Departamento de Ciências e Tecnologias da Informação

Migração de bases de dados relacionais para NoSQL - Métodos de Análise

Fábio Vieira de Oliveira

Dissertação submetida como requisito parcial para obtenção do grau de

Mestre em Engenharia Informática

Orientador:

Doutor Abílio Oliveira, Professor Auxiliar

ISCTE-IUL

Setembro, 2017

“A tarefa não é tanto ver aquilo que ninguém viu, mas pensar o que ninguém ainda pensou sobre aquilo que todo mundo vê.”

Arthur Schopenhauer

AGRADECIMENTOS

Quero iniciar agradecendo a minha família, fonte de inspiração e motivação para todos os dias, independente da tarefa ou do objetivo, estarão sempre em primeiro lugar, apesar de durante alguns momentos ter de me refugiar no silencio, os vossos sons sempre tocam o meu coração, á vocês eu dedico este trabalho, sei que viveram ele comigo.

Durante a minha caminha nestes anos de mestrado, muitos foram os desafios que surgiram e que foram todos ultrapassados, alguns com menos, outros com mais entusiamo, mas a verdade é que todos juntos contribuíram para que fosse possível chegar nesse momento final de curso, tenho a certeza de que saio com novos conhecimentos e uma visão diferente de tudo.

Nesse momento, jamais poderia faltar os meus sinceros agradecimentos ao meu orientador, o professor Doutor Abílio Oliveira que em todos os momentos contribuiu não somente com conhecimento académico, mas também com uma motivação contagiante e exemplar, com uma simplicidade tão genuína e majestosa, fruto de uma paixão notável pelo seu trabalho que nos contagia, o meu muito obrigado.

RESUMO

Vivemos numa época onde o crescimento previsto para o atual volume de dados disponíveis, em qualquer empresa ou organização, é muito elevado. Junta-se a este facto o atual crescimento na adoção por modelos de bases de dados NoSQL e Big Data, em detrimento de modelos relacionais. Perante esta realidade, levanta-se a questão de como evoluir ou migrar dos atuais, e ainda muito frequentes, sistemas relacionais, para os novos sistemas NoSQL. Esta investigação tem como grande objetivo validar os métodos de migração existentes (e já utilizados) e realizar uma adaptação aos mesmos, de forma a perceber qual o método mais eficiente para proceder à migração de uma base de dados relacional para uma base de dados NoSQL. A parte teórica deste trabalho mostra como proceder à implementação e análise do ambiente. Os resultados revelam que neste processo de migração, o método mais eficiente é o que se designa como offline automático. Contudo, o mesmo necessita de uma janela de indisponibilidade maior do que o método de migração online, que por sua vez exige mais recursos por parte do sistema operativo para proceder à migração. Portanto, o método mais eficiente para migrar uma base de dados vai depender da disponibilidade aplicacional, e dos recursos computacionais disponíveis, para a mesma. Esperamos dar um importante contributo no sentido de auxiliar na escolha de um método de migração a utilizar, e das métricas que podem ser coletadas para melhor avaliar a performance de uma migração.

Palavras-Chave

Migração, Métodos, Cenários, Métricas, Bases de dados Relacionais, NoSQL

ABSTRACT

We live in an age where the expected growth for the current volume of data available in any company or organization is very high. Added to this is the current growth in the adoption of NoSQL and Big Data database models, to the detriment of relational models. Given this reality, the question arises as to how to evolve or migrate from the current, and still very frequent, relational systems, to the new NoSQL systems. The purpose of this research is to validate existing migration methods (and already used) and to adapt them, to understand the most efficient method to migrate from a relational database to a NoSQL database. The theoretical part of this work shows how to proceed to the implementation and analysis of the environment. The results show that in this migration process, the most efficient method is what is referred to as automatic offline. However, it requires a window of unavailability greater than the method of online migration, which in turn requires more resources from the operating system to migrate. Therefore, the most efficient method to migrate a database will depend on the application availability, and the computational resources available, for the same. We hope to make an important contribution in the sense that the tests carried out in this thesis can help in choosing which method of migration to use and which metrics can be collected to better analyse the performance of a migration.

Keywords

Migration, Methods, Metrics, Relational Databases, NoSQL

Lista de Abreviaturas

ACID - Atomicity, Consistency, Isolation, Durability

API - Application Programming Interface

APP - Application

ASCII - American Standard Code for Information Interchange

BASE - Basically Available, Soft state, Eventual consistency

BI - Business Intelligence

BSON - Binary JSON

C1 - Coordinator

CAP - Consistency, Availability, Partition Tolerance

CPU - Central Processing Unit

CRUD - Create, read, update, delete

CSN - Commit Sequence Number

CSV - Comma Separated Values

DB - Database

DBA - Database Administrator

DDL - Data Definition Language

DML- Data Manipulation Language

DSS - Decision Support System

DW - Data Warehouse

ECSV - Extract to CSV

EI - Extract Initial

EO - Extract Online

ETL - Extract, Transform e Load

GUI - Graphical User Interface

HDFS - Hadoop Distributed File System

HTAP - Hybrid Transactional/Analytical Processing

I/O - Input/Output

ICSV - Importation from CSV

IDE - Integrated Development Environment

INFO - Information

ISO - International Standard Organization

IT - Information Technology

JDBC - Java Database Connectivity

JSON - Java Script Object Notation

KPI - Key Performance Indicator

N/A - Não aplicável

NoSQL - Not Only Structured Query Language

OLAP - Online Analytical Processing

OLTP - Online Transaction Processing

PMI - Project Management Institute

RDBMS - Relational Database Management System

RFID - Radio Frequency Identification

RI - Replication Initial

RO - Replication Online

SCN - System Change Number

SGBD - Sistema Gerenciador de Base de dados

SLA - Service Level Agreement

SQL - Structured Query Language

TPC - Transaction Processing Performance Council

TPC-C - Transaction Processing Performance Council - On-Line Transaction/OLTP

TPC-H - Transaction Processing Performance Council - Decision Support/OLAP

KPI - Key Performance Indicator

Capítulo I – Introdução Geral

1.1. Contextualização

Hoje em dia parece haver dados por todo o lado. Dados que se organizam e estruturam em informação. Como guardá-los, geri-los, transferi-los e usá-los em tempo considerado útil nos mais variados contextos, nomeadamente nas grandes organizações ou nas empresas?

Neste sentido, abordamos aqui o contexto do trabalho que nos propomos a desenvolver. Salientamos os pontos relevantes para a sua conceptualização e realização, assim como abordamos a motivação pessoal e académica, que contribuíram para impulsionar o mesmo. Na parte final desta introdução teremos uma visão geral da estrutura deste documento.

A evolução tecnológica que se tem verificado nos últimos anos levou a que as empresas sentissem a necessidade de se adaptar a um novo universo tecnológico, em progressivo desenvolvimento. É preciso otimizar processos, gerir adequadamente recursos, melhorar performances. O que implica também recolher, tratar e analisar grandes volumes de dados, para possibilitar uma correta gestão de informação, o que é essencial em qualquer empresa. Estima-se que o volume de dados esteja a crescer 40% por ano, e prevê-se um crescimento de 44 vezes entre 2009 e 2020 (Manyika et al., 2011).

Diante da quantidade de bases de dados construídas com base no modelo relacional[1], ainda predominante, exemplificar e contextualizar a possibilidade de migração para o modelo NoSQL (*Not Only Structured Query Language*), que vão sendo gradualmente adotados em médias e grandes empresas, é um desafio estimulante, a ser trabalhado. A notória escassez de informação sobre este tema da migração, que nos propomos abordar, fez-se sentir durante a elaboração deste projeto, embora tenhamos constatado que existem vários métodos e técnicas para validar uma migração (Neto, Neto, Junior, & Oliveira, 2015), que serão por nós apresentadas.

Atualmente a importância dada aos sistemas gerenciadores de bases de dados é indiscutível, verificando-se que estas ferramentas contêm todas as informações e os dados das empresas.

[1] Cf. Capítulo II, na secção 2.3 Bases de dados Relacionais.

Existem grandes e conhecidos sistemas que utilizam o modelo NoSQL, como p.e., o *Google Maps*[2], e até mesmo as redes sociais, como o *Twitter*[3] e o *Facebook*[4].

No caso de uma área empresa não conseguir aceder aos seus dados durante uma negociação, ou se durante a faturação não for possível aceder às faturas, devido a algum problema de conexão em qualquer uma das camadas, até se chegar à base de dados, i.e., se de alguma forma os dados de uma base de dados forem perdidos, isso pode ser o suficiente para levar ao encerramento da empresa. Como exemplo, verifique-se as empresas que foram à falência após o ataque terrorista às torres gémeas no dia 11 de setembro de 2001 nos Estados Unidos da América (EUA), onde várias empresas tinham os seus dados replicados de uma torre para a outra (Spaniol, 2015).

A utilização dos então conhecidos sistemas gerenciadores de bases de dados (SGBD), ou simplesmente bases de dados (BD), é alvo de grandes avanços. A estes, na maioria dos casos, atribuem a tarefa de apenas realizar o armazenamento dos dados que possam vir a tornar-se informação – ou seja, dados organizados de uma forma coerente, fiável e consistente (Alturas, 2013). No entanto, esta depende da aplicação de regras e da correta extração dos dados para consolidar em informação útil, entrando aqui os sistemas de consolidação, ou mesmo o processamento de informação em bases de dados, sendo que ainda precisamos realizar o correto compartilhamento da informação para os indivíduos interessados (Lage & Alturas, 2012). A verdade é que as bases de dados são essenciais para muitas empresas e fazem parte, quase que diretamente, dos seus negócios.

No caso de haver indisponibilidade das bases de dados, um negócio pode ser diretamente afetado, levando a penalizações financeiras, sejam elas por incumprimento ou quebra ao nível de serviço – do inglês *service level agreement* (SLA) –, ou devido a negócios não finalizados.

Em pesquisas realizadas anteriormente [5], pouco conteúdo foi encontrado sobre como migrar do modelo relacional para o modelo NoSQL, ou mesmo exemplificando com métodos e métricas[6], que podem ser medidas em qualquer SGBD durante a migração, para avaliar a sua performance.

---

[2] Serviço de pesquisa e visualização de mapas, pode ser consultado em https://maps.google.com.
[3] Serviço de rede social, pode ser consultado em https://twitter.com.
[4] Serviço de rede social, pode ser consultado em https://www.facebook.com.
[5] Consultar Capítulo II na secção 2.8 Trabalhos anteriores relacionados
[6] O livro *The data warehouse toolkit: the complete guide to dimensional modelling* (Kimball & Ross, 2011), é bastante elucidativo neste tópico e aborda ainda outras questões como *Key Performance Indicator.(KPI).*

A pouca informação disponível sobre o tema aumenta significativamente o nível de dificuldade e exigência no estudo que iremos desenvolver. Em concordância com Neto, Neto e Junior (2015), existem artigos e trabalhos académicos sobre migrações de sistemas específicos, realizados através da confeção de *scripts* desenvolvidos pelo desenvolvedor responsável pela aplicação, ou pelo DBA, mas nada em âmbito geral. De acordo com Kimball e Ross (2011), o acesso rápido às informações é essencial para a correta tomada de decisão num projeto, no caso de se pretender trabalhar com desenvolvimento de aplicações ou extração de informação.

Em sistemas de *Data Warehouse* (DW), será certamente preciso trabalhar os dados da base de dados (BD), ou interagir com o administrador da base de dados (DBA), em algum momento da vida útil do sistema (Kimball & Ross, 2011). Conhecer a história dos sistemas de bases de dados, e como estes têm evoluído, é muito importante para compreender o modo como estes são organizados (Tanenbaum, 1996)[7] e a própria dinâmica das empresas.

1.2. Motivação

O tema principal desta dissertação é a migração de um modelo relacional para um modelo NoSQL, e as formas de o assegurar. Trata-se de um tema atual, crítico e de grande valia técnica, que emerge de motivações pessoais, profissionais – pela necessidade de lidar com o mesmo no dia-a-dia – e académicas elevadas, refletindo-se no aumento da pesquisa de cunho técnico.

Como profissional de tecnologia da informação e amante da mesma, as bases de dados NoSQL são, naquilo que diz respeito a Big Data – ou grandes volumes de dados –, uma tendência tecnológica, visto que os diversos modelos existentes e as varias possibilidades levantadas, por si só, constituem motivações suficientes para um investigador humilde. Segundo a maioria dos autores consultados para a realização deste trabalho, o modelo de bases de dados NoSQL é atualmente muito difundido[8].

[7] O livro *Computer Networks* de *Andrew S. Tanenbaum* (Tanenbaum, 1996), é bastante elucidativo neste tópico. Aborda ainda outras questões como o porquê da necessidade de termos sistemas distribuídos de bases de dados, o porquê de termos não somente um modelo para todas as atuais necessidades de armazenamento de dados.

[8] Cf. Capítulo II na secção 2.8 Trabalhos anteriores relacionados.

Várias empresas estão atentas a esta tecnologia e, consequentemente, as Universidades acabam por promover a pesquisa sobre esta temática, de modo a expandirem os seus conhecimentos e contribuírem para a sua divulgação. Por ser uma área de estudo recente, e com novas tecnologias a surgir, esta é uma tendência para os próximos anos.

De modo a analisar e propor um procedimento, juntamente com um conjunto de métricas, para a migração de um modelo relacional para um modelo do tipo NoSQL, serão investigadas questões relativas à performance dos possíveis métodos utilizados no decorrer dessa migração. Durante a migração, iremos medir quais os impactos dos principais métodos de migração, bem como a sua usabilidade, em comparação com outros modelos disponíveis ou utilizados.

Antaño, Castro e Valencia (2014) apresentaram um estudo sobre migração de bases de dados relacional para NoSQL, evidenciando um conjunto de algumas das técnicas a aplicar durante o processo, que aqui iremos considerar, juntamente com outras [9]. Será também analisada a fiabilidade no processo, ou seja, quão seguro se apresenta o método da migração, e cada técnica utilizada. Poderá ainda verificar-se se uma ferramenta informática utiliza ou não uma *stage area*[10], que é normalmente usada para a extração, transformação e carregamento (do inglês *ETL*), ou se envia diretamente as informações extraídas da origem para o destino.

Pretende-se também compreender se os problemas de rede afetam ou não a migração ou, caso seja necessário, se se consegue reiniciar o processo em caso de perda dos ficheiros antes da aplicação na base de dados final. Num ambiente virtual, repleto de *hackers* e ciberataques, é importante prevermos todo o tipo de situações, tendo também a segurança como ponto a avaliar.

1.3. Questão de investigação e objetivos gerais

Os modelos de bases de dados NoSQL estão a ser gradualmente mais utilizados. No entanto, sendo ainda recentes, pretende-se contribuir para uma maior divulgação e esclarecimento sobre o que é este conceito, e qual o seu papel no mundo tecnológico atual.

[9] Cf. Capítulo 2, na secção 2.7 Trabalhos anteriores relacionados.

[10] Cf. Capítulo III, na secção 3.6, subsecção Requisitos para migração.

Assim, como muitos sistemas de informação se baseiam e sustentam num modelo relacional, espera-se com este estudo contribuir para um melhor conhecimento dos processos, das dificuldades e problemas que possam surgir durante a migração entre o modelo relacional e o NoSQL. Dada a criticidade de utilização destes modelos, e visando a sua utilidade, o presente estudo será realizado num contexto prático, pois é inviável, ou questionável, ter-se uma verdadeira noção dos processos necessários para a realização desta tarefa se nos mantivéssemos somente num plano teórico. Neste trabalho desenvolver-se-á uma migração real de um modelo relacional para um modelo NoSQL, em laboratório, com simulação de carga real baseada em ferramentas oficiais e normalizadas para esse efeito.

Esperamos também que este estudo seja útil para investigadores e organizações que venham a considerar esta mudança de paradigma de armazenamento exequível, e que pretendam melhor analisar os impactos em seus sistemas durante a migração. Deste modo, consideramos pertinente a seguinte questão de investigação:

- Qual o método mais eficiente para proceder à migração de uma base de dados relacional para uma base de dados NoSQL?

De acordo com esta questão, e necessidade prática, definiram-se os seguintes objetivos gerais:

- Identificar quais são os requisitos e as várias fases de uma migração.
- Verificar os possíveis métodos de migração de uma base de dados relacional para uma NoSQL.
- Avaliar cada um destes métodos, de acordo com as etapas seguidas em cada fase da migração – recorrendo a métricas específicas e adequadas, em cada caso.
- Comparar os vários métodos de acordo com as métricas estabelecidas.
- Determinar o método de migração mais eficiente para cada um dos cenários de bases de dados estudados.

O processo de elaboração de uma dissertação é um processo bastante dinâmico, em que novos conhecimentos e curiosidades académicas vão surgindo com naturalidade ao longo do processo de desenvolvimento. Pelo que alguns objetivos secundários, ou complementares aos objetivos principais, serão_definidos ao longo da investigação. Um desses objetivos será compreender qual a relação entre o modelo *NoSQL* e *Big Data*, e quais as mudanças que têm de ser levadas a cabo de forma a conseguirmos integrar e analisar dados *Big Data* contidos num sistema de

bases de dados NoSQL, ou num sistema relacional. O desenvolvimento deste objetivo poderá trazer pistas pertinentes para estudos futuros.

1.4. Abordagem Metodológica

De forma a responder à questão de investigação por nós levantada e dar seguimento aos objetivos traçados, procederemos inicialmente a um levantamento do estado da arte e revisão de literatura. Esta irá permitir-nos identificar as várias fases de uma migração, bem como os possíveis métodos de migração de uma base de dados relacional para uma NoSQL.

De seguida, avançaremos com a instalação do software necessário para procedermos às configurações e validações prementes na nossa pesquisa. Posteriormente às configurações executadas nos servidores, construiremos os vários cenários (OLTP, OLAP e HTAP) de forma a podermos aplicar os métodos de migração e compará-los, no que diz respeito à sua rentabilidade, analisando para isso as métricas definidas.

Aplicaremos assim os métodos *online, offline* automático e *offline* manual de migração, realizando testes que nos permitirão avaliar as métricas, e medir o desempenho de cada um dos métodos. De acordo com as métricas apuradas, procederemos então à comparação dos métodos de migração, de modo a determinarmos qual deles é de facto o mais eficiente no processo de migração de uma base de dados relacional para uma base de dados NoSQL.

No final da migração, e após a análise das métricas registadas, poderemos confirmar qual é o método mais eficiente para proceder à migração de uma base de dados relacional para uma NoSQL e, concomitantemente, responder aos objetivos a que nos propusemos.

1.5. Organização do Documento

O presente trabalho está dividido em duas partes principais.

A primeira parte inclui os capítulos I e II.

No primeiro capítulo – introdução geral – é apresentada uma breve contextualização da temática central, as bases de dados, e sua importância, bem como as motivações para realizarmos este projeto, demostrando o interesse pessoal e académico, e citando alguns trabalhos considerados relevantes. Enunciamos a questão de investigação, alicerçada numa pergunta de partida, bem como os objetivos principais associados à problemática a desenvolver, no decorrer de um caso

de estudo pesquisado – nos vários cenários analisados – e a abordagem metodológica seguida. No capítulo II, em termos de revisão de literatura, são apresentados alguns trabalhos anteriores e referências teórico-concetuais importantes, com ligação à temática central, desenvolvendo-se o estado da arte. Focam-se as tecnologias de bases de dados que têm ligação com este trabalho, quer as que serão objeto de estudo, mas também os modelos que têm ligação de cunho técnico com a mesma. É feita uma introdução ao modelo relacional e uma análise à sua fundamentação, explicando qual a finalidade da propriedade ACID e exemplificando com alguns SGBD, segundo a modelação relacional. É também apresentado o modelo NoSQL, bem como o teorema de CAP e a propriedade BASE, descrevendo alguns dos exemplos atualmente disponíveis nesse modelo. Definimos ainda o conceito de *Big Data*, com apresentação de alguns modelos atualmente disponíveis, descrevendo as suas características e classificação.

A segunda parte, de índole empírica, é formada pelos capítulos III e IV, onde apresentamos o estudo realizado, bem como os resultados encontrados, e a discussão dos mesmos. Por fim, apresentamos as conclusões deste trabalho.

Em suma, desenvolveram-se cinco capítulos: (I) Introdução Geral; (II) Revisão da literatura; (III) Caso de Estudo; (IV) Apresentação e discussão de resultados; e (V) Conclusões.

Capítulo II – Revisão da Literatura

2.1. Introdução ao Capítulo

Neste capítulo abordam-se as tecnologias de bases de dados consideradas neste estudo, informando quais estão presentes em cada uma das fases da migração, e qual a finalidade de cada uma delas no processo total, bem como o que se pode esperar das mesmas.

2.2. Conceitos fundamentais sobre bases de dados relacionais e NoSQL

Todas as empresas, de modo geral, utilizam e trabalham com bases de dados, quer sejam próprias ou adquiridas através de algum fornecedor, estando estas presentes no nosso dia a dia (Caldeira, 2015). Atualmente utiliza-se muito o modelo relacional, que obedece a um conjunto específico de regras, garantindo que uma transação na base de dados é consistente até o seu final (Codd, 1982) . Um dos princípios básicos que um SGBD (Sistema de Gestão de Bases de Dados) precisa seguir para ser do tipo relacional (cf. secção 2.3), é conhecido como *ACID*, sendo este controlo totalmente transparente para o utilizador final ou mesmo para a aplicação.

Para os modelos de bases de dados de grandes dimensões, podem ser analisados os sistemas de *Data Warehouse (DW)*, *Data Mining*, *Decision Support System* (DSS) *e Business Intelligence* (BI). Estes são sistemas de armazenamento que servem como recetores de dados, para que as informações sejam extraídas conforme os critérios necessários ou, em alguns casos, conforme as necessidades do utilizador (Chaudhuri, Dayal, & Narasayya, 2011). Na atual situação, numa empresa, as infraestruturas tecnológicas são compostas por um ou mais servidores, para o processamento, gestão da base de dados, e um sistema de armazenamento. Este último é específico e conhecido como *storage*.

Na realidade, os sistemas passaram a ter que armazenar e gerir uma grande quantidade de dados, levando a que seja necessário, não só dispor de mais espaço para os mesmos, assim como evoluir os códigos de busca de informação dentro desses sistemas, de modo a otimizar a sua pesquisa e, consequentemente, o sucesso da empresa, em termos de processamento de informação. Os atuais SGBD com maior quantidade de dados passaram a ser designados como *Big Data*, obedecendo a critérios específicos para entrarem nessa categoria (Mysore, Khupat, & Jain, 2014).

