
        <xsl:value-of select="Country" /><xsl:text>|</xsl:text>
    </xsl:for-each>

    <xsl:for-each select="TourismRegion">

        <xsl:text>ID,</xsl:text>
        <xsl:value-of select="ID" />
        <xsl:text>:</xsl:text>

        <xsl:text>Name,</xsl:text>
```

```
    <xsl:value-of select="Name" />
    <xsl:text>:</xsl:text>

    <xsl:text>Type,TourismRegion:</xsl:text>

    <xsl:text>NameMLS,</xsl:text>
    <xsl:value-of select="NameMLS" />
    <xsl:text>:</xsl:text>

    <xsl:text>BelongsTo,</xsl:text>
    <xsl:value-of select="Town" /><xsl:text>,</xsl:text>
    <xsl:value-of select="State" /><xsl:text>,</xsl:text>
    <xsl:value-of select="Region" /><xsl:text>,</xsl:text>
    <xsl:value-of select="Country" /><xsl:text>|</xsl:text>
</xsl:for-each>

<xsl:for-each select="State">

    <xsl:text>ID,</xsl:text>
    <xsl:value-of select="ID" />
    <xsl:text>:</xsl:text>

    <xsl:text>Name,</xsl:text>
    <xsl:value-of select="Name" />
    <xsl:text>:</xsl:text>

    <xsl:text>GPSLon,</xsl:text>
    <xsl:value-of select="GPSLon" />
    <xsl:text>:</xsl:text>

    <xsl:text>GPSLat,</xsl:text>
    <xsl:value-of select="GPSLat" />
    <xsl:text>:</xsl:text>

    <xsl:text>Type,State:</xsl:text>

    <xsl:text>Capital,</xsl:text>
    <xsl:value-of select="Capital" />
    <xsl:text>:</xsl:text>

    <xsl:text>Abrev,</xsl:text>
    <xsl:value-of select="Abrev" />
    <xsl:text>:</xsl:text>

    <xsl:text>ISO,</xsl:text>
    <xsl:value-of select="ISO" />
    <xsl:text>:</xsl:text>
```

```
    <xsl:text>BelongsTo,</xsl:text>
    <xsl:value-of select="Region" /><xsl:text>,</xsl:text>
    <xsl:value-of select="Country" /><xsl:text>|</xsl:text>
</xsl:for-each>

<xsl:for-each select="Region">

    <xsl:text>ID,</xsl:text>
    <xsl:value-of select="ID" />
    <xsl:text>:</xsl:text>

    <xsl:text>Name,</xsl:text>
    <xsl:value-of select="Name" />
    <xsl:text>:</xsl:text>

    <xsl:text>GPSLon,</xsl:text>
    <xsl:value-of select="GPSLon" />
    <xsl:text>:</xsl:text>

    <xsl:text>GPSLat,</xsl:text>
    <xsl:value-of select="GPSLat" />
    <xsl:text>:</xsl:text>

    <xsl:text>Type,Region:</xsl:text>

    <xsl:text>Continent,</xsl:text>
    <xsl:value-of select="Continent" />
    <xsl:text>:</xsl:text>

    <xsl:text>BelongsTo,</xsl:text>
    <xsl:value-of select="Country" /><xsl:text>|</xsl:text>
</xsl:for-each>

<xsl:for-each select="Country">

    <xsl:text>ID,</xsl:text>
    <xsl:value-of select="ID" />
    <xsl:text>:</xsl:text>

    <xsl:text>Name,</xsl:text>
    <xsl:value-of select="NamePT" />
    <xsl:text>:</xsl:text>

    <xsl:text>GPSLon,</xsl:text>
    <xsl:value-of select="GPSLon" />
    <xsl:text>:</xsl:text>
```

```
            <xsl:text>GPSLat,</xsl:text>
            <xsl:value-of select="GPSLat" />
            <xsl:text>:</xsl:text>

            <xsl:text>Type,Country:</xsl:text>

            <xsl:text>ISO,</xsl:text>
            <xsl:value-of select="ISO" />
            <xsl:text>:</xsl:text>

            <xsl:text>ISO2,</xsl:text>
            <xsl:value-of select="ISO2" />
            <xsl:text>:</xsl:text>

            <xsl:text>ISO3,</xsl:text>
            <xsl:value-of select="ISO3" />
            <xsl:text>:</xsl:text>

            <xsl:text>IsActive,</xsl:text>
            <xsl:value-of select="IsActive" />
            <xsl:text>:</xsl:text>

            <xsl:text>NameMLS,</xsl:text>
            <xsl:value-of select="NameMLS" />
            <xsl:text>|</xsl:text>
        </xsl:for-each>

    </xsl:template>
</xsl:transform>
```

B.8 Portals

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:transform
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    version="2.0">

    <xsl:template match="root">

        <xsl:for-each select="row">

            <xsl:text>ID,</xsl:text>
            <xsl:value-of select="ID" />
            <xsl:text>:</xsl:text>

            <xsl:text>Name,</xsl:text>
            <xsl:value-of select="Name" />
            <xsl:text>:</xsl:text>

            <xsl:text>Enable,</xsl:text>
            <xsl:value-of select="Enable" />
            <xsl:text>:</xsl:text>

            <xsl:text>PortalFlags,</xsl:text>
            <xsl:value-of select="PortalFlags" />
            <xsl:text>|</xsl:text>

        </xsl:for-each>
    </xsl:template>
</xsl:transform>
```

B.9 RealEstate

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:transform
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    version="2.0">

    <xsl:template match="root">

        <xsl:for-each select="row">
            <xsl:choose>
                <xsl:when test="position()=1">

