UNIVERSIDADE PAULISTA – UNIP

SABRINA JÚLIA SANTOS SÉRGIO

BANCO DE DADOS EM NUVEM: UMA MUDANÇA NO PARADIGMA NAS BASES DE DADOS

LIMEIRA
2018

UNIVERSIDADE PAULISTA – UNIP

SABRINA JÚLIA SANTOS SÉRGIO

BANCO DE DADOS EM NUVEM: UMA MUDANÇA NO PARADIGMA NAS BASES DE DADOS

Trabalho de conclusão de curso apresentado à banca examinadora da Universidade Paulista - UNIP, como requisito parcial à obtenção do Bacharelado em Ciência da Computação sob a orientação do Professor Me. Antônio Mateus Locci e Professor Me. Sérgio Eduardo Nunes.

LIMEIRA
2018

SABRINA JÚLIA SANTOS SÉRGIO

BANCO DE DADOS EM NUVEM: UMA MUDANÇA NO PARADIGMA NAS BASES DE DADOS

Trabalho de conclusão de curso apresentado à banca examinadora da Universidade Paulista - UNIP, como requisito parcial à obtenção do Bacharelado em Ciência da Computação sob a orientação do Professor Me. Antônio Mateus Locci e Professor Me. Sérgio Eduardo Nunes.

Aprovada em ____ de _____ de 20___.

BANCA EXAMINADORA

_____

Prof. Dr. _____

_____

Prof. Me. _____

_____

Prof. Esp. _____

DEDICATÓRIA

A Deus, por ser essencial em minha vida, autor de meu destino, meu guia, socorro presente na hora da angústia. À minha mãe Naudeci Carvalho Santos, e aos meus irmãos.

AGRADECIMENTOS

Agradeço à Deus por ter me sustentado até o presente momento, por cada instante ao meu lado me dando graça e sabedoria em todas as circunstâncias de minha vida.

Agradeço a minha linda família que em tudo me apoiou, principalmente minha mamãe, Naudeci Carvalho Santos, pelo cuidado especial em cada detalhe.

Ao meu amigo Eduardo Gabriel dos Santos, que me ajudou durante esses 4 anos de curso.

Ao meu Pastor Natalino Corrêa e sua amada família, que por diversas vezes me aconselhou e orientou da melhor maneira possível.

Aos professores que me ensinaram, orientaram e de certa forma mostraram o caminho ideal a ser seguido.

Agradeço aos meus professores Orientadores Me. Sérgio Eduardo Nunes e Me. Antônio Mateus Locci pelo conhecimento e esforço empregado.

Também ao professor Me. Marcos Vinicius Gialdi que contribuiu de uma forma carinhosa para conclusão desse trabalho.

RESUMO

SABRINA, Júlia Santos Sérgio. *Banco de Dados em Nuvem: Uma Mudança no Paradigma nas Bases de Dados*. Trabalho de conclusão de curso, 2018.

Muitas organizações coletam grandes quantidades de dados de clientes, dados científicos, vendas e outros dados para análise futura. Tradicionalmente, a maioria dessas organizações armazena dados estruturados em bancos de dados relacionais para acesso e análise subsequentes. No entanto, um número crescente de desenvolvedores e usuários começaram a utilizar vários tipos de bancos de dados não relacionais, agora frequentemente chamados de banco de dados NoSQL. Bases de dados relacionais, incluindo bancos de dados hierárquicos, gráficos e orientados a objeto, existem desde o final dos anos 60. Atualmente, novos tipos de bancos de dados NoSQL estão sendo desenvolvidos e começando a ganhar atração no mercado. Bancos de dados NoSQL diferentes usam abordagens diferentes. O que eles têm em comum é que eles não são relacionais. Sua principal vantagem é que, ao contrário dos bancos de dados relacionais, eles lidam com dados não estruturados, como arquivos de processamento de texto, e-mail, multimídia e mídia social de forma eficiente. E com a inclusão do armazenamento em nuvem (*cloud*) tornando qualquer banco de dados móvel, mudando as especificações e até a estrutura de dados. Esta pesquisa, classificada como Exploratória e Quantitativa, apresenta conceitos sobre bancos de dados MySQL e NoSQL, e discute questões como limitações, vantagens, desvantagens e dúvidas em relação aos bancos de dados citados.

Palavra-Chave: Armazenamento em Nuvem. Banco de Dados. NoSQL. Relacional.

ABSTRACT

SABRINA, Júlia Santos Sérgio. Cloud Database: A Paradigm Change in Databases BSc. Thesis, 2018.

Many organizations collect large amounts of customer data, scientific data, sales and other data for future analysis. Traditionally, most of these organizations store structured data in relational databases for subsequent access and analysis. However, a growing number of developers and users have begun to use various types of non-relational databases, now often called the NoSQL database. Relational databases, including hierarchical, graphical, and object-oriented databases, have existed since the late 1960s. New types of NoSQL databases are now being developed and beginning to gain market appeal. Different NoSQL databases use different approaches. What they have in common is that they are not relational. Its main advantage is that, unlike relational databases, they deal with unstructured data such as word processing, email, multimedia, and social media files efficiently. And with the inclusion of cloud storage making any database mobile, changing specifications and even data structure. This research, classified as Exploratory and Quantitative, presents concepts about MySQL and NoSQL databases, and discusses issues such as limitations, advantages, disadvantages and doubts regarding the cited databases.

Keywords: Cloud Storage. Database. NoSQL. Relational.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

ACID	Atomicidade, Consistência, Isolamento, Durabilidade
AWS	Association World Server
BLOB	Binary Large Object
BSON	Binary JSON
CC	Cloud computing
CCN	Conceito de Computação em Nuvem
CRM	Customer Relationship Management
DBA	Database Administrator
DBaaS	Serviço de Banco de Dados
DBMS	Database Management Systems
DML	Data Manipulation Language
DMS	Database Migration Service
HVAC	Heating, Ventilation and Air Conditioning
JSON	Javascript Object Notation
RDBMS	Relational Database Management System
RDS	Relation Database Service
SaaS	Software-as-a-Service
SGBD	Sistema de Gerenciamento de Banco de Dados
SLA	Service Level Agreement
SQL	Structured Query Language
SQL	Structures Query Language
VPC	Virtual Private Cloud
VPN	Virtual Private Network
WWW	World Wide Web

SUMÁRIO

1 INTRODUÇÃO

O departamento de Tecnologia da informação (TI) de qualquer organização é responsável por fornecer informações confiáveis na computação, como, armazenamento, backup e instalações de rede no custo mais baixo possível (BARROS, 2008).

Buya et al. (2008) diz, enorme investimento em infraestrutura de TI funciona como obstáculo na sua adoção, especialmente para pequenas organizações. Empresas que desejam economizar dinheiro procuram alternativas que podem reduzir seus investimentos de capital envolvido na compra e manutenção de hardware e software para que eles possam obter o máximo de benefícios da TI. *Cloud Computing* (CC) ou Conceito de Computação em Nuvem (CCN) torna-se uma escolha natural e ideal para tais corporações e clientes.

Computação em nuvem aproveita muitas tecnologias, como a consolidação de servidores, maior capacidade de armazenamento e de forma mais rápida, computação em grade (GRID), virtualização, arquitetura N-camada e redes robustas. Isto oferece uma infraestrutura altamente escalável e configuração mínima com baixo custo de implementação e pouca mão de obra com a manutenção (CHANG et al. 2006).

Fornecendo assim serviços relacionais a TI, como *Software-as-a-Service* (SaaS) ou Software como Serviço, Plataformas de Desenvolvimento e Infraestrutura através da rede *on-demand* (que pode ser acessado a qualquer momento a partir de qualquer lugar) com base no modelo "*pay-as-you-go*" (pagando apenas o que é utilizado).

Chang et al. (2006) fala que o conceito de rápido crescimento mudara as percepções dos usuários relacionados com TI. Flexibilidade, escalabilidade, alta disponibilidade, uso de precificação e *multi-tenancy*, essas são as principais características da Computação em Nuvem. Reduz os custos dos recursos até o final do provedor devido a economias de escala. Provisionamento rápido e implementação imediata das últimas aplicações a menor custo são os benefícios que forçam as pessoas a adotar a computação em nuvem.

Com toda essa infraestrutura, atualmente mais opções de banco de dados estão se tornando disponíveis para atender as necessidades de processamento de dados. As tecnologias NoSQL estão invadindo a predominância de bancos de dados

relacionais, que garantem a integridade de dados e transações, mas normalmente impõem esquemas rígidos baseados em SQL para estruturar e armazenar dados. Agora, a medida que as empresas observam as opções, os inúmeros bancos de dados NoSQL que surgiram nos últimos anos oferecem recursos atraentes para gerenciar conjuntos de grandes volumes de dados variados e em constante mudança. Enquanto isso, muitos dos principais fornecedores de Sistema de Gerenciamento de Banco de Dados (SGBD) estão aumentando suas ofertas de banco de dados com recursos NoSQL (TAMANE, 2016).

Embora os produtos do sistema de gerenciamento de banco de dados relacional continuem a dominar a demanda em termos de instalações e uso, a tecnologia de banco de dados NoSQL representa o tipo de SGBD de crescimento mais rápido adotado atualmente (Ll; MA; CHEN, 2014).

Srivastava et al. (2015) dizem que o NoSQL descreve uma ampla categoria de sistemas de bancos de dados que, em alguns casos, podem ter recursos e casos de uso drasticamente diferentes. No final dos anos 90, o termo originalmente significava, literalmente, nada de SQL.

Com o passar do tempo, a medida que a realidade se instalou e a praticidade de expor dados em uma interface SQL tornou-se aparente, o NoSQL mudou seu significado para NO SQL, onde NO se refere a *Not Only Hope*, os bancos de dados NoSQL são considerados bancos de dados da próxima geração, pois geralmente não são relacionais, distribuídos de código aberto e dimensionáveis horizontalmente.

1.1 Objetivos

1.1.1 Objetivo Geral

Este estudo tem como objetivo analisar as dificuldades de migração dos bancos de dados relacional e não relacional para a nuvem, explorando as vantagens e desvantagens da migração, aludindo os recursos envolvidos no processo.

1.1.2 Objetivos Específicos

- Analisar as dificuldades de migração de um banco de dados relacional e de um não relacional para a *cloud*;
- Levantar as vantagens e desvantagens dos bancos de dados relacionais e não relacionais;
- Realizar estudo de viabilidade de migração de um banco de dados relacional e de um NoSQL para a *cloud*;

1.2 Justificativa

A justificativa deste estudo se dá devido a crescente necessidade do aumento de análises e armazenamento de dados no mundo corporativo de hoje, juntamente com uma correspondência de arquitetura nas opções de implantação disponíveis no momento.

A fim de baixar os custos para as empresas e aumentar a velocidade de transferência de dados é preciso delinear uma pesquisa para análise de dados em grande escala na nuvem, mostrando porque os sistemas atualmente disponíveis não são ideais para a implantação *Cloud*, para poder argumentar que existe a necessidade de um DBMS projetado e arquitetado especificamente para plataformas de computação em nuvem para melhor desempenho, fazendo-se necessário a comparação de banco de dados relacionais (NoSQL) com banco de dados em nuvem.

1.3 Metodologia

Este estudo é resultado de uma pesquisa exploratória, baseada em um levantamento bibliográfico, composto de livros, artigos e demais materiais científicos de autores conceituados na área. Essa apresentação de dados é de suma importância para a utilização da comunicação via rede elétrica expondo as características, conceitos, utilidades, o lado positivo e negativo dessa transmissão de dados. As fontes que serão utilizadas nesse trabalho para fornecerem as respostas adequadas as soluções dos problemas propostos. Foram utilizados livros

online e em idioma português, entre 2008 a 2018, que abordam as características de banco de dados e armazenamento em nuvem, o conceito da implantação da utilização *Cloud* e possíveis problemas de implementação. Também na pesquisa foram utilizados artigos encontrados sobre a temática, nacionais e internacionais.

A coleta de dados tem o seguinte procedimento: pesquisa exploratória dos materiais selecionados e leitura seletiva e objetiva, a fim de verificar se os materiais consultados abordam a proposta do tema e se o registro das informações extraídas tem relevância, para isso foi analisado com cautelas os autores, ano de publicação, métodos, resultados e conclusões. Essa etapa tem como finalidade compreender as informações e sumariá-las, de acordo com as fontes pesquisadas, para possibilitar as respostas aos problemas em questão. Conforme as respostas dos problemas foram emergindo, forma também analisadas, abrindo espaço para discussão dos resultados obtidos a partir do referencial teórico relativo a temática do estudo.

2 CAPÍTULO 1 – BANCO DE DADOS RELACIONAIS

Nesta seção serão apresentados os conceitos de Banco de Dados Relacionais, bem como as vantagens e desvantagens da sua utilização.

2. 1 Introdução a Banco de Dados Relacionais

Um banco de dados é uma coleção organizada de dados, armazenados e acessados eletronicamente. Os projetistas de banco de dados normalmente organizam os dados para modelar aspectos da realidade de uma maneira que suporte processos que exigem informações, como, por exemplo, modelar a disponibilidade de quartos em hotéis de forma a apoiar a localização de um hotel com vagas (DATE, 2004).