Big Data é o termo comumente utilizado para descrever grande volume de dados, sejam eles estruturados ou não estruturados. O mais importante nestes não é a quantidade em si, mas sim o valor dos dados, ajudando as empresas a tomarem decisões baseados nos dados retirados destes sistemas (SAS Institute, 2016).

Existem sistemas de *Big Data* a serem executados em SGBD do tipo relacional e do tipo não relacional. De referir que cada um possui sua própria estrutura de metadados[11], e armazenamento interno, sendo necessário inúmeras alterações ao nível de código[12] para podermos utilizá-los (Gomes, 2011). Na Figura 1 estão representadas as principais diferenças (entre os dois tipos) no que se refere à importação e processamento de dados, em uma base de dados do modelo relacional e uma do modelo NoSQL (Dijcks, 2013).

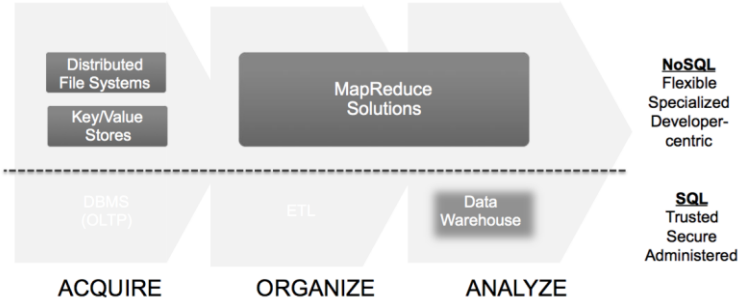


Figura 1 - Comparação entre processamento aplicacional em SGBD relacional e NoSQL.

Fonte: (Dijcks, 2013)

O tipo de linguagem usada, e a forma de armazenamento utilizada por estes dois modelos, são também pontos de diferenciação entre eles. O modelo relacional utiliza o padrão SQL ANSI[13], linguagem padrão para o SGBD relacionais (American National Standards Institute, 2017). Já os modelos NoSQL (Weber & Strauch, 2010) utilizam padrões como JSON (Java *Script Object Notation*) para aceder aos dados, o que pode, contudo, variar consoante o modelo NoSQL

[11] Metadados são habitualmente definidos simplesmente como dados descrevendo outros. (Campos, 2007)

[12] Em computação, código é um conjunto de instruções a serem interpretadas e executadas por um programa computacional.

[13] É um padrão para a linguagem SQL adotado pelas bases de dados relacionais. (American National Standards Institute, 2017)

utilizado, é frequente encontramos o termo BSON *(Binary JSON*), utilizado para armazenar os dados em formato binário, como por exemplo na base de dados *MongoDB* (MongoDB, 2015). A Tabela 1 apresenta a comparação entre o modelos relacional SQL, e o NoSQL/Mongodb (MongoDB, 2016).

Tabela 1 - Comparativo das definições entre os modelos relacionais e NoSQL/Mongodb

SQL	MONGODB
DATABASE	Database
TABLE	Collection
ROW	Document
COLUMN	Field
INDEX	Index
FOREIGN KEY	Embedded Document
JOIN	Embedded document, document references or $lookup to combine data from different collections
PRIMARY KEY	Primary Key(_id)

Fonte: (MongoDB, 2015, p. 3)

A Tabela 2 constituída por comandos de apresentação de dados, apresenta uma comparação entre estes dois tipos de modelos, relacional e NoSQL, centrando-se naquilo que difere os componentes num SGBD (MongoDB, 2015).

Tabela 2 - Mapeando comandos SQL e MongoDB

SQL SELECT Statements	MongoDB find() Statements
`SELECT *` `FROM people`	`db.people.find()`

`SELECT id,`
`  user_id,`
`  status`
`FROM people`
	`db.people.find(`
	`  { },`
	`  { user_id: 1, status: 1 }`
	`)`

`SELECT user_id, status`
`FROM people`
	`db.people.find(`
	`  { },`
	`  { user_id: 1, status: 1, _id: 0 }`
	`)`

Fonte: (MongoDB, 2016)

As bases de dados NoSQL surgiram como uma solução de escalabilidade ao modelo relacional, pois estas não seguem a consistência ACID[14], mas sim o Teorema de CAP[15] (Moniruzzaman & Hossain, 2013).

Segundo Forbes (2010), existem aplicações que não necessitam de utilizar a consistência existente nas bases de dados relacionais:

"The truth is that you don't need ACID for Facebook status updates or tweets or Slashdot's comments. So long as your business and presentation layers can robustly deal with inconsistent data, it does not really matter. It is not ideal, obviously, and preferably, you see zero data loss, inconsistency, or service interruption, however accepting data loss or inconsistency (even just temporary) as a possibility, breaking free of by far the biggest scaling "hindrance" of the RDBMS world, can yield dramatic flexibility. (. . .) This is the case for many social media sites: data integrity is largely optional, and the expense to guarantee it is an unnecessary expenditure. When you yield pennies for ad clicks after thousands of users and hundreds of thousands of transactions, you start to look to optimize" (Forbes, 2010).

Diante das opções aqui demostradas, e do poder de cada uma delas, é de realçar o fato de termos diversos artigos sobre ambos os tipos de SGBD e tão limitado material sobre como migrar um SGBD relacional para um SGBD NoSQL, daí, em boa medida, o interesse do presente trabalho((Davenport et al., 2014), (Gomes, 2011)).

É importante exemplificar a possibilidade de migração de bases de dados relacionais para outro modelo que não o relacional, nomeadamente para o modelo NoSQL. Existem técnicas que podem ser utilizadas para efetuar, e validar, esse tipo de migração, como por exemplo, Migração Incremental, reengenharia de dados, entre outras(Neto et al., 2015).

De um modo geral, a informação encontrada em artigos científicos e dissertações, retrata situações de migrações de sistemas específicos – como por exemplo o trabalho de (Davenport

[14] Cf. capítulo II, secção 2.4. Bases de dados relacionais.

[15] Cf. capítulo II, secção 2.5. Bases de dados NoSQL.

et al., 2014), (Gomes, 2011)) – e não do processo de migração de uma forma geral (Neto et al., 2015). Esta pesquisa pretende fornecer indicadores seguros sobre como proceder à migração de uma BD relacional para uma não relacional. Para tal, recorreremos a métricas específicas que permitem quantificar e assim avaliar o modelo em uso e a eficácia do método de migração a utilizar, em alguns cenários específicos, durante todo o processo.

Poderemos assim analisar e propor um conjunto de técnicas estatísticas, bem como um modelo de migração, isto segundo alguns tipos específicos de necessidades, e conforme cenários de migração devidamente definidos. Estes procedimentos podem ser replicados para outros tipos de cenários que não os aqui estudados, e devidamente ajustados conforme o caso.

Howard, (2007) em um dos seus trabalhos, desenvolveu uma pesquisa onde identificou os principais motivos que levam à necessidade de recorrer a um processo de migração de bases de dados; na Figura 2 são sintetizados os resultados dessa pesquisa.

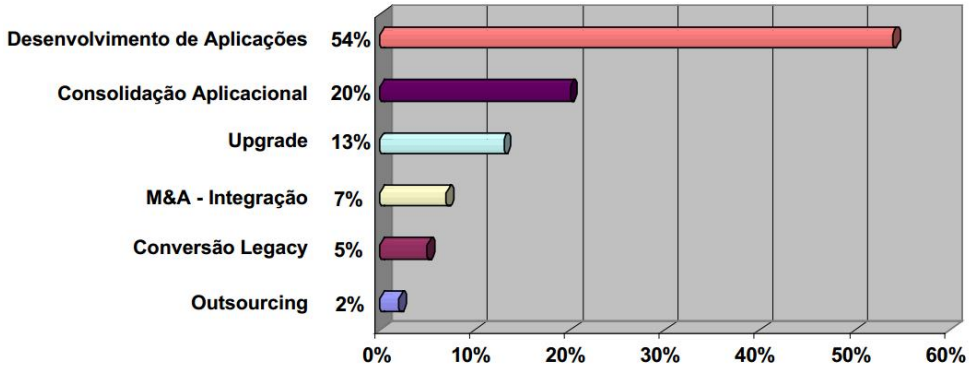


Figura 2 - Principais motivos para migração de bases de dados.

Fonte: (Howard, 2007, p. 3)

Dos dados na Figura 2 podemos aferir que mais de metade das migrações (cerca de 54%) ocorrem devido ao desenvolvimento de aplicações em outros modelos, e que a consolidação dos dados nesse mesmo modelo vem em segundo lugar com 20% das preferências. Segundo a pesquisa apresentada (Antaño et al., 2014), no âmbito da migração de bases de dados relacionais para NoSQL, um conjunto de técnicas devem ser aplicadas durante o processo de migração.

A confiabilidade no processo é um ponto muito importante, pois reflete o quão seguro é a técnica e o meio. No caso de, por exemplo, a técnica utilizada recorrer a uma *stage area* [16], ou se esta envia diretamente as informações extraídas da origem para o destino, tem-se que nesses casos relatar se a existência de problemas na rede pode ou não afetar a migração (Antaño et al., 2014).

As questões sobre o método, a performance e a confiabilidade são o foco deste trabalho, sendo que durante os trabalhos de laboratório, todos os testes foram executados de igual forma e tendo em consideração a sua infraestrutura, onde foram definidos roteiros conforme as necessidades de migração e testados nos cenários envolvidos.

Na Figura 3 podemos ver as etapas que envolvem a migração oficial do SGBD *MongoDB*, ilustrando um fluxo que considera a aplicação das melhores práticas para este tipo de migração, seguindo as indicações do próprio mantenedor do SGBD(MongoDB, 2015).

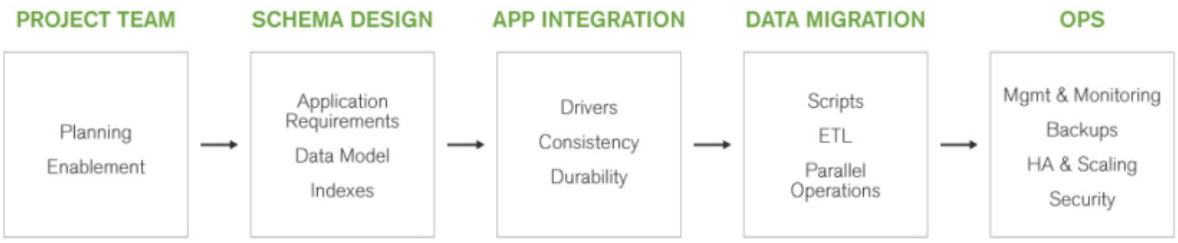


Figura 3 – MongoDB *Migration Roadmap*.

Fonte: (MongoDB, 2015, p. 2)

Todo o processo de migração está associado a uma etapa do planeamento. Para além do planeamento outras etapas serão abordadas, e em todas as análises tentaremos vinculá-las ao processo de execução, e à fase em que foram desenvolvidas.

[16] Uma staging area é um espaço temporário dentro ou fora de uma base de dados, utilizado para armazenamento temporário de informação, para o intuito do mesmo ser processado para posteriormente, ser transformado em informação útil para o sistema OLAP (Kimball & Ross, 2011).

2.3. Análise comparativa entre SBGDS relacionais e NoSQL

Durante décadas, as bases de dados relacionais têm sido a escolha como sistema de armazenamento padrão, no entanto, este paradigma tem vindo a mudar e muitas aplicações tem alterando sua maneira de armazenar os dados, devido à escalabilidade proporcionada pelos modelos NoSQL (Nascimento, 2016).

Assim, a grande motivação para usar modelos NoSQL é resolver o problema de escalabilidade apresentada nos modelos relacionais. Entretanto, é necessário analisar ambos os modelos, a necessidade do negócio e o objetivo da migração. Isto porque para que a migração seja necessária é preciso conhecer o objetivo a alcançar migrando para um modelo NoSQL. Deste modo há de ter em conta algumas considerações e explicar conceitos básicos sobre os modelos, nomeadamente:

- Escalonamento
- Consistência de dados
- Flexibilidade

O escalonamento consiste horizontal consistem em podermos adicionar outros servidores a base de dados, sendo considerado como uma das principais motivações para utilização quando se pensa em utilizar os modelos NoSQL (Nascimento, 2016).

A consistência dos dados, como já referimos, tem a ver com o estado dos dados dentro da base de dados. A consistência deve estar diretamente ligada ao modo como o sistema precisa realizar as suas transações, sendo que o sistema deve fazer uso dessa eventual consistência. Por exemplo, no caso de um sistema bancário atual, não seria possível migrar para um sistema NoSQL, para não se correr o risco de não haver consistência numa transação, que conduziria a problemas graves no sistema (Nascimento, 2016).

Quando consideramos a modelação de dados, entramos na parte mais sensível, e uma das mais importantes no que se refere ao desempenho de uma aplicação, pois é fundamental considerar a sua eventual flexibilidade. A flexibilidade consiste na em não seguir toda as propriedades ACID (Lóscio, Oliveira, & Pontes, 2011).

2.4. Bases de dados relacionais

As bases de dados relacionais normalmente referenciada apenas por RDBMS, surgiram em meados da década de 80, no entanto, somente alguns anos mais tarde as empresas passaram a utilizar bases de dados hierárquicas, em vez de arquivos simples (do inglês *flat file*) como havia na década de 70 (Caldeira, 2015). Em 1985, Edgar Frank Codd, criador do modelo relacional, publicou um artigo onde definiu treze regras para que um SGBD seja considerado relacional (Codd, 1990), essas regras podem sintetizar-se como:

1. Regra Fundamental
2. Regra da informação
3. Regra da garantia de acesso
4. Tratamento sistemático de valores nulos
5. Catálogo dinâmico online baseado no modelo relacional
6. Regra da sub-linguagem abrangente
7. Regra da atualização de visões
8. Inserção, atualização e eliminação de alto nível
9. Independência dos dados físicos
10. Independência lógica de dados
11. Independência de integridade
12. Independência de distribuição
13. Regra da não-subversão.

Propriedade ACID

A propriedade ACID – que no original designa Atomicidade, Consistência, Isolamento e Durabilidade envolve quatro caraterísticas importantes que devem ser tidas em conta no decorrer de uma transação dentro da base de dados, conforme descrevemos de forma resumida na Tabela 3:

Tabela 3 - Propriedade ACID

Caraterística	Significado

A – Atomicidade	Todas as tarefas de uma transação são executadas ou nenhuma delas é. Não há transações parciais. Por exemplo, se uma transação se inicia utilizando 10 linhas, mas o sistema falhar após 2 atualizações, em seguida, a base de dados deve reverter as alterações a estas 2 linhas; caso uma transação não possa atingir um estado estável, o sistema deve retornar ao seu estado inicial.
C – Consistência	A operação leva a base de dados de um estado consistente para outro igualmente consistente. A transação deve deixar o sistema num estado correto ou, em alternativa, abortar, no caso de uma transação não conseguir atingir um estado estável; nesse caso, o sistema deverá retornar ao seu estado inicial.
I – Isolamento	O efeito de uma transação não está visível para outras transações até que esta seja confirmada. O comportamento de uma transação não é afetado por outras operações que estão sendo executadas em simultâneo. A transação deve serializar todo o acesso aos recursos compartilhados, e garantir que os programas concorrentes não alterem informações ou interfiram em operações uns dos outros.
D – Durabilidade	As alterações feitas por transações confirmadas são permanentes. Após uma transação ser concluída, a base de dados garante, através dos seus mecanismos de recuperação, que as mudanças a partir da transação não são perdidas. Mesmo se o sistema falhar, as alterações resultantes de uma transação são permanentes e duráveis.

Fonte: (Robinson, Webber, & Eifrem, 2015)

A aplicação deste conjunto de características é chamada de escrita consistente e estável em disco. Esta é uma camada de abstração que não é vista, nem mesmo manipulada pela camada aplicacional(camada esta fora da base de dados), todo o trabalho é executado pelo SGBD num caminho normal (Robinson et al., 2015).

Exemplos de SGBD

Tal como podemos ver na Tabela 4, o *ranking* dos modelos de bases de dados atualmente existentes mostra que o modelo relacional é o mais utilizado (DB-Engines, 2016).

Tabela 4 – *Ranking* dos modelos de bases de dados mais utilizados

Rank					**Score**		
May 2016	**Apr 2016**	**May 2015**	**DBMS**	**Database Model**	**May 2016**	**Apr 2016**	**May 2015**
1.	1.	1.	Oracle	Relational DBMS	1462.02	-5.51	+19.93
2.	2.	2.	MySQL	Relational DBMS	1371.83	+1.72	+77.56
3.	3.	3.	Microsoft SQL Server	Relational DBMS	1142.82	+7.77	+11.79
4.	4.	4.	MongoDB	Document store	320.22	+7.78	+42.90

5.	5.	5.	PostgreSQL	Relational DBMS	307.61	+3.89	+34.09
6.	6.	6.	DB2	Relational DBMS	185.96	+1.87	-15.09
7.	8.	8.	Cassandra	Wide column store	134.50	+4.83	+27.95
8.	7.	7.	Microsoft Access	Relational DBMS	131.58	-0.39	-14.00
9.	9.	10.	Redis	Key-value store	108.24	-3.00	+13.51
10.	10.	9.	SQLite	Relational DBMS	107.26	-0.70	+2.10

Fonte: (DB-Engines, 2016)

O *ranking* apresenta uma lista dos sistemas de gestão de base de dados classificados pela sua atual popularidade. A popularidade de um sistema é medida, de acordo com DB-Engines (2016), segundo os seguintes parâmetros:

1. Número de menções do sistema em *web sites*: medido o número de resultados em consultas dos motores de busca. Foram utilizados os sistemas da *Google* e *Bing* para esta medição. Para contar apenas resultados relevantes, procura-se <nome do sistema> em conjunto com o banco de dados, por exemplo, "Oracle" e "base de dados".
2. Interesse geral no sistema: para esta medição, foi utilizada a frequência de pesquisas no sistema *Google Trends*.
3. Frequência de discussões técnicas sobre o sistema - o número de questões sobre o sistema e o número de utilizadores interessados através dos conhecidos fóruns *IT-related Q&A StackOverflow* e *DBA Stack Exchange*.
4. Número de ofertas de emprego, em que o sistema é mencionado - foi analisado o número de ofertas sobre os principais motores de busca de emprego *Indeed* e *Simply Hired*.
5. Número de perfis em redes profissionais, em que o sistema é mencionado - foi utilizado o *LinkedIn*[17] para as buscas.
6. Relevância em redes sociais - foram contados no *Twitter* o número de *tweets*, em que o sistema é mencionado.

[17] O LinkedIn é uma ferramenta social para uso profissional, acessível em linkedin.com

2.5. Bases de dados NoSQL

Podemos considerar, atualmente, quatro tipos de bases de dados NoSQL. A Figura 4 apresenta os tipos de bases de dados NoSQL, e alguns exemplos de SGBD que utilizam neste formato:

Figura 4 - Bases de dados NoSQL.

Fonte: (Lanni, 2016)

Quanto às diferenças no modo de aceder dos dados (Cf. Tabela 5).

Tabela 5 - Comparação entre os sistemas NoSQL: Queries

	Suporte *Map Reduce*	REST	Queries "quase" *SQL*	Outros Métodos de Acesso	*Server side Scripts*	Índices Secundários
	Armazenamento do tipo Documentos					
MongoDB	✓	✓	×	Linguagem proprietária, CLI e APIs em várias linguagens	✓ *JavaScript*	✓
CouchDB	✓	✓	×	*APIs* em várias linguagens	✓ *JavaScript*	✓
RavenDB	✓	✓	×/✓ *LINQ Queries*	*API .NET*	✓	✓
	Armazenamento do tipo Família de Colunas					
Hbase	✓	✓	×/✓ Através de outros sistemas	*CLI, API Java* e *Thrift*	✓ *Java*	×
Cassandra	✓	×/✓ *APIs* de terceiros	✓ *CQL*	*CLI* e *APIs* em várias linguagens, incluindo *Thrift*	×	✓/×
HyperTable	✓	×	× Em Desenvolvimento	Suporte para *C++*, *Java*, *Perl*, *PHP*, *Python*, *Ruby* e *Thrift*	×	×
	Armazenamento do tipo Par Chave-Valor					
Riak	✓	✓	×	*CLI* e *APIs* em várias linguagens	✓ *JavaScript* e *Erlang*	✓
Redis	×	×/✓ *APIs* de terceiros	×	*CLI* e *APIs* em várias linguagens	✓ *Lua*	×
Voldemort	✓	✓	×	*API* Java e clientes *C++*, *Python* e *Ruby*	?	?
DynamoDB	✓ *Amazon Elastic Map Reduce*	✓	×	Linguagem propretária, *APIs* em várias linguagens	×	×
	Armazenamento do tipo Grafo					
Neo4J	×	✓	×	Linguagem *Cypher*, *CLI* e *APIs* em várias linguagens	✓ *Java*	✓
Titan	×/✓ através do sistema *Faunus*	×	×	*API Java* e *TinkerPop stack,* suporte para *Clojure* e *Python*	✓	✓
OrientDB	×	×	✓	Suporte para várias linguagens incluindo *TinkerPop Stack*	✓ *Java* e *JavaScript*	✓

Fonte: (Pereira, 2014, p. 47)

É importante termos em atenção as diferenças encontradas entre as propriedades seguidas por cada SGBD, iremos agora fazer uma breve comparação entre as propriedades de forma a facilitar o entendimento. Na Tabela 6 é possível visualizar uma adaptação de Neo4j (2016), sendo esta uma breve comparação entre as propriedades ACID e BASE.

Tabela 6 - Comparação entre ACID e BASE

ACID	BASE
CONSISTÊNCIA FORTE	Fraca consistência
ISOLAMENTO	Foco em disponibilidade
ATOMICIDADE	Respostas aproximadas
DISPONIBILIDADE	Disponibilidade forte

Teorema de CAP

O teorema de CAP refere-se à consistência, disponibilidade e tolerância à partição de dados (Diana & Gerosa, 2010). Salientam-se as três propriedades possíveis de um modelo NoSQL:

C – *Consistency* - Consistência

A – *Availability* - Alta disponibilidade[18]

P – *Network Partition Tolerance* - Tolerância à partição dos dados na rede

Assim, para conseguir assegurar uma destas características, a base de dados deve garantir que podemos racionar (por partição) os dados em nós, de modo a que estes continuem a garantir o acesso correto a esses mesmos dados através da rede estabelecida. O Teorema de CAP defende que somente duas das três propriedades referidas podem ser asseguradas em simultâneo, sendo que assim temos três possíveis conjugações:

- Sistemas CP - este modelo assegura consistência e tolerância a particionamento, abrindo mão da alta disponibilidade. Exemplos que seguem este padrão: *BigTable*[19], *HBase*[20], *MongoDB*[21], entre outros.
- Sistemas AP - este modelo assegura acessibilidade e o particionamento em rede, sendo que quando possível faz a atualização dos dados entres os nós. O intervalo é definido internamente pelo SGBD de modo a garantir as restantes características.