                    <xsl:text>RealEstateID,</xsl:text>
                    <xsl:value-of select="ID" />
                    <xsl:text>:</xsl:text>

                    <xsl:text>SystemID,</xsl:text>
                    <xsl:value-of select="SystemID" />
                    <xsl:text>:</xsl:text>

                    <xsl:text>Type,</xsl:text>
                    <xsl:value-of select="Type" />
                    <xsl:text>:</xsl:text>

                      <xsl:text>EID,</xsl:text>
                    <xsl:value-of select="EID" />
                    <xsl:text>:</xsl:text>

                    <xsl:text>Clid,</xsl:text>
                    <xsl:value-of select="Clid" />
                    <xsl:text>:</xsl:text>

                    <xsl:text>Client_ID,</xsl:text>
                    <xsl:value-of select="Client_ID" />
                    <xsl:text>:</xsl:text>

                    <xsl:text>MasterClient_ID,</xsl:text>
                    <xsl:value-of select="MasterClient_ID" />
                    <xsl:text>:</xsl:text>

                    <xsl:text>SerialNumb,</xsl:text>
                    <xsl:value-of select="SerialNumb" />
                    <xsl:text>:</xsl:text>

                    <xsl:text>Category_Name,</xsl:text>
                    <xsl:value-of select="Category_Name" />
```

```
<xsl:text>:</xsl:text>

<xsl:text>Proptype_Name,</xsl:text>
<xsl:value-of select="Proptype_Name" />
<xsl:text>:</xsl:text>

<xsl:text>Business_Name,</xsl:text>
<xsl:value-of select="Business_Name" />
<xsl:text>:</xsl:text>

<xsl:text>Zone_Name,</xsl:text>
<xsl:value-of select="Zone_Name" />
<xsl:text>:</xsl:text>

<xsl:text>Neighborhood_Name,</xsl:text>
<xsl:value-of select="Neighborhood_Name" />
<xsl:text>:</xsl:text>

<xsl:text>Town_Name,</xsl:text>
<xsl:value-of select="Town_Name" />
<xsl:text>:</xsl:text>

<xsl:text>State_Name,</xsl:text>
<xsl:value-of select="State_Name" />
<xsl:text>:</xsl:text>

<xsl:text>Status_Name,</xsl:text>
<xsl:value-of select="Status_Name" />
<xsl:text>:</xsl:text>

<xsl:text>Year,</xsl:text>
<xsl:value-of select="Year" />
<xsl:text>:</xsl:text>

<xsl:text>AreaU,</xsl:text>
<xsl:value-of select="AreaU" />
<xsl:text>:</xsl:text>

<xsl:text>AreaB,</xsl:text>
<xsl:value-of select="AreaB" />
<xsl:text>:</xsl:text>

<xsl:text>AreaT,</xsl:text>
<xsl:value-of select="AreaT" />
<xsl:text>:</xsl:text>

<xsl:text>Price,</xsl:text>
```

```
<xsl:value-of select="Price" />
<xsl:text>:</xsl:text>

<xsl:text>SecPrice,</xsl:text>
<xsl:value-of select="SecPrice" />
<xsl:text>:</xsl:text>

<xsl:text>ShowPic,</xsl:text>
<xsl:value-of select="ShowPic" />
<xsl:text>:</xsl:text>

<xsl:text>Image_EID,</xsl:text>
<xsl:value-of select="Image_EID" />
<xsl:text>:</xsl:text>

<xsl:text>HighLight,</xsl:text>
<xsl:value-of select="HighLight" />
<xsl:text>:</xsl:text>

<xsl:text>Web,</xsl:text>
<xsl:value-of select="Web" />
<xsl:text>:</xsl:text>

<xsl:text>Score,</xsl:text>
<xsl:value-of select="Score" />
<xsl:text>:</xsl:text>

<xsl:text>Date,</xsl:text>
<xsl:value-of select="Date" />
<xsl:text>:</xsl:text>

<xsl:text>LayoutFile,</xsl:text>
<xsl:value-of select="LayoutFile" />
<xsl:text>:</xsl:text>

<xsl:text>Category_NameEN,</xsl:text>
<xsl:value-of select="Category_NameEN" />
<xsl:text>:</xsl:text>

<xsl:text>Proptype_NameEN,</xsl:text>
<xsl:value-of select="Proptype_NameEN" />
<xsl:text>:</xsl:text>

<xsl:text>Business_NameEN,</xsl:text>
<xsl:value-of select="Business_NameEN" />
<xsl:text>:</xsl:text>
```

```
<xsl:text>Status_NameEN,</xsl:text>
<xsl:value-of select="Status_NameEN" />
<xsl:text>:</xsl:text>

<xsl:text>WebChat,</xsl:text>
<xsl:value-of select="WebChat" />
<xsl:text>:</xsl:text>

<xsl:text>tipopreco,</xsl:text>
<xsl:value-of select="tipopreco" />
<xsl:text>:</xsl:text>

<xsl:text>promoter,</xsl:text>
<xsl:value-of select="promoter" />
<xsl:text>:</xsl:text>

<xsl:text>conclusion_month,</xsl:text>
<xsl:value-of select="conclusion_month" />
<xsl:text>:</xsl:text>

<xsl:text>contact_email,</xsl:text>
<xsl:value-of select="contact_email" />
<xsl:text>:</xsl:text>

<xsl:text>groups,</xsl:text>
<xsl:value-of select="groups" />
<xsl:text>:</xsl:text>

<xsl:text>GPSLon,</xsl:text>
<xsl:value-of select="GPSLon" />
<xsl:text>:</xsl:text>