Segundo Elmasri et al. (2011) o sistema de gerenciamento de banco de dados (SGBD) é o software que interage com usuários finais, aplicativos e com próprio banco de dados para capturar e analisar dados. Um DBMS de propósito geral permite a definição, criação, consulta e administração de banco de dados.

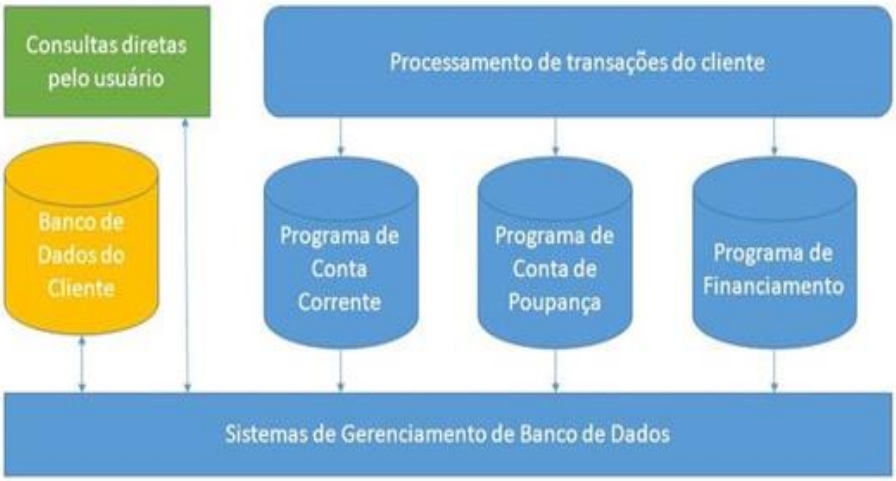

Figura 1 – Abordagem do Gerenciamento de Banco de Dados.
Fonte: Adaptado de SlidePlayer.

Um banco de dados é geralmente armazenado em um formato específico do DBMS que não é portátil, mas DBMSs diferentes podem compartilhar dados usando padrões como SQL e ODBC ou JDBC. A soma total do banco de dados, o DBMS e seus aplicativos associados podem ser chamados de “sistema de banco de dados” (LEAVITT, 2010).

Muitas vezes o termo “banco de dados” é usado para se referir livremente a qualquer um dos DBMS, o sistema de banco de dados ou um aplicativo associado ao banco de dados. Para Elmasri e Navathe, banco de dados representa:

> Um banco de dados representa algum aspecto do mundo real, às vezes chamado de minimundo ou de universo de discurso (UoD – *Universe of Dscourse*). As mudanças no minimundo são refletidas no Banco de Dados. Um banco de dados e uma coleção logicamente coerente de dados com algum significado inerente. Uma variedade aleatória de dados não pode ser corretamente chamada de banco de dados. Um banco de dados é projetado, construído é populado com dados para uma finalidade específica. Ele possui um grupo definido de usuários e algumas aplicações previamente concebidas nas quais esses usuários estão interessados (ELMASRI; NAVATHE, 2011, p. 3).

Gunter et al. (2013) diz que os cientistas da computação podem classificar os sistemas de gerenciamento de banco de dados de acordo com os modelos de banco de dados que eles suportam. Os bancos de dados relacionais se tornaram dominantes na década de 1980. Eles modelam dados como linhas e colunas em uma série de tabelas, e a grande maioria usa SQL para escrever e consultar dados. Nos anos 2000, houve a evolução nos bancos de dados não relacionais tornaram-se populares e conhecidos como NoSQL, porque usam linguagens de consulta diferentes.

2.2 Banco de Dados Relacional

Marcário (2007) explica que o banco de dados relacional foi inventado em 1970 por EF Codd, um jovem programador da IBM. Em seu artigo, “Um modelo relacional de dados para grandes bancos de dados compartilhados”, Codd propôs a mudança do armazenamento de dados em estruturas hierárquicas ou de navegação para a organização de dados em tabelas contendo linhas e colunas.

Cada tabela, que é chamada de relação, em um banco de dados relacional contém um ou mais categorias de dados em colunas, também chamadas de atributos. Cada linha, também chamada de registro ou tupla, contém uma instância única de dados, ou chave, para as categorias definidas pelas colunas. Cada tabela possui uma chave primária exclusiva, que identifica as informações em uma tabela. A relação entre tabelas pode então ser definida através do uso de chaves

estrangeiras – um campo em uma tabela que se vincula a chave primaria de outra tabela (MACÁRIO et al. 2007).

Empregado

NumEmp	NomeEmp	Salário	Dept
032	J Silva	380	21
074	M Reis	400	25
089	C Melo	520	28
092	R Silva	480	25
112	R Pinto	390	21
121	V Simão	905	28
130	J Neves	640	28

Departamento

NumDept	NomeDept	Ramal
21	Pessoal	142
25	Financeiro	143
28	Técnico	144

Figura 2 – Banco de Dados Relacional.
Fonte: adaptado de CULTURAMIX.

Por exemplo, um típico banco de dados de entrada de pedidos comerciais incluiria uma tabela que descreveria um cliente com colunas para nome, endereço, número de telefone e assim por diante. Pinheiro et al. (2009), diz que um usuário de um banco de dados relacional pode, então, obter uma visualização do banco de dados para atender as suas necessidades. Por exemplo, um gerente de filial pode gostar de visualizar ou relatar todos os clientes que compraram produtos após uma determinada data. Um gerente de serviços financeiros na mesma empresa, poderia, nas mesmas tabelas, obter um relatório sobre as contas que precisam ser pagas.

Ao criar um banco de dados relacional, você pode definir o domínio de valores possíveis em uma coluna de dados e outras restrições que podem se aplicar a esse valor de dados. Por exemplo, um domínio de possíveis clientes poderia permitir até 10 nomes possíveis de clientes, mas ser restrito em uma tabela a permitir que apenas três desses nomes de clientes sejam especificados. Duas restrições estão relacionadas a integridade dos dados e as chaves primarias e estrangerias:

- A integridade da entidade garante que a chave primária em uma tabela seja exclusiva e que o valor não seja definido como nulo.

- A integridade referencial requer que todo valor em uma coluna de chave estrangeira seja encontrado na chave primária da tabela da qual se originou.

2.2.1 Tipos de Dados

Há várias categorias de banco de dados, desde arquivos simples básicos que não são relacionais ao NoSQL, até bancos de dados gráficos mais novos que não são considerados ainda mais relacionais do que os bancos de dados relacionais padrão (MySQL, 2018).

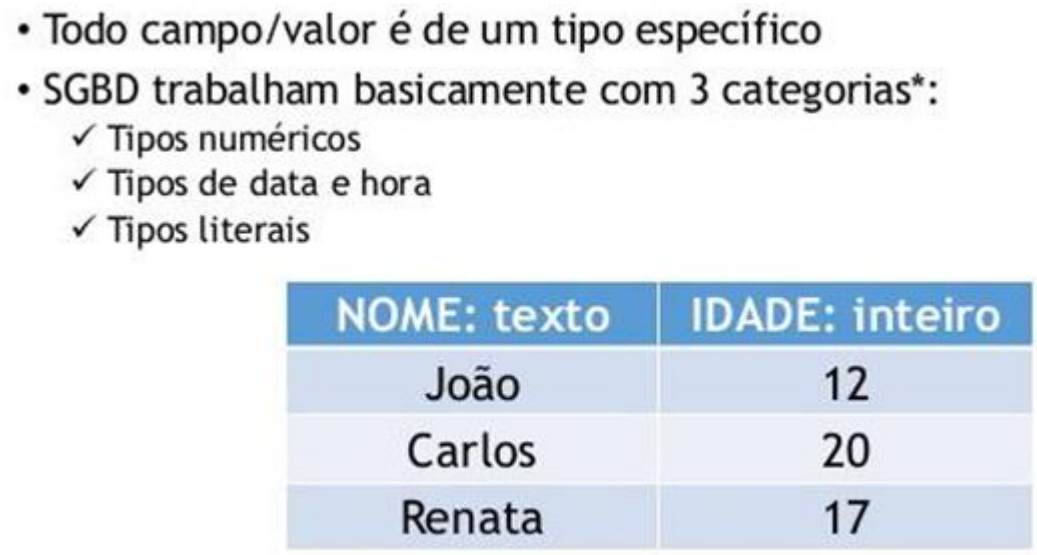

Figura 3 – Tipos de Dados.
Fonte: MySQL, 2018.

Um banco de dados de arquivo simples consiste em uma única tabela de dados sem inter-relação – normalmente arquivos de texto. Esse tipo de arquivo permite que os usuários especifiquem atributos de dados, como colunas e tipos de dados.

Os bancos de dados relacionais padrão permitem que os usuários gerenciem relacionamentos de dados predefinidos em vários bancos de dados. Os bancos de dados relacionais populares incluem o Microsoft SQL Server, o Oracle Database, o MySQL e o IBM DB2.

Todos usaram um banco de dados de uma forma ou de outra. Uma lista telefônica até se qualifica como uma. Os bancos de dados armazenam informações e as organizam de forma prática. Aplicativos de software, como o Microsoft Access,

informam funções de banco de dados, oferecendo ao usuário grande flexibilidade com seus dados.

2.2.1.1 Tabelas

Popescu (2015) compreende que as informações básicas armazenadas em um programa de dados são inseridas em uma tabela. De aparência semelhante a uma planilha, como o Microsoft Excel ou o Lotus 123, as tabelas fornecem uma maneira organizada de inserir informações. Os usuários podem criar nomes de campo, como nome, endereço e número de identificação do cliente. Os usuários também podem formatar os campos na tabela. Por exemplo, um campo numérico pode aparecer com formatação monetária, como dólares ou euros.

Silberschatz et al. (2006) continua, a maioria dos programas de software de banco de dados tem modelos disponíveis para formatos usados com frequência, como gerenciamento de contatos, vendas ou acompanhamento de inventário. Os usuários podem modificar esses modelos para personaliza-los para suas próprias finalidades especificas.

2.2.1.2 Formulários

O software de banco de dados inclui a opção de criar formulários fáceis de usar para facilitar a tarefa de inserir dados aos olhos do operador. Os trabalhadores têm o poder de adaptar formulários para atender as suas necessidades. O uso de modelos personalizáveis para formulários torna o processo de criação mais rápido e fácil.

Para ajudar aqueles que inserem dados no formulário, o uso de mensagens fornece orientação. Por exemplo, quando esses dados de entrada precisam escolher em um menu suspenso para um determinado campo, o criador do banco de dados tem a opção de adicionar uma mensagem *pop-up* quando esse campo é selecionado, lembrando os usuários de selecionar a partir do menu suspenso (SILBERSCHATZ, 2006).

2.2.1.3 Relatórios

Programas de banco de dados incluem funções de criação de relatórios. Os relatórios permitem que os usuários manipulem seus dados de várias maneiras. Os usuários podem inserir funções em relatórios para ajudar na análise dos dados. Por exemplo, um departamento de vendas pode criar um relatório que mostre apenas os dados do primeiro trimestre e fornece os totais para cada mês separadamente.

Para Elmasri et al. (2011) a criação de relatórios normalmente permite criatividade ao projetar a aparência do relatório. Aspectos visuais, como opções de fontes, localização de campos, inserção de gráficos e manipulação de cores permitem que o designer do relatório o torne visualmente agradável e profissional.

2.2.1.4 Funções de Dados

O software de banco de dados fornece aos usuários recursos para organizar suas informações de maneira simples e específica. Por exemplo, o uso de funções de classificação permite a sequência alfa ou reversa-alfa, geralmente clicando em um ícone. As funções de filtragem permitem que os usuários extraiam informações por critérios especificados.

Figura 4 – Funções de Dados.
Fonte: autoria própria.

2.3 Vantagens do Banco de Dados Relacional

As principais vantagens dos bancos de dados relacionais são permitir que os usuários categorizem e armazenem facilmente em dados que podem ser consultados e filtrados posteriormente para extrair informações específicas para relatórios.

Bancos de dados relacionais também são fáceis de estender e não dependem de organização física. Após a criação do banco de dados original, uma nova categoria de dados pode ser adicionada sem que todos os aplicativos existentes sejam modificados (CHANG et al, 2006).

Outras vantagens do banco de dados relacional incluem:

- Preciso: Os dados são armazenados apenas uma vez, o que elimina a de duplicação de dados.
- Flexível: Consultas complexas são fáceis para os usuários realizarem.
- Colaborativo: Vários usuários podem acessar o mesmo banco de dados.
- Confiável: Os modelos de banco de dados relacionais são maduros e bem compreendidos.
- Seguro: Os dados em tabelas nos sistemas de gerenciamento de banco de dados relacionais (RDBMSes) podem ser limitados para permitir o acesso apenas a usuários específicos.

2.4 Desvantagens do Banco de Dados Relacional

Para Gyorodi et al. (2015), os bancos de dados relacionais são amplamente utilizados em muitos setores para armazenar registros financeiros, acompanhar o inventário e manter registros dos funcionários. Em um banco de dados relacional, as informações são armazenadas em tabelas (geralmente chamadas relações), que ajudam a organizar e estruturar os dados. Embora sejam amplamente utilizados, os bancos de dados relacionais têm algumas desvantagens.

Uma alternativa para o banco de dados relacional é um sistema de banco de dados usando uma linguagem de programação orientada a objetos como o Java.