[18] Um sistema de alta disponibilidade do inglês High-Availability (HA) é um sistema computacional resistente a falhas de hardware, software ou mesmo energia, tendo como objetivo manter os serviços disponibilizados o máximo de tempo possível. Maiores informações podem ser obtidas no projeto Open-source *Linux-HA* http://www.linux-ha.org/wiki/Main_Page

[19] https://cloud.google.com/bigtable/

[20] https://hbase.apache.org/

[21] https://www.mongodb.com/

- Sistemas CA - nos modelos que utilizam este sistema temos a consistência e a alta disponibilidade em prática, sendo o mesmo aplicado aos conhecidos sistemas de bases de dados relacionais. Como exemplos deste modelo tem-se as bases de dados *Oracle*[22], *MySQL*[23], *Sybase*[24] e *SQL Server*[25].

Na Figura 5 é apresentado o mapa ilustrativo do modelo CAP, juntamente com alguns exemplos de bases de dados que seguem cada uma das opções antes referidas.

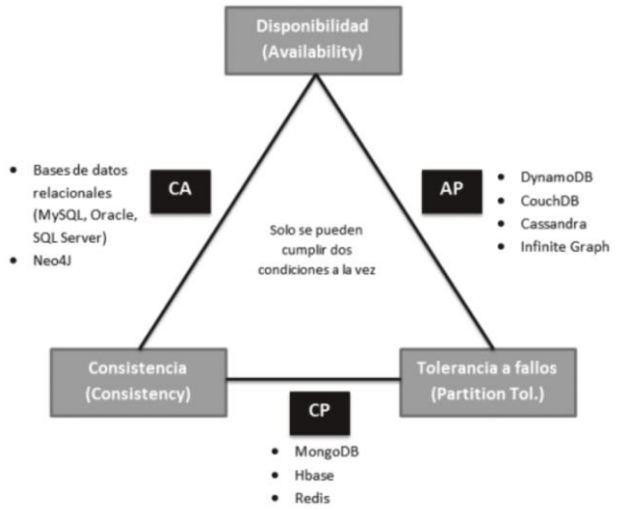


Figura 5 - Teorema de CAP – exemplos de bases de dados.

Fonte: (Antaño et al., 2014)

Propriedade BASE

A propriedade Base engloba um conjunto de propriedades a serem seguidas nos modelos NoSQL, que tem como foco a disponibilidade e o desempenho, não sendo o foco a consistência nem o isolamento dos dados. Pode-se comparar o ACID como sendo o equivalente para os modelos relacionais, sendo que a sigla BASE significa:

[22] https://www.oracle.com
[23] https://www.mysql.com
[24] http://infocenter.sybase.com
[25] https://www.microsoft.com/en-us/sql-server/

- BA – *Basically Available* – disponibilidade básica.
- S – *Soft-State* – não precisa ser consistente o tempo todo.
- E – *Eventually Consistent* – consistente no momento atual, sendo que não valida transações.

Tipos de bases de dados NoSQL

Na Figura 6 podemos ver os quatro tipos de bases de dados NoSQL atualmente disponíveis:

1- Key-Value ou Valor chave;
2- Graph DB ou Grafos
3- Column Family ou Família de colunas
4- Document ou documento

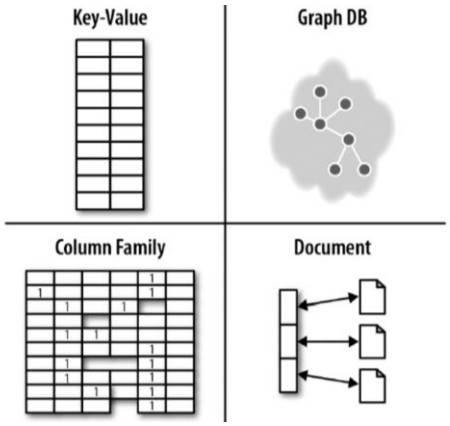


Figura 6 - Quadrante NoSQL.

Fonte: (Robinson et al., 2015, p. 197)

Segue agora uma explicação mais detalhada de cada um destes quatro tipos, com base em um exemplo de SGBD atualmente utilizado.

Grafos

Os grafos são muito utilizados na programação para a construção de algoritmos (conhecidos como algoritmos de grafos ou, no inglês, *Graph Algorithms*), no sentido de resolver problemas complexos relacionados com correlação de dados ou mesmo dados complexos (Neo4j, 2016).

Existem também bases de dados que são baseadas na teoria dos grafos, sendo que estas utilizam-se dos nós, relacionamentos e propriedades:

- *Nodes* ou nós: representam as entidades, como por exemplo pessoas, empresas, contas, ou qualquer outro item que seja necessário manter sob controlo. Eles são o equivalente do registo ou linha em uma base de dados relacional, ou o documento em uma base de dados de documentos.
- Relacionamentos: são as linhas que conectam os nós entre si, isto é, representam a relação entre estes. Padrões significativos emergem ao examinar as conexões e interconexões de nós, propriedades e relacionamentos. Os relacionamentos são o conceito-chave nas bases de dados de grafos, o que representa uma abstração que não está diretamente implementado noutros sistemas.
- As propriedades: são informações pertinentes que dizem respeito aos nós.

Na Figura 7 temos um exemplo de base de dados segundo este modelo, exemplificando a mesma com uma estrutura de dados.

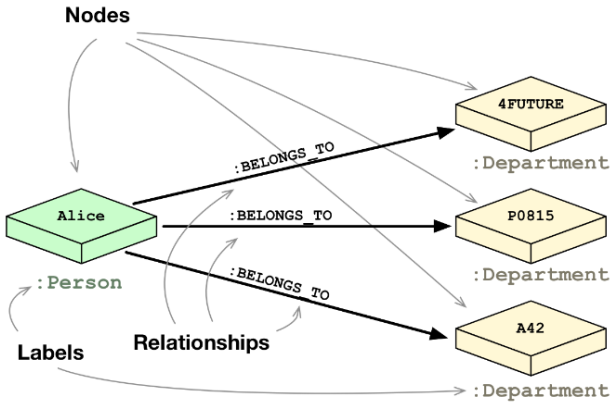


Figura 7 - Exemplo BD grafo.

Fonte: (Neo4j, 2016)

Outros exemplos de bases de dados usando o modelo grafos são os SGBDs Neo4j[26] e OrientDB[27].

Apresentamos na Figura 8 uma comparação entre os modelos relacional e de grafo, naquilo que tange a modelação de dados aplicacional, sendo aqui representada a mesma aplicação para ambos os modelos.

[26] https://neo4j.com/
[27] http://orientdb.com/orientdb/

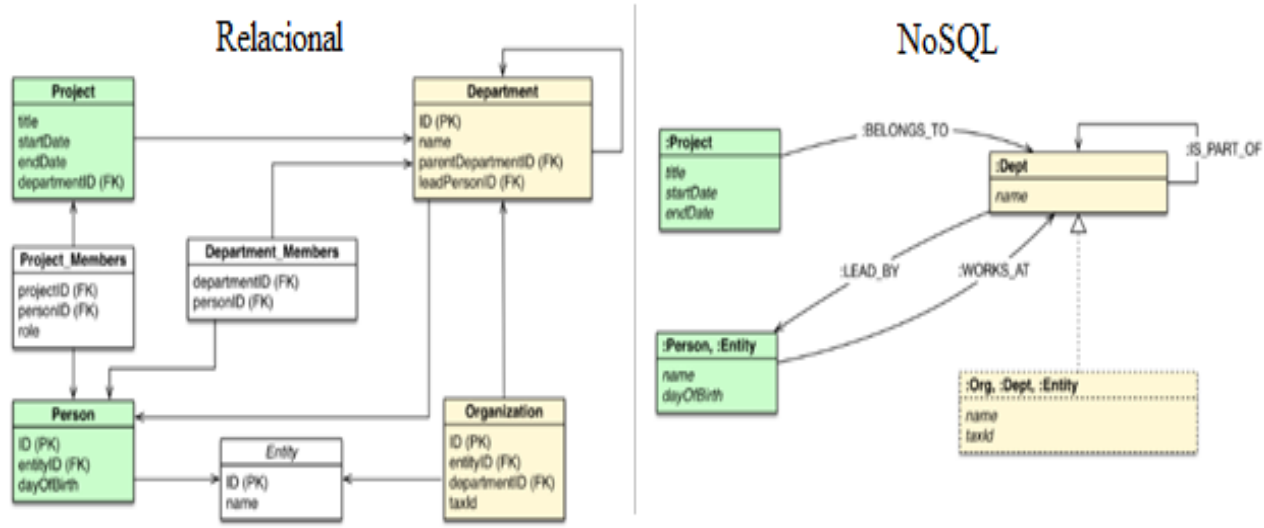


Figura 8 – Exemplo comparativo entre modelo relacional e modelo com grafos.

Fonte: (Neo4j, 2016)

Colunas

Os sistemas de bases de dados NoSQL baseados no tipo de família de colunas são conhecidos por possuírem uma grande capacidade escalar do tipo horizontal. Estes conseguem incluir quantidades enormes de dados (McCreary & Kelly, 2013). Sendo muito utilizada a função *MapReduce*[28]. Os modelos atualmente existentes são quase todos baseados no SGBD da *Google*[29] chamado *Bigtable*.

Este é um sistema de armazenamento distribuído para gerir dados estruturados em grande escala, e com uma latência mínima, sendo utilizado nos sistemas da própria *Google*, nomeadamente o *Google Earth* – como exemplo de grande utilização e baixa latência no tempo de resposta ao utilizar (McCreary & Kelly, 2013).

As estruturas de armazenamento do tipo família de colunas possuem quatro componentes principais utilizados na sua estrutura: colunas, super-colunas, família de colunas e superfamília de colunas (Robinson et al., 2015).

[28] https://static.googleusercontent.com/media/research.google.com/pt-PT//archive/mapreduce-osdi04.pdf
[29] https://www.google.com

A Figura 9 apresenta um exemplo de uma estrutura do tipo *Bigtable*.

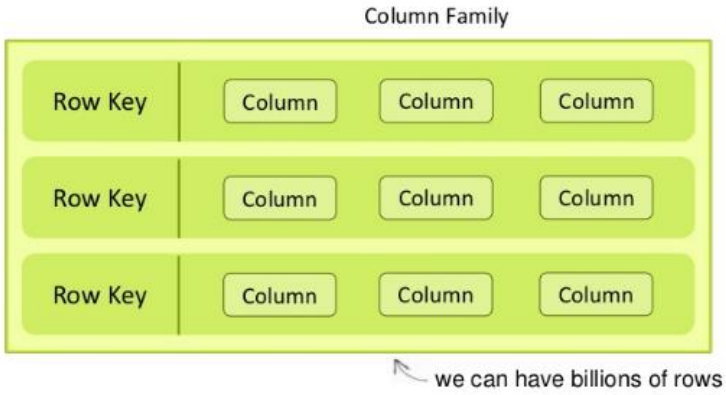


Figura 9 - Exemplo estrutura *Bigtable*.

Fonte: (Robinson et al., 2015)

Podemos ver agora alguns exemplos de bases de dados NoSQL pertencentes a este modelo, sendo que o *Hypertable* é altamente escalável e baseado no modelo *Bigtable* da *Google*.

São ainda citados outros conhecidos SGBD no âmbito deste modelo:

- Cassandra[30]
- *Hypertable*[31]
- *Google Bigtable*

Documentos

Este tipo de SGBD NoSQL é utilizado aquando da necessidade de se armazenar documentos, sendo estes indexados e o mecanismo de busca é simples. Eis alguns exemplos:

- RavenDB[32]
- CouchDB[33]
- MongoDB
- MarkLogic Server[34]
- eXist[35]
- MySQL[36]

[30] http://cassandra.apache.org/
[31] http://www.hypertable.org/
[32] https://ravendb.net/
[33] http://couchdb.apache.org/
[34] http://www.marklogic.com/what-is-marklogic/
[35] http://exist-db.org/
[36] https://www.mysql.com

Chave-Valor

Este modelo permite a visualização dos dados de forma particionada, ou seja, os dados são apresentados em uma tabela *hash*. De maneira simples, a base de dados é composta por um conjunto de chaves que estão associadas a um único valor, podendo ser do tipo *string* ou binário (Lóscio, Oliveira e Pontes, 2011). Segundo Miguel e Fraga (2013), as informações são armazenadas em objetos indexados por chaves, responsáveis por identificá-los unicamente, e realizar a sua posterior consulta. Podemos ver na Figura 10 um exemplo de armazenamento que utiliza o modelo chave-valor (do inglês *Key-Value Store)*, tal como é originalmente conhecido.

Figura 10 - Modelo NoSQL Key-*Value Store*.

Fonte: (Barriviera & Fraga, 2013)

Outros SGBD que seguem este modelo são:

- Chave/Valor (*Key/Value*)
- Memcachedb
- Project Voldemort
- Redis
- SimpleDB
- Hbase

2.6. Big Data

Embora o termo "*big data*" seja relativamente novo, o ato de recolher e armazenar grandes quantidades de informações para eventual análise de dados é bem antigo. O conceito ganhou força no início dos anos 2000 quando, segundo Sicular (2013), um analista famoso deste setor, Doug Laney, articulou a definição de *Big Data* como os três V's ou três fatores chave – volume, velocidade e variedade.

É importante lembrar que o valor principal de *Big Data* não vem dos dados em sua forma bruta, mas do processamento e análise destes dados e os insights, produtos e serviços que surgem desta análise. As mudanças radicais nas tecnologias e abordagens de gestão de *Big Data* devem ser acompanhadas, de forma semelhante, por mudanças dramáticas na forma como os dados suportam decisões e geram inovação de produtos e serviços (Davenport & Dyche, 2013).

O conceito de *Big Data*

Esta classificação é dada devido ao tamanho e complexidade da informação a ser armazenada, e não somente ao tipo de SGBD utilizado. A *Big Data* preocupa-se com a quantidade de dados (volume, variedade e velocidade) para o processamento dos mesmos, e concebe-se como o principal desafio a vencer, saber o que guardar e fazê-lo de modo cada vez mais rápido. *Big Data* é um termo amplamente usado para referir um conjunto de dados muito grande ou complexo num sistema de bases de dados. De referir que existem três fatores chave que classificam um sistema como um sistema de *Big Data*:

- Volume
- Velocidade
- Variedade

Na Tabela 7 podemos ver uma análise comparativa entre os sistemas de bases de dados com maior necessidade de armazenamento.

Tabela 7 - Comparativo entre BI, DM, *Big Data*

Características	BI (Business Intelligence)	Data Mining	Big Data
1 - Virtude da solução	Volumetria - Monitorar o desempenho dos indicadores das operações.	Metodologia científica e algoritmos. Descobrir padrões de comportamento de dados. Detecção de pontos cegos da gestão. Análise estatística intensa e pontual.	Data mining em grande escala. Geração de conhecimento de gestão apoiado por inteligência e capacidade computacional. Análise estatística intensa e contínua.
2 - Tipos de Dados	Dados estruturados em planilhas, banco de dados relacionais e dimensionais, etc.	Dados estruturados em planilhas, banco de dados relacionais e dimensionais, etc.	Dados estruturados, semiestruturados e não estruturados em bancos de dados NoSQL ou TripleStores.
3 - Estilo da análise	Reflete apenas o passado dos dados em **pequena** ou **grande** escala. Não há inteligência no sistema, sendo necessário profissionais da gestão para interpretar as informações e tomada de decisão.	Permite fazer a predição e descoberta de fatores relevantes ao negócio em **pequena** escala usando inteligência computacional. Necessita de profissionais da gestão trabalhando em colaboração com cientistas da informação.	Permite fazer a predição e descoberta de fatores relevantes ao negócio em **grande** escala usando inteligêcia computacional. Necessita de profissionais da gestão trabalhando em colaboração com cientistas da informação.
4 - Resultados esperados	Diversas visualizações de gráficos consolidadas em painéis de controle conhecidos como *dashboards*.	Relatório de recomendação estratégica.	Painéis de controle com indicadores preditivos e recomendações estratégicas.
5 - Foco	Monitorar indicadores tais como preço, valor, temperatura, custo total, etc.	Identificar padrões de comportamento dos dados, criando novos indicadores de análise para o BI.	Extração de conhecimento de grandes massas de dados com fontes e tipos variados.
6 - Comercialização	Custo de implantação, integração do sistema e mensalidade por usuário.	Valor por projeto, envolvendo o custo da produção do relatório.	Custo de implantação, integração do sistema e mensalidade e/ou comissionamento sobre o resultado do faturamento.
7 - Volume de dados	Alto, porém limitado ao processamento dos bancos de dados relacionais/dimensionais.	Baixo, trabalho por amostragem (pequenas parcelas) de dados com alto custo de processamento.	Alto com estruturas de processamento distribuídos e grande demanda de processamento.
			www.aquare.la

Fonte: (Aquarela, 2016)

Volume

As organizações coletam dados de uma grande variedade de fontes, incluindo transações comerciais, redes sociais e informações de sensores ou dados transmitidos de máquina para máquina. No passado, armazenar tamanha quantidade de informação teria sido um problema, mas o aparecimento de novas tecnologias, como o *Hadoop*[37], tem aliviado a carga (Aquarela, 2016).

Velocidade

Os dados fluem numa velocidade sem precedentes e devem ser tratados em tempo hábil. *Tags* de RFID, sensores, celulares e contadores inteligentes estão impulsionado a necessidade de lidar com imensas quantidades de dados em tempo real, ou quase real.

[37] http://hadoop.apache.org

Variedade

Os dados que são extraídos deste sistema precisam, acima de tudo, de retornar valor a seu utilizador, ou seja, além das qualidades acima descritas, os dados extraídos precisam de acrescentar valor ao negócio, caso contrário o sistema por si mesmo não teria fundamentação e real necessidade, isso quanto a classificação de *Big data*.

Tecnologias *Big Data*

O *Hadoop* é uma plataforma *open source*[38] desenvolvida especialmente para processamento e análise de grandes volumes de dados, sejam eles estruturados ou não. O projeto é mantido pela *Apache Foundation*[39], mas conta com a colaboração de várias empresas, como a *Yahoo*[40], *Facebook*, *Google* e *IBM*[41].

Pode-se dizer que o projeto teve início em meados de 2003, quando a *Google* criou um modelo de programação que distribui o processamento entre vários computadores para ajudar o seu mecanismo de busca a ficar mais rápido e livre da necessidade de servidores poderosos (e bastante caros). Esta tecnologia recebeu o nome de *MapReduce* (Davenport & Dyche, 2013).

A *Google* apresentou o *Google File System*[42] (GFS), um sistema de arquivos especialmente preparado para lidar com processamento distribuído e, como não poderia deixar de ser no caso de uma empresa como esta, grandes volumes de dados (na ordem de *terabytes* ou mesmo *petabytes*). O *Hadoop*, cuja implementação do sistema de arquivos recebeu o nome de *Hadoop Distributed File System*[43] (HDFS).

Segundo Mysore, Khupat & Jain (2014) podemos classificar os problemas de negócio de acordo com o tipo de *Big Data*, conforme indicado na Tabela 8.

Tabela 8 - Problemas de negócios relacionados com *Big Data*

[38] https://opensource.org/
[39] https://www.apache.org/
[40] https://www.yahoo.com/
[41] https://www.ibm.com
[42] https://research.google.com/archive/gfs.html
[43] https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html

Problemas de negócios	Tipo de *Big Data*	Descrição
Serviços públicos: prever o consumo de energia	Dados gerados por máquina	Concessionárias de serviços públicos implementaram medidores inteligentes para medir o consumo de água, gás e eletricidade a intervalos regulares de uma hora ou menos. Esses medidores inteligentes geram enormes volumes de dados de intervalo que precisam ser analisados. As concessionárias utilizam sistemas grandes, caros e complicados para gerar energia. Cada rede contém sensores sofisticados que monitorizam voltagem, corrente, frequência e outras características operacionais importantes. Para ter eficiência operacional, a empresa precisa monitorizar os dados entregues pelo sensor. Uma solução de *Big Data* pode analisar os dados de geração de energia (fornecimento) e de consumo de (demanda) usando medidores inteligentes.
Telecomunicações: analítica de perda de clientes	Dados da *WEB* e sociais	Operadores de telecomunicações precisam criar modelos detalhados de perda de clientes que incluam dados de médias sociais e de transação, como CDR's, para estar à frente da concorrência. O valor dos modelos de perda de clientes depende da qualidade dos atributos e do comportamento social destes.
	Dados de transação	Provedores de telecomunicações que implementam uma estratégia analítica preditiva, podem gerenciar e prever a perda analisando os padrões de chamada dos assinantes.
Marketing: Análise de sentimento	Dados da *WEB* e sociais	Departamentos de *marketing* usam *feeds* do *Twitter* para realizar análise de sentimento e determinar o que os utilizadores estão falando sobre a empresa e seus produtos ou serviços, especialmente após o lançamento de um novo produto ou *release*. O sentimento do cliente deve ser integrado nos seus dados de perfil para derivar resultados significativos. O *feedback* do cliente pode variar de acordo com seus aspetos demográficos.

Atendimento ao cliente: Monitoramento de chamada	Gerado por humanos	Departamentos de TI estão usando soluções de *Big Data* para analisar *logs* de aplicativo e obter *insight* que possa melhorar o desempenho do sistema. Arquivos de *log* de diferentes fornecedores de aplicativos estão em formatos diferentes e precisam ser padronizados para uso pelos departamentos de TI.
	Biométrica	
Varejo e *marketing*: Dados de dispositivos móveis e direcionamento com base em localização	gerados por máquina	Varejistas podem atingir seu público-alvo com promoções específicas e cupões com base em dados de localização. As soluções são geralmente projetadas para detetar a localização de um usuário ao entrar em uma loja ou através de um GPS.
		Dados de localização combinados a dados de preferência do cliente obtidos em redes sociais permitem que os varejistas direcionem campanhas de *marketing online* e nas lojas com base no histórico de compras. As notificações são entregues por meio de aplicativos remotos, SMS e *e-mail*.
	Dados de transação	

Fonte: (Mysore et al., 2014)

Podemos ainda ver na Figura 11, de acordo com Mysore et al. (2014), as várias categorias e utilizações aplicadas para classificar os sistemas de *Big Data* e a sua utilização.