<xsl:text>GPSLat,</xsl:text>
<xsl:value-of select="GPSLat" />
<xsl:text>:</xsl:text>

<xsl:text>ImgFlag,</xsl:text>
<xsl:value-of select="ImgFlag" />
<xsl:text>:</xsl:text>

<xsl:text>SMS,</xsl:text>
<xsl:value-of select="SMS" />
<xsl:text>:</xsl:text>

<xsl:text>Vendedor_ID,</xsl:text>
<xsl:value-of select="Vendedor_ID" />
<xsl:text>:</xsl:text>
```

```
        <xsl:text>OtherPrice,</xsl:text>
        <xsl:value-of select="OtherPrice" />
        <xsl:text>:</xsl:text>

        <xsl:text>OtherPriceSec,</xsl:text>
        <xsl:value-of select="OtherPriceSec" />
        <xsl:text>:</xsl:text>

        <xsl:text>DateCreated,</xsl:text>
        <xsl:value-of select="DateCreated" />
        <xsl:text>:</xsl:text>

        <xsl:text>Transporter,</xsl:text>
        <xsl:value-of select="Transporter" />
        <xsl:text>:</xsl:text>

        <xsl:text>BusinessPriceString,</xsl:text>
        <xsl:value-of select="BusinessPriceString" />
        <xsl:text>:</xsl:text>

        <xsl:text>Images,</xsl:text>
        <xsl:value-of select="ImageID" />
    </xsl:when>
    <xsl:when test="position()=last()">
        <xsl:choose>

            <xsl:when test="preceding::ID[1]!=ID">
                <xsl:text>|</xsl:text>

                <xsl:text>RealEstateID,</xsl:text>
                <xsl:value-of select="ID" />
                <xsl:text>:</xsl:text>

                <xsl:text>SystemID,</xsl:text>
                <xsl:value-of select="SystemID" />
                <xsl:text>:</xsl:text>

                <xsl:text>Type,</xsl:text>
                <xsl:value-of select="Type" />
                <xsl:text>:</xsl:text>

                <xsl:text>EID,</xsl:text>
                <xsl:value-of select="EID" />
                <xsl:text>:</xsl:text>

                <xsl:text>Clid,</xsl:text>
```

```
<xsl:value-of select="Clid" />
<xsl:text>:</xsl:text>

<xsl:text>Client_ID,</xsl:text>
<xsl:value-of select="Client_ID" />
<xsl:text>:</xsl:text>

<xsl:text>MasterClient_ID,</xsl:text>
<xsl:value-of select="MasterClient_ID" />
<xsl:text>:</xsl:text>

<xsl:text>SerialNumb,</xsl:text>
<xsl:value-of select="SerialNumb" />
<xsl:text>:</xsl:text>

<xsl:text>Category_Name,</xsl:text>
<xsl:value-of select="Category_Name" />
<xsl:text>:</xsl:text>

<xsl:text>Proptype_Name,</xsl:text>
<xsl:value-of select="Proptype_Name" />
<xsl:text>:</xsl:text>

<xsl:text>Business_Name,</xsl:text>
<xsl:value-of select="Business_Name" />
<xsl:text>:</xsl:text>

<xsl:text>Zone_Name,</xsl:text>
<xsl:value-of select="Zone_Name" />
<xsl:text>:</xsl:text>

<xsl:text>Neighborhood_Name,</xsl:text>
<xsl:value-of select="Neighborhood_Name" />
<xsl:text>:</xsl:text>

<xsl:text>Town_Name,</xsl:text>
<xsl:value-of select="Town_Name" />
<xsl:text>:</xsl:text>

<xsl:text>State_Name,</xsl:text>
<xsl:value-of select="State_Name" />
<xsl:text>:</xsl:text>

<xsl:text>Status_Name,</xsl:text>
<xsl:value-of select="Status_Name" />
<xsl:text>:</xsl:text>
```

```
<xsl:text>Year,</xsl:text>
<xsl:value-of select="Year" />
<xsl:text>:</xsl:text>

<xsl:text>AreaU,</xsl:text>
<xsl:value-of select="AreaU" />
<xsl:text>:</xsl:text>

<xsl:text>AreaB,</xsl:text>
<xsl:value-of select="AreaB" />
<xsl:text>:</xsl:text>

<xsl:text>AreaT,</xsl:text>
<xsl:value-of select="AreaT" />
<xsl:text>:</xsl:text>

<xsl:text>Price,</xsl:text>
<xsl:value-of select="Price" />
<xsl:text>:</xsl:text>

<xsl:text>SecPrice,</xsl:text>
<xsl:value-of select="SecPrice" />
<xsl:text>:</xsl:text>

<xsl:text>ShowPic,</xsl:text>
<xsl:value-of select="ShowPic" />
<xsl:text>:</xsl:text>

<xsl:text>Image_EID,</xsl:text>
<xsl:value-of select="Image_EID" />
<xsl:text>:</xsl:text>

<xsl:text>HighLight,</xsl:text>
<xsl:value-of select="HighLight" />
<xsl:text>:</xsl:text>

<xsl:text>Web,</xsl:text>
<xsl:value-of select="Web" />
<xsl:text>:</xsl:text>

<xsl:text>Score,</xsl:text>
<xsl:value-of select="Score" />
<xsl:text>:</xsl:text>

<xsl:text>Date,</xsl:text>
<xsl:value-of select="Date" />
<xsl:text>:</xsl:text>
```

```
<xsl:text>LayoutFile,</xsl:text>
<xsl:value-of select="LayoutFile" />
<xsl:text>:</xsl:text>