Justin James, da Tech Republic destacou várias desvantagens do sistema de banco de dados relacional. Desenvolvedores de banco de dados tendem a adicionar camadas ao banco de dados para novas funções e usam serviços da Web para acoplar camadas de dados.

Para Politowski et al. (2014), as desvantagens surgem das limitações da linguagem. De acordo com James, o atual banco de dados relacional força os desenvolvedores a recriar a lógica, encontrar soluções para incompatibilidades entre aplicativos e seu sistema de banco de dados, aperfeiçoar ou depurar o banco de dados existente por causa do tempo e dinheiro investidos e trabalhar com um sistema muito complexo.

2.4.1 Custo

Uma desvantagem dos bancos de dados relacionais é o desperdício de tempo em configurar e manter o sistema de banco de dados. Para configurar um banco de dados relacional, geralmente precisa adquirir um software especial.

Se não for implantado por um programador, poderá usar qualquer quantidade de produtos para configurar um banco de dados relacional. Leva tempo para inserir todas as informações e configurar o programa.

Se a empresa for grande e precisar de um banco de dados mais robusto, precisará contratar um programador para criar um banco de dados relacional usando SQL (*Structures Quert Language*) e um administrador de banco de dados para manter o banco de dados depois de compilados. Independentemente dos dados que usar, terá que importá-lo de outros dados, como arquivos de texto ou planilhas do Excel, ou ter os dados digitados no teclado, não importando o tamanho da empresa (MySQL, 2018).

2.4.2 Abundância de Informação

Politowski et al. (2014) e Gyorodi et al. (2015) concordam que os avanços na complexidade da informação causam outra desvantagem aos bancos de dados relacionais. Bancos de dados relacionais são feitos para organizar dados por características comuns. Imagens complexas, números, designs e produtos multimídia desafiam a fácil categorização, abrindo caminho para um novo tipo de

banco de dados chamado de sistemas de gerenciamento de banco de dados objeto-relacional. Esses sistemas são projetados para lidar com aplicativos mais complexos e tem a capacidade de serem escalonáveis.

2.4.3 *Limites Structured*

Alguns bancos de dados relacionais têm limites nos comprimentos de campo. Ao projetar o banco de dados, você precisa especificar a quantidade de dados que pode ser ajustada em um campo. Alguns nomes ou consultas de pesquisa são mais curtos do que os reais, e isso pode levar a perda de dados (MySQL, 2018).

2.4.4 Base de Dados Isolados

Sistemas de banco de dados relacionais complexos podem levar esses bancos de dados a se tornarem “ilhas de informação”, onde as informações não podem ser compartilhadas facilmente de um sistema grande para outro.

Muitas vezes, com grandes empresas ou instituições, você encontra bancos de dados relacionais crescidos em divisões separadas de maneira diferente. Por exemplo, talvez o departamento de faturamento do hospital usasse um banco de dados, enquanto o departamento de pessoal do hospital usava um banco de dados diferente. Obter esses bancos de dados para “conversar” uns com os outros pode ser um empreendimento grande e caro, mas em um sistema hospitalar complexo, todos os bancos de dados precisam ser envolvidos para bons cuidados com pacientes e funcionários.

2.5 Armazenamento em Nuvem

Embora nem todos concordem com a definição exata da computação em nuvem, a maioria concorda que a visão abrange uma mudança geral de processamento de computador, armazenamento e entrega de software longe dos servidores desktop e locais, em toda a rede e em *data centers* da próxima geração hospedados por grandes empresas de infraestrutura, como Amazon, Google, Yahoo, Microsoft ou Sun (AMAZON WEB SERVICES, 2018).

Assim como a rede elétrica revolucionou o acesso a eletricidade há cem anos, liberando as corporações de terem que gerar seu próprio poder e permitindo que elas se concentrassem em seus diferenciadores de negócios, a computação em nuvem é considerada revolucionária, liberando as corporações de grandes empresas de TI, investimentos de capital, e permitindo que elas se conectem a recursos de computação extremamente poderosos na rede (BARROS, 2008).

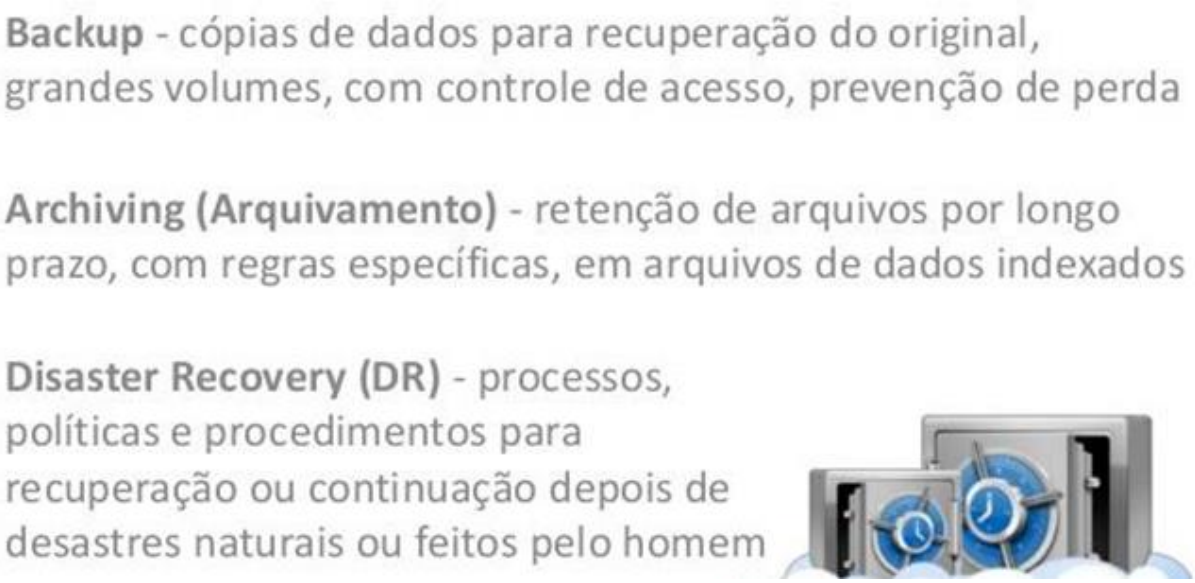

Figura 5 – Conceito-Chave, Armazenamento em Nuvem.
Fonte: AMAZON WEB SERVICES (2018).

2.5.1 Mudança de Paradigma no Armazenamento em Nuvem

Aplicativos de gerenciamento de dados são candidatos potenciais para implantação na nuvem. Isto é porque o sistema de banco de dados corporativo geralmente vem com um custo inicial alto, as vezes proibitivo, tanto em *hardware* quanto em *software* (AZURE, 2016).

Para muitas empresas (especialmente para *start-ups* e médias empresas), o *pay-as-you-go* que é o modelo de computação em nuvem, além de ter outra pessoa se preocupando em manter o hardware, tornando-se muito atraente. Desta forma, a computação em nuvem é reminiscência do provedor de serviços de aplicativos (ASP) e paradigmas de banco de dados como serviço (DaaS) (XPLENTY, 2018).

Na prática, as plataformas de computação em nuvem, como as oferecidas pela Amazon Web Services, Synaptic Hosting da AT & T, AppNexus, GoGrid, Rackspace Cloud Hosting e até certo ponto, O HP, Yahoo, Intel Cloud Computing

Testbed e a iniciativa de nuvem IBM e Google, funcionam de maneira diferente dos ASPs e DaaS. Em vez de possuir, instalar e manter o software de banco de dados para você (geralmente em uma arquitetura de multilocação), os fornecedores de computação em nuvem normalmente mantem pouco mais do que o *hardware* e oferecem aos clientes conjuntos de máquinas virtuais nas quais podem instalar seus próprios softwares (CHANG et al. 2006).

A disponibilidade de recursos é tipicamente elástica, com um poder de computação e armazenamento de quantidade aparentemente infinita disponível sob demanda, em um pagamento apenas pelo que você usa, criando um novo modelo de precificação.

Bancos de dados relacionais baseados em nuvem ou banco de dados como serviço (DBaaS), são amplamente usados porque permitem que as empresas terceirizem os requisitos de manutenção de banco de dados, correções e suporte a infraestrutura.

2.6 Banco de Dados em Nuvem

Segundo Barros (2008), o crescimento massivo de dados digitais, mudança de armazenamento de dados requisitados, melhores instalações de banda larga e menor custo de implementação e manutenção levou ao surgimento de bancos de dados em nuvem (*Cloud Storage*), dados como serviço (DaaS) e banco de dados como serviço (DBaaS) são os diferentes termos usados para gestão de dados na nuvem. Eles diferem como os dados são armazenados e gerenciados. Armazenamento em nuvem é um armazenamento virtual que permite aos usuários armazenar “N” dados.

Atualmente existem algumas empresas que prestam o serviço de armazenamento em nuvem, como Dropbox, iCloud, Google Cloud, entre outros que são serviços populares de armazenamento em nuvem. DaaS permite ao usuário armazenar dados em um disco remoto disponível através da Internet. É usado principalmente para *backup* e gerenciamento básico de dados.

Armazenamento na nuvem não pode funcionar sem serviços básicos de gerenciamento de dados. Assim, esses dois termos são usados de forma intercambiável. O DBaaS está “*one step*” adiante. Oferece funcionalidade completa de banco de dados e permite que os usuários acessem e armazenem seus bancos

de dados em discos a qualquer momento de qualquer lugar através da Internet. As empresas atualmente que prestam esses serviços são, Amazon’s SimpleDB, Amazon RDS, Big Table do Google, Yahoo, O Sherpa e o banco de dados SQL Azure da Microsoft são bancos de dados comumente usados na nuvem (AZURE, 2016).

3 CAPÍTULO 2 – BANCO DE DADOS NoSQL

Nesta seção serão abordados os conceitos de Banco de Dados NoSQL, tipos de armazenamento, bem como suas vantagens e desvantagens.

3.1 Banco de Dados NoSQL

Um NoSQL, originalmente referindo-se a “não SQL” ou “não relacional” de banco de dados, fornece um mecanismo de armazenamento e recuperação de dados, que é modelado em outros bancos de dados e tem relações tabulares utilizados em meios de banco de dados relacionais.

Esses bancos de dados existem desde o final da década de 1960, mas não obtiveram e apelido “NoSQL” até uma onda de popularidade no início do século XXI, desencadeada pelas necessidades das empresas da Web 2.0. Os bancos de dados NoSQL são cada vez mais usados em aplicativos Web de dados grandes e em tempo real. Os sistemas NoSQL também são as vezes chamados de “Não apenas SQL” para enfatizar que eles podem suportar linguagens de consulta do tipo SQL, ou podem se assentar ao lado do banco de dados SQL em uma arquitetura de persistência poliglota (CATTELL, 2010).

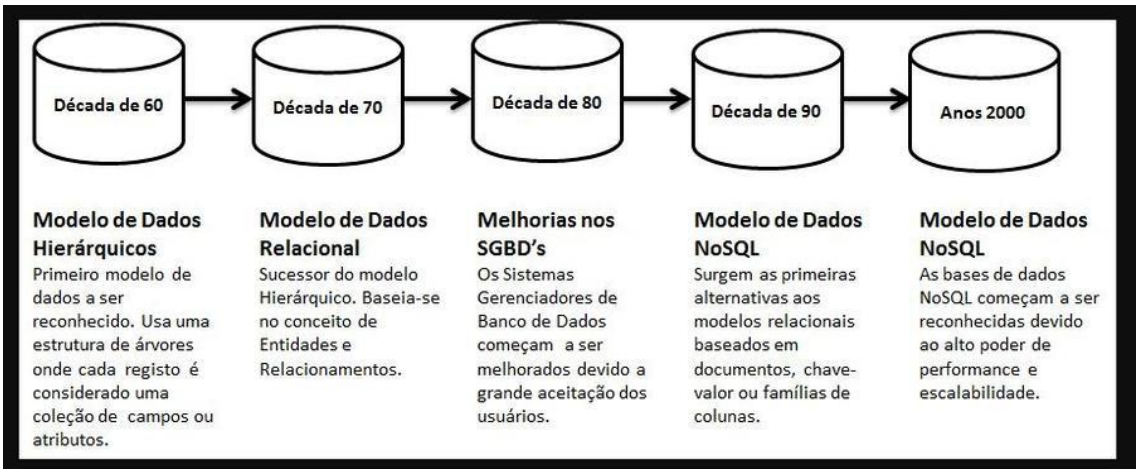

Figura 5 – Evolução do banco de Dados Relacional ao NoSQL.
Fonte: DEVMEDIA, 2018.

Clarence et al. (2012) diz que as motivações para essa abordagem incluem, simplicidade de design, escalonamento “horizontal” mais simples para clusters de máquinas (o que é um problema para bancos de dados relacionais) e controle mais

refinado sobre a disponibilidade. As estruturas de dados usadas pelos bancos de dados NoSQL (por exemplo, valor-chave, coluna larga, gráfico ou documento) são diferentes daquelas usadas por padrão nos bancos de dados relacionais, tornando algumas operações mais rápidas no NoSQL. A adequação particular de um determinado banco de dados NoSQL depende do problema que ele deve resolver. Às vezes, as estruturas de dados usadas pelos bancos de dados NoSQL também são vistas como “mais flexíveis” do que as tabelas de bancos de dados relacionais.