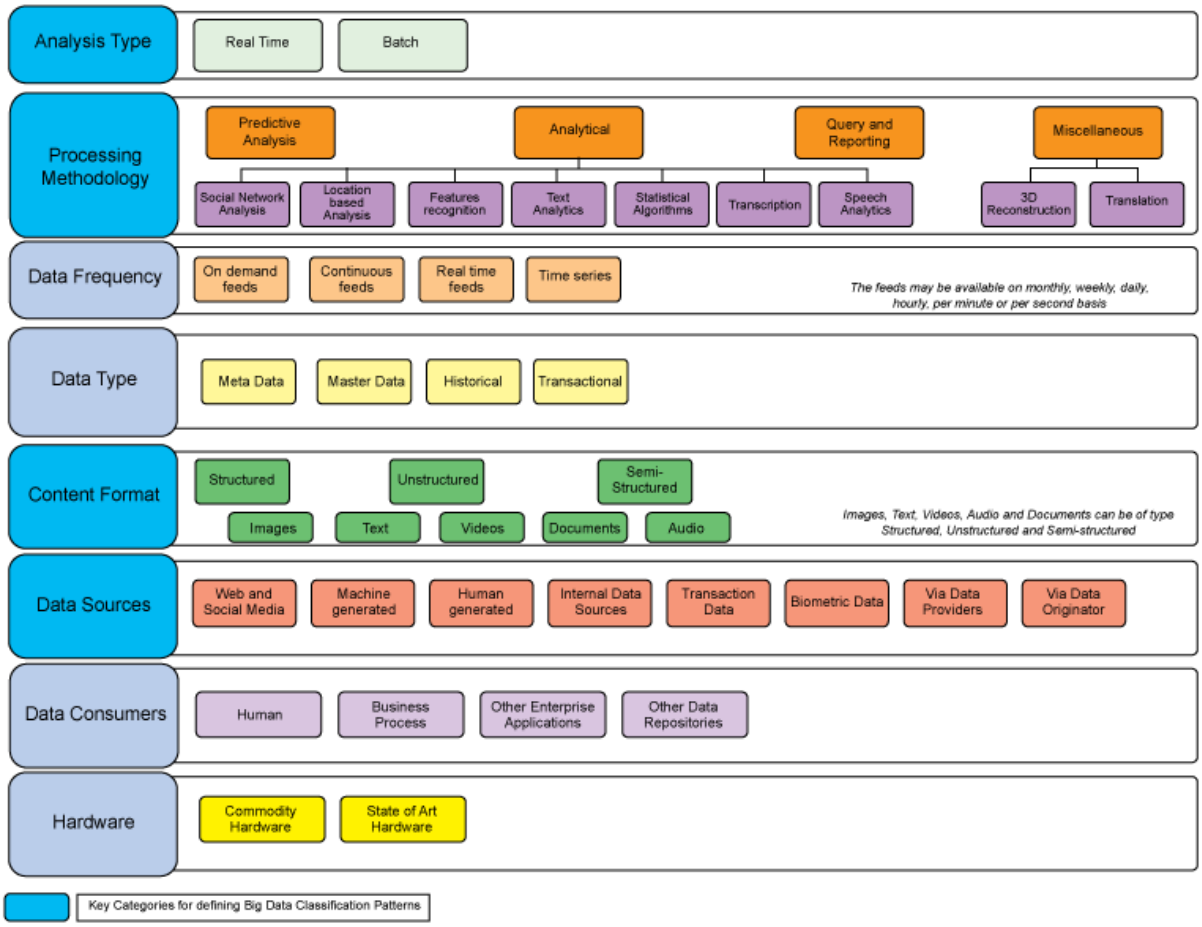


Figura 11 - Classificação do *Big Data*.

Fonte: (Mysore et al., 2014)

2.7. Migração de bases de dados relacionais para NoSQL

Nesta secção serão apresentadas as informações inerentes ao processamento e à compreensão do processo de migração. Na Figura 12 (MongoDB, 2015) enumeram-se alguns dos projetos que sofreram migração do modelo relacional para o modelo NoSQL.

Organization	Migrated From	Application
eHarmony	Oracle & Postgres	Customer Data Management & Analytics
Shutterfly	Oracle	Web and Mobile Services
Cisco	Multiple RDBMS	Analytics, Social Networking
Craigslist	MySQL	Archive
Under Armour	Microsoft SQL Server	eCommerce
Foursquare	PostgreSQL	Social, Mobile Networking Platforms
MTV Networks	Multiple RDBMS	Centralized Content Management
Buzzfeed	MySQL	Real-Time Analytics
Verizon	Oracle	Single View, Employee Systems
The Weather Channel	Oracle & MySQL	Mobile Networking Platforms

Figura 12 - Projetos que migraram do SGBD relacional para NoSQL.

Fonte: (MongoDB, 2015)

Na análise ao artigo *RDBMS to MongoDB Migration Guid* (MongoDB, 2015), pode-se verificar que não são identificadas as técnicas e os métodos utilizados para executar cada migração.

Podemos também validar que não são divulgadas as métricas, janelas de paragem (*downtime aplicacional)*, ou mesmo problemas encontrados durante a migração.

A partir desta figura podemos considerar que existe uma real possibilidade de pesquisa e uma oportunidade de validar esses possíveis métodos de migração entre estes dois modelos de sistemas de gestão de bases de dados (SGBD), de modo a criar um contexto onde seja possível validar o meio da migração.

2.8. Trabalhos anteriores relacionados

Na sua tese de mestrado Gomes (2011) estudou a diferença entre os modelos relacional e não relacional. Analisou assim a diferença de estrutura e o mapeamento de objetos necessário para realizar uma migração de uma base de dados de um modelo para o outro, mas não fez teste de

caso ou uso de ferramentas existentes para esta migração – enquanto nós o iremos fazer[44] –, usou apenas *scripts* manualmente desenvolvidos em algumas *application programming interfaces* (API's).

No trabalho intitulado "Requisitos para Ferramentas de Migração de Dados" (Neto et al., 2015) os autores apresentam uma tabela com requisitos para migração de bases de dados, com base nas suas próprias experiências profissionais, e com embasamento teórico de outros trabalho existentes na área. Esta tabela será uma boa referência durante o desenvolvimento desta dissertação.

Em 2014 foi publicado um artigo com o título "Migracion de Bases de Datos SQL a NoSQL" (Antaño et al., 2014), que indica algumas métricas usadas, tal como foram propostas pelo autor, num âmbito genérico no que se refere às migrações das bases de dados relacionais para NoSQL. Este trabalho server igualmente como referência, chamando a atenção para alguns pontos importantes nas métricas que serão analisadas nesta dissertação.

Outros autores apresentam a migração de uma aplicação *web* para uma base de dados NoSQL localizada na nuvem (Costa, Rocha, Mendonc, & Maia, n.d.). Cuer (2014) realizou uma comparação de desempenho entre os modelos relacional e NoSQL, não se focando este na migração em si. No entanto, focando-se no desempenho destes dois distintos modelos.

Portanto, concluímos que estes trabalhos foram diretamente aplicados em casos específicos de migração, salientando-se que o objetivo da migração estava definido. Porém, na nossa proposta iremos montar um modelo que se consiga aplicar em qualquer âmbito de migração de BD, do modelo relacional para o NoSQL, obedecendo, é claro, às características dos modelos citados.

[44] Cf. descrito na parte relativa à investigação – Estudo de Caso

Capitulo III - Estudo de Caso

3.1. Introdução ao Capítulo

Para responder à questão principal, e aos objetivos, desta pesquisa, iremos centrar-nos no que podemos designar como um estudo de caso (cf. secção 3.3.). Serão abordados os pontos relevantes para a avaliação e validação da migração (de uma BD relacional para NoSQL). Será ainda explicada a metodologia seguida, e apresentados os cenários aplicacionais simulados, o modo como os mesmos foram configurados, bem como quais os procedimentos adotados em cada etapa da migração, e quais as métricas que foram usadas para recolher os dados necessários – para avaliar a migração –, para análise e comparação, em determinados períodos de tempo (ou intervalos).

3.2. Objetivos

A revisão de literatura permitiu-nos entender o funcionamento dos modelos de bases de dados (BD) que vamos estudar, e qual o seu papel, em cada momento, no processo de migração. Os métodos de migração e as métricas serão coletados em todos os cenários migrados (cf. secção 3.6 Componentes da migração), permitindo assim a comparação destes modelos de BD. Tentaremos responder à questão de partida, e entender qual é o método mais eficiente para proceder à migração de uma base de dados relacional para uma NoSQL – conforme os métodos a propor e as métricas a usar. O que passa por analisar e compreender como estes modelos podem ser úteis, de forma a podermos executar e gerar informação de cariz técnico e analítico, e posteriormente responder aos objetivos inicialmente propostos, e que aqui recordamos:

- Identificar quais são os requisitos e as várias fases de uma migração.
- Verificar os possíveis métodos de migração de uma base de dados relacional para uma NoSQL.
- Avaliar cada um destes métodos, de acordo com as etapas seguidas em cada fase da migração – recorrendo a métricas específicas e adequadas, em cada caso.
- Comparar os vários métodos de acordo com as métricas estabelecidas.
- Determinar o método de migração mais eficiente para cada um dos cenários de bases de dados estudados (OLTP, OLAP e HTAP).

3.3. Metodologia

Para responder à questão formulada e atingir estes objetivos, recorremos ao método de estudo de caso. Carmo e Ferreira (1998) define um estudo de caso como uma abordagem empírica que:

- Investiga um fenómeno atual no seu contexto real;

- Considera que os limites entre determinados fenómenos e o seu contexto não são claramente evidentes;

- utiliza variadas fontes de dados (Carmo & Ferreira, 1998).

Num estudo de caso o investigador não pode controlar os acontecimentos, focalizando-se na investigação de um fenómeno atual no contexto onde o mesmo ocorre (Carmo & Ferreira, 1998). É assim que procedemos neste trabalho, uma vez que após termos uma aplicação cliente[45] conectada na base de dados, em execução, e sem manipularmos os processos aplicacionais em si mesmos, decorrem os processos que executam a migração desejada, seguindo um fluxo normal aplicacional, sendo então coletadas as informações referentes às métricas que serão por nós analisadas, de forma a podermos medir a eficiência de todo o processo em curso.

Os dados recolhidos num estudo de caso podem ser de natureza qualitativa, quantitativa ou ambas (Carmo & Ferreira, 1998). No nosso caso, temos um estudo de caso de natureza qualitativa, em que os investigadores definem a questão de partida em função da sua própria experiência, ou de situações ligadas à sua vida prática. Embora em um estudo de caso se possam utilizar diferentes técnicas de recolha de dados, nesta pesquisa privilegiamos a técnica da observação, pois submetemos os dados a testes e observamos os resultados dai provenientes.

A fiabilidade do estudo de caso é garantida através de uma descrição pormenorizada e rigorosa da forma como o estudo foi realizado, a qual implica, não só uma explicitação dos pressupostos e da teoria subjacentes ao próprio estudo, mas também uma descrição do processo de recolha de dados, e da forma como se obtiveram os resultados (Carmo & Ferreira, 1998).

Podemos assim, em resumo, e em termos metodológicos, apresentar os passos a dar e o procedimento a seguir, neste estudo de caso. Tomar-se-ão, sequencialmente as ações:

[45] Neste trabalho a aplicação é descrita na secção 3.4 Descrição da arquitetura em Descrição da carga aplicacional.

- Selecionamos previamente entre os métodos de migração existentes, quais serão e replicados (*online, offline* automático e *offline* manual) para analisar e avaliar uma migração.
- Serão identificados e enumerados os requisitos necessários para uma migração de uma base de dados relacional para NoSQL, independentemente do procedimento seguido.
- Serão enumeradas todas as fases que devem ser seguidas durante a aplicação de cada um dos três métodos a replicar, no decorrer de uma migração.
- Será feita a comparação de métricas observadas e analisadas, quebrando em períodos iguais a cada etapa da migração, de forma a poder interpretar os valores medidos em casa etapa.
- Serão comparados os valores de todas as métricas, possibilitando caracterizar qual a importância de cada métrica, e o que traduz, em cada método.

Para que os objetivos sejam alcançados, devemos garantir que:

- A migração acontecerá em sua totalidade, seguindo os requisitos, e as fases estipuladas.
- Serão aplicados os métodos durante os testes em laboratório, sendo os mesmos analisados e validados em termos da sua exequibilidade para uma migração.
- Após análise das métricas coletadas, verificar qual a métrica mais utilizada (e mais relevante) a medir durante uma migração.
- Comparar o valor de cada métrica coletada, em cada um dos diferentes métodos usados (cf. secção 3.6 Componentes da migração em Métricas para migração), e mostrar qual o melhor método a seguir.
- Mostrar o resultado global obtido através de todos os métodos, em cada um dos cenários (cf. secção 3.4 Descrição da arquitetura em Cenário OLTP / OLAP e HTAP) avaliados, podendo assim analisar-se qual o comportamento, eme termos de performance, associado a cada método de migração.

A abordagem concetual, e o respetivo procedimento a adotar, foi formulado de acordo com o quadro teórico que resumimos na revisão da literatura. Propondo-se assim um modelo que nos indica a execução de testes em laboratório, seguindo um conjunto de fatores previamente

definidos, de acordo com as normas e padrões vigentes[46]. Após a construção e elaboração dos vários cenários[47], serão revisados os detalhes a atender na migração, e este modelo concetual será aplicado igualmente em cada um deles, coletando as métricas estabelecidas e computando os valores obtidos, individualmente.

Após a execução da migração, será possível comparar os resultados obtidos em cada uma das etapas desse processo, em cada um dos (nove) cenários propostos (cf. Tabela 17), e validar os valores de cada métrica usada.

Na Tabela 9 especificamos o procedimento seguido, em cada fase deste estudo, de acordo com a abordagem metodológica já descrita.

Tabela 9 - Objetivos e opções metodológicas associadas

OBJETIVO	**OPÇÃO METODOLÓGICA**
Identificar quais são os requisitos e as várias fases de uma migração.	Efetuar a revisão da literatura, pesquisar em teses da área, validar as documentações dos principais desenvolvedores de software da área temática e filtrar aqueles que melhor auxiliam no enquadramento deste trabalho.
Verificar os diferentes métodos de migração de uma base de dados relacional para uma NoSQL.	
Avaliar cada um destes métodos, de acordo com as etapas seguidas em cada fase da migração – recorrendo a métricas específicas e adequadas, em cada caso	Realizar testes de laboratórios e aplicar controladamente cada um dos métodos identificados, possibilitando a extração de métricas para medir desempenho.
Comparar os vários métodos de acordo com as métricas estabelecidas.	Armazenar e possibilitar a comparação de todas as métricas entre todos os métodos.
Determinar o método de migração mais eficiente para cada um dos cenários de bases de dados estudados (OLTP, OLAP, HTAP)	Analisar os resultados das métricas e comparar os valores entre todos os métodos, respondendo ao objetivo.

[46] Cf. Capítulo III, secção 3.4 Descrição da arquitetura.
[47] Cf. Capítulo III, secção 3.6 Componentes da migração.

3.4. Descrição da arquitetura

Vamos descrever a arquitetura e o modo como esta foi configurada – em cada servidor da expressa – e utilizada durante o processo de migração, de forma a que os testes previstos possam ser devidamente realizados e, quando necessário, replicados. A arquitetura será toda virtualizada, utilizando o portátil pessoal disponível para os testes. Estes testes podem a qualquer momento ser realizados em ambiente físico, seguindo as mesmas configurações que iremos detalhar.

Na Tabela 10 indicamos a configuração do ambiente hospedeiro, sendo este um portátil de uso do próprio aluno onde os testes se realizaram.

Tabela 10 - Configuração do ambiente físico (*hardware*)

Ambiente	Componente	Explicação	Valor
Físico:	*Hostname*	Identificação do servidor	*fabolive*
	RAM	Quantidade de memória utilizada	16 GB
	InternalNetwork	Rede interna somente para comunicação entre os servidores	192.168.56.150
	Storage	Capacidade de armazenamento	1 TB
	SisOp	Sistema operativo utilizado	*Windows* 10 *Enterprise*
	CPU	Processador	Intel(R) Core(TM) i5 2.4GHz

A configuração utilizada em cada servidor virtual de base de dados será a mesma, detalhada na Tabela 11, onde indicamos também a versão de *software* utilizada nos servidores[48]:

Tabela 11 - Configuração dos servidores

Ambiente	Componente	Explicação	Valor
Origem	*Hostname*	Identificação do servidor	Host01
	RAM	Quantidade de memória utilizada	4096 MB
	InternalNetwork	Rede interna somente para comunicação entre os servidores 100MB	192.168.0.1

[48] Cf. Capítulo III, secção 3.4 Descrição da arquitetura em Software utilizado.

	Storage	Capacidade de armazenamento	120 GB
	SisOp	Sistema operativo utilizado	Oracle Linux 6.5
	Base de dados	Oracle *enterprise edition* 12c	12.1.0.1
Destino	*Hostname*	Identificação do servidor	Host02
	RAM	Quantidade de memória utilizada	4096 MB
	InternalNetwork	Rede interna somente para comunicação entre os servidores 100MB	192.168.0.2
	Storage	Capacidade de *storage*	120GB
	SisOp	Sistema operativo utilizado	Oracle Linux 6.5
	Base de dados	MongoDB	MongoDB 3.2.9

Todo o fluxo e conectividade serão realizados automaticamente pela camada virtual, ou seja, o *software* de virtualização foi também responsável pela comunicação entre os servidores de bases de dados. A infraestrutura a utilizar pode ser consultada na Figura 13[49], sendo que esta é a mesma para todos os cenários[50], onde os dados serão extraídos da origem, e inseridos no destino, mudando apenas o caminho utilizado para a migração.

Figura 13 - Infraestrutura utilizada.

De forma a aceder às bases de dados nos respetivos servidores, foram utilizadas as portas padrão de todo o *software*, podendo esses tipos de *software* ser consultados na documentação original

[49] Sempre que numa figura não seja indicada a Fonte, isso significa que essa mesma Figura é da autoria do investigador.
[50] Cf. secção 3.4 Descrição da arquitetura em Descrição da carga aplicacional.

de cada fornecedor, nas versões listadas na secção de *software*[51]. Nesta arquitetura, a rede é emulada através do *software* de virtualização, configurada para responder com uma velocidade igual ou inferior a 100MB. Existem placas de rede que funcionam numa velocidade maior, mas neste trabalho, esta foi a arquitetura de referência; no entanto, se for necessário, poderemos aplicar outras velocidades sem necessidade de alteração das métricas, ou mesmo do plano.

Descrição da carga aplicacional

Nesta secção são apresentados os cenários utilizados durante os testes de laboratório. Aqui foram aplicadas as técnicas e coletadas as métricas, possibilitando assim que as mesmas sejam devidamente analisadas e os resultados analisados.

Iremos também introduzir os conceitos macros utilizados durante a migração e os modelos de coletas de informações, análise e execução. Durantes cada teste, foi realizado um carregamento de ambiente aplicacional, ou seja, foi realizado um carregamento de dados reais durante os testes. Este tipo de simulação gera uma carga na base de dados de origem, sendo esta conhecida como teste de *Benchmark*, segundo o *site* (TPC, 1992a). Estas cargas são classificadas conforme o modelo de dados, existindo uma grande quantidade de modelos. Neste caso utilizaremos os modelos:

- *On-line Transaction Processing – TPC-C*
- *On-line Analytical Processing - TPC-H*
    - TPC-C[52]

O modelo "TPC-C simulates a complete computing environment where a population of users executes transactions against a database. The benchmark is centred around the principal activities (transactions) of an order-entry environment. These transactions include entering and delivering orders, recording payments, checking the status of orders, and monitoring the level of stock at the warehouses. While the benchmark portrays the activity of a wholesale supplier, TPC-C is not limited to the activity of any particular business segment, but, rather represents any industry that must manage, sell, or distribute a product or service" (TPC, 1992b, p. 1).

[51] Cf. Capítulo III, secção 3.4, subsecção Software utilizado.
[52] http://www.tpc.org/tpcc/

- TPC-H[53]

Este modelo pode ser definido conforme a entidade consultada, "The TPC Benchmark™H (TPC-H) is a decision support benchmark. It consists of a suite of business oriented ad-hoc queries and concurrent data modifications. The queries and the data populating the database have been chosen to have broad industry-wide relevance. This benchmark illustrates decision support systems that examine large volumes of data, execute queries with a high degree of complexity, and give answers to critical business questions" (TPC, 1992b).

Para os testes executados neste trabalho, serão utilizados os modelos referente aos carregamentos TPC-C e TPC-H, simulando o comportamento de um sistema OLTP e OLAP, respetivamente.

As bases de dados envolvidas foram os SBGDs *Oracle database* e *MongoDB.* Ambas possuem grande quantidade de informação e são de fácil acessibilidade, sendo inclusive ambas consideradas as mais utilizadas em cada género (tal como antes referenciado). No enquadramento estão também as atividades que possuem ligação indireta com este conteúdo, no entanto (não estão previstas neste estudo as etapas de arquitetura e modelo de dados). Ambas as atividades – arquitetura e modelo de dados – devem ser realizadas antes da migração se iniciar, porque este é um trabalho a ser realizado pela arquitetura do produto, e não está propriamente ligado à execução da migração em si. Pode ainda necessitar de alterações ao modelo de dados em qualquer momento, que não durante a migração.

Foquemo-nos agora nos cenários considerados.