<xsl:text>Category_NameEN,</xsl:text>
<xsl:value-of select="Category_NameEN" />
<xsl:text>:</xsl:text>

<xsl:text>Proptype_NameEN,</xsl:text>
<xsl:value-of select="Proptype_NameEN" />
<xsl:text>:</xsl:text>

<xsl:text>Business_NameEN,</xsl:text>
<xsl:value-of select="Business_NameEN" />
<xsl:text>:</xsl:text>

<xsl:text>Status_NameEN,</xsl:text>
<xsl:value-of select="Status_NameEN" />
<xsl:text>:</xsl:text>

<xsl:text>WebChat,</xsl:text>
<xsl:value-of select="WebChat" />
<xsl:text>:</xsl:text>

<xsl:text>tipopreco,</xsl:text>
<xsl:value-of select="tipopreco" />
<xsl:text>:</xsl:text>

<xsl:text>promoter,</xsl:text>
<xsl:value-of select="promoter" />
<xsl:text>:</xsl:text>

<xsl:text>conclusion_month,</xsl:text>
<xsl:value-of select="conclusion_month" />
<xsl:text>:</xsl:text>

<xsl:text>contact_email,</xsl:text>
<xsl:value-of select="contact_email" />
<xsl:text>:</xsl:text>

<xsl:text>groups,</xsl:text>
<xsl:value-of select="groups" />
<xsl:text>:</xsl:text>

<xsl:text>GPSLon,</xsl:text>
<xsl:value-of select="GPSLon" />
```

```
            <xsl:text>:</xsl:text>

            <xsl:text>GPSLat,</xsl:text>
            <xsl:value-of select="GPSLat" />
            <xsl:text>:</xsl:text>

            <xsl:text>ImgFlag,</xsl:text>
            <xsl:value-of select="ImgFlag" />
            <xsl:text>:</xsl:text>

            <xsl:text>SMS,</xsl:text>
            <xsl:value-of select="SMS" />
            <xsl:text>:</xsl:text>

            <xsl:text>Vendedor_ID,</xsl:text>
            <xsl:value-of select="Vendedor_ID" />
            <xsl:text>:</xsl:text>

            <xsl:text>OtherPrice,</xsl:text>
            <xsl:value-of select="OtherPrice" />
            <xsl:text>:</xsl:text>

            <xsl:text>OtherPriceSec,</xsl:text>
            <xsl:value-of select="OtherPriceSec" />
            <xsl:text>:</xsl:text>

            <xsl:text>DateCreated,</xsl:text>
            <xsl:value-of select="DateCreated" />
            <xsl:text>:</xsl:text>

            <xsl:text>Transporter,</xsl:text>
            <xsl:value-of select="Transporter" />
            <xsl:text>:</xsl:text>

            <xsl:text>BusinessPriceString,</xsl:text>
            <xsl:value-of select="BusinessPriceString" />
            <xsl:text>:</xsl:text>

            <xsl:text>Images,</xsl:text>
            <xsl:value-of select="ImageID" />

        </xsl:when>
        <xsl:otherwise>
            <xsl:text>,</xsl:text>
            <xsl:value-of select="ImageID" />
        </xsl:otherwise>
    </xsl:choose>
```

```
    <xsl:text>|</xsl:text>
</xsl:when>

<xsl:when test="preceding::ID[1]!=ID">
    <xsl:text>|</xsl:text>

    <xsl:text>RealEstateID,</xsl:text>
    <xsl:value-of select="ID" />
    <xsl:text>:</xsl:text>

    <xsl:text>SystemID,</xsl:text>
    <xsl:value-of select="SystemID" />
    <xsl:text>:</xsl:text>

    <xsl:text>Type,</xsl:text>
    <xsl:value-of select="Type" />
    <xsl:text>:</xsl:text>

    <xsl:text>EID,</xsl:text>
    <xsl:value-of select="EID" />
    <xsl:text>:</xsl:text>

    <xsl:text>Clid,</xsl:text>
    <xsl:value-of select="Clid" />
    <xsl:text>:</xsl:text>

    <xsl:text>Client_ID,</xsl:text>
    <xsl:value-of select="Client_ID" />
    <xsl:text>:</xsl:text>

    <xsl:text>MasterClient_ID,</xsl:text>
    <xsl:value-of select="MasterClient_ID" />
    <xsl:text>:</xsl:text>

    <xsl:text>SerialNumb,</xsl:text>
    <xsl:value-of select="SerialNumb" />
    <xsl:text>:</xsl:text>

    <xsl:text>Category_Name,</xsl:text>
    <xsl:value-of select="Category_Name" />
    <xsl:text>:</xsl:text>

    <xsl:text>Proptype_Name,</xsl:text>
    <xsl:value-of select="Proptype_Name" />
    <xsl:text>:</xsl:text>

    <xsl:text>Business_Name,</xsl:text>
```

```
<xsl:value-of select="Business_Name" />
<xsl:text>:</xsl:text>

<xsl:text>Zone_Name,</xsl:text>
<xsl:value-of select="Zone_Name" />
<xsl:text>:</xsl:text>

<xsl:text>Neighborhood_Name,</xsl:text>
<xsl:value-of select="Neighborhood_Name" />
<xsl:text>:</xsl:text>

<xsl:text>Town_Name,</xsl:text>
<xsl:value-of select="Town_Name" />
<xsl:text>:</xsl:text>

<xsl:text>State_Name,</xsl:text>
<xsl:value-of select="State_Name" />
<xsl:text>:</xsl:text>