Segundo Cunha (2011) muitas lojas NoSQL comprometem a consistência (no sentido do teorema CAP) em favor da disponibilidade, da tolerância à partição e da velocidade. As barreiras para a maior adoção de lojas NoSQL incluem o uso de linguagens de consulta de baixo nível (em vez de SQL, por exemplo, a falta de capacidade de realizar junções ad-hoc em tabelas), falta de interfaces padronizadas e grandes investimentos anteriores em relacionamentos relacionais existentes nas bases de dados. A maioria das lojas NoSQL carece de verdadeiras transações ACID, embora algumas bases de dados, como MarkLogic, Aerospike, FairCom, c-TreeACE, Goole Spanner (embora tecnicamente um NesSQL banco de dados), Symas LMDB e OrientDB tê-los feito central para seus projetos.

Em vez disso, a maioria dos bancos de dados NoSQL oferece um conceito de “consistência eventual” em que as alterações do banco de dados são propagadas para todos os nós “eventualmente” (geralmente em milissegundos), portanto, as consultas de dados podem não retornar dados atualizados imediatamente ou podem resultar na leitura de dados não precisos, um problema conhecido como leitura obsoleta.

Além disso, alguns sistemas NoSQL podem exibir gravações perdidas e outras formas de perda de dados. Alguns sistemas NoSQL fornecem conceitos como o log *write-ahead* para evitar a perda de dados. Para processamento de transações distribuídas em vários bancos de dados, a consistência de dados é um desafio ainda maior que é difícil para bancos de dados relacionais e NoSQL. Mesmo os bancos de dados relacionais atuais “não permitem que restrições de integridade referencial abranjam os bancos de dados”. Existem poucos sistemas que mantêm transações ACID e Padrões X/Open XA para processamento de transações distribuídas (SHREINER et al. 2015).

Um banco de dados NoSQL é uma alternativa aos bancos de dados relacionais que é especialmente útil para trabalhar com grandes conjuntos de dados

distribuídos. Esses bancos de dados podem suportar uma variedade de modelos de dados, incluindo formatos de valor-chave, documentos, colunas e gráficos.

Evans (2009) explica que um banco de dados gráfico se expande além dos modelos de dados relacionais baseados em colunas e linhas tradicionais; esse banco de dados NoSQL usa nós e arestas que representam conexões entre relacionamentos de dados e podem descobrir novos relacionamentos entre os dados. Os bancos de dados de gráficos são mais sofisticados do que os bancos de dados relacionais e, portanto, seus usos incluem detecção de fraudes ou mecanismos de recomendação da Web.

3.2 Tipos de Armazenamento de Dados NoSQL

3.2.1 Armazenamento de Valor Chave

No tipo de armazenamento *Key Value*, uma tabela de *hash* é usada, na qual uma chave exclusiva aponta para um item. As chaves podem ser organizadas em grupos lógicos de chaves, exigindo apenas que as chaves sejam exclusivas dentro de seu próprio grupo. Isso permite chaves idênticas em diferentes grupos lógicos (DEVMEDIA, 2018).

A tabela a seguir mostra um exemplo de um armazenamento de valor-chave em que a chave é o nome da cidade e o valor é o endereço da Universidade de Ulster nessa cidade. Algumas implementações do armazenamento de valores-chave fornecem mecanismos de armazenamento em cache, que melhoram muito seu desempenho (JSON, 2018).

Tudo o que é necessário para lidar com os itens armazenados no banco de dados é a chave. Os dados são armazenados na forma de uma *string*, JSON ou BLOB (*Binary Large Object)* (JSON, 2018).

Uma das maiores falhas nessa forma de banco de dados é a falta de consistência no nível do banco de dados. Isso pode ser adicionado pelos desenvolvedores com seu próprio código, mas isso adiciona mais esforço, complexidade e tempo. O banco de dados NoSQL mais famoso que é construído em um armazenamento de valor chave é o DynamoDB da Amazon (DEVMEDIA, 2018).

3.2.2 Document Store

Lennon (2013) explica que os armazenamentos de documentos são semelhantes aos principais armazenamentos de valores, pois são sem esquema e baseados em um modelo de valor-chave. Ambos, portanto, compartilham muitas das mesmas vantagens e desvantagens. Ambos não têm consistência no nível do banco de dados, o que abre caminho para que os aplicativos forneçam mais recursos de confiabilidade e consistência. Existem, no entanto, diferenças fundamentais entre os dois.

Em *Document Stores*, os valores (documentos) fornecem codificação para os dados armazenados. Essas codificações podem ser XML, JSON ou BSON (JSON codificado em binário). Além disso, a consulta baseada em dados pode ser feita. O aplicativo de banco de dados mais popular que se baseia em um armazenamento de documentos é o MongoDB.

3.2.3 Loja de Colunas

Em um banco de dados de Armazenamento de Colunas, os dados são armazenados em colunas, ao contrário de serem armazenados em linhas, como é feito na maioria dos sistemas de gerenciamento de banco de dados relacional.

Um Armazenamento de Colunas é composto por uma ou mais Famílias de Colunas que agrupam logicamente determinadas colunas no banco de dados. Uma chave é usada para identificar e apontar para um número de colunas no banco de dados, com um atributo *keyspace* que define o escopo dessa chave. Cada coluna contém tuplas de nomes e valores, ordenadas e separadas por virgulas (POPESCU, 2015).

Os Armazenamentos de Colunas têm acesso rápido de leitura/gravação aos dados armazenados. Em um armazenamento de coluna, as linhas que correspondem a uma única coluna são armazenadas como uma única entrada de disco. Isso possibilita um acesso mais rápido durante as operações de leitura/gravação. Os bancos de dados mais populares que usam o armazenamento de colunas incluem o Bigtable, o Base e o Cassandra do Google (TAMANE, 2016).

3.2.4 Base de Gráfico

Sandalage e Fowler (2013) explica que um banco de dados NoSQL com base em gráfico, uma estrutura de gráfico direcionada é usada para representar os dados. O gráfico é composto de arestas e nós. Formalmente, um gráfico é uma representação de um conjunto de objetos, onde alguns pares de objetos são conectados por links. Os objetos interconectados são representados por abstrações matemáticas, chamados vértices e os links que conectam alguns pares de vértices são chamados de arestas. Um conjunto de vértices e as arestas que os conectam é considerado um gráfico.

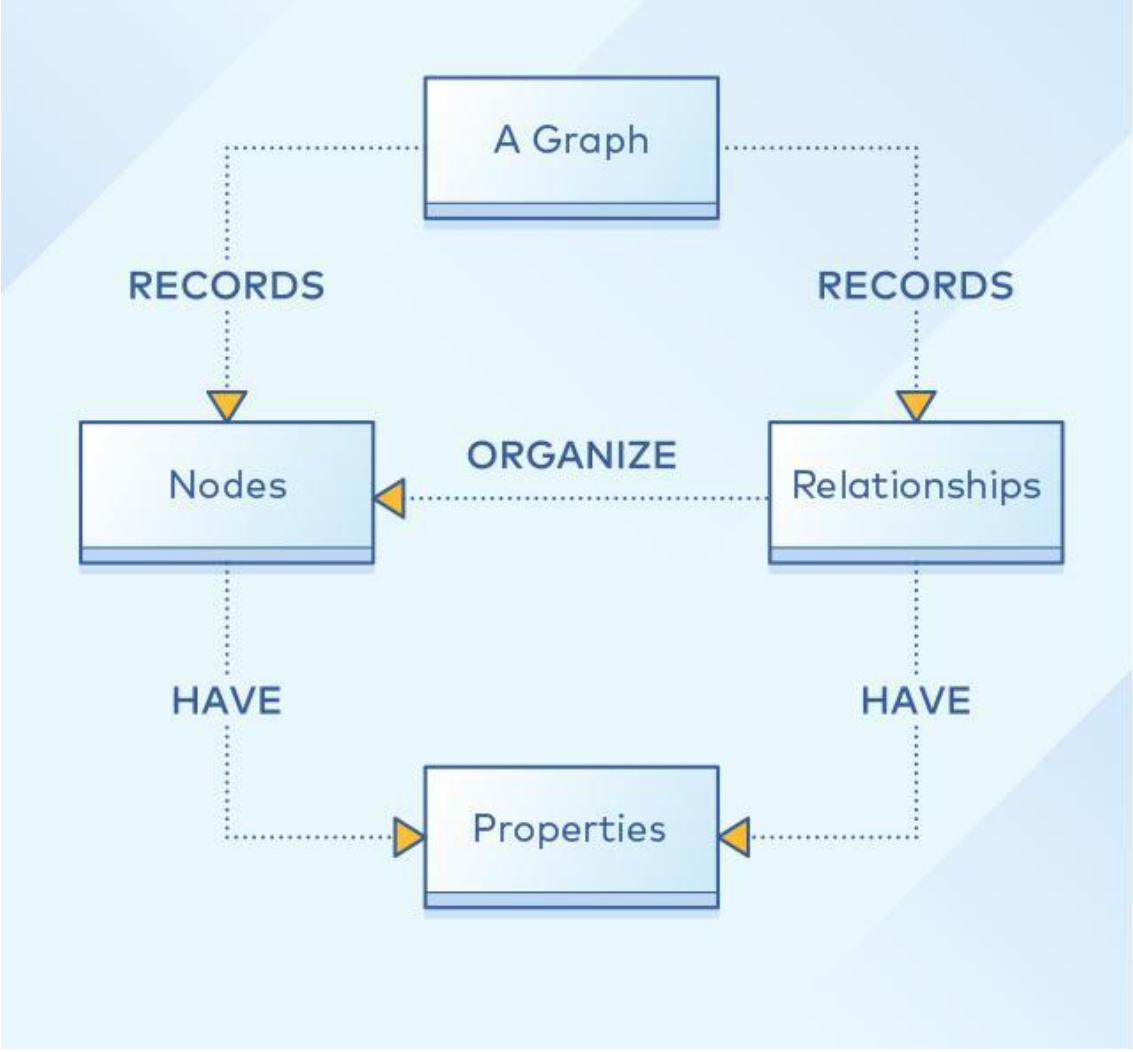

Figura 6 – Nós, arestas e relação entre objetos.
Fonte: ALTARADE, 2018.

Isso ilustra a estrutura de um banco de dados de base de gráfico que usa arestas e nós para representar e armazenar dados. Esses nós são organizados por alguns relacionamentos uns com os outros, o que é representado por arestas entre os nós. Ambos os nós e o relacionamentos possuem algumas propriedades definidas.

Bancos de dados de gráficos são normalmente usados em aplicativos de redes sociais. Os bancos de dados gráficos permitem que os desenvolvedores se concentrem mais nas relações entre os objetos do que nos próprios objetos. Neste contexto, eles de fato permitem um ambiente escalável e fácil de usar. Atualmente, o InfoGrid e o InfiniteGraph são os bancos de dados gráficos mais populares (SUISSA, 2010).

3.2.5 Manipulando Dados Relacionais

Como a maioria dos bancos de dados NoSQL não tem capacidade para junções em consultas, o esquema do banco de dados geralmente precisa ser projetado de forma diferente. Existem três técnicas principais para manipular dados relacionais em um banco de dados NoSQL.

3.2.5.1 Múltiplas Consultas

Em vez de recuperar todos os dados com uma consulta, é comum fazer várias consultas para obter os dados desejados. As consultas do NoSQL geralmente são mais rápidas que as consultas SQL tradicionais, portanto, o custo de fazer consultas adicionais pode ser aceitável. Se um número excessivo de consultas for necessário, uma das outras duas abordagens é mais apropriada.

3.2.5.2 Cache, Replicação e Dados Não Normalizados

Em vez de apenas armazenar chaves estrangeiras, é comum armazenar valores estrangeiros reais junto com os dados do modelo. Por exemplo, cada comentário de blog pode incluir o nome de usuário, além de um ID de usuário, fornecendo acesso fácil ao nome de usuário sem exigir outra consulta. Quando um nome de usuário é alterado, isso precisará ser alterado em muitos lugares no banco

de dados. Assim, essa abordagem funciona melhor quando as leituras são muito mais comuns do que as escritas.

3.2.5.3 Alinhando Dados

Com bancos de dados de documentos como o MongoDB, é comum colocar mais dados em um número menor de coleções. Por exemplo, em um aplicativo de blog, pode-se optar por armazenar comentários no documento de postagem do blog para que, com uma única recuperação, todos os comentários sejam obtidos. Assim, nessa abordagem, um único documento contém todos os dados necessários para uma tarefa específica.

3.3 Vantagens do Banco de Dados NoSQL

Srivastava et al. (2015) referem que hoje, as vantagens dos bancos de dados NoSQL não são um segredo, especialmente quando a computação em nuvem ganhou ampla adoção. Os bancos de dados NoSQL foram criados em resposta as limitações da tecnologia de banco de dados relacional tradicional. Quando comparados com os bancos de dados relacionais, os bancos de dados NoSQL são mais escaláveis e fornecem desempenho superior e seu modelo de dados aborda várias deficiências do modelo relacional.