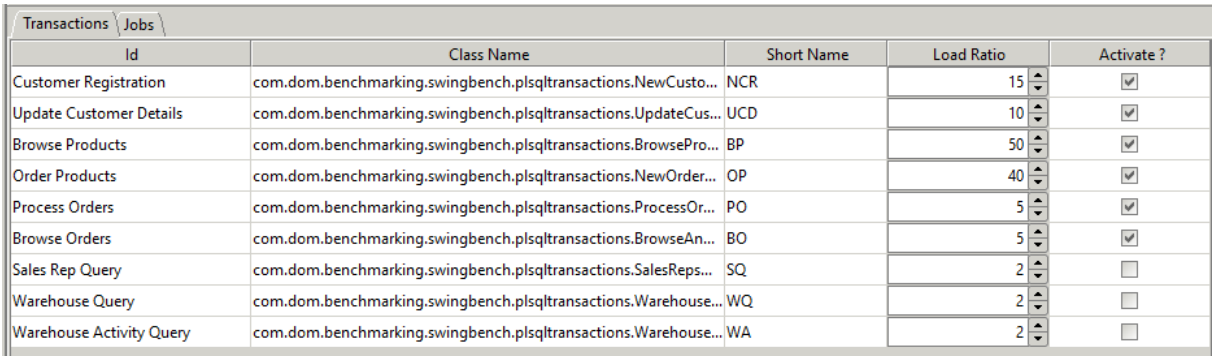
Cenário OLTP

O cenário OLTP consiste numa base de dados relacional com aplicação única executando *front-end* em java[54] caracterizado como carregamento do tipo OLTP (TPC-C). A aplicação utilizada para gerar toda a carga e simular o cenário acima descrito é o *SwingBench*[55], que está detalhado

[53] http://www.tpc.org/tpch
[54] https://www.java.com/pt_BR/
[55] Swingbench is a free load generator (and benchmarks) designed to stress test an Oracle database (11g, 12c)

na secção do *software*[56]. Neste tipo de carga é utilizado o *schema*[57] SOE (*SchemaOrderEntry*), que caracteriza as transações que ocorrem numa base de dados transacional do modelo relacional, segundo o *site* TPC (TPC, 1992) que juntamente com diversas entidades tecnológicas publicam vários artigos na área.

Para sabermos quais as transações que são geradas e em que quantidade, podemos consultar a carga que será gerada pela ferramenta na Figura 14, sendo que o "*load ratio*" é a quantidade de transações envolvendo a tabela na coluna "*Id*". Aqui o valor da coluna representada é o valor que será utilizado para dividir os ciclos entre todos os objetos ativos.


Transactions | Jobs

Id	Class Name	Short Name	Load Ratio	Activate ?
Customer Registration	com.dom.benchmarking.swingbench.plsqltransactions.NewCusto...	NCR	15	☑
Update Customer Details	com.dom.benchmarking.swingbench.plsqltransactions.UpdateCus...	UCD	10	☑
Browse Products	com.dom.benchmarking.swingbench.plsqltransactions.BrowsePro...	BP	50	☑
Order Products	com.dom.benchmarking.swingbench.plsqltransactions.NewOrder...	OP	40	☑
Process Orders	com.dom.benchmarking.swingbench.plsqltransactions.ProcessOr...	PO	5	☑
Browse Orders	com.dom.benchmarking.swingbench.plsqltransactions.BrowseAn...	BO	5	☑
Sales Rep Query	com.dom.benchmarking.swingbench.plsqltransactions.SalesReps...	SQ	2	☐
Warehouse Query	com.dom.benchmarking.swingbench.plsqltransactions.Warehouse...	WQ	2	☐
Warehouse Activity Query	com.dom.benchmarking.swingbench.plsqltransactions.Warehouse...	WA	2	☐

Figura 14 - Controle de transações Cenário OLTP.

De modo a entender a modelação de dados utilizada, na Figura 15 podemos visualizar o modelo de dados correspondente ao sistema OLTP.

[56] Cf. Capítulo III, secção 3.4, subsecção Software utilizados.
[57] https://docs.oracle.com/cd/B19306_01/server.102/b14220/schema.htm#i22627

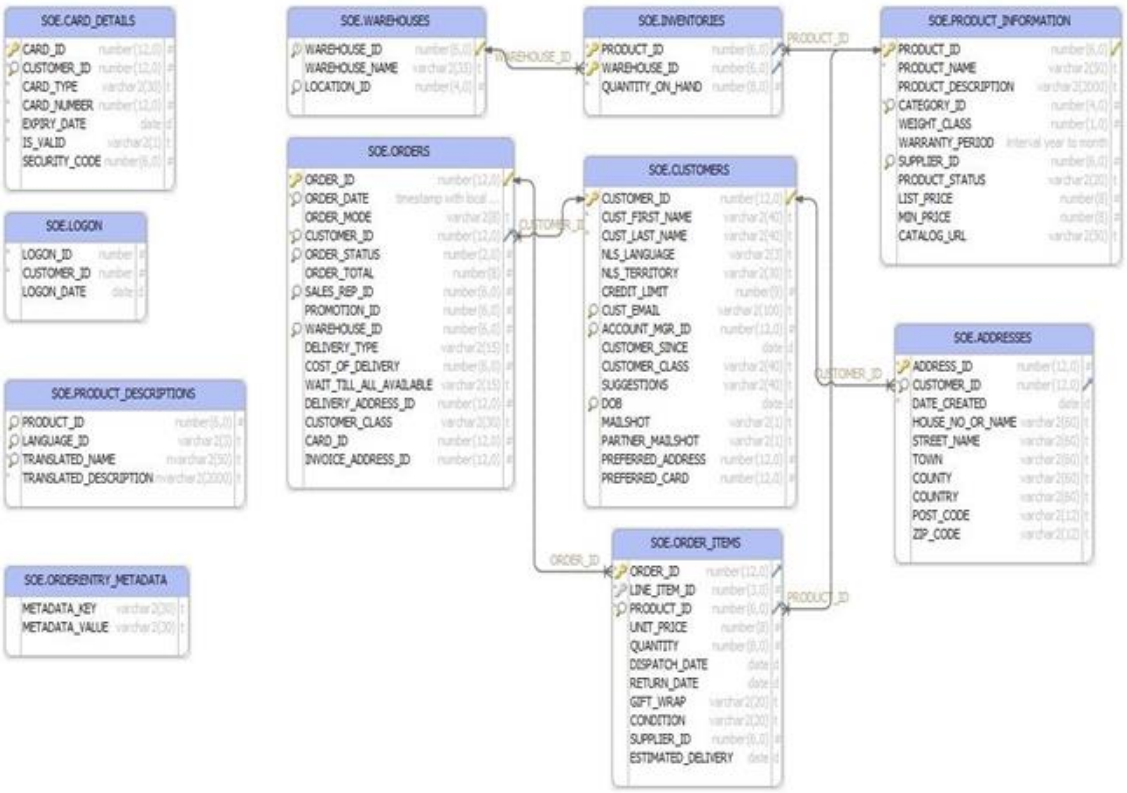


Figura 15 - *Schema* SOE - Modelo Relacional.

O modelo de dados convertido para o SGBD NoSQL referente ao sistema OLTP ficou com a modelação que pode ser consultada na Figura 16:

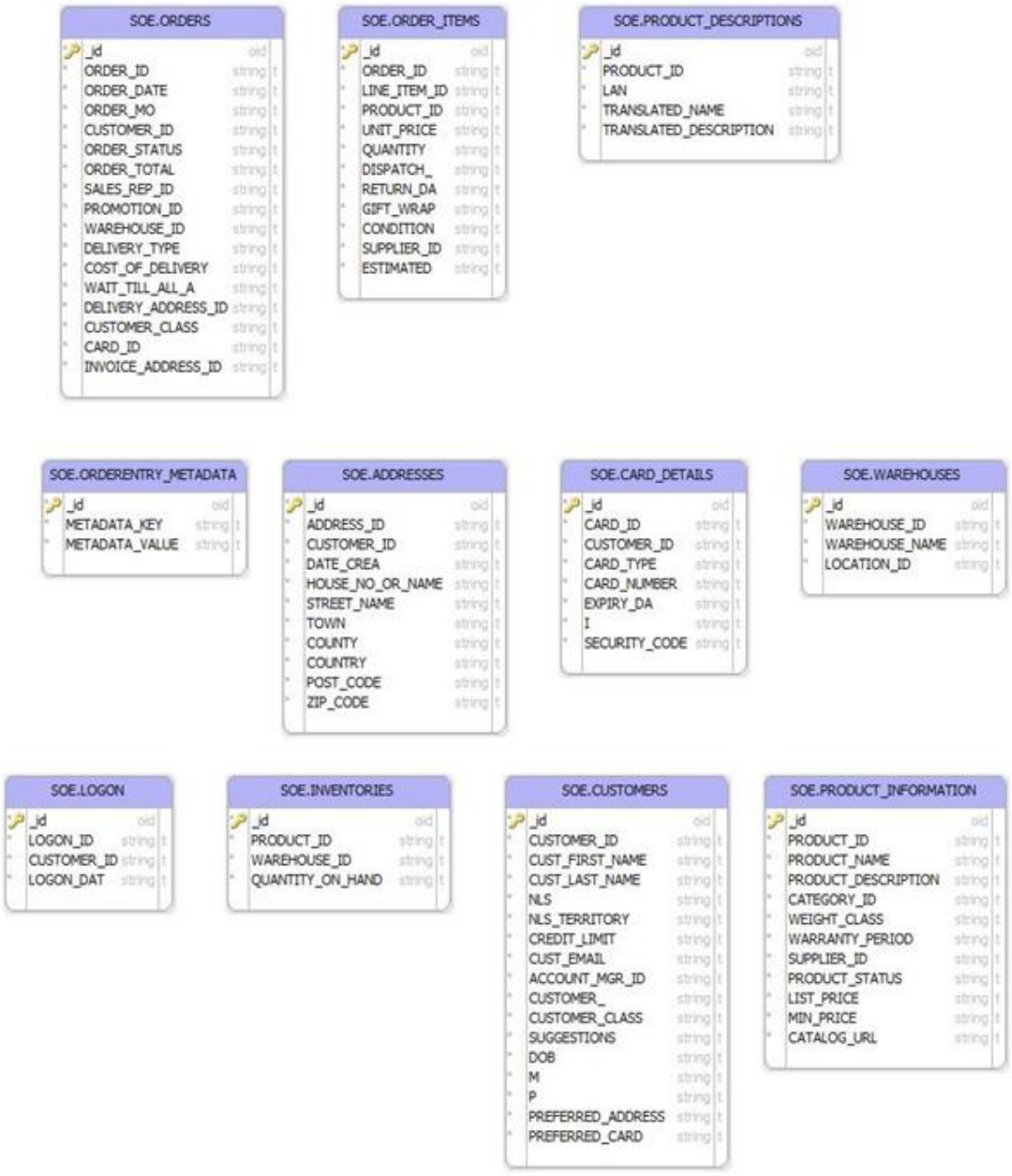


Figura 16 - *Schema* SOE – MongoDB.

Este é o modelo de dados final que estará presente na base de dados mongo; a modelação de dados será detalhada na secção sobre modelação de dados (mas para uma melhor visualização das diferenças do modelo de dados nos diferentes SGBD, apresentamo-lo aqui).

Cenário OLAP

Este cenário consiste numa base de dados utilizada para armazenamento de dados com carga do tipo *DSS ou OLAP*, onde são feitos todos os carregamentos dos diversos *front-ends* aplicacionais para posterior processamento dos dados. Estas bases de dados são processadas em lotes para resumo dos facturamentos diários, agregados semanais e/ou fechos mensais.

Para simulação desta carga, foi novamente utilizada a ferramenta *SwingBench* (detalhada na secção do *software),* sendo que para este tipo de carga foi utilizado o *schema* SH (*SalesHistory*), que caracteriza as transações que ocorrem no modelo DSS, DW, BI. Este emula um carregamento do tipo TPC-H e isto com base em pesquisas realizadas pela entidade TPC. (TPC, 1992c). Podemos consultar a carga que foi gerada pela ferramenta utilizada na Figura 17, sendo que o "*load ratio*" é a quantidade de transações envolvendo a tabela da coluna "*Id*". O valor da coluna representada é o valor que foi utilizado para dividir os ciclos entre todos os objetos ativos.

Transactions | Jobs

Id	Class Name	Short Name	Load Ratio	Activate ?
Sales Rollup by Month and Channel	com.dom.benchmarking.swingbench.dsstransactions.SalesRollup...	SRMC	65	☑
Sales Rollup by Week and Channel	com.dom.benchmarking.swingbench.dsstransactions.SalesRollup...	SRWC	75	☑
Sales Cube by Month and Channel	com.dom.benchmarking.swingbench.dsstransactions.SalesCube...	SCMC	60	☑
Sales Cube by Week and Channel	com.dom.benchmarking.swingbench.dsstransactions.SalesCube...	SCWC	50	☑
Product Sales Cube and Rollup b...	com.dom.benchmarking.swingbench.dsstransactions.ProductSal...	PSCR	25	☑
Product Sales Cube and Rollup b...	com.dom.benchmarking.swingbench.dsstransactions.ProductMo...	PMSCR	20	☑
Sales Moving Average	com.dom.benchmarking.swingbench.dsstransactions.SalesMovin...	SMA	95	☑
Period to Period Sales Comparison	com.dom.benchmarking.swingbench.dsstransactions.PeriodToP...	PPSC	15	☑
Week to Week Sales Comparison	com.dom.benchmarking.swingbench.dsstransactions.WeekToWe...	WWSC	15	☑
Top Sales by Quarter	com.dom.benchmarking.swingbench.dsstransactions.TopSalesWi...	TSQ	55	☑
Top Sales by Week	com.dom.benchmarking.swingbench.dsstransactions.TopSalesWi...	TSW	40	☑
Sales within Quarter by Country	com.dom.benchmarking.swingbench.dsstransactions.SalesByQua...	SQC	75	☑
Sales within Week by Country	com.dom.benchmarking.swingbench.dsstransactions.SalesByWe...	SWC	100	☑

Figura 17 - Controle de transações Cenário OLAP

De modo a entender a modelação de dados utilizada, na Figura 18 podemos visualizar o modelo de dados correspondente ao sistema OLAP.

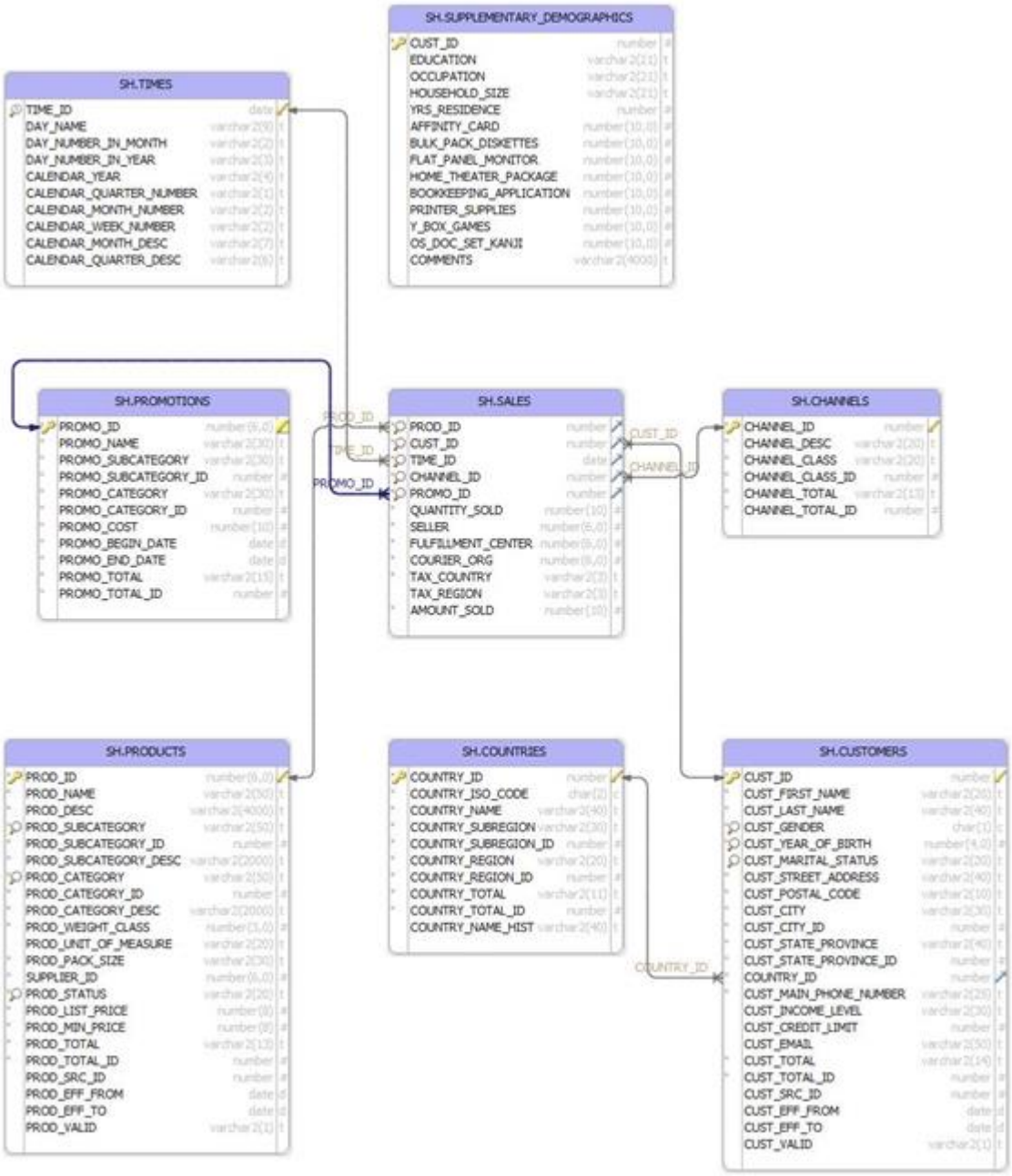


Figura 18 - Schema SOE - Modelo Relacional.

O modelo de dados convertido para o SGBD NoSQL referente ao sistema OLAP ficou com a modelação que pode ser consultada na Figura 19:

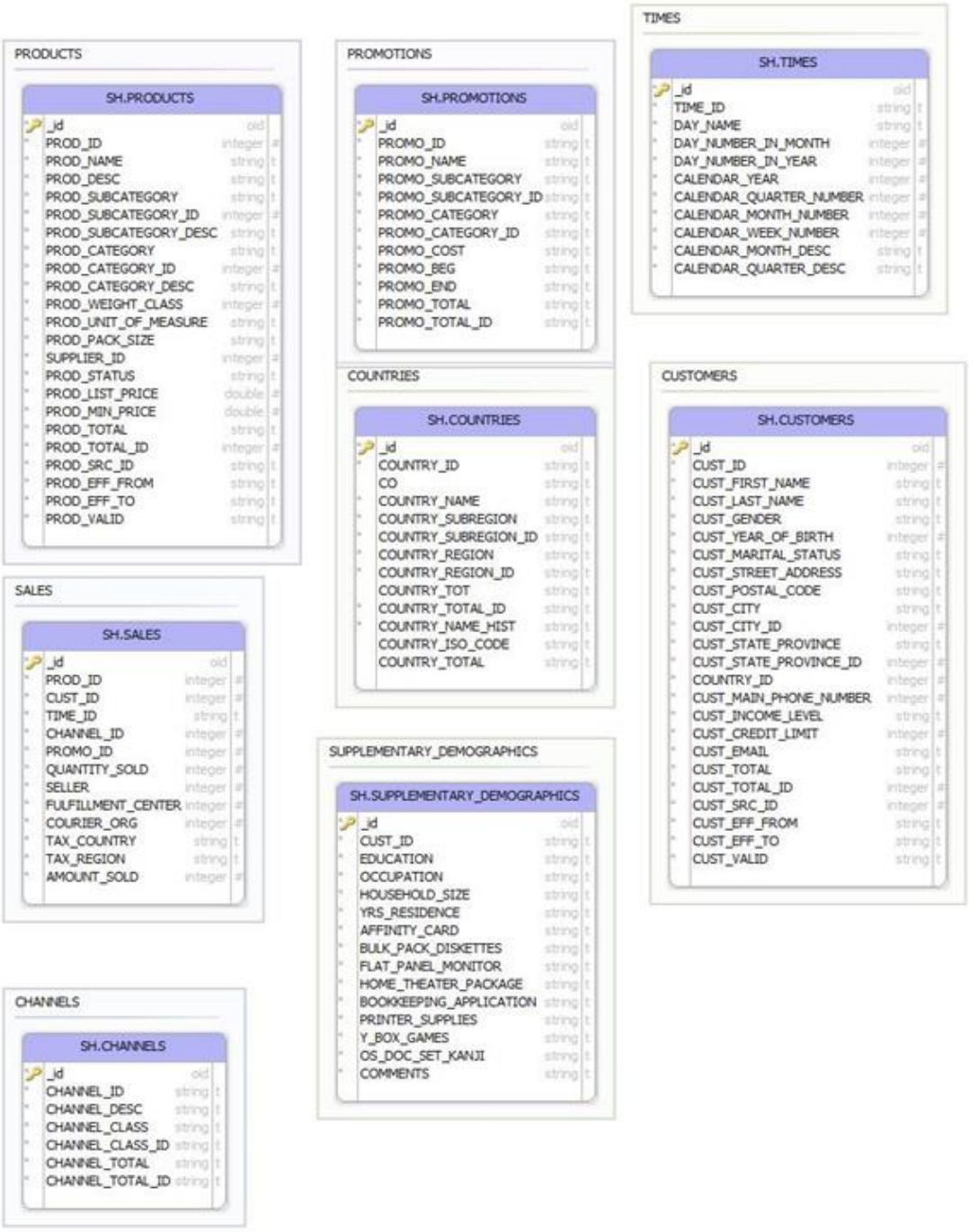


Figura 19 - *Schema* SH – MongoDB.

Este é o modelo de dados final que estará presente na base de dados mongo, o tópico referente a modelação de dados (cf. secção 3.6 Componentes da migração em modelação de dados), ficando aqui, no entanto para uma melhor visualização das diferenças para o modelo de dados nos diferentes SGBD.

Cenário HTAP

O cenário HTAP (*hybrid transactional/analytical processing*) é o cenário mais complexo, isto porque este é a junção de ambos os cenários anteriores. Esta é uma base de dados relacional suportando os sistemas OLTP e OLAP em simultâneo, fazendo com que a utilização de recursos seja a mais pesada dos cenários aqui propostos.

Para simulação desta carga, usar-se-á igualmente a ferramenta *SwingBench*, sendo inicializadas duas instâncias desta ferramenta: numa delas o cenário OLTP, e na outra o cenário OLAP. Ambas serão configuradas com os mesmos parâmetros dos cenários anteriores. Recorde-se o significado de HTAP: "Hybrid transaction/analytical processing (HTAP) is an emerging application architecture that "breaks the wall" between transaction processing and analytics. It enables more informed and "in business real time" decision making" (Gartner, 2016, p. 1).