<xsl:text>Status_Name,</xsl:text>
<xsl:value-of select="Status_Name" />
<xsl:text>:</xsl:text>

<xsl:text>Year,</xsl:text>
<xsl:value-of select="Year" />
<xsl:text>:</xsl:text>

<xsl:text>AreaU,</xsl:text>
<xsl:value-of select="AreaU" />
<xsl:text>:</xsl:text>

<xsl:text>AreaB,</xsl:text>
<xsl:value-of select="AreaB" />
<xsl:text>:</xsl:text>

<xsl:text>AreaT,</xsl:text>
<xsl:value-of select="AreaT" />
<xsl:text>:</xsl:text>

<xsl:text>Price,</xsl:text>
<xsl:value-of select="Price" />
<xsl:text>:</xsl:text>

<xsl:text>SecPrice,</xsl:text>
<xsl:value-of select="SecPrice" />
<xsl:text>:</xsl:text>
```

```
<xsl:text>ShowPic,</xsl:text>
<xsl:value-of select="ShowPic" />
<xsl:text>:</xsl:text>

<xsl:text>Image_EID,</xsl:text>
<xsl:value-of select="Image_EID" />
<xsl:text>:</xsl:text>

<xsl:text>HighLight,</xsl:text>
<xsl:value-of select="HighLight" />
<xsl:text>:</xsl:text>

<xsl:text>Web,</xsl:text>
<xsl:value-of select="Web" />
<xsl:text>:</xsl:text>

<xsl:text>Score,</xsl:text>
<xsl:value-of select="Score" />
<xsl:text>:</xsl:text>

<xsl:text>Date,</xsl:text>
<xsl:value-of select="Date" />
<xsl:text>:</xsl:text>

<xsl:text>LayoutFile,</xsl:text>
<xsl:value-of select="LayoutFile" />
<xsl:text>:</xsl:text>

<xsl:text>Category_NameEN,</xsl:text>
<xsl:value-of select="Category_NameEN" />
<xsl:text>:</xsl:text>

<xsl:text>Proptype_NameEN,</xsl:text>
<xsl:value-of select="Proptype_NameEN" />
<xsl:text>:</xsl:text>

<xsl:text>Business_NameEN,</xsl:text>
<xsl:value-of select="Business_NameEN" />
<xsl:text>:</xsl:text>

<xsl:text>Status_NameEN,</xsl:text>
<xsl:value-of select="Status_NameEN" />
<xsl:text>:</xsl:text>

<xsl:text>WebChat,</xsl:text>
<xsl:value-of select="WebChat" />
<xsl:text>:</xsl:text>
```

```
<xsl:text>tipopreco,</xsl:text>
<xsl:value-of select="tipopreco" />
<xsl:text>:</xsl:text>

<xsl:text>promoter,</xsl:text>
<xsl:value-of select="promoter" />
<xsl:text>:</xsl:text>

<xsl:text>conclusion_month,</xsl:text>
<xsl:value-of select="conclusion_month" />
<xsl:text>:</xsl:text>

<xsl:text>contact_email,</xsl:text>
<xsl:value-of select="contact_email" />
<xsl:text>:</xsl:text>

<xsl:text>groups,</xsl:text>
<xsl:value-of select="groups" />
<xsl:text>:</xsl:text>

<xsl:text>GPSLon,</xsl:text>
<xsl:value-of select="GPSLon" />
<xsl:text>:</xsl:text>

<xsl:text>GPSLat,</xsl:text>
<xsl:value-of select="GPSLat" />
<xsl:text>:</xsl:text>

<xsl:text>ImgFlag,</xsl:text>
<xsl:value-of select="ImgFlag" />
<xsl:text>:</xsl:text>

<xsl:text>SMS,</xsl:text>
<xsl:value-of select="SMS" />
<xsl:text>:</xsl:text>

<xsl:text>Vendedor_ID,</xsl:text>
<xsl:value-of select="Vendedor_ID" />
<xsl:text>:</xsl:text>

<xsl:text>OtherPrice,</xsl:text>
<xsl:value-of select="OtherPrice" />
<xsl:text>:</xsl:text>

<xsl:text>OtherPriceSec,</xsl:text>
<xsl:value-of select="OtherPriceSec" />
```

```
                    <xsl:text>:</xsl:text>

                    <xsl:text>DateCreated,</xsl:text>
                    <xsl:value-of select="DateCreated" />
                    <xsl:text>:</xsl:text>

                    <xsl:text>Transporter,</xsl:text>
                    <xsl:value-of select="Transporter" />
                    <xsl:text>:</xsl:text>

                    <xsl:text>BusinessPriceString,</xsl:text>
                    <xsl:value-of select="BusinessPriceString" />
                    <xsl:text>:</xsl:text>

                    <xsl:text>Images,</xsl:text>
                    <xsl:value-of select="ImageID" />
                </xsl:when>
                <xsl:otherwise>
                    <xsl:text>,</xsl:text>
                    <xsl:value-of select="ImageID" />
                </xsl:otherwise>
            </xsl:choose>