As vantagens do NoSQL incluem poder para manipular:

- Grandes volumes de dados estruturados, semiestruturados e não estruturados;
- *Sprints* ágeis, interação rápida e *push* de código frequente;
- Programação orientada a objetos, fácil de usar e flexível;
- Arquitetura eficiente de *scale-out* em vez de arquitetura monolítica e cara;

Atualmente, as empresas utilizam bancos de dados NoSQL para um número crescente de casos de uso. Os bancos de dados NoSQL também tendem a ser de código aberto e isso significa uma maneira relativamente barata de desenvolver, implementar e compartilhar *software* (LEAVITT, 2010).

Segundo Shreiner et al. (2015), as empresas escolhem o MongoDB para desenvolver aplicativos modernos, pois oferecem as vantagens dos bancos de dados relacionais, juntamente com as inovações do NoSQL.

3.4 Desvantagens do Banco de Dados NoSQL

Gyorodi et al. (2015) explica que a maioria dos bancos de dados NoSQL não suporta recursos de confiabilidade que são suportados nativamente por sistemas de bancos de dados relacionais. Esses recursos de confiabilidade podem ser resumidos como atomicidade, consistência, isolamento e durabilidade. Isso também significa que os bancos de dados NoSQL, que não suportam esses recursos, trocam consistência por desempenho e escalabilidade.

Na continuação de Gyorodi et al. (2015), para suportar recursos de confiabilidade e consistência, os desenvolvedores devem implementar seu próprio código proprietário, o que adiciona mais complexidade ao sistema. Shreiner et al. (2015) diz que isso pode limitar o número de aplicativos que podem confiar nos bancos de dados NoSQL para transações seguras e confiáveis, como sistemas bancários.

Chang et al. (2016) diz que outras formas de complexidade encontradas na maioria dos bancos de dados NoSQL incluem incompatibilidade com consultas SQL. Isso significa que uma linguagem de consulta manual ou proprietária é necessária, adicionando ainda mais tempo e complexidade.

3.5 Sistemas de Gerenciamento de Banco de Dados NoSQL

Para uma breve comparação dos bancos de dados, a Tabela 1 fornece uma breve comparação entre diferentes sistemas de gerenciamento de banco de dados NoSQL.

Tabela 1 – Sistemas de Gerenciamento de Banco de Dados NoSQL.

	Storage Type	Query Method	Interface	Programming Language	Open Source	Replication
Cassandra	Column Store	Thrift API	Thrift	Java	Yes	Async
MongoDB	Document Store	Mongo Query	TCP/IP	C++	Yes	Async
HyperTable	Column Store	HQL	Thrift	Java	Yes	Async
CouchDB	Document Store	MapReduce	REST	Erlang	Yes	Async
BigTable	Column Store	MapReduce	TCP/IP	C++	No	Async
HBase	Column Store	MapReduce	REST	Java	Yes	Async

Fonte: ALTARADE, 2018.

O MongoDB possui um armazenamento de esquema flexível, o que significa que os objetos armazenados não são necessariamente obrigados a ter a mesma estrutura ou campos. O MongoDB também possui alguns recursos de otimização, que distribuem as coleções de dados, resultando em melhoria geral de desempenho e um sistema mais equilibrado (SILBERSCHATZ, 2006).

Outros sistemas de banco de dados NoSQL, como o Apache CouchDB também são do tipo banco de dados de armazenamento de documentos e compartilham muitos recursos com o MongoDB, com exceção de que o banco de dados pode ser acessado usando APIs RESTful (DEVMEDIA, 2018).

REST é um estilo arquitetural que consiste em conjunto coordenado de restrições arquiteturais aplicadas a componentes, conectores e elementos de dados, dentro da *World Wide Web.* Ele se baseia em um protocolo de comunicações que pode ser armazenado em estado, servidor-cliente e armazenável em cache (por exemplo, o protocolo HTTP).

Os aplicativos RESTful usam solicitações HTTP para publicar, ler dados e excluir dados. Quanto aos bancos de dados de base de coluna, o Hypertable é um banco de dados NoSQL escrito em C ++ e baseado no BigTable do Google.

O Hypertable suporta a distribuição de armazenamentos de dados entre nós para maximizar a escalabilidade, assim como o MongoDB e o CouchDB. Um dos bancos de dados NoSQL mais utilizados é o Cassandra, desenvolvido pelo Facebook.

O Cassandra é um banco de dados de armazenamento de colunas que inclui muitos recursos voltados para confiabilidade e tolerância a falhas. Em vez de fornecer uma visão detalhada de cada NoSQL DBMS, o Cassandra e o MongoDB, dois dos sistemas de gerenciamento de banco de dados NoSQL mais utilizados, serão explorados nas próximas subseções (CASSANDRA, 2016).

4 CAPÍTULO 3 – BANCO DE DADOS NoSQL VS. RELACIONAIS

No Capítulo 3 serão apresentados dados comparativos entre os bancos de dados NoSQL e Relacionais.

4.1 Comparativo entre Banco de Dados NoSQL e Relacionais

Esta Tabela 2 fornece uma breve comparação de recursos entre NoSQL e bancos de dados relacionais:

Tabela 2 – Banco de dados NoSQL vs. Relacionais.

Característica	Bancos de dados NoSQL	Bancos de dados relacionais
atuação	Alto	Baixo
Confiabilidade	Pobre	Boa
Disponibilidade	Boa	Boa
Consistência	Pobre	Boa
Armazenamento de dados	Otimizado para dados enormes	De tamanho médio a grande
Escalabilidade	Alto	Alta (mas mais cara)

Fonte: ALTARADE, 2018.

Deve-se observar que a tabela 2 mostra uma comparação no nível do banco de dados, não nos vários sistemas de gerenciamento de banco de dados que implementam os dois modelos. Esses sistemas fornecem suas próprias técnicas proprietárias para superar alguns dos problemas e deficiências em ambos os sistemas e, em alguns casos, melhoram significativamente o desempenho e a confiabilidade.

Começando com alguns conceitos-chaves de bancos de dados relacionais e NoSQL. Abaixo está um banco de dados estruturado em gráfico para relacionamentos humanos. No diagrama, (a) mostra uma estrutura sem esquema e

(b) mostra como ela pode ser estendida para um esquema estruturado normal (GYORODI et al. 2015).

Sem esquema significa que dois documentos em uma estrutura de dados NoSQL não precisam de campos comuns e podem armazenar diferentes tipos de dados.

{\color{000000}var cars = [
{ Model: "BMW", Color: "Red", Manufactured: 2016 },
{ Model: "Mercedes", Type: "Coupe", Color: "Black", Manufactured: “1-1-2017” }
]; }

Em um mundo relacional, você precisa armazenar dados em uma estrutura definida a partir da qual você pode recuperar dados. Por exemplo (usando um operador JOIN entre duas tabelas):

{\color{000000}SELECT Orders.OrderID, Customers.Name, Orders. Date
FROM Orders
INNER JOIN Customers
ON Orders.CustID = Customers.CustID}

Como um tópico mais avançado e uma demonstração de quando o SQL é um candidato melhor que o NoSQL, usarei o algoritmo de compactação rápida. Este algoritmo NoSQL recentemente proposto mostra que é difícil lidar com a geração continua de tabelas de strings ordenadas (chamadas stables).

Essas tabelas são sequencias de valores-chave classificadas por chave. Sua geração, ao longo do tempo, faz com que a operação de leitura crie um gargalo de E/S de disco, e as leituras ficam mais lentas do que as gravadas em bancos de dados NoSQL, negando assim uma das principais vantagens dos bancos de dados NoSQL. Em uma tentativa de reduzir esse efeito, os sistemas NoSQL executam os protocolos de compactação em segundo plano, tentando mesclar várias tabelas em uma única tabela. No entanto, essa fusão é muito intensiva em recursos (GYORODI et al. 2015).

4.1.1 Linguagem

Pense em uma cidade – vamos chamar de Cidade A – onde todos falam a mesma língua. Todos os negócios são construídos em torno dele, toda forma de comunicação usa – em suma, é a única maneira que os moradores entendem e interagem com o mundo ao seu redor. Mudar essa linguagem em um só lugar seria confuso e perturbador para todos (DEVMEDIA, 2018).

Agora, pense em outra cidade, a Cidade B, onde cada lar pode falar uma língua diferente. Todos interagem com o mundo de maneira diferente e não há um entendimento “universal” ou uma organização definida. Se uma casa é diferente, isso não afeta ninguém. Isso ajuda a ilustrar uma das diferenças fundamentais entre os bancos de dados não relacionais NoSQL e relacionais de SQL e essa distinção tem grandes implicações.

4.1.1.1 O Banco de Dados Relacional (SQL)

Sadalage et al. (2013) explica que os bancos de dados relacionais usam a linguagem de consulta estruturada (SQL) para definir e manipular dados. Por um lado, isso é extremamente poderoso: o SQL é uma das opções mais versáteis e amplamente utilizadas, tornando-o uma opção segura e especialmente ótima para consultas complexas.

Politowsk (2014) indaga, por outro lado, pode ser restritivo. O SQL exige que use esquemas predefinidos para determinar a estrutura de seus dados antes de trabalhar com eles. Além disso, todos os dados devem seguir a mesma estrutura. Isso pode exigir uma preparação inicial significativa e, como na Cidade A, pode significar que uma mudança na estrutura seria difícil e prejudicial para todo o sistema.

4.1.1.2 O Banco de Dados NoSQL

Um banco de dados NoSQL, por outro lado, possui um esquema dinâmico para dados não estruturados e os dados são armazenados de várias maneiras: podem ser orientados por colunas, orientados a documentos, baseados em gráficos ou organizados como um armazenamento *KeyValue*. Essa flexibilidade significa que:

- Pode criar documentos sem precisar primeiro definir sua estrutura;
- Cada documento pode ter sua própria estrutura única;
- A sintaxe pode variar de banco de dados para banco de dados e pode adicionar campos conforme avança;

4.1.2 Escalabilidade

Na maioria das situações, os bancos de dados SQL são escalonáveis verticalmente, o que significa que você pode aumentar a carga em um único servidor aumentando as coisas como CPU, RAM ou SSD.

Banco de dados NoSQL, por outro lado, são escalonáveis horizontalmente. Isso significa que você manipula mais trafego fragmentado ou adicionando mais servidores ao seu banco de dados NoSQL. É como adicionar mais andares ao mesmo prédio do que adicionar mais prédios ao bairro. O último pode se tornar maior e mais poderoso, tornando os bancos de dados NoSQL a escolha preferida para conjuntos de dados grandes ou em constante mudança (EVANS, 2009).

Sendo uma das principais diferenças entre NoSQL e SQL é que os bancos de dados NoSQL são considerados mais escaláveis que os bancos de dados SQL. O MongoDB, por exemplo, tem suporte embutido para replicação e particionamento (particionamento horizontal de dados) para suportar escalabilidade. Embora recursos, até certo ponto, estejam disponíveis em bancos de dados SQL, eles exigem um investimento significativo de recursos humanos e de hardware.

Para uma comparação detalhada das duas opções, pode-se referenciar a classificação proposta pela Cattell para os armazenamentos de dados. Este relatório está resumindo abaixo: O teste foi realizado usando três parâmetros principais: simultaneidade, armazenamento de dados e replicação. A simultaneidade foi avaliada analisando o mecanismo de bloqueio de dados, o controle de simultaneidade de múltiplas versões e o ACID. O armazenamento de dados foi testado em armazenamento físico e em modos de memória e a replicação foi testada em seu suporte a modos síncronos ou assíncronos (CATTELL, 2010).

Usando dados recuperados desses testes, os autores concluíram que os produtos SQL com capacidade de cluster mostraram desempenho promissor por nó

e também a capacidade de escalabilidade, dando vantagem aos sistemas RDBMs sobre NoSQL, devido a sua total conformidade com ACID.

4.1.3 Estrutura

Bancos de dados da Estrutura SQL são baseados em tabela, enquanto os bancos de dados NoSQL são baseados em documentos, pares de valores-chave, bancos de dados de gráficos ou armazenamentos de colunas amplas. Isso torna os bancos de dados relacionais SQL uma opção melhor para aplicativos que exigem transações de várias linhas – como um sistema de contabilidade – ou para sistemas legados que foram criados para uma estrutura relacional (GYORODI et al. 2015).

Alguns exemplos de bancos de dados SQL incluem MySQL, Oracle, PostgreSQL e Microsoft SQL Server. Exemplos de bancos de daos NoSQL incluem MongoDB, BigTable, Redis, Cassandra RavenDB, HBase, Neo4j e CouchDB.

Gunter et al. (2013) diz que agora que as principais diferenças estruturais entre os bancos de dados SQL e NoSQL foram divididas, vamos nos aprofundar nas principais diferenças funcionais entre os dois, olhando especificamente para o MySQL e o MongoDB como exemplos.

4.1.4 Indexação

Nos sistemas RDBMS, os índices são usados para acelerar as operações de recuperação de dados. Um índice ausente significa que uma tabela precisará ser completamente verificada para atender a uma consulta de leitura.

Em SQL e NoSQL, os índices de banco de dados servem ao mesmo proposito, recuperação mais rápida e otimizada de dados. Mas, como isso é diferente, devido a diferentes arquiteturas de banco de dados e diferenças na forma como os dados são armazenados no mecanismo de banco de dados (DEVMEDIA, 2018).

Enquanto em índices SQL estão em forma de *B-Trees* que mostram estrutura hierárquica de dados relacionais, em bancos de dados NoSQL eles apontam para documentos ou partes de documentos que, em geral, não têm nenhuma relação entre eles.