Software utilizado

Os tipos de *software* utilizados durante os testes foram os seguintes:

- *Oracle Database 12c Enterprise Edition Release* 12.1.0.2.0 - 64bit *Production*[58].
    - O Oracle 12c é uma base de dados do modelo relacional, o "c" significa "cloud" para indicar que 12c é "cloud enabled".
- Mongodb-org-server-3.2.9-1.el5[59]
    - O MongoDB é uma base de dados NoSQL do tipo *document-oriented*, sendo disponibilizada como *open-source software* via licença AGPL v3.0.
- *Oracle GondenGate* 12c *for Big Data Version* 12.2.0.1.1[60]
    - Oracle *GoldenGate* é uma ferramenta de replicação heterogenia e multi-plataforma.
- *Swingbench* 2.5[61]

[58] http://www.oracle.com/technetwork/database/enterprise-edition/downloads/database12c-linux-download-2240591.html
[59] https://repo.mongodb.org/yum/redhat/5/mongodb-org/3.2/x86_64/RPMS
[60] https://blogs.oracle.com/dataintegration/oracle-goldengate-for-big-data-122011-update-is-available-now
[61] http://www.dominicgiles.com/downloads.html

    - *Swingbench* é um gerador de carga livre e *benchmarks* projetado para testar bases de dados *Oracle*.
- *VirtualBox* 5.1.4 *for Windows hosts*[62]
    - O *VirtualBox* é um *software* de virtualização, disponível sobre a licenças *open-source* sob os termos da GNU *General Public License* (GPL) *version* 2.
- OSWatcher Black Box 1.1.7-1(OSWBB) [63]
    - O *OSWatcher Black Box* (OSWbb) coleta e arquiva as métricas do sistema operativo e da rede, e pode ser utilizado para diagnosticar questões de desempenho. *OSWatcher* invoca uma série de utilitários UNIX/*Linux*, dependendo sempre da plataforma (Solaris, HP/UX, *Linux and* Tru64.
- *Oracle Linux Server release* 6.5[64]
    - *Oracle Linux* é uma distribuição *Linux*, assim como a distribuição CentOS ela é baseada no *Red Hat Enterprise Linux* (RHEL), *re-packaged*, mantida e distribuída pela *Oracle*, disponível sob os termos GNU *General Public License* (GPL).
- PuTTY Release 0.67[65]
    - PuTTY é uma implementação gratuita para os protocolos SSH e Telnet, funciona nas plataformas Microsoft *Windows* e *Unix*, juntamente com um emulador de terminal *xterm*.

3.5. Procedimentos seguidos

A tarefa de migração de uma base de dados é sempre uma tarefa crítica, pois como é de se esperar nos ambientes produtivos, sejam eles bases de dados académicas ou corporativas, os dados precisam estar sempre disponíveis para utilização.

[62] https://www.virtualbox.org/wiki/Download_Old_Builds_5_1
[63] https://docs.oracle.com/cd/E37670_01/E37355/html/ol_oswatcher_diag.html
[64] https://www.oracle.com/linux/products.html
[65] https://www.chiark.greenend.org.uk/~sgtatham/putty/

Quando se trata de uma migração entre modelos que seguem diferentes normalizações, tem-se ainda um maior desafio e uma necessidade acrescida de cautelar, ou mesmo planear e validar, diferentes possibilidades antes de se iniciar o processo de migração ou de testes de migração.

A migração, como todo o projeto, é dividida em fases – ou janelas temporais –, sendo que cada fase – ou janela – é constituída por um conjunto de processos com princípio, meio e fim. De seguida, apresentamos uma breve descrição sobre as diferentes fases/janelas que caracterizam um processo de migração.

- INFO - Pré-migração: ocorre antes do início da migração propriamente dito e no servidor de origem. Tem como função reunir informações sobre o estado da base de dados e dos dados a serem migrados.
- Fases 1 a 5: são fases temporárias, com período de execução determinado, onde existirão processos específicos e definidos pelo projeto, sendo as mesmas específicas a um servidor durante a existência da migração.
- INFO - Pós-migração: sucede após a migração ser finalizada. Nela é realizado um levantamento de informações e validações sobre os processos de replicação e toda a informação migrada, servindo como evidência sobre o estado final da migração a ser comparada com a informação retirada no processo de pré-migração.

Os processos de migração são separados entre as fases (ou janelas de execução), dependendo da técnica utilizada; para descrever cada um dos processos, segue-se a seguinte composição:

- Origem: base de dados relacional de onde são extraídos os dados para a migração.
- Destino: base de dados NOSQL onde são inseridos os dados durante a migração.
- Extração inicial (EI): processo que realiza a extração inicial dos dados já existentes na base de dados origem e envia para o processo de replicação inicial. Este é um processo especial que está somente em execução durante o carregamento inicial.
- Files: são os dados extraídos na origem e já no formato esperado pelo processo de migração a serem consumidos ou inseridos na base de dados destino.
- Coletor (C1): processo de comunicação do *software* de replicação entre os servidores de origem e destino, existindo no servidor de destino, sendo controlado de forma automática pelo *software* de migração.
- Extração *online* (EO): processo que ao ser iniciado extrai todas as DDL e DML realizadas na base de dados origem desde o seu início, com registo do CSN inicial do

processo. Este é enviado para o destino que serve como referencia para continuidade do processo migração.

- Replicação inicial (RI): este processo lê e aplica toda a informação que chega ao destino e que está previamente configurada para ser replicada. Este processo existe somente numa fase e a sua função é aplicar os ficheiros extraídos pelo processo EI.
- Replicação *online* (RO): o processo de RO aplica todos os ficheiros extraídos pelo processo EO, ficando este *online* durante a sincronização dos dados. Este processo aplica somente as transações após o CSN ter registado no processo de EO, para que não exista aplicação de informação duplicada, ou fora da ordem, das mesmas que aconteceram na origem.
- Exportação para CSV (ECSV): este processo é utilizado juntamente com a técnica de migração *offline* manual, responsável pela geração e envio dos ficheiros no formato CSV do servidor da origem para o servidor de destino.
- Importação do CSV (ICSV): processo responsável pela importação dos ficheiros gerados através do processo ECSV no servidor de destino.

Na Tabela 12 podemos visualizar todos os processos utilizados e em que servidor irão existir.

Tabela 12 - Componentes e localização

ONDE	PROCESSO
ORIGEM	APP
ORIGEM	EI
ORIGEM	EO
DESTINO	RI
DESTINO	RO
DESTINO	C1
ORIGEM	ECSV
DESTINO	ICSV
DESTINO	INFO
ORIGEM	INFO

Cada migração terá o seu procedimento a ser desenvolvido e seguido, que será detalhado juntamente com o método, ficando com toda a informação centralizada referente a cada método, na sua própria secção.

3.6. Componentes da migração

Aqui descrevemos os requisitos prévios que possibilitam um processo de migração[66] de dados, e as métricas usadas para a avaliar, detalhando a configuração que a permite efetuar.

Requisitos para migração

As migrações de bases de dados são por norma baseadas em processos disponibilizados pelo próprio fornecedor, ou caso contrário, são processo manuais desenvolvidos por técnicos internos envolvidos com o processo de migração. Quando essa documentação ou mesmo a técnica proposta não é suficiente, quer seja por limitação quer por incompatibilidade, são necessárias validações de outras técnicas, ou ferramentas para esse efeito. Essa premissa foi o nosso ponto de partida para os testes de migração e planeamento.

Após o levantamento do estado de arte, foram reunidos vários pontos a seguir durante uma migração de base de dados relacional para NoSQL, com base em outros estudos, e também em processos que os fornecedores ou mantenedores do *software* indicam como caminho a seguir – podendo consultar a referência na listagem abaixo. Foram reunidos os seguintes pontos para serem validados como requisitos para uma migração, vindo alguns de pesquisas efetuadas, e outros adicionados pelo autor:

- Planeamento (Antaño et al., 2014; Oliveira & Marcelino, 2012)
- Quantidade de registros / Situação inicial (Antaño et al., 2014)
- Mapear os tipos de dados (data type) / Equivalência entre metadados (Davenport et al., 2014; Gomes, 2011; Pereira, 2014)
- Restrições e trigger (Antaño et al., 2014)
- Codificação de caracteres (Antaño et al., 2014; Sancho & Oliveira, 2015; Santos, 2015)
- Ensaios
- Implementação
- Monitorização / Validação parcial e Final
- Staging area/ Sem staging area

[66] Assim como existem requisites de *software*, existem os requisitos para uma migração, ou seja, precisa estar garantida uma determinada configuração ou estado antes da migração iniciar ou acontecer.

- Falhas durante a migração
- Modelação de dados (Gomes, 2011)

Planeamento

A etapa do planeamento é crucial para o bom início do projeto, bem como para o funcionamento do mesmo. Aqui são planeadas todas as atividades a serem executadas e as métricas que deverão ser coletadas antes, durante e depois da migração. Esta etapa envolve as equipas funcionais, as de desenvolvimento e infraestrutura para uma completa validação das necessidades e constituição do plano de migração a ser seguido, bem como a validação antes da execução final.

Sugere-se seguir uma metodologia de gestão de projetos, podendo-se citar o *Project Management Institute* (PMI)[67] como muito divulgado, tendo sido aplicado durante a realização desta tese, adequando os tópicos à realidade do presente projeto e à abordagem metodológica.

Quantidade de registos / Situação inicial

Uma métrica simples e básica é a quantidade de registos a serem migrados, esta contagem deve acontecer antes da migração se possível, caso a migração seja online, sugere-se a utilização de uma técnica de validação temporal.

Para as bases de dados Oracle, podemos aplicar a tecnologia *Oracle Flashback Technology*[68], onde várias opções de recuperação temporal são oferecidas. Para outros modelos de bases de dados, deve-se sempre validar a documentação oficial do fornecedor de forma a saber qual a técnica que deve ser utilizada para se obter essa contagem inicial.

Mapear os tipos de dados (*data type*) / Equivalência entre *metadados*

Os *metadados* devem ser analisados na origem e convertidos para o seu correspondente no destino. No caso do mesmo não ter um equivalente existem algumas opções, nomeadamente:

a) ser convertido para um tipo de dado que seja aceite pela aplicação no destino antes da migração;

[67] http://www.pmi.org/about/learn-about-pmi
[68] *Oracle Flashback Technology is a group of Oracle Database features that let you view past states of database objects or to return database objects to a previous state without using point-in-time media recovery*. https://docs.oracle.com/database/121/ADFNS/adfns_flashback.htm#ADFNS01001

b) ser convertido durante a migração;
c) se assim for entendido pela camada aplicacional, pode ser excluído.

As tarefas de ajustes dos *metadados* podem ocorrer antes ou durante a migração, não sendo possível acontecer depois. Esta situação deve-se ao facto de que se é aceite pelo negócio, ou seja, pela aplicação. É possível realizar a conversão destes *metadados* antes da migração para algum tipo que seja aceite no destino, e assim migrar para que todo o ambiente compatível.

Existe a possibilidade de se fazer a conversão durante a migração, para tal utiliza-se a ferramenta de migração, caso a mesma assim o possibilite. No caso de existirem caracteres não compatíveis, além de erros inesperados durante a migração ainda é possível que alguns ou todos os dados sejam perdidos, podendo assim comprometer todo o processo de migração.

Restrições e *trigger*

Os *triggers* nos atuais SGBDs têm a função de "disparar" uma ação após a ocorrência de uma outra, contra uma tabela, ou com base num evento na base de dados, sendo este objeto previamente criado. Um dos exemplos que temos sobre *triggers* é a possibilidade de após à alteração de uma determinada informação numa tabela, se poder realizar o cadastro desta alteração numa outra tabela. Esta operação é utilizada por vezes em sistemas onde é necessária uma auditoria ou histórico de alterações realizadas.

Existem técnicas já disponibilizadas pelo próprio fornecedor do SGBD que capturam toda a informação. No entanto, o método manual é por vezes preferível em sistemas pequenos, devido à flexibilidade de poder ser alterado pelo próprio desenvolvedor do sistema. Tal permite que não seja necessário conhecer a técnica do fornecedor específico, devendo apenas ser desativada durante a migração, para que no final não se tenha uma auditoria da nossa própria manutenção. Esta situação apenas se verifica caso o método de migração execute alguma alteração na origem, de forma a controlar os dados já enviados.

É importante salientar que a etapa da validação ocorre somente na origem, não está aqui a ser discutida a utilização de *triggers* na base de dados no destino.

No SGBD mongo não existe *trigger*, como existe no modelo relacional. Esta validação pode ser executada de outra forma. No entanto, apenas para relato nessa secção, podendo ser validada diretamente na documentação oficial do produto em docs.mongodb.com/manual.

Codificação de caracteres

Todo o armazenamento da informação é feito num formato específico do SGBD e apresentado numa forma visual, tal como é dado a conhecer ao utilizador. Contudo, é importante ter em consideração que os utilizadores podem ser provenientes de várias partes do mundo e consequentemente ter linguagens distintas. Portanto, a linguagem ou a codificação dos dados a apresentar deve ter em conta essa informação. Podemos ter os dados armazenados na origem em um formato, e no destino o formato esperado ser outro. É importante dar atenção a essa variável, de forma a ser analisada, ajustada e convertida, caso seja necessário.

Ensaios

Os ensaios são testes que devem realizar-se sempre, antes de uma migração com os dados reais de produção, simulando com a maior representatividade possível o ambiente real. Em alguns casos, onde existe uma base de dados muito grande e não existe disponibilidade de infraestrutura para que se consiga replicar toda a sua estrutura, sugere-se validar sempre através de uma amostragem de dados, o mais próxima possível da real. É ainda necessário agendar testes em produção antes da migração, onde todo o ambiente seja testado e seja possível retroceder ao ambiente original, após os testes. Esta técnica é conhecida como *snapshot database* e pode esta ser utilizada em conjunto com tecnologias de *storage* ou virtualização[69].

Implementação

Para a implementação ou execução, todo o ambiente deve ser controlado, bem como todas as execuções e interações do mesmo, assim como entre utilizadores, caso necessário. Todas as atividades devem ser previamente cadastradas e as atividades seguidas de forma singular conforme o plano. O mais adequado é ter um gestor de projetos como ponto focal de modo a orquestrar todas as atividades executadas e a respetiva ordem.

Devemos ainda controlar quando devem ser abandonadas as atividades e iniciado o plano de reposição do ambiente – no caso de ser necessário cancelar a migração durante a execução.

[69]http://en.community.dell.com/techcenter/enterprise-solutions/w/oracle_solutions/4621.oracle-database-backup-and-recovery-using-dell-storage-snapshot-technologies

Monitorização / Validação parcial e Final

Durante todo o ciclo de vida da migração é indicado validar o ambiente original e de destino, a fim de confirmar que todos os dados foram migrados e que podemos dar por finalizada a migração. Os valores na origem e destino devem ser qualificados de acordo com a linha atual da migração, seja essa, início, meio ou fim.

Staging area/ Sem staging area

A utilização de uma *stage area* é comum em sistema de *extract*, *transform* e *load* (ETL), podendo esta ser aplicada para a migração de uma base de dados. Esta técnica consistente em enviar dados que estão a ser extraídos na origem para uma localização provisória, antes de serem aplicados no destino. Se não for possível ter uma área disponível para utilização desta técnica, é possível realizar a mesma enviando os dados diretamente da base de dados origem para a de destino, isto sem recorrer a uma área intermédia para alocação dos dados.

Falhas durante a migração

O comportamento do método utilizado será analisado no caso de existirem falhas durante a migração. Havendo falhas de rede ou problemas em qualquer um dos servidores envolvidos durante a migração, essas falhas serão analisadas e registadas para análise dos resultados e possíveis impactos.

Modelação de dados

Durantes os testes serão utilizados dois modelos de dados distintos, sendo um deles para gerar um tipo de carga classificada como transacional (*TPC-C*) e o outro para um modelo tipicamente de BI(*TPC-H*), tal como visto anteriormente[70].

A modelação de dados relacional segue um conjunto de regras específicos para esse modelo, sendo a mais básica delas o relacionamento entre as entidades. Uma vez que este tema é extremamente complexo e extenso, não será visto na sua totalidade, apenas será feita uma leve introdução para que se possa entender esta temática.

[70] Cf capítulo III, secção 3.4. Descrição da arquitetura

Para o sistema NoSQL a modelação de dados vai depender do tipo de base de dados utilizada. No caso desta tese, o SGBD escolhido é o MongoDB que é do tipo "*document*" e a sua modelação é a nível dos documentos que são armazenados (MongoDB, 2015).

O SGBD MongoDB possui o sistema *Schemaless* de modelação ou como citada na própria documentação, "*flexibe schema*". Aqui, será utilizada a modelação "*Model One-to-One Relationships with Embedded Documents*". Esta é escolhida pelas suas características de facilidade de implementação.

Será realizada a migração duma tabela para uma coleção sem necessidade de transformação de dados, isto porque a quantidade de testes a realizar e o tempo necessário não são compatíveis com o prazo de realização desta tese.

No trabalho "Um Estudo sobre Modelação Lógica para Bancos de Dados NoSQL", Lima & Mello (2015) fazem uma abordagem aprofundada a este tema, descrevendo com detalhe cada uma das estruturas e modelações que comummente são encontradas nos modelos NoSQL.

Métodos de migração

Uma vez satisfeitos os requisitos[71], é então possível proceder a uma migração dos dados seguindo um determinado método. Podem considerar-se dois métodos de migração distintos.

Na Figura 20 podemos ver os possíveis caminhos para uma migração:

- OFFLINE
- ONLINE

[71] Cf. capítulo III, secção 3.6. Requisitos de migração.

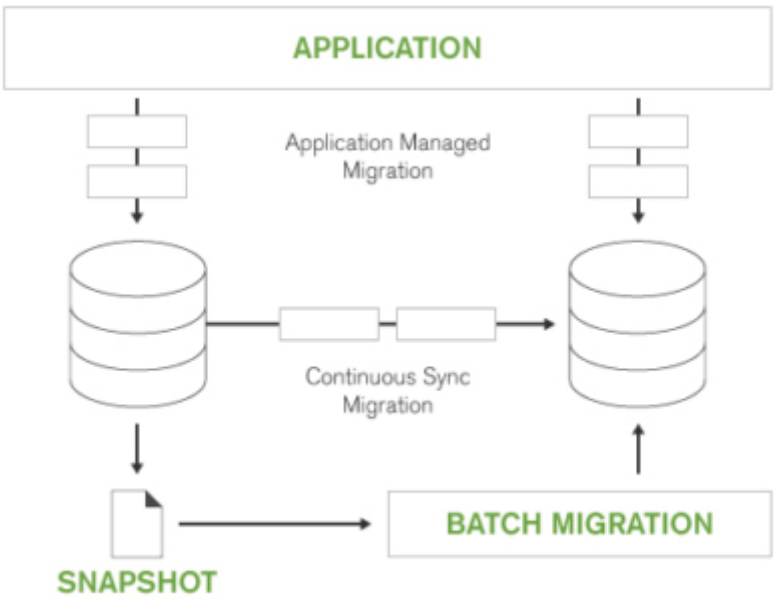


Figura 20 - Opções para migração de dados NoSQL

Fonte: (MongoDB, 2015)

Portanto, esses métodos podem acontecer no modo *online*, ou seja, com o sistema aplicacional a funcionar e com sincronismo contínuo, durante o tempo necessário do projeto. Já no sistema *offline*, teremos o sistema aplicacional indisponível e toda a aplicação será migrada e disponibilizada, aqui não havendo sincronismo posterior ao carregamento. Iremos adotar os métodos *online* e *offline* (cf. Figura 20) para replicar um processo de migração, adequando-os ao estudo de caso realizado.

Os métodos que adotaremos no decorrer dos testes a realizar neste estudo, são:

- Migração *online* com sincronização contínua;
- Migração *offline* através de ferramenta;
- Migração *offline* manual através de *scripts*.

Os três métodos indicados serão testados, e os resultados comparados, entre os ambientes, com o intuito de averiguar a questão de investigação e os objetivos propostos.

Iremos agora analisar em detalhe cada um destes métodos, no que respeita à sua execução, rotinas e aos impactos de cada cenário[72].

Migração *online* com sincronização contínua

Esta migração consiste em ter a aplicação sempre a utilizar a base de dados de forma ininterrupta. No final, a aplicação pode ser redirecionada para a nova infraestrutura com uma perda de serviço mínima ou praticamente nula – este fluxo é representado na Figura 21.

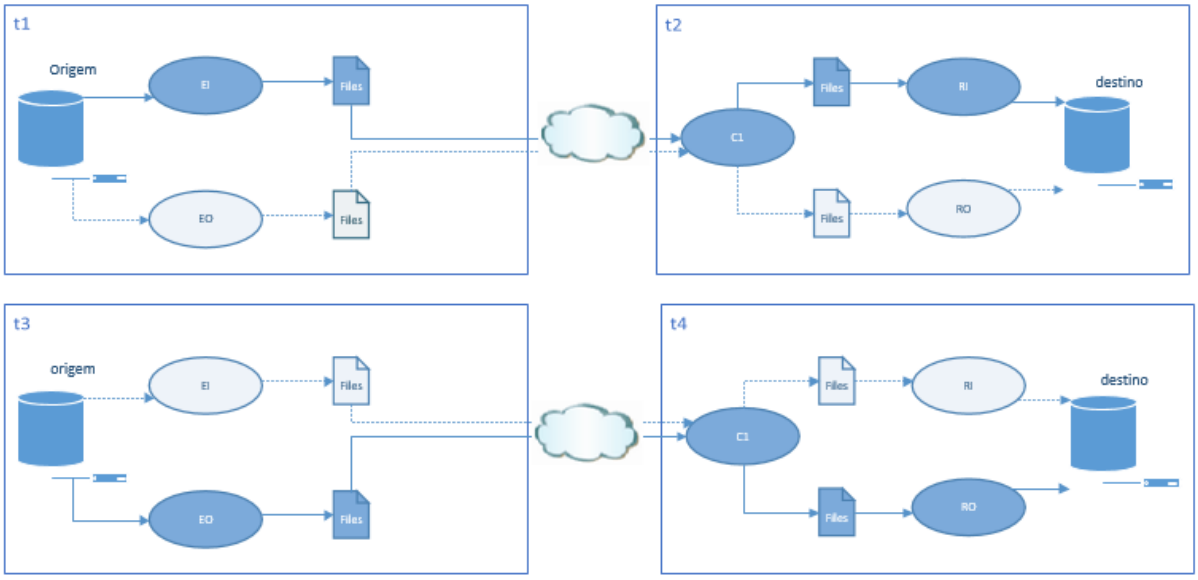


Figura 21 - Arquitetura e fluxo da migração *online*.