        </xsl:for-each>
    </xsl:template>
</xsl:transform>
```

B.10 RealEstateBusiness

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:transform
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    version="2.0">

    <xsl:template match="root">

        <xsl:for-each select="row">

            <xsl:text>RealEstateID,</xsl:text>
            <xsl:value-of select="Realestate_ID" />
            <xsl:text>:</xsl:text>

            <xsl:text>Business_ID,</xsl:text>
            <xsl:value-of select="Business_ID" />
            <xsl:text>:</xsl:text>

            <xsl:text>SystemID,</xsl:text>
            <xsl:value-of select="SystemID" />
            <xsl:text>:</xsl:text>

            <xsl:text>Price,</xsl:text>
            <xsl:value-of select="Price" />
            <xsl:text>:</xsl:text>

            <xsl:text>PriceOtherCurrency,</xsl:text>
            <xsl:value-of select="PriceOtherCurrency" />
            <xsl:text>:</xsl:text>

            <xsl:text>RentTypeID,</xsl:text>
            <xsl:value-of select="RentType_ID" />

            <xsl:if test="RentType_ID!=0">
                <xsl:text>:RentTypeNameMLS,</xsl:text>
                <xsl:value-of select="Name" />
                <xsl:text>:</xsl:text>

                <xsl:text>RentTypeName,</xsl:text>
                <xsl:value-of select="NamePT" />
                <xsl:text>:</xsl:text>

                <xsl:text>Days,</xsl:text>
                <xsl:value-of select="Days" />

            </xsl:if>
```

```
            <xsl:text>|</xsl:text>

        </xsl:for-each>
    </xsl:template>
</xsl:transform>
```

B.11 RealEstateCatalogs

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:transform
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    version="2.0">

    <xsl:template match="root">

        <xsl:for-each select="row">

            <xsl:text>RealEstateID,</xsl:text>
            <xsl:value-of select="RealEstateID" />
            <xsl:text>:</xsl:text>

            <xsl:text>RealEstateType,</xsl:text>
            <xsl:value-of select="RealEstateType" />
            <xsl:text>:</xsl:text>

            <xsl:text>CountryID,</xsl:text>
            <xsl:value-of select="CountryID" />
            <xsl:text>:</xsl:text>

            <xsl:text>SystemID,</xsl:text>
            <xsl:value-of select="SystemID" />
            <xsl:text>:</xsl:text>

            <xsl:text>PortalFlags,</xsl:text>
            <xsl:value-of select="PortalFlags" />
            <xsl:text>:</xsl:text>

            <xsl:text>Transporter,</xsl:text>
            <xsl:value-of select="Transporter" />
            <xsl:text>|</xsl:text>