4.1.5 Aplicativo de CRM (Customer Relationship Management)

Os aplicativos de CRM são um dos melhores exemplos de ambientes de *big data*, com enormes volumes de dados diários e números de transações. Todos os fornecedores desses aplicativos usam SQL e NoSQL e, embora os dados transacionais ainda sejam armazenados principalmente em bancos de dados SQL, com aprimoramentos de serviços DBaaS (serviço de banco de dados) publicamente disponíveis, como AWS DynamoDB e Azure DocumentDB, muito mais processamento de dados poderia passar para o mundo NoSQL rodando nas nuvens (LAZZARETTI, 2013).

Embora esses serviços de gerenciamento removam desafios de segurança e acesso técnico, também é uma área em que os bancos de dados NoSQL são usados para o que são feitos principalmente – armazenamento analítico e mineração de dados. A quantidade de dados armazenados em enormes CRMs em empresas de telecomunicações ou finanças seria quase impossível de analisar usando ferramentas de mineração de dados, como SAS ou R, porque a demanda de recursos de hardware em um mundo transacional seria enorme.

Dados não estruturados, semelhantes a documentos, que constituem a entrada para modelos estatísticos, que então dão as empresas a capacidade de fazer análises de rotatividade ou de marketing, são o principal benefício desses sistemas. O CRM também é um dos melhores exemplos em que esses dois sistemas não são concorrentes, mas existem em harmonia, cada um desempenhando seu papel na maior arquitetura de dados (POPESCU, 2015).

4.2 MySQL: O Banco de Dados Relacional SQL

Alguns benefícios e pontos fortes do MySQL:

- Maturidade: O MySQL é um banco de dados extremamente estabelecido, o que significa que há uma enorme comunidade, testes extensivos e bastante estabilidade.
- Compatibilidade: O MySQL está disponível para todas as principais plataformas, incluindo Linux, Windows, Mac, BSD e Solaris. Ele também possui conectores para linguagens como Node.js, Ruby, C#, C++, Java,

Perl, Python e PHP, o que significa que não está limitado a linguagem de consulta SQL.

- Custo-benefício: O banco de dados é de código aberto e gratuito.
- Replicável: O banco de dados MySQL pode ser replicado em vários nós, o que significa que a carga de trabalho pode ser reduzida e a escalabilidade e a disponibilidade do aplicativo podem ser aumentadas.
- Sharding: Embora o sharding não possa ser feito na maioria dos bancos de dados SQL, isso pode ser feito em servidores MySQL. Isso é rentável e bom para os negócios.

4.3 MongoDB: O Banco de Dados não Relacional NoSQL

Alguns benefícios e pontos fortes do MongoDB:

- Esquema dinâmico: Conforme mencionado, isso proporciona flexibilidade para alterar o esquema de dados sem modificar nenhum dado existente.
- Escalabilidade: O MongoDB é dimensionável horizontalmente, o que ajuda a reduzir a carga de trabalho e dimensionar seus negócios com facilidade.
- Capacidade de gerenciamento: O banco de dados não requer um administrador de banco de dados. Como é bastante amigável para o usuário dessa maneira, ele pode ser usado por desenvolvedores e administradores.
- Velocidade: É de alto desempenho para consultas simples.
- Flexibilidade: Você pode adicionar novas colunas ou campos no MongoDB sem afetar as linhas existentes ou o desempenho do aplicativo.

4.4 Razões para usar Banco de Dados SQL

Nem todo banco de dados atende a todas as necessidades de negócios. É por isso que muitas empresas dependem de bancos de dados relacionais e não relacionais para diferentes tarefas (MySQL, 2018).

Embora os bancos de dados NoSQL tenham ganhado popularidade por sua velocidade e escalabilidade, ainda existem situações em que um banco de dados SQL altamente estruturado pode ser preferível. Dois motivos pelos quais pode-se considerar um banco de dados SQL são:

1) Precisando-se de conformidade com o ACID (Atomicidade, Consistência, Isolamento, Durabilidade). A conformidade com o ACID reduz anomalias e protege a integridade do seu banco de dados. Ele faz isso definindo exatamente como as transações interagem com o banco de dados, o que não é o caso dos bancos de dados NoSQL, que têm um objetivo principal de flexibilidade e velocidade, em vez de 100% de integridade de dados.
2) Seus dados são estruturados e imutáveis. Se o seu negócio não está crescendo exponencialmente, pode não haver razão para usar um sistema projetado para suportar uma variedade de tipos de dados e alto volume de trafego.

Indicadores para projetos em que o SQL é ideal:

- Requisitos de dados discretos relacionados a lógica que podem ser identificados antecipadamente.
- Integridade de dados é essencial.
- Tecnologia comprovada baseada em padrões, com boa experiência de desenvolvedor e suporte.

4.5 Razões Para Usar Um Banco de Dados NoSQL

Para evitar que o banco de dados se torne um gargalo no sistema, especialmente em ambientes de alto volume, os bancos de dados NoSQL funcionam de maneira que os bancos de dados relacionais não podem (MySQL, 2018).

Os seguintes recursos estão impulsionando a popularidade dos bancos de dados NoSQL, como MongoDB.

1) Armazenando grandes volumes de dados sem estrutura. Um banco de dados NoSQL não limita os tipos de dados armazenáveis. Além disso, pode adicionar novos tipos à medida que as necessidades de negócios mudam
2) Usando computação em nuvem e armazenamento. O armazenamento baseado em nuvem é uma ótima solução, mas requer que os dados sejam distribuídos facilmente em vários servidores para escalonamento. Usar hardware acessível no local para testes e, em seguida, para produção na nuvem é o objetivo do banco de dados NoSQL.
3) Desenvolvimento rápido, se estiver desenvolvendo usando metodologias ágeis modernas, um banco de dados relacional irá atrasa-lo. Um banco de dados NoSQL não requer o nível de preparação normalmente necessário para bancos de dados relacionais.

Indicadores para projetos em que o NoSQL é ideal:

- Requisitos de dados não relacionais, indeterminados ou em evolução.
- Objetivos de projeto mais simples ou mais frouxos, capazes de iniciar a codificação imediatamente.
- Velocidade e escalabilidade são imperativas.

4.6 Implantação de Banco de Dados em Nuvem

Um banco de dados em nuvem é uma coleção de conteúdo, estruturado ou não estruturado, que reside em uma plataforma de infraestrutura de computação em

nuvem privada, publica ou hibrida. Existem dois modelos de ambiente de banco de dados em nuvem: tradicional e banco de dados como um serviço (DBaaS) (XPLENTY, 2018).

Em um modelo de nuvem tradicional, um banco de dados é executado na infraestrutura de um departamento de TI por meio de uma máquina virtual. Tarefas de supervisão e gerenciamento de banco de dados que recaem sobre os funcionários de TI da organização.

4.6.1 O que é Banco de Dados com um Serviço (DBaaS)

Politowski et al. (2014), por comparação, o modelo DBaaS é um serviço de assinatura baseado em taxas, no qual o banco de dados é executado na infraestrutura física do provedor de serviços. Diferentes níveis de serviço geralmente estão disponíveis. Em um arranjo DBaaS clássico, o provedor mantém a infraestrutura física e o banco de dados, deixando o cliente gerenciar o conteúdo e a operação do banco de dados.

Como alternativa, um cliente pode configurar um acordo de hospedagem gerenciada, no qual o provedor lida com manutenção e gerenciamento do banco de dados. Essa última opção pode ser especialmente atraente para as pequenas empresas que têm necessidades de banco de dados, mas não possuem o conhecimento em TI (XPLENTY, 2018).

4.6.2 Benefícios do Banco de Dados em Nuvem

Segundo Winmanm (2012) em comparação com a operação de um banco de dados tradicional em um servidor físico e arquitetura de armazenamento no local, um banco de dados em nuvem oferece as seguintes vantagens distintas:

- Eliminação de infraestrutura física: Em um ambiente de banco de dados em nuvem, o provedor de servidores, armazenamento e outras infraestruturas de computação em nuvem são responsáveis pela manutenção e disponibilidade. A organização que possui e opera o banco de dados é responsável apenas pelo suporte e manutenção do software de banco de dados e seu conteúdo. Em um ambiente DBaaS, o

provedor de serviços é responsável por manter e operar o software de banco de dados, deixando os usuários do DBaaS responsáveis apenas por seus próprios dados.

- Redução de custos: Através da eliminação de uma infraestrutura física pertencente e operada por um departamento de TI, economias, significativas podem ser obtidas com gastos de capital reduzidos menos funcionários, custos operacionais e de eletricidade e HVAC diminuídos e uma quantidade menor de espaço físico necessário.

4.6.3 Benefício do DBaaS

Para Vogels (2009) além dos benefícios de empregar um modelo de ambiente de banco de dados em nuvem, a contratação com um provedor DBaaS oferece benefícios adicionais:

- Escalabilidade instantânea: Se a capacidade do banco de dados for necessária devido a picos comerciais sazonais ou picos inesperados na demanda, um provedor de DBaaS pode oferecer rapidamente capacidade adicional de taxa, taxa de transferência e largura de banda de acesso por meio de sua própria infraestrutura. Um banco de dados operando em uma infraestrutura tradicional no local provavelmente precisaria esperar semanas ou meses pela aquisição e instalação de recursos adicionais de servidor, armazenamento ou comunicações.
- Garantias de desempenho: Por meio de um acordo de nível de serviço (SLA), um provedor DBaaS pode ser obrigado a fornecer garantias que normalmente quantificam a disponibilidade mínima de tempo de atividade e os tempos de resposta das transações. Um SLA especifica remédios monetários e jurídicos se esses limites de desempenho não forem atingidos.
- Conhecimentos especializados: Em um ambiente de TI corporativo, com exceção das maiores empresas multinacionais, encontrar especialistas em banco de dados de classe mundial pode ser difícil e mantê-los na equipe pode ter um custo proibitivo. Em um ambiente DBaaS o provedor

pode atender milhares de clientes; assim, encontrar, ofertas e manter talentos de classe mundial é um desafio menor.

- A mais recente tecnologia: Para permanecerem competitivos, os provedores de DBaaS trabalham duro para garantir que todos os softwares de banco de dados, sistemas operacionais de servidor e outros aspectos da infraestrutura geral sejam mantidos atualizados com atualizações de segurança e recursos regularmente emitidas pelos fornecedores de software.
- Suporte a *failover*: Para que um provedor de serviços de banco de dados atenda as garantias de desempenho e disponibilidade, cabe a esse provedor garantir operação ininterrupta caso o data center principal falhe por qualquer motivo. O suporte a *failover* geralmente abrange a operação de vários servidores de imagem espelhada e recursos de armazenamento de dados. Manipulando corretamente, o *failover* para um data center de backup deve ser imperceptível para qualquer cliente desse serviço.
- Preços decrescentes: Com os avanços na tecnologia e um mercado intensamente competitivo entre os principais provedores de serviços, o preço de uma ampla gama de serviços de computação em nuvem sofre uma recalibração continua. A queda dos preços é um grande impulso para migrar bancos de dados no local e outras infraestruturas de TI para a nuvem.

4.6.4 Arquitetura de Banco de Dados em Nuvem

Os bancos de dados em nuvem, como seus ancestrais tradicionais, podem ser divididos em duas grandes categorias: relacional e não relacional. Um banco de dados relacional, geralmente escrito em linguagem de consulta estruturada (SQL), é composto de um conjunto de tabelas inter-relacionadas organizadas em linhas e colunas. O relacionamento entre tabelas e colunas (campos) é especificado em um esquema. Os bancos de dados SQL, por design, dependem de dados altamente consistentes em seu formato, como transações bancarias ou uma lista telefônica. Escolhas populares incluem MySQL, Oracle, IBM DB2 e Microsoft SQL Server (BUYYA et al. 2008).

Barros (2008) diz que bancos de dados não relacionais, as vezes chamados de NoSQL, não empregam um modelo de tabela. Em vez disso, eles armazenam conteúdo, independentemente de sua estrutura, como um único documento. Essa tecnologia é adequada para dados não estruturados, como conteúdo de mídia social, fotos e vídeos.

4.6.5 Migrando Banco de Dados Legados para Nuvem

Um banco de dados local pode migrar para uma implementação em nuvem. Existem várias razões para fazer isso, incluindo o seguinte:

- Permite que a TI remova a infraestrutura de armazenamento e servidor físico no local;
- Preenche a lacuna de talentos quando a TI não tem experiência adequada em banco de dados interno;
- Melhora a eficiência do processamento, especialmente quando aplicativos e analises que acessam os dados também residem na nuvem;
- Consegue economia de custos por vários meios, incluindo:
    - Redução da equipe interna de TI;
    - Declínio continuo do preço do serviço em nuvem;
    - Pagando apenas os recursos realmente consumidos, conhecidos como preços de pagamento conforme o uso.

A realocação de um banco de dados para a nuvem pode ser uma maneira eficaz de ativar ainda mais o desempenho de aplicativos corporativos como parte de uma implantação mais ampla de software como serviço. Isso simplifica os processos necessários para disponibilizar informações por meio de conexões baseadas na Internet.

A consolidação de armazenamento também pode ser um benefício de mover os bancos de dados de uma empresa para a nuvem. Bancos de dados em vários departamentos de uma grande empresa, por exemplo, podem ser combinados na nuvem em um único sistema hospedado de gerenciamento de banco de dados.