Neste modelo de migração a aplicação está sempre a utilizar a base de dados. No entanto, caso necessário, pode-se realizar uma paragem para as validações finais e o sincronismo total. É importante referir que esta replicação não é em tempo real, ou seja, os dados não serão criados e posteriormente replicados. Existindo um atraso entre a base de dados de origem e a base de dados de destino; esse atraso é configurável, pode ser muito pequeno ou quase inexistente, no entanto, o mesmo deve ser tido em consideração. A Tabela 13 mostra onde cada processo é executado durante a migração, de acordo com o fluxo referido.

[72] Cf. capítulo III, secção 3.4 na subsecção Cenário OLTP, OLAP e HTAP.

Tabela 13 - Fluxo migração *online*

t_0	t_1	t_2	t_3	t_4	t_5
Pré-migração	**Fase 1**	**Fase 2**	**Fase 3**	**Fase 4**	**Pós-migração**
APP	APP EI	APP RI	APP EO	APP RO	APP
INFO	INFO	INFO	INFO	INFO	INFO

Neste fluxo, podemos validar quais os processos que temos em cada fase. A coleta de métricas será executada em todas as fases, com o intuito de se ter um relatório final com as medidas verificadas, em cada uma das fases. Para uma melhor explicação dos passos seguidos, podemos criar uma linha temporal e ligar as etapas (cf. Figura 22).

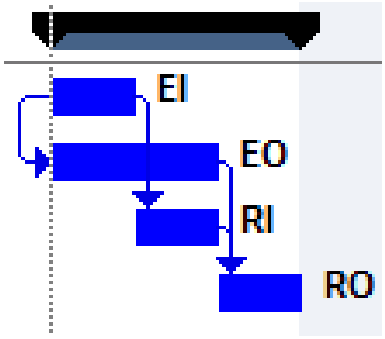


Figura 22 - Fluxograma e dependências no modelo de migração *online*.

O método de migração *online* (cf. Figura 22) é descrito da seguinte forma:

1. Os processos EI, EO, RI e RO são configurados;
2. O processo EI é iniciado e irá extrair todos os dados existentes, até ao momento em que foi iniciado, para ficheiros e finaliza automaticamente;
3. Os ficheiros serão enviados para o servidor de destino automaticamente pelo processo EI e rececionados pelo processo C1;
4. O processo EO é iniciado e irá extrair todos os anteriores ao início do processo EI e permanecer em execução até que seja manualmente finalizado;
5. O processo RI irá aplicar todos os ficheiros extraídos pelo processo EI e finalizar automaticamente;
6. O processo RO irá aplicar todos os ficheiros extraídos pelo processo RI e permanecer em execução até que seja manualmente finalizado;

7. Após estarem somente os processos EO e RO em execução, já estaremos a replicar de forma *online*. Estando aqui apenas com o delta entre os processos de extração EI e EO a serem aplicados, juntamente com novas transações que estão a ser inseridas;
8. Nesse momento a única opção é aguardar até à decisão finalizar a migração. Para um perfeito sincronismo será necessário parar a aplicação para que nenhuma informação seja inserida na origem, e todas sejam replicadas para o destino, podendo agora executar a fase de pós-migração.

Migração *offline* através de ferramenta

Este método de migração é baseado no modelo *online*. Aqui, a diferença é a aplicação não estar em execução e a migração ser toda executada somente num carregamento, caracterizando desta forma o modelo *offline* de migração. Este processo ocorre através do *software* de migração[73], sendo controlado de forma manual em cada passo da execução.

Os processos envolvidos podem ser observados na Figura 23.

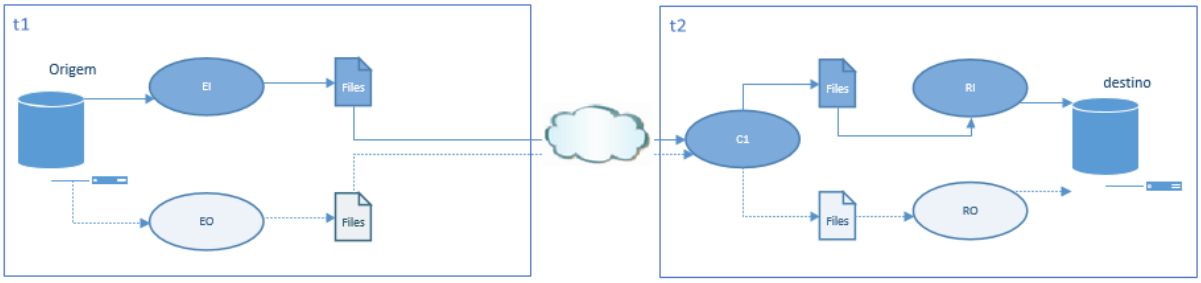


Figura 23 - Arquitetura migração *offline* automática.

Podemos verificar onde cada processo é executado, bem como o fluxo que é seguido neste método (cf. Tabela 14) e, ainda, quais os processos que vão ser avaliados através das métricas, em cada fase, e servidor.

Tabela 14 - Fluxo migração *offline* através de ferramenta

[73] Cf. capítulo III, secção 3.4 na subsecção *software* utilizado.

t_0	t_1	t_2	t_3
Pré-migração	**Fase 1**	**Fase 2**	**Pós-migração**
APP			APP
	EI	RI	
INFO	INFO	INFO	INFO

Cada uma das fases é executada somente num servidor. Aqui as coletas são executadas em um servidor específico na fase correspondente, conforme podemos ver no fluxo na Figura 24.

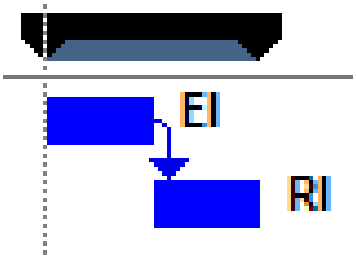


Figura 24 - Fluxograma e dependências no modelo de migração *offline* automática.

Este fluxograma é descrito da seguinte forma:

1. Os processos EI e RI são configurados;
2. O processo EI é iniciado e extrai todos os dados existentes até o momento em que foi iniciado para ficheiros e finaliza automaticamente;
3. Os ficheiros são enviados para o servidor destino automaticamente pelo processo EI e rececionados pelo processo C1;
4. O processo RI consistente em aplicar todos os ficheiros extraídos pelo processo EI e finaliza automaticamente;
5. Nesse momento a replicação esta finalizada e todos os dados já estão no servidor destino, devendo então ser iniciada a fase de pós-migração.

Migração *offline* manual através de scripts

Este processo é todo manual e requer interação do técnico durante toda a migração. O técnico deve analisar como os dados serão extraídos, em que formato serão armazenados e enviados para o servidor de destino, bem como a forma de carregar a informação.

Podemos visualizar na Figura 25 os processos envolvidos neste tipo de migração.

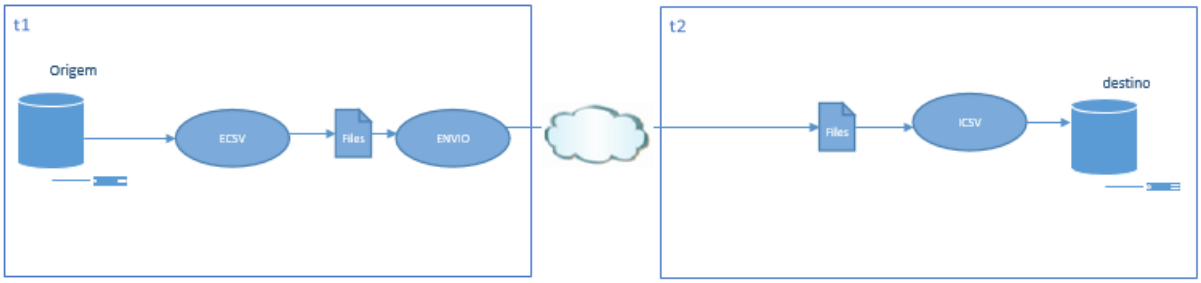


Figura 25 - Arquitetura migração *offline* manual.

Na Tabela 15 podemos validar os processos existentes nessa migração e onde é executado cada processo durante a migração, bem como as fases de pré e pós-migração.

Tabela 15 - Fluxo de migração *offline* manual através de scripts

t_0	t_1	t_2	t_3
Pré-migração	**Fase 1**	**Fase 2**	**Pós-migração**
APP			APP
	ECSV	ICSV	
INFO	INFO	INFO	INFO

Cada uma das fases é executada somente num servidor. As coletas são executadas por servidor na fase correspondente (cf. Figura 26).

Figura 26 - Fluxograma e dependências no modelo de migração *offline* automática.

Na Figura 26 é apresentado o fluxograma correspondente ao modelo de migração *offline* manual através de *scripts* que podem ser descritos da seguinte forma:

1. Os processos ECSV e ICSV são configurados;
2. O processo ECSV é iniciado e extrai todos os dados existentes até o momento em que foi iniciado para ficheiros, enviados no final e finaliza automaticamente;
3. O processo ICSV irá aplicar todos os ficheiros extraídos pelo processo ECSV e finaliza automaticamente;
4. Neste momento a replicação está finalizada e todos os dados já estão no servidor de destino, devendo agora dar-se início à fase de pós-migração.

Métricas para migração

Nesta secção são abordadas as métricas – valores ou características que vamos medir no decorrer de cada passo no processo de migração, ou cenário executado – que serão coletadas e analisadas. Com estas será possível realizar uma análise comportamental do ambiente migrado, desde o sistema operativo até ao impacto aplicacional, durante e proveniente de cada migração – e poderemos comparar os vários casos estudados.

As métricas aqui sugeridas são os consumos de recursos dos sistemas operativos, juntamente com métricas genéricas para validação da migração. Estas serão de seguida enumeradas e analisadas graficamente durante cada caso executado.

Durante o levantamento do estado da arte, foi encontrado um conjunto de métricas comuns a vários dos autores citados e que estão representadas na Tabela 16.

Tabela 16 – Principais referências para métricas

Métrica	Principais referências
Baseline	(Yaqub, 2012)
CPU	(Alexandre, 2009; Yaqub, 2012)
Disco(I/O)	(Alexandre, 2009; Yaqub, 2012)
Rede	(Alexandre, 2009)
Memória	(Alexandre, 2009; Yaqub, 2012)
Latência	adicionada pelo investigador
Intervenção manual	adicionada pelo investigador
Indisponibilidade	adicionada pelo investigador

Procedemos agora à definição de cada uma das métricas.

Baseline

Esta métrica é uma das mais importantes, dado que irá revelar todos os desvios existentes na migração. A mesma quantidade de linhas presentes na origem deve estar presente no destino após a migração. Esta métrica possui dois comportamentos distintos: para os métodos *offline*, deverá apresentar o mesmo valor de linhas migradas; já para os sistemas online, a mesma terá um acompanhamento inicial e no final da migração.

Teremos então valores diferenciados, o que se deve ao facto de possuir tantos valores superiores como inferiores, dependendo do comportamento aplicacional – caso sejam adicionadas ou excluídas linhas após a coleta da métrica e início da migração.

CPU

Coleta o tempo gasto da CPU em processamento de sistema e de utilizador. Como o nome sugere, são capturados e analisados no sistema operativo os consumos de cada um desses componentes, de forma individual e correlacionando numa linha temporal, de acordo com cada fase da migração. Podemos assim detetar qualquer comportamento anormal que afete o processo em curso. Mostra o percentual de utilização de CPU executando tarefas de utilizador, nesse caso aplicações em geral, sendo que será realizada a análise das quatro métricas diretamente ligadas a CPU[74]:

- cpu %user
    - Percentual de CPU utilizado por processos que não sejam originais do sistema operativo, tal como a nossa aplicação.
- cpu %system
    - Total do percentual de CPU utilizado pelo sistema operativo, o core do sistema.
- cpu %idle
    - Total de CPU não utilizado.
- cpu %iowait
    - Total de CPU utilizado por operações de I/O.

Disco(I/O)

Neste caso, a coleta de dados é feita nos discos locais. Quando se necessita de uma análise de discos de redes, indica-se a medida diretamente no sistema de *storage*. Na Figura 27 podemos ver a interface do sistema FreeNAS, um sistema *free* e *open source*:

[74] As definições abaixo são retiradas do manual incluído juntamente com a distribuição Linux utilizada, pode ser consultado através do terminal com o comando: ex.: *man mpstat*.

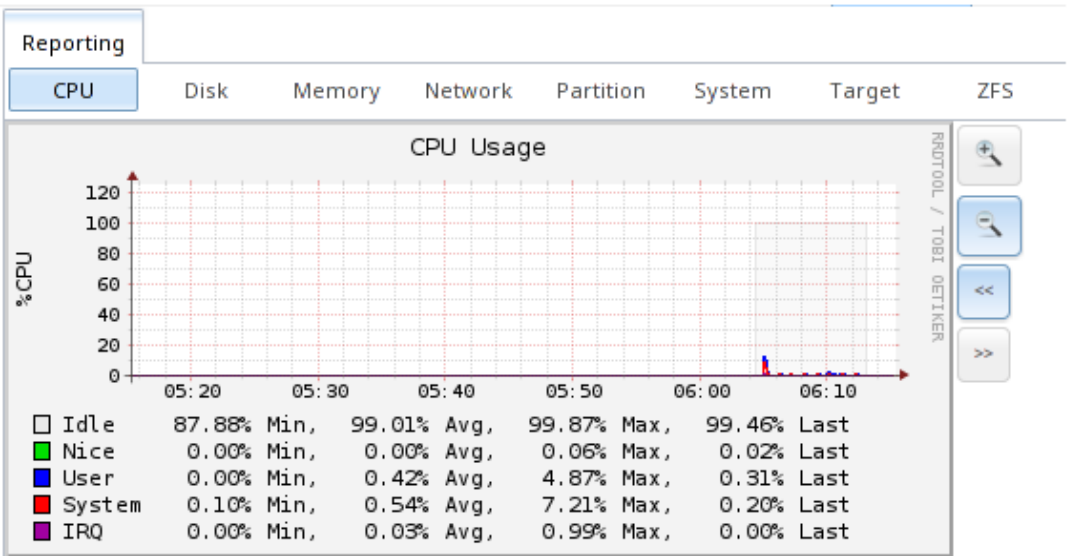


Figura 27 - Interface FreeNAS.

Fonte: (FreeNas, 2017)

As métricas específicas que foram coletadas e analisadas para o disco são[75]:

- rkB/s
    - Este representa o número de kilobytes lidos no disco específico por segundo.
- wkB/S
    - Este representa o número de kilobytes escritos no disco específico por segundo.

Rede

É indicada pela monitorização do envio e receção de pacotes de redes. Permite observar também os pacotes com erros e fragmentação. Esta situação acontece quando um bloco maior do que o pacote de rede é enviado, tendo consequências na performance aquando da sua ocorrência. Serão coletados apenas dois eventos através desta métrica[76]:

- Tcp: segments received
    - Segmentos de rede recebidos utilizando o protocolo TCP
- Tcp: segments send out

[75] As definições abaixo são retiradas do manual incluído juntamente com a distribuição Linux utilizada, pode ser consultado através do terminal com o comando: ex.:*man iostat*.

[76] As definições abaixo são retiradas do manual incluído juntamente com a distribuição Linux utilizada, pode ser consultado através do terminal com o comando: ex.:*man netstat*.

- Segmentos de rede enviados utilizando o protocolo TCP

Este é um tema que pode ser muito vasto, sendo que mais informações podem ser consultadas no livro de Tanenbaum (2003), intitulado "Redes de Computadores".

Memória

É responsável pela coleta da utilização da memória RAM disponível no servidor. Integra as seguintes métricas[77]:

- MemFree
  - Memória livre não utilizada no exato momento da coleta.
- Buffers
  - Com metadados do *filesystem* e não conteúdo de dados propriamente dito.
- Cached
  - Memória ocupada por informação dos *filesystem* ou seus ocupantes, a informação propriamente dita.

Esta métrica é bastante complexa, existindo várias aplicações de sua utilização. Na nossa implementação focamo-nos nos pontos acima, no entanto, vários outros podem ser utilizados (uma boa leitura desta pode ser realizada no manual de documentação fornecido para RedHat[78]). "Linux always tries to use RAM to speed up disk operations by using available memory for buffers (file system metadata) and cache (pages with actual contents of files or block devices)" (RedHat, n.d.).

Latência

A latência da migração corresponde ao levantamento das informações referentes ao atraso ou ao tempo desde que a linha (dado/informação) é inserida na base de dados origem até à sua disponibilização na base de dados destino, ficando esta com a caracterização do tempo até à disponibilização dos dados na sua estrutura final (base de dados destino). Esta métrica é válida para os sistemas *online*.

[77] As definições abaixo são retiradas do manual incluído juntamente com a distribuição Linux utilizada, pode ser consultado através do terminal com o comando: ex.: *man vmstat*.
[78] A Red Hat é uma empresa que disponibiliza soluções baseadas em sistema operativo sobre a licença GNU/Linux, pode ser consultado em redhat.com.

Para os sistemas *offline* ela é um indicativo do tempo total da migração dos dados. Possibilita-nos validar o tempo de aplicação da carga inicial, bem como a quantidade de registos na mesma, sendo aplicável somente ao modelo *online*, devido à sua natureza incremental, não sendo aplicado para os métodos *offline*, pois os mesmos têm apenas um carregamento total correspondente ao tamanho total da base de dados.

Intervenção manual

É a métrica que reflete o rastreio de toda a intervenção humana necessária durante a migração, associada à facilidade e simplicidade de utilização da técnica de migração. É uma métrica simples, mas importante, devido à visão direta que nos dá sobre a aplicação do conjunto "técnica e ferramenta". A intervenção manual será computada conforme o número de iterações executadas durante a migração, de forma a que cada iteração acrescente um valor ao contador.

Indisponibilidade

A métrica de indisponibilidade visa medir o tempo em que o sistema (centrado na base de dados) fica indisponível durante a execução da migração. Para os métodos de migração offline, esta métrica revela o tempo total da migração. Para o método *online*, a mesma irá traduzir o tempo necessário para sincronizar a latência (métrica já citada), com todos os dados inseridos no servidor origem.

Capitulo IV – Apresentação e Discussão dos Resultados

4.1. Introdução à Discussão

Será efetuada uma análise dos dados obtidos através dos testes realizados, no decorrer da migração de uma base de dados relacional (origem) para uma base de dados NoSQL (destino), sendo as informações extraídas da base de dados origem e aplicadas no destino, realizando durante a migração a coleta das métricas. Serão apresentadas as informações necessárias para se compreender os resultados desta pesquisa, pelos dados obtidos na análise realizada.

Na Tabela 17 representam-se os vários casos definidos a partir da aplicação dos métodos de migração, em cada cenário.

Tabela 17 – Identificação dos casos de teste

Cenários / Métodos	**OLTP**	**OLAP**	**HTAP**
ONLINE	Caso 1	Caso 2	Caso 3
OFFLINE 1	Caso 4	Caso 5	Caso 6
OFFLINE 2	Caso 7	Caso 8	Caso 9

Na validação e apresentação das métricas, algumas serão apresentadas por completo em cada apresentação do caso, sendo que outras, por serem mais simples ou por terem menor tamanho, podendo ser representadas em uma mesma tabela, serão assim apresentadas. Na Tabela 18 indicamos como cada métrica será apresentada em cada caso.

Tabela 18 – Relacionamento entre os casos de teste e as métricas

	Caso 1	**Caso 2**	**Caso 3**	**Caso 4**	**Caso 5**	**Caso 6**	**Caso 7**	**Caso 8**	**Caso 9**
BASELINE	1	1	1	1	1	1	1	1	1
CPU	X	X	X	X	X	X	X	X	X
DISCO(I/O)	X	X	X	X	X	X	X	X	X
REDE	X	X	X	X	X	X	X	X	X
MEMÓRIA	X	X	X	X	X	X	X	X	X
LATÊNCIA	X	X	X	N/A	N/A	N/A	N/A	N/A	N/A
INTERVENÇÃO MANUAL	1	1	1	1	1	1	1	1	1
INDISPONIBILIDADE	N/A	N/A	N/A	1	1	1	1	1	1

De forma a fazermos uma leitura adequada da Tabela cima consideramos a legenda seguinte:

- X – Será apresentada separadamente durante apresentação de cada caso.
- N/A – Métrica não aplicável ao caso atual.
- 1 – Será apresentada em quadro resumido, enquadrando todos os casos analisados em uma única apresentação (cf. secção 4.3 – Discussão dos casos apresentados).

Na Tabela 19 mostramos uma classificação a nível técnico, dividindo/classificando as métricas por tecnologia. Cada métrica é específica de uma camada tecnológica única, ou seja, cada uma é executada somente em um dos componentes de software.

Tabela 19 - Relacionamento das métricas com a componente tecnológica

Pré-Migração	**Sistema Operativo**	**Replicação**	**Pós-Migração**
Oracle	**Linux**	**GoldenGate, Scripts**	**Mongo**
BASELINE	CPU	LATÊNCIA	BASELINE
	DISCO(I/O)	INTERVENÇÃO MANUAL	
	REDE	INDISPONIBILIDADE	
	MEMÓRIA		

Conforme podemos ver referido na Tabela 19, as métricas referentes ao sistema operativo foram retiradas do Linux e a métricas referente a replicação, são aquela provenientes da replicação dos dados.

Iremos agora realizar a visualização de todos os casos e, posteriormente, analisar as métricas avaliadas e expor os resultados obtidos.

4.2. Apresentação dos casos apresentados

De forma a responder aos objetivos propostos, nesta secção iremos analisar as métricas. Começaremos por analisar as métricas que foram possíveis agregar num único quadro/tabela, de forma a facilitar o entendimento, e apresentar clara e objetivamente os valores a que chegámos (cf. Tabela 20).

Tabela 20 - Métricas agregadas para os métodos propostos.

	Caso 1	**Caso 2**	**Caso 3**	**Caso 4**	**Caso 5**	**Caso 6**	**Caso 7**	**Caso 8**	**Caso 9**
LATÊNCIA				N/A	N/A	N/A	N/A	N/A	N/A
INTERVENÇÃO MANUAL	4	4	4	2	2	2	2	2	2
INDISPONIBILIDADE	N/D	N/D	N/D	03:10:40	05:24:47	08:09:58	02:29:28	01:43:39	04:58:30

A legenda para o quadro acima é a seguinte:

- X – Será apresentada separadamente durante apresentação de cada caso
- N/A – Métrica não aplicável ao caso atual.
- N/D – Não definido, pois a mesma pode ou não ser utilizada, sendo que neste trabalho optámos por não utilizar e, assim, diferenciar totalmente dos métodos *offline* no quesito indisponibilidade.