        </xsl:for-each>
    </xsl:template>
</xsl:transform>
```

B.12 RealEstateFeature

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:transform
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    version="2.0">

    <xsl:template match="root">

        <xsl:for-each select="row">

            <xsl:text>RealEstateID,</xsl:text>
            <xsl:value-of select="ID" />
            <xsl:text>:</xsl:text>

            <xsl:text>SystemID,</xsl:text>
            <xsl:value-of select="SystemID" />
            <xsl:text>:</xsl:text>

            <xsl:text>FeatureType,</xsl:text>
            <xsl:value-of select="FeatureType" />
            <xsl:text>:</xsl:text>

            <xsl:text>Transporter,</xsl:text>
            <xsl:value-of select="Transporter" />
            <xsl:text>:</xsl:text>

            <xsl:text>EID,</xsl:text>
            <xsl:value-of select="EID" />
            <xsl:text>:</xsl:text>

            <xsl:text>Clid,</xsl:text>
            <xsl:value-of select="Clid" />
            <xsl:text>:</xsl:text>

            <xsl:text>Client_ID,</xsl:text>
            <xsl:value-of select="Client_ID" />
            <xsl:text>:</xsl:text>

            <xsl:text>MasterClient_ID,</xsl:text>
            <xsl:value-of select="MasterClient_ID" />
            <xsl:text>:</xsl:text>

            <xsl:text>SerialNumb,</xsl:text>
            <xsl:value-of select="SerialNumb" />
            <xsl:text>:</xsl:text>
```

```
<xsl:text>Category_ID,</xsl:text>
<xsl:value-of select="Category_ID" />
<xsl:text>:</xsl:text>

<xsl:text>Business_ID,</xsl:text>
<xsl:value-of select="Business_ID" />
<xsl:text>:</xsl:text>

<xsl:text>Zone_ID,</xsl:text>
<xsl:value-of select="Zone_ID" />
<xsl:text>:</xsl:text>

<xsl:text>Neighborhood_ID,</xsl:text>
<xsl:value-of select="Neighborhood_ID" />
<xsl:text>:</xsl:text>

<xsl:text>Town_ID,</xsl:text>
<xsl:value-of select="Town_ID" />
<xsl:text>:</xsl:text>

<xsl:text>State_ID,</xsl:text>
<xsl:value-of select="State_ID" />
<xsl:text>:</xsl:text>

<xsl:text>Region_ID,</xsl:text>
<xsl:value-of select="Region_ID" />
<xsl:text>:</xsl:text>

<xsl:text>Status_ID,</xsl:text>
<xsl:value-of select="Status_ID" />
<xsl:text>:</xsl:text>

<xsl:text>AreaU,</xsl:text>
<xsl:value-of select="AreaU" />
<xsl:text>:</xsl:text>

<xsl:text>AreaB,</xsl:text>
<xsl:value-of select="AreaB" />
<xsl:text>:</xsl:text>

<xsl:text>AreaT,</xsl:text>
<xsl:value-of select="AreaT" />
<xsl:text>:</xsl:text>

<xsl:text>Price,</xsl:text>
<xsl:value-of select="Price" />
<xsl:text>:</xsl:text>
```

```
<xsl:text>ShowPic,</xsl:text>
<xsl:value-of select="ShowPic" />
<xsl:text>:</xsl:text>

<xsl:text>Highlight,</xsl:text>
<xsl:value-of select="Highlight" />
<xsl:text>:</xsl:text>

<xsl:text>Web,</xsl:text>
<xsl:value-of select="Web" />
<xsl:text>:</xsl:text>

<xsl:text>Score,</xsl:text>
<xsl:value-of select="Score" />
<xsl:text>:</xsl:text>

<xsl:text>Date,</xsl:text>
<xsl:value-of select="Date" />
<xsl:text>:</xsl:text>

<xsl:text>AsGarage,</xsl:text>
<xsl:value-of select="AsGarage" />
<xsl:text>:</xsl:text>

<xsl:text>AsPool,</xsl:text>
<xsl:value-of select="AsPool" />
<xsl:text>:</xsl:text>

<xsl:text>AsFurniture,</xsl:text>
<xsl:value-of select="AsFurniture" />
<xsl:text>:</xsl:text>

<xsl:text>IsOpenSpace,</xsl:text>
<xsl:value-of select="IsOpenSpace" />
<xsl:text>:</xsl:text>

<xsl:text>HeightVal,</xsl:text>
<xsl:value-of select="HeightVal" />
<xsl:text>:</xsl:text>

<xsl:text>CountFeatures,</xsl:text>
<xsl:value-of select="CountFeatures" />
<xsl:text>:</xsl:text>

<xsl:text>HasDescription,</xsl:text>
<xsl:value-of select="HasDescription" />
```

```
<xsl:text>:</xsl:text>

<xsl:text>Leilao_UID,</xsl:text>
<xsl:value-of select="Leilao_UID" />
<xsl:text>:</xsl:text>

<xsl:text>SecPrice,</xsl:text>
<xsl:value-of select="SecPrice" />
<xsl:text>:</xsl:text>

<xsl:text>hasPrice,</xsl:text>
<xsl:value-of select="hasPrice" />
<xsl:text>:</xsl:text>

<xsl:text>hasOrderPrice,</xsl:text>
<xsl:value-of select="hasOrderPrice" />
<xsl:text>:</xsl:text>

<xsl:text>OrderPrice,</xsl:text>
<xsl:value-of select="OrderPrice" />
<xsl:text>:</xsl:text>

<xsl:text>BusinessFlags,</xsl:text>
<xsl:value-of select="BusinessFlags" />
<xsl:text>:</xsl:text>

<xsl:text>CountryID,</xsl:text>
<xsl:value-of select="CountryID" />
<xsl:text>:</xsl:text>

<xsl:text>PortalFlags,</xsl:text>
<xsl:value-of select="PortalFlags" />
<xsl:text>:</xsl:text>

<xsl:text>GPSLon,</xsl:text>
<xsl:value-of select="GPSLon" />
<xsl:text>:</xsl:text>

<xsl:text>GPSLat,</xsl:text>
<xsl:value-of select="GPSLat" />
<xsl:text>:</xsl:text>

<xsl:text>EnergyCertification,</xsl:text>
<xsl:value-of select="EnergyCertification" />
<xsl:text>:</xsl:text>

<xsl:text>FeatureFlags,</xsl:text>
```

```
<xsl:value-of select="FeatureFlags" />
<xsl:text>:</xsl:text>

<xsl:text>PriceChangeDate,</xsl:text>
<xsl:value-of select="PriceChangeDate" />
<xsl:text>:</xsl:text>

<xsl:text>PriceChangeVariation,</xsl:text>
<xsl:value-of select="PriceChangeVariation" />
<xsl:text>:</xsl:text>

<xsl:text>PriceChangeValue,</xsl:text>
<xsl:value-of select="PriceChangeValue" />
<xsl:text>:</xsl:text>

<xsl:text>hasGPS,</xsl:text>
<xsl:value-of select="hasGPS" />
<xsl:text>:</xsl:text>

<xsl:text>RealEstateVersion,</xsl:text>
<xsl:value-of select="RealEstateVersion" />
<xsl:text>:</xsl:text>

<xsl:text>ImgFlag,</xsl:text>
<xsl:value-of select="ImgFlag" />
<xsl:text>:</xsl:text>

<xsl:text>TourismRegion_ID,</xsl:text>
<xsl:value-of select="TourismRegion_ID" />
<xsl:text>:</xsl:text>

<xsl:text>MinPax,</xsl:text>
<xsl:value-of select="MinPax" />
<xsl:text>:</xsl:text>

<xsl:text>MaxPax,</xsl:text>
<xsl:value-of select="MaxPax" />
<xsl:text>:</xsl:text>

<xsl:text>PriceChangeLastPrice,</xsl:text>
<xsl:value-of select="PriceChangeLastPrice" />
<xsl:text>:</xsl:text>

<xsl:text>Status,</xsl:text>
<xsl:value-of select="status" />
<xsl:text>:</xsl:text>
```

```
            <xsl:text>StatusMLS,</xsl:text>
            <xsl:value-of select="StatusMLS" />
            <xsl:text>:</xsl:text>

            <xsl:text>PropTypeName,</xsl:text>
            <xsl:value-of select="PropTypeName" />
            <xsl:text>:</xsl:text>

            <xsl:text>PropTypeMLS,</xsl:text>
            <xsl:value-of select="PropTypeMLS" />
            <xsl:text>:</xsl:text>

            <xsl:text>Rooms,</xsl:text>
            <xsl:value-of select="Rooms" />
            <xsl:text>:</xsl:text>

            <xsl:text>Convertibles,</xsl:text>
            <xsl:value-of select="Convertibles" />
            <xsl:text>:</xsl:text>

            <xsl:text>Floors,</xsl:text>
            <xsl:value-of select="Floors" />
            <xsl:text>:</xsl:text>

            <xsl:text>MasterPropType,</xsl:text>
            <xsl:value-of select="Type" />
            <xsl:text>:</xsl:text>

            <xsl:text>MasterPropTypeMLS,</xsl:text>
            <xsl:value-of select="TypeMLS" />
            <xsl:text>|</xsl:text>

        </xsl:for-each>
    </xsl:template>
</xsl:transform>
```