4.6.6 Como funciona o Banco de Dados em Nuvem

Do ponto de vista estrutural e de design, um banco de dados em nuvem não é diferente daquele que opera nos servidores locais de uma empresa. A principal diferença está em onde reside.

Quando um banco de dados local é conectado a usuários locais por meio de rede local (LAN) interna de uma corporação, um banco de dados em nuvem reside em servidores e armazenamento fornecidos por um provedor de nuvem ou DBaaS e é acessado exclusivamente pela Internet. Para um aplicativo de software, por exemplo, um banco de dados SQL residente no local ou na nuvem deve parecer idêntico.

Acessado por meio de consultas diretas (como instruções SQL) ou por meio de chamadas de API, o comportamento do banco de dados deve ser o mesmo. No entanto, pode ser possível discernir pequenas diferenças no tempo de resposta. Um banco de dados local, acessado por meio de uma LAN, provavelmente fornecerá uma resposta um pouco mais rápida do que banco de dados baseado em nuvem, que requer uma viagem de ida e volta na Internet para cada interação com o banco de dados. Na prática, as diferenças provavelmente são pequenas.

5 CAPÍTULO 4 – MIGRAÇÃO DOS BANCOS DE DADOS MYSQL E MONGODB PARA A *CLOUD*

Neste capítulo serão discutidas algumas dificuldades de migração de um banco de dados MySQL e MongoDB para cloud, no caso, a AWS. Esta discussão se desenvolverá a partir da análise técnica dos bancos de dados MySQL e MongoDB, simulação de custos para uma possível migração para a AWS, além da segurança envolvida neste processo.

5.1 Dificuldade de migração dos bancos de dados MySQL e MongoDB para a cloud

O processo de migração de um banco de dados nem sempre é uma tarefa fácil de ser executada, precisando de uma avaliação prévia do estado atual do banco antes de iniciar qualquer tipo de planejamento de migração. A migração de dados de um banco de dados para outro pode ser bastante desafiadora com relação à consistência dos dados, ao tempo de inatividade das aplicações e às principais diferenças de *design* entre os bancos de dados de origem e de destino. Toda migração de banco de dados, seja relacional ou não relacional, será necessário realizar um levantamento dos pontos de criticidade do negócio envolvido, pois este ditará como será conduzido a migração.

Gerenciar uma migração de um banco de dados é uma tarefa que agrega riscos ao negócio, pois há a necessidade de se certificar se a operação de migração não irá afetar o negócio enquanto é executada. Uma migração pode levar horas ou até mesmo dias, dependendo do tamanho do banco de dados e também do tipo de negócio envolvido. Quando há a necessidade de migrar um banco de dados em produção, sempre haverá a preocupação com os dados que estão a todo momento em transformação, podendo causar algum tipo de discrepância nos mesmos ou ainda problemas ao negócio.

Em se tratando do processo de migração de um banco de dados MySQL *on-premises* para o Amazon AWS RDS (*Relation Database Service*) MySQL, é possível seguir uma das estratégias abaixo:

- Usando o AWS *Database Migration Service* (AWS DMS) – O banco de dados pode estar *on-line* em Produção;
- Através da exportação do *dump* do banco de dados MySQL *on-premises* e importando no AWS RDS MySQL com as próprias ferramentas do banco de dados – O banco de dados deve estar *offline*;
- Também pode ser usado o modo *binlog replication* do banco de dados MySQL – O banco de dados pode estar em *on-line* em Produção.

Com base na estratégia escolhida para a realização da migração do banco de dados, será necessário alocar recursos com conhecimentos adequados para a execução da tarefa. E para que o projeto de migração não corra o risco de não ser finalizado, é interessante que os recursos humanos envolvidos pelo menos trabalhem em par, pois caso haja uma desistência, a continuidade do projeto estará garantida.

Com relação a migração de um banco de dados NoSQL para a Amazon AWS, é necessário um prévio levantamento da situação do banco de dados atual *on-premises* e também dos modelos ofertados pela *Cloud*. Com base nestas informações, é possível desenvolver uma estratégia para a execução do projeto. No caso da escolha da migração de um banco de dados NoSQL como o MongoDB, há a possibilidade de migração para uma instância em *cluster*. No entanto, este *cluster* deverá ser implementado do “zero”, pois a AWS não fornece como serviço o MongoDB. Uma opção compatível com o MongoDB seria o DynamoDB. O DynamoDB segue uma estrutura de dados parecida com o MongoDB.

Como no processo de migração do banco MySQL, a migração do banco de dados NoSQL MongoDB segue as mesmas criticidades, diferenciando somente os tipos de tecnologias envolvidas e recursos alocados.

Para realizar a migração de um banco de dados NoSQL, como o MongoDB, é possível seguir algumas das estratégias abaixo:

- Usando o AWS Database Migration Service (AWS DMS) - O banco de dados pode estar *on-line* em Produção;

- Realizar o *dump/exporter* de todas as *collection* do banco de dados *on-premises* MongoDB e importando no cluster MongoDB de destino – O banco de dados deve estar *offline*;
- Desenvolver uma ferramenta de migração usando as *features* do MongoDB para o envio incremental das *collections* para o banco de dados de destino.

A alocação dos recursos envolvidos será baseada na estratégia escolhida, pois o conhecimento necessário será dirigido a partir disto. Em ambos os processos de migração haverá a necessidade de treinamento de equipes, contrato de consultoria, contratos a serem seguidos e conviver por algum tempo com ambientes duplicados.

5.1.1 Vantagens

A migração de um banco de dados para uma *cloud* proporciona diversos ganhos, como:

- Diminuição de custos para manter o banco de dados – o cliente pagará somente pelo poder computacional, armazenamento e outros serviços que vier a usar, sem contrato de longo prazo ou compromissos prévios;
- Flexibilidade – possibilidade de ter uma infraestrutura elástica a um custo baixo;
- Confiável – poderá usufruir de uma estrutura global e segura, podendo ter o banco de dados hospedado em regiões diferentes para garantir a disponibilidade do negócio.
- Escalável – tem a possibilidade de escalar o banco de dados;
- Segurança – a *cloud* utiliza uma abordagem de ponta a ponta para proteger e fortalecer a infraestrutura, incluindo medidas físicas de segurança, operacionais e de *software*.
- Produtividade – acesso aos servidores de qualquer lugar do mundo, facilitando o gerenciamento.

5.1.2 Segurança

Em se tratando da segurança do uso do AWS RDS MySQL e MongoDB, a AWS proporciona o uso do modelo de segurança baseada em grupos, sendo que um grupo de segurança controla o acesso a instância do banco de dados que não esteja em uma mesma VPC (*Virtual Private Cloud*). Além disso, há a possibilidade do uso de servidores *bastion* (*host* configurado para desempenhar um papel crítico na segurança da rede interna, seja na *cloud* ou *on-premises*) para que o acesso ao banco de dados seja somente através deste servidor. A AWS também proporciona o uso de um VPN (*Virtual Private Network*) para uso de acesso aos servidores de banco de dados e servidor *bastion*, através do uso de um certificado digital gerado através da plataforma da AWS.

Como se pode perceber, a estrutura em *cloud*, no caso a AWS, proporciona diversos benefícios, sendo um deles a de segurança de modo muito eficiente. Em um ambiente *on-premises*, nem sempre é possível ter toda a segurança que uma *cloud* como a AWS fornece, pois gera grandes custos para se manter.

A migração de um banco de dados para *cloud* proporciona um benefício direto aos custos do negócio. Um banco de dados como MySQL ou MongoDB, precisará ter pelo menos um DBA para cada banco. Além disso, uma equipe de infraestrutura para manter o servidor onde o banco de dados está alojado.

Na AWS o uso da infraestrutura é de acordo com o uso necessário que o negócio despende, não precisando se preocupar com contratos de fornecedores ou até mesmo com uma gama de problemas relacionados a infraestrutura, *links* de Internets, indisponibilidade de infraestrutura, etc.

5.2 Análise Técnica dos Bancos MYSQL e MongoDB

O banco de dados MySQL utiliza tabelas para armazenar os seus dados e a linguagem SQL para acessar os dados (MYSQL, 2018). Já o banco de dados MongoDB armazena os seus dados no formato de documentos JSON e os codifica em formato binário BSON. O BSON estende o JSON para fornecer tipos de dados adicionais, campos ordenados para ser eficiente na codificação e decodificação em diferentes idiomas (MONGODB, 2018).

O MySQL utiliza *schemas* para definir a estrutura do objeto no banco de dados, exigindo que todas as linhas dentro de uma tabela tenham a mesma estrutura com valores sendo representados por um tipo de dado específico. Já o MongoDB é um banco de dados livre de *schemas*, permitindo que seja criado documentos sem necessidade de definir uma estrutura para o documento. Esses documentos podem ser facilmente alterados adicionando ou excluindo campos. No caso de um banco de dados relacional, seria necessária uma reestruturação da base de dados para executar a mesma tarefa.

Em relação a modelo de dados, no MongoDB é possível representar relacionamentos hierárquicos, matrizes de dados e outras estruturas complexas. Em alguns casos, o desempenho do MongoDB é melhor em relação ao MySQL porque o MongoDB não usa *joins* para conectar dados, acarretando em um melhor desempenho das *queries* executadas.

Tanto o MySQL quanto o MongoDB fazem uso de índices para proporcionar eficiência no processo de busca de dados. No MySQL, se um índice não for definido, o banco de dados deverá realizar um *full scan* na tabela para localizar todas as linhas relevantes da busca executada. Já no MongoDB, se um índice não for encontrado, todos os documentos de uma *collection* deverão ser verificados para selecionar os documentos que fornecem uma correspondência para a instrução da consulta. Ambos os bancos de dados MySQL e MongoDB, detém terminologias e conceitos parecidos, como demonstrado na Tabela 3:

Tabela 3 – Terminologias do MySQL e MongoDB.

MySQL	MongoDB
ACID Transação	ACID Transação
Tabelas	Collection
linhas	Documentos
Colunas	Campos
Indices secundários	Indices secundários
JOINs	$lookup & $graphLookup
GROUP_BY	Aggregation Pipeline

Fonte: MONGODB-MYSQL, 2018.

Como o MySQL, o MongoDB oferece um rico conjunto de recursos e funcionalidades, muito além daqueles oferecidos por armazenamentos de dados NoSQL. O MongoDB possui uma linguagem de consulta avançada, índices secundários altamente funcionais incluindo pesquisas de textos e geoespacial, uma poderosa estrutura de agregação para análise de dados, *faceted search*, processamento de gráficos, etc. Com o MongoDB, é possível usar esses recursos em tipos de dados diversos, o que não ocorre com um banco de dados relacional, e além disso tê-los em escala. Na Tabela 4, tem-se uma comparação de características entre o MySQL e MongoDB:

Tabela 4 – Características entre o MySQL e MongoDB.

	MySQL	MongoDB
Open source	sim	sim
Transações ACID	sim	sim
Modelo de dados flexível	não	sim
Schemas	sim	sim
Expressive joins, faceted search, graphs queries, powerful aggregations	sim	sim
Drivers idiomáticos de idioma nativo	não	sim
Expansão horizontal com controles de localidade de dados	não	sim
Suporte a Analytics e BI	sim	sim
Segurança de nível empresarial e ferramentas de gerenciamento maduras	sim	sim
Banco de dados como um serviço em todas as principais cloud de mercado	sim	sim

Fonte: MONGODB-MYSQL, 2018.

5.2.1 Forma de busca de dados

Em relação aos recursos/ferramentas de busca de dados, o MySQL e MongoDB diferem em sua sintaxe e semântica, pois um segue o padrão SQL e o outro, linguagem própria. Segue abaixo exemplos que representam comandos DML

(*Data Manipulation Language*) de dados MySQL (MYSQL, 2018) e MongoDB (MONGODB, 2018):

- Consulta:

  MySQL => select * from cliente;

  MongoDB => db.cliente.find()

- Inserindo dados:

  MySQL => insert into cliente(cliente_id, nome, sobrenome) values (1, “Maria”, “Silva”)

  MongoDB => db.cliente.insert({cliente_id, nome, sobrenome})

- Update:

  MySQL => update cliente set nome = ‘João’ where cliente_id = 1

  MongoDB => db.cliente.update ({cliente_id: {$eq = 1}, {$set {nome = ‘João’}}}

5.2.2 Suporte a Sistemas Operacionais

O MySQL (MYSQL, 2018) e o MongoDB (MONGODB, 2018) têm uma gama considerável de compatibilidade/suporte a sistema operacional:

- Mysql: Microsoft Windows, Mac OS X, GNU/Linux, AIX, FreeBSD, HP-UX, NetBSD;
- MongoDB: GNU/Linux, Mac OS X, Solaris e Microsoft Windows.

Também suportam modelos de replicação/cluster:

- MySQL: O MySQL suporta replicação *master-slave* e replicação *master- master* (a partir do MySQL 5.7.6);
- MongoDB: O MongoDB suporta replicação, *sharding* (escalonamento horizontal, complex de implentar no MySQL) e *auto-elections*.