As métricas serão analisadas e detalhadas em seguida.

Nota importante: Todas as Figuras referentes aos casos estudados – e aqui enunciados como Caso 01, Caso 02, ... e Caso 09 – estão em Anexo.

- LATÊNCIA

Podemos concluir que o caso 01 foi 51% mais rápido no transporte da informação na janela (ou fase) T1 quando comparado ao caso 02 na mesma janela (Cf. Tabelas 21 e 22). Já o caso 03 tal como esperado[79], representa quase o valor de ambos os modelos somados. Realizando uma análise comparativa no período, ou fase, T2, podemos ver que esta diferença cai para 49%, sendo que podemos ver nas demais métricas algumas diferenças, tais como uso de CPU e disco nos mesmos períodos (cf. Caso 01, Caso 02 e Caso 03, em Anexo).

- INTERVENÇÃO MANUAL

Para essa métrica foram computados somente os passos executados durante a migração, dadas as inúmeras implementações possíveis, não se fez a contagem dessa métrica antes do início da migração, visto que ao desenvolver os *scripts* foram surgindo várias alterações a estes, cada qual com mais aprimoramentos, sendo que em caso de existir em uma migração alguém que necessite de criar um processo de migração e que não conheça exatamente o que se pretende e como lá chegar, pode-se poluir a métrica de forma a esconder os passos realmente importantes desse processo.

Podemos com esta métrica ver claramente que o método de migração *online* exige maior iteração homem/processo (Cf. Tabela 20), visto que a explicação para isso é encontrada ao analisarmos o fluxo dos processos, pois o mesmo é composto por um passo extra em cada se comparado com os processos *offline*. Entendemos, assim, que a complexidade deste cenário online é um diferencial se comparado com os demais cenários online, ficando este como um ponto negativo ao mesmo.

- INDISPONIBILIDADE

Foram computadas e posteriormente comparadas todas as indisponibilidades durantes os vários casos migrados. Logo notou-se que o tempo de indisponibilidade é proporcional ao tamanho

[79] Ver Capítulo III, secção 3.4 em Cenário HTAP. Este cenário representa a os cenários OLTP e OLAP em execução ao mesmo tempo, sendo espectável o tempo ser a soma ou com uma diferença marginal.

dos *schemas*[80] migrados dentro de um mesmo método de migração, sendo que o método mais rápido foi o do método *offline* automático, para todos os casos utilizados.

A métrica de indisponibilidade, tem grande impacto nos sistemas, dado que durante a ocorrência dessa métrica a aplicação estará indisponível para uso. O tempo total de migração para o método *offline automático* e o tempo total de indisponibilidade registrada para este estão disponíveis na Tabela 21.

Tabela 21 - Tempo total e indisponibilidade *offline* automático

Cenário OLTP – Migração *offline* automático	
T1	00:58:16
T2	02:12:24
Indisponibilidade total:	03:10:40
Cenário OLAP – Migração *offline* automático	
T1	01:41:07
T2	03:43:40
Indisponibilidade total:	05:24:47
Cenário OLAP – Migração *offline* automático	
T1	02:18:37
T2	05:51:21
Indisponibilidade total:	08:09:58

Para o método de migração *offline* manual, podemos consultar a Tabela 22:

Tabela 22 - Tempo total e indisponibilidade *offline* manual

Cenário OLTP – Migração *offline* manual	
T1	01:28:40
T2	01:10:56
Indisponibilidade total:	02:29:28
Cenário OLAP – Migração *offline* manual	
T1 – buscar esse tempo nos logs	01:58:37
T2	01:43:39
Indisponibilidade total:	03:42:16
Cenário customizado – Migração *offline* manual	

[80] Segundo a documentação oficial do Oracle, schema é uma coleção lógica de dados ou objetos dentro do schema. Um schema pertence sempre a um único utilizador de base de dados e cada utilizador possui somente um schema, com um relacionamento um para um entre ambos. *http://docs.oracle.com/database/122/SQLRF/Database-Objects.htm#SQLRF20003*

T1	02:38:56
T2	02:46:19
Indisponibilidade total:	05:25:15

Iremos agora realizar a análise da métrica CPU e iniciando pelo cenário **OLTP** (cf. casos 01, 04 e 07, em Anexo), onde passaremos por todos os três métodos de migração, comparando a utilização deste recurso e ainda descrevendo o uso deste para o cenário mais otimizado e o menos otimizado a nível de melhor utilização deste recurso.

O Caso 01 apresenta uma utilização de CPU extremamente alta quando comparada com os casos 04 e 07, isso porque, como podemos observar na Figura 28 em Anexo, temos muitos picos do evento "cpu %iowait", sendo esse uma característica de esgotamento deste recurso, sendo que quanto mais tivermos este evento, menor a performance na utilização deste recurso.

Podemos também validar que o caso 07, possui a melhor utilização desse recurso, isso porque, quando nos referimos a utilização, não estamos referindo a utilização máxima deste recurso, mas sim a não termos falta deste recurso durante a migração, e este é por nós considerado o melhor caso de uso. Yaqub (2012) em sua tese media a quantidade de execuções executadas, nós, diferentemente, medimos a utilização do recurso, de modo a podermos validar a utilização deste recurso entre os diversos métodos.

Ao analisarmos o evento de "cpu %idle" no sistema, o mesmo caracteriza que a CPU não está a ser utilizada no momento, sendo que a melhor utilização é caracterizada pelo evento "cpu %user" onde a CPU é usada pelas aplicações alocadas no sistema operativo, chegando este a 80% no caso 07 e com inúmeros picos no caso 01, ficando entre 80% e 10% de utilização durante a janela medida.

Durante a janela T3 do caso 01, temos a maior défice de CPU; podemos ver claramente este comportamento ao validar o evento "cpu %iowait" quase sempre a 100% de utilização, ou seja, caracterizando assim a falta deste recurso para processamento e totalmente alocado para escrita em disco. Este recurso apresenta uma utilização totalmente diferente se comparada a janela T2 do caso 04, onde 80% da CPU esta apresentada no evento "cpu %user" e o evento "cpu %iowait" teve apenas um pico de utilização perto do 50% de utilização e em média ficando em 20% de utilização.

O caso 07 e o caso 04 apresentam uma utilização quase que equivalente deste recurso, sendo que temos momentos de oscilação na janela T2 do cenário 07 no momento que temos o

carregamento dos ficheiros para esse caso, posteriormente durante o carregamento das tabelas não se notou diferença na utilização deste recurso nos casos aqui discutidos.

Podemos ver que o recurso de CPU é notoriamente mais eficiente no caso 07 ao analisarmos a métrica com mesmo nome, sendo que esta apresenta os maiores picos de utilização após a leitura na base de dados e escrita nos ficheiros em CSV para envio para o servidor destino, sendo que no caso 07 o maior pico de utilização foi entre os 50 e 60 mil eventos de "rk/Bs" durante o evento de escrita.

Esta métrica tem um comportamento muito diferente nos casos 01 e 04, isso porque esses casos são aplicações complexas a ser executadas, e não simplesmente *scripts*, como é o caso 07. Nestes casos, o recurso de disco é consumido em sua maioria pelo evento de leitura, isso porque os ficheiros são lidos na base de dados e escritos em um formato proprietário pelo software de replicação, e enviado para o servidor destino, esse fluxo é caracterizado pela leitura que pode ser vista através do evento "rk/Bs".

Na parte final da janela T1 do caso 07, podemos ver que a utilização passou a ser toda para o evento "rkB/s", evento esse esperado, pois nesse momento temos a leitura dos processos para envio, procedimento esse que para o caso 07 foi automatizado através do uso de ferramentas disponíveis no sistema operativo, nesse caso foi utilizado o *scp* para envio dos ficheiros, caracterizando somente leitura neste momento.

Para a métrica DISCO, tivemos o maior pico de utilização no evento "wkB/S" da janela T2 no caso 01, chegando esta quase às 35 mil unidades deste evento, sendo que para o caso 04 chegou às 50 mil e no caso 07, ultrapassou as 60 mil neste evento, caracterizando uma maior escrita do que leitura.

Podemos constatar no caso 01, durante a janela T3, o maior pico do evento "rkB/s", chega às 6 mil unidades, sendo que nos outros casos, somente durante o carregamento dos ficheiros para o método *offline* manual é que tivemos um pico de utilização acentuado, no entanto curto, logo após o inicio do próximo passo, a maior utilização passou a ser a escrita, caracterizada pelo evento "wkB/s". Ambos os eventos de leitura são esperados, pois no caso 01 na janela T3, estamos a realizar o sincronismo final dos ficheiros, sendo que é realizada uma leitura em todos os ficheiros em busca da janela temporal para início da aplicação do delta da replicação.

Durante a análise da métrica memória, poucas foram as conclusões acerca da utilização deste recurso pela migração, ou mesmo ao impacto causado por esta, isso porque a memória se

manteve em constante utilização a nível de sistema operativo. Para o caso 07 a memória manteve-se com alguma oscilação em torno de 10% e durante a janela T3 deste caso, fixou-se nos 2,5GB de utilização. A utilização desta métrica teve maior oscilação no caso 07, onde foi aplicado o método *offline* manual, onde um *script* que foi automatizado utilizou-se de recursos dos sistema operativo e do SGBD para realizar a migração, existindo vários momento de oscilação entre os eventos "Cached" e "MemFree", que resumidamente foi a tabela antes carregada em memória para escrita em disco, após o fluxo final, que é descarregada e uma nova entra em seu lugar, seguindo este mesmo fluxo para todas as tabelas migradas.

A métrica de rede foi pouco informativa durante os nossos testes, confirmando-se o comportamento já esperado dos casos, ou seja nos casos 01 e 04, temos a aplicação (GoldenGate) sempre a enviar informação e receber a confirmação do envio, já no caso 07, podemos ver que a utilização de rede é mínima se comparada com os demais casos, ficando a mesma sempre sempre perto das 160 mil unidades, sendo que como a utilização de rede é cumulativa; esse cenário aconteceu após o carregamento do cenário 04, ficando aqui o reflexo da utilização anterior; outros testes foram executados por forma a comprovar essa utilização e o mesmo confirmou-se, ou seja, a utilização é sempre mínima para o caso 07, onde a migração é manual e este recurso (rede) fica quase na sua totalidade sem utilização, sendo utilizado somente ao final do processo.

Para o caso 01 na fase T3, podemos claramente visualizar a não utilização de rede, comportamento esse que condiz com o que acontece durante o processo, sendo que nessa fase, os ficheiros já se encontram no servidor destino e estão a ser lidos localmente, sendo que não temos aqui utilização de rede.

Podemos concluir que o caso 01 é o maior consumidor de CPU, igualmente o maior consumidor dos recursos de disco, sendo que o caso menos intrusivo para o sistema operativo é o caso 07.

Quanto à análise centrada na migração do cenário **OLAP**, representados nos casos 02, 05 e 08 (cf. Caso 02, Caso 05 e Caso 08, em Anexo), podemos concluir o seguinte:

A Figura 40 em Anexo demonstra uma grande utilização de CPU, o mesmo comportamento que observámos ao analisar o caso 01, ambos sendo migrados pelo método online, diferenciando apenas o cenário, sendo um OLTP e o outro OLAP.

Durante a execução do caso 02, simulámos durante a janela 03 um erro aleatório, de forma a que pudéssemos validar se um erro é percetível pelas métricas propostas, sendo que foi possível notar o esgotamento de CPU e a falha dos demais processos, onde não houve carregamento de dados, sendo visível este comportamento, ilustrado nas Figuras 42, 45 e 51 (cf. Anexos); na Figura 48 não se notou qualquer movimentação dos blocos em memória, reflexo da falta de CPU para proceder a escrita dos dados em disco.

Já no caso 03, para que fosse possível validar a finalização de uma migração, executámos a paragem da replicação durante a coleta das métricas, foi possível constatar a mudança no comportamento da métrica (cf. Caso 03, Figuras 57, 59 e 63, em Anexo).

4.3. Discussão dos resultados

Sucintamente, para que uma migração seja possível precisamos de garantir uma série de requisitos – nomeadamente, Planeamento, Quantidade de registos / Situação inicial, Mapeamento dos tipos de dados (data type) / Equivalência entre metadados, Restrições e *trigger*, Codificação de caracteres, Ensaios, Implementação, Monitorização / Validação parcial e Final, Staging area/ Sem staging area, Falhas durante a migração, Modelação de dados – de acordo com o definido por vários autores (e.g. Antaño et al., 2014; Sancho & Oliveira, 2015; Santos, 2015; Oliveira & Marcelino, 2012; Davenport et al., 2014; Gomes, 2011; Pereira, 2014)[81], e com algumas novas métricas, por nós definidas.

Pudemos desta forma verificar que, nos cenários por nós testados[82], foi concluída com êxito a migração de dados de BD relacional para NoSQL, para posterior avaliação das métricas – podendo assim identificar-se os requisitos e as várias fases de uma migração durante a execução real de uma migração.

Os resultados verificados permitem-nos sustentar a premência deste estudo, e denotam o seu carácter inovador, porque, tratando-se de migração de bases de dados (ou mesmo de aplicações legadas alocadas em bases de dados relacionais para bases de dados NoSQL), e ainda que existam algumas teses que de algum modo abordem o tema – como o trabalho de Gomes (2011)

[81] Ver capítulo III, secção 3.6 em Requisitos para migração
[82] Ver capítulo III, secção 3.4 em m Cenário OLTP, OLAP, HTAP

–, a verdade é que, naquilo que se refere a um método genérico, que possa ser aplicado em vários casos de migração, a literatura que encontrámos é muito escassa e incompleta. Também por isso abordámos e desenvolvemos esta temática na presente dissertação.

Através dos métodos de migração que apresentámos e desenvolvemos neste trabalho, permitiram-nos verificar, e testar, os diferentes métodos de migração de uma base de dados relacional para NoSQL, de forma a que cada um deles mostrou-se mais eficiente em um quesito. No caso do método de migração *online*, o consumo de recursos de sistema operativo é superior ao observado nos métodos *offline*, sendo que a métrica de indisponibilidade foi sempre melhor (ponto positivo quando a aplicação precisa de estar *online*, sendo um exemplo desta situação, uma aplicação bancária), sendo este indicado para os casos onde tenhamos este tipo de recurso disponível (recursos no sistema operativo) e em que não tenhamos uma grande janela de indisponibilidade. A mesma lógica pode ser inversamente aplicada para ambos os métodos *offline*, ou seja, onde existe pouco recurso disponível e uma grande janela temporal para indisponibilidade é executada.

As métricas aqui estudadas mostraram-se bastante adequadas ao proposto, sendo que pela sua análise, podemos entender o comportamento do sistema operativo. Yaqub (2012), em sua tese, foca-se nas métricas de sistema operativo para uma migração entre sistemas de virtualização e físico; essas métricas foram aqui utilizadas e, adicionalmente, por termos neste trabalho o envolvimento da camada de redes, considerámos ainda uma métrica específica para as redes.

A análise de cada uma das métricas, permitiu-nos distinguir os consumos de recursos em cada uma das fases da migração, respondendo assim a alguns dos objetivos propostos.

Através da revisão da literatura e das pesquisas realizadas foi possível identificar quais são os requisitos e as várias fases de uma migração, procedendo então para a validação e verificação dos possíveis métodos de migração de uma base de dados relacional para uma NoSQL, utilizando aqui os métodos fornecidos pelo próprio desenvolvedor da base de dados (Cf. Figura 20 - Opções para migração de dados NoSQL) e realizando uma adaptação para o surgimento de um terceiro método. De modo a avaliar cada um destes métodos, de acordo com as etapas seguidas em cada fase da migração – recorrendo a métricas específicas e adequadas, em cada caso, foi necessário realizar a análise das mesmas, sendo que todos os resultados podem ser consultados no Anexo 1, onde podemos avaliar os vários métodos de acordo com as métricas estabelecidas, realizando assim a comparação entre os métodos, e podendo validar cada uma das métricas, em cada cenário. Com base nos resultados e posterior a análise foi assim possível

determinar o método de migração mais eficiente para cada um dos cenários de bases de dados estudados (OLTP, OLAP e HTAP).

Comparando os vários casos[83] de migração estudados, foram validados os consumos dos recursos através da análise das métricas, verificando-se qual o melhor método de migração, consoante as necessidades ou disponibilidades de recursos, sendo que a métrica indisponibilidade é uma das mais críticas, pois o SLA (Cf. Capítulo I na secção 1.1 Contextualização) a que uma aplicação está associada, poderá por vezes ditar qual o método de migração a ser utilizado. Podemos citar aqui um sistema bancário, onde a aplicação precisa operar em um regime 24h X 7dias, sem paragens, ficando aqui impraticável ou inviável para o negócio uma paragem para realizar a migração.

Foi também possível validar a existência de erros durante uma migração, e a mudança de comportamento, pois quando executámos a finalização de um processo de migração, as métricas indicaram o tempo exato desta paragem, respondendo assim a mais um dos objetivos propostos. Ou seja, os erros no decorrer da migração podem ser percetíveis, e detetados, com a utilização das métricas aplicadas neste trabalho.

[83] Cf. Tabela 17, na secção 4.1 Discussão dos resultados.

Capitulo V - Conclusões

Tendo em consideração o grande crescimento das bases de dados e a progressiva adoção do modelo NoSQL, a necessidade de migrar os modelos relacionais vigentes para NoSQL, ou de proceder à integração entre os diferentes sistemas, é notória. Foi este facto que mais nos motivou a desenvolver e orientar o presente trabalho, de forma a responder à pergunta de partida: Qual o método mais eficiente para proceder à migração de uma base de dados relacional para uma base de dados NoSQL?

Podemos assim responder que o método de migração mais eficiente é o método de migração *offline* automático. No entanto, este requer uma grande janela de indisponibilidade, sendo que, caso não seja possível realizar a paragem da aplicação ou mesmo da base de dados, durante o processo de migração, o método *online* será o indicado.

Adicionalmente, conseguimos fazer uso de uma integração entre os SGBDs relacionais e NoSQL. Os resultados por nós obtidos permitiram não somente responder a todos os objetivos propostos, mas também possibilitaram encontrar contribuições para o contexto académico e profissional/prático, propondo métodos de migração, e meios de validação dos mesmos. Pela análise às métricas coletadas, respondemos aos objetivos propostos. Identificaram-se os requisitos e as várias fases de uma migração, procedendo então para a validação e verificação dos possíveis métodos de migração de uma base de dados relacional para uma NoSQL, utilizando aqui os métodos fornecidos pelo próprio desenvolvedor da base de dados (Cf. Figura 20 - Opções para migração de dados NoSQL), e realizando uma adaptação para o surgimento de um terceiro método. De modo a avaliar cada um destes métodos, de acordo com as etapas seguidas em cada fase da migração – recorrendo a métricas específicas e adequadas, em cada caso –, foi necessário realizar a análise das mesmas. Os resultados obtidos permitiram-nos determinar o método de migração mais eficiente para cada um dos cenários de bases de dados estudados (OLTP, OLAP e HTAP).

Após a execução dos testes, e determinação do melhor método de migração, percebemos a necessidade de responder a uma outra pergunta: “Qual a janela de indisponibilidade disponível para a migração?”. Devemos obter informações sobre a disponibilidade do ambiente, pois caso tenhamos uma grande janela de indisponibilidade, onde o sistema aplicacional possa ser interrompido durante um longo período de tempo, então, nesse caso, verificámos que é o método *offline* automático que tem melhor performance.

Quanto à métrica mais utilizada durante a aplicação dos métodos de migração, foi a que se refere à utilização de CPU, em relação ao qual sempre se observou um maior impacto, seguindo-se a utilização de disco. O método de migração online foi o maior consumidor destes dois tipos de recursos, o que se repercutiu nos valores observados nas duas métricas respetivas, durante a execução das migrações.

Do ponto de visto teórico, a maior contribuição deste estudo é mostrar como podem ser utilizados os métodos de migração de bases de dados relacionais para NoSQL, e as suas métricas associadas, pois estes métodos podem ser aplicados em vários sistemas de SGBD, não sendo vinculados somente a este trabalho, ou ao *software* aqui utilizado.

Esta investigação visa contribuir, ainda que com algumas limitações, para o enriquecimento dos métodos de migração já existentes, de forma a poder-se, mais assertivamente, controlar-se os recursos durante um processo migração. Dando visibilidade às várias métricas disponíveis, e mensurando os impactos durante a realização de uma migração – respondendo assim, também, aos objetivos propostos.

O processo de elaboração de uma dissertação é um processo bastante dinâmico, em que novos conhecimentos e curiosidades académicas vão surgindo, com naturalidade, ao longo do seu desenvolvimento. Pelo que, alguns objetivos secundários, ou complementares aos principais, foram surgindo, e sendo definidos, ao longo da investigação. Podendo-se aqui referir o modelo de migração *online*, onde o mesmo pode ser utilizado para integração entre os sistemas relacionais e NoSQL. Permitindo assim a existência de ambos os sistemas em simultâneo, e recebendo os mesmos dados dos sistemas aplicacionais através do sistema relacional, e enviados pelo *software* de migração, tal como o modelo *online* indica.

Consideramos que, no futuro, uma migração utilizando os mesmos métodos ou adaptações semelhantes para a *Cloud* seria um trabalho interessante, de forma a permitir realizar uma migração entre *Datacenters* distintos, inclusive distribuídos entre continentes. Outro desafio gratificante para um futuro pesquisador será a realização da migração destes cenários utilizando outras técnicas de migração, e compará-las utilizando as mesmas métricas, de forma a realizar uma adaptação destes métodos.

Em suma, verificámos a performance de três métodos de migração possíveis entre bases de dados relacional e NoSQL, recorrendo a um conjunto de métricas que nos fornecem uma visão detalhada do consumo de recursos e também detalhes sobre o processo de migração, em cada

uma das suas fases, mantendo com isso um controlo pormenorizado a nível do desempenho da migração, seja qual for o método escolhido. Com este trabalho esperamos ter dado um bom contributo, em termos científicos, para este domínio das tecnologias de informação e, em particular, dos sistemas de informação. Os resultados que obtivemos podem revelar-se importantes para qualquer profissional que trabalhe com SGBD, e precise de efetuar migrações de bases de dados e/ou avaliar a sua performance.