B.13 RealEstatePortal

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:transform
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    version="2.0">

    <xsl:template match="root">

        <xsl:for-each select="row">

            <xsl:text>RealEstateID,</xsl:text>
            <xsl:value-of select="ID" />
            <xsl:text>:</xsl:text>

            <xsl:text>SystemID,</xsl:text>
            <xsl:value-of select="SystemID" />
            <xsl:text>:</xsl:text>

            <xsl:text>PortalID,</xsl:text>
            <xsl:value-of select="PortalID" />
            <xsl:text>:</xsl:text>

            <xsl:text>ClientID,</xsl:text>
            <xsl:value-of select="ClientID" />
            <xsl:text>:</xsl:text>

            <xsl:text>Clid,</xsl:text>
            <xsl:value-of select="Clid" />
            <xsl:text>:</xsl:text>

            <xsl:text>CountryID,</xsl:text>
            <xsl:value-of select="CountryID" />
            <xsl:text>|</xsl:text>
        </xsl:for-each>
    </xsl:template>
</xsl:transform>
```

B.14 System

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:transform
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    version="2.0">

    <xsl:template match="root">

        <xsl:for-each select="row">

            <xsl:text>SystemID,</xsl:text>
            <xsl:value-of select="SystemID" />
            <xsl:text>:</xsl:text>

            <xsl:text>Name,</xsl:text>
            <xsl:value-of select="Name" />
            <xsl:text>:</xsl:text>

            <xsl:text>ConfigKey,</xsl:text>
            <xsl:value-of select="ConfigKey" />
            <xsl:text>|</xsl:text>

        </xsl:for-each>
    </xsl:template>
</xsl:transform>
```

B.15 Video

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:transform
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    version="2.0">

    <xsl:template match="root">

        <xsl:for-each select="row">

            <xsl:text>ID,</xsl:text>
            <xsl:value-of select="ID" />
            <xsl:text>:</xsl:text>

            <xsl:text>SystemID,</xsl:text>
            <xsl:value-of select="SystemID" />
            <xsl:text>:</xsl:text>

            <xsl:text>EID,</xsl:text>
            <xsl:value-of select="EID" />
            <xsl:text>:</xsl:text>

            <xsl:text>Client_ID,</xsl:text>
            <xsl:value-of select="Client_ID" />
            <xsl:text>:</xsl:text>

            <xsl:text>RealEstate_ID,</xsl:text>
            <xsl:value-of select="RealEstate_ID" />
            <xsl:text>:</xsl:text>

            <xsl:text>RealEstate_Type,</xsl:text>
            <xsl:value-of select="RealEstate_Type" />
            <xsl:text>:</xsl:text>

            <xsl:text>Description,</xsl:text>
            <xsl:value-of select="Description" />
            <xsl:text>:</xsl:text>

            <xsl:text>Priority,</xsl:text>
            <xsl:value-of select="Priority" />
            <xsl:text>:</xsl:text>

            <xsl:text>Ext,</xsl:text>
            <xsl:value-of select="Ext" />
            <xsl:text>:</xsl:text>
```

```
            <xsl:text>Transporter,</xsl:text>
            <xsl:value-of select="Transporter" />
            <xsl:text>:</xsl:text>

            <xsl:text>DescriptionMLS,</xsl:text>
            <xsl:value-of select="DescriptionMLS" />
            <xsl:text>|</xsl:text>

        </xsl:for-each>
    </xsl:template>
</xsl:transform>
```

B.16 Videos

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:transform
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    version="2.0">

    <xsl:template match="root">
        <xsl:for-each select="row">
            <xsl:choose>
                <xsl:when test="position()=1">

                    <xsl:text>RealEstateID,</xsl:text>
                    <xsl:value-of select="RealEstate_ID" />
                    <xsl:text>:</xsl:text>

                    <xsl:text>Videos,</xsl:text>
                    <xsl:value-of select="ID" />

                </xsl:when>
                <xsl:when test="position()=last()">
                    <xsl:choose>

                        <xsl:when test="preceding::ID[1]!=ID">
                            <xsl:text>|</xsl:text>

                            <xsl:text>RealEstateID,</xsl:text>
                            <xsl:value-of select="RealEstate_ID" />
                            <xsl:text>:</xsl:text>

                            <xsl:text>Videos,</xsl:text>
                            <xsl:value-of select="ID" />

                        </xsl:when>
                        <xsl:otherwise>
                            <xsl:text>,</xsl:text>
                            <xsl:value-of select="ID" />
                        </xsl:otherwise>
                    </xsl:choose>
                    <xsl:text>|</xsl:text>
                </xsl:when>

                <xsl:when test="preceding::ID[1]!=ID">
                    <xsl:text>|</xsl:text>

                    <xsl:text>RealEstateID,</xsl:text>
                    <xsl:value-of select="RealEstate_ID" />
```

```
                    <xsl:text>:</xsl:text>

                    <xsl:text>Videos,</xsl:text>
                    <xsl:value-of select="ID" />

                </xsl:when>
                <xsl:otherwise>
                    <xsl:text>,</xsl:text>
                    <xsl:value-of select="ID" />
                </xsl:otherwise>
            </xsl:choose>

        </xsl:for-each>
    </xsl:template>
</xsl:transform>
```