5.2.3 Serviços de suporte do MySQL e MongoDB

Ambos os bancos de dados aqui comentados, oferecem serviços *Enterprises* de suporte. No MySQL temos a *Oracle Lifetime Support* (ORACLE, 2018) em três níveis:

- *Premier* para versões de 1 a 5 anos;
- *Extended* para versões de 6 a 8 anos;
- *Sustain*, para aqueles que desejam usar a mesma versão por mais de 9 anos. Cada nível oferece suporte 24x7 com base de conhecimento, versões de manutenção, correções de *bugs*, *patches* e atualizações.

Já o O MongoDB (MONGODB-SUPPORT, 2018) oferece suporte de nível corporativo que se estende além do intervalo/*patched:*

- Oferece suporte 24/7;
- Suporte ao ciclo de vida estendido do banco de dados, que permite a flexibilidade de atualizar para versões mais novas no seu próprio ritmo;
- Acesso ilimitado ao suporte, correções de segurança, atualizações, etc.

5.3 Simulação de custos dos bancos de dados MySQL e MongoDB na *cloud* AWS

A utilização de banco de dados na *cloud* está muitas vezes atrelada ao custo da hora em uso do recurso do que exatamente no licenciamento, isto por causa de parcerias que as *clouds* detém com as empresas desenvolvedoras destes produtos. Essas parcerias ajudam a diminuir os custos de licenciamento, pois trabalham em conjunto com as *clouds* para fornecer uma espécie de serviço agrupado com os produtos.

Os custos dos bancos de dados na *cloud* podem variar, pois existe alguns fatores que devem ser levados em conta no momento da estimativa de valores, baseado no modelo/tipo/tamanho do projeto que se deseja implementar. Em se tratando de um ambiente na AWS, alguns itens básicos deverão sempre ser levados

em consideração no processo de tomada de decisão para construção de um ambiente de banco de dados, como:

- Processamento: a cobrança de instância/hora do Amazon EC2, tanto *on-demand* como *Reserved* (*Light, Medium* ou *Heavy*), como modelos de *Spot*;
- Armazenamento: volumes EBS (*Standard* e *Provisioned* IOPS) para o seu armazenamento (tanto capacidade de armazenamento atual quanto I/O para disco);
- Transferência de dados: Tráfego de rede, cobranças diferenciadas para comunicação entre *Availability Zone* com configurações que abrangem várias zonas);
- *Backups*: Armazenamento no Amazon S3 para *snapshots* de configurações de volumes (EBS) de dados;
- Réplicas em diferentes zonas para a garantia de continuidade do negócio.

Após a definição e planejamento do projeto de migração/ambiente novo de banco de dados na AWS, sabendo o que exatamente será necessário para a definição da arquitetura do banco de dados, é possível através da ferramenta AWS *Simple Monthly Calculator,* oferecida pela AWS, modelar o projeto conforme a necessidade planejada no escopo do projeto e projetar os seus custos mensais de forma a viabilizar uma melhor estratégia para o negócio (AWS, 2018).

Com base nas informações descritas acima e usando a calculadora da AWS, é possível realizar uma simulação de custos de uma determinada instância/grupo de instâncias de banco de dados MySQL e MongoDB na AWS, (AWS, 2018), conforme as simulações desenvolvidas a seguir.

a) Simulação de custos de um banco de dados Amazon RDS MySQL (AWS, 2018):

- Tipo de Instância: 3 x db.m4.xlarge
- Multi-AZ

- Storage: 1Tb (provisioned IOPS – 3000 IOPS, uma para cada instância)
- Backup space: 1000Gb/mês
- Intra-Region Data Transfer: 40Gb/Mês
- Com suporte Business

As Figuras 7 e 8 ilustram o cálculo de custo referente ao banco de dados **Amazon RDS MySQL na AWS** utilizando o modelo ***on-demand***.

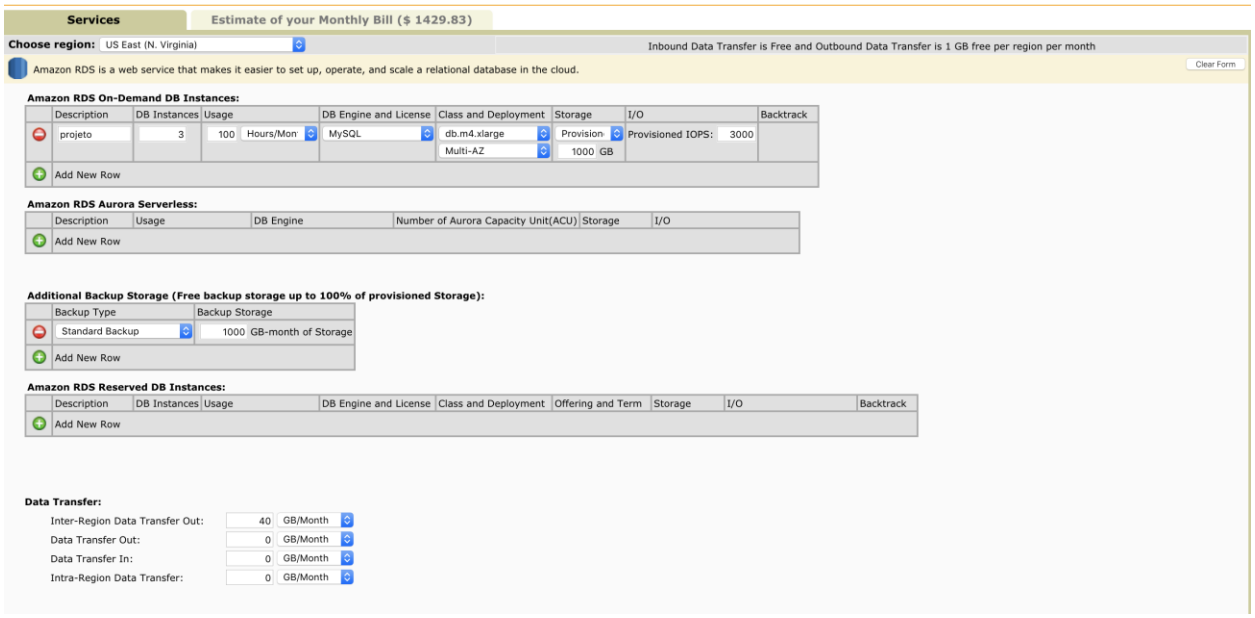

Figura 7 – Amazon RDS MySQL na AWS no modelo *on-demand.*
Fonte: AWS. Simply Monthly Calculator. 2018.

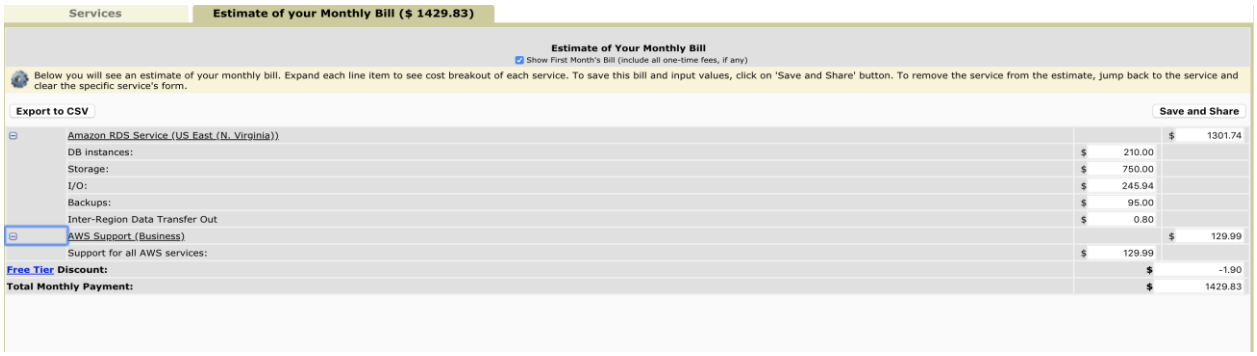

Figura 8 – Cálculo do custo Amazon RDS MySQL na AWS no modelo *on-demand.*
Fonte: AWS. Simply Monthly Calculator. 2018.

Conforme as Figuras 7 e 8, o custo atual *On-demand* instâncias é:

$ 1429,83/Mês.

As Figuras 9 e 10 ilustram o cálculo de custo referente ao banco de dados **MySQL RDS na AWS** utilizando **instâncias reservadas.**

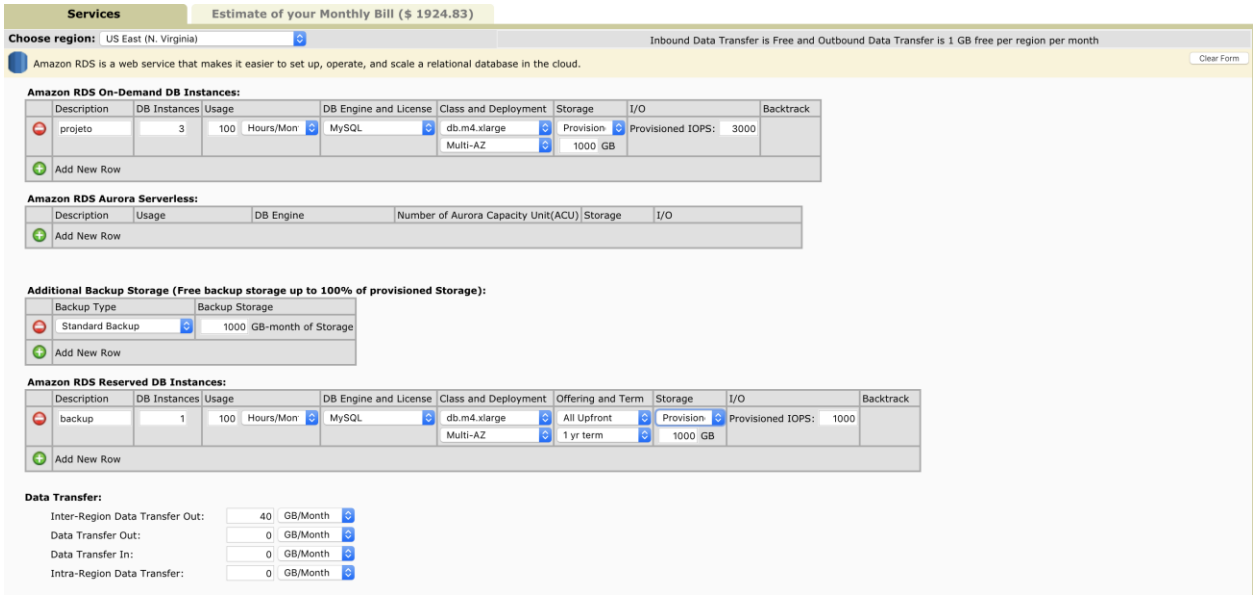

Figura 9 – MySQL RDS na AWS utilizando instâncias reservadas.
Fonte: AWS. Simply Monthly Calculator. 2018.

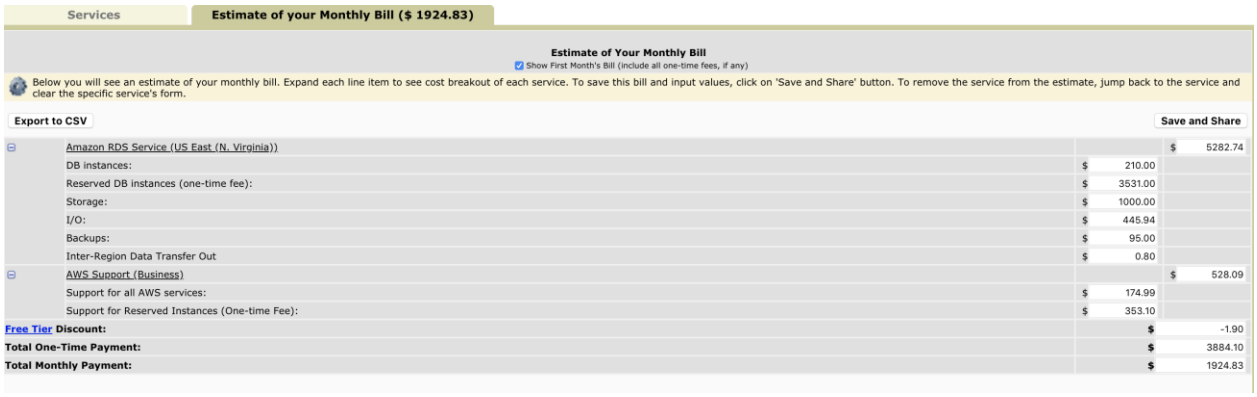

Figura 10 – Cálculo do custo MySQL RDS na AWS utilizando instâncias reservadas
Fonte: AWS. Simply Monthly Calculator. 2018.

Conforme as Figuras 9 e 10, o custo atual Instâncias reservadas (1 ano, parcial *upfront*) é:

$ 1924,83/Mês.

b) Simulação de custos de um banco de dados MySQL em instâncias EC2 (AWS, 2018):

- Tipo de instância: 3 x Linux m4.4xlarge

- Multi-AZ
- Storage: 1Tb (provisioned IOPS – 3000 IOPS, one for each instance)
- Backup space: 1000Gb/month
- Intra-Region Data Transfer: 40Gb/Month
- Com suporte Business.

As Figuras 11 e 12 ilustram o cálculo de custo referente ao banco de dados **MySQL utilizando instâncias EC2 na AWS** no **modo *on-demand***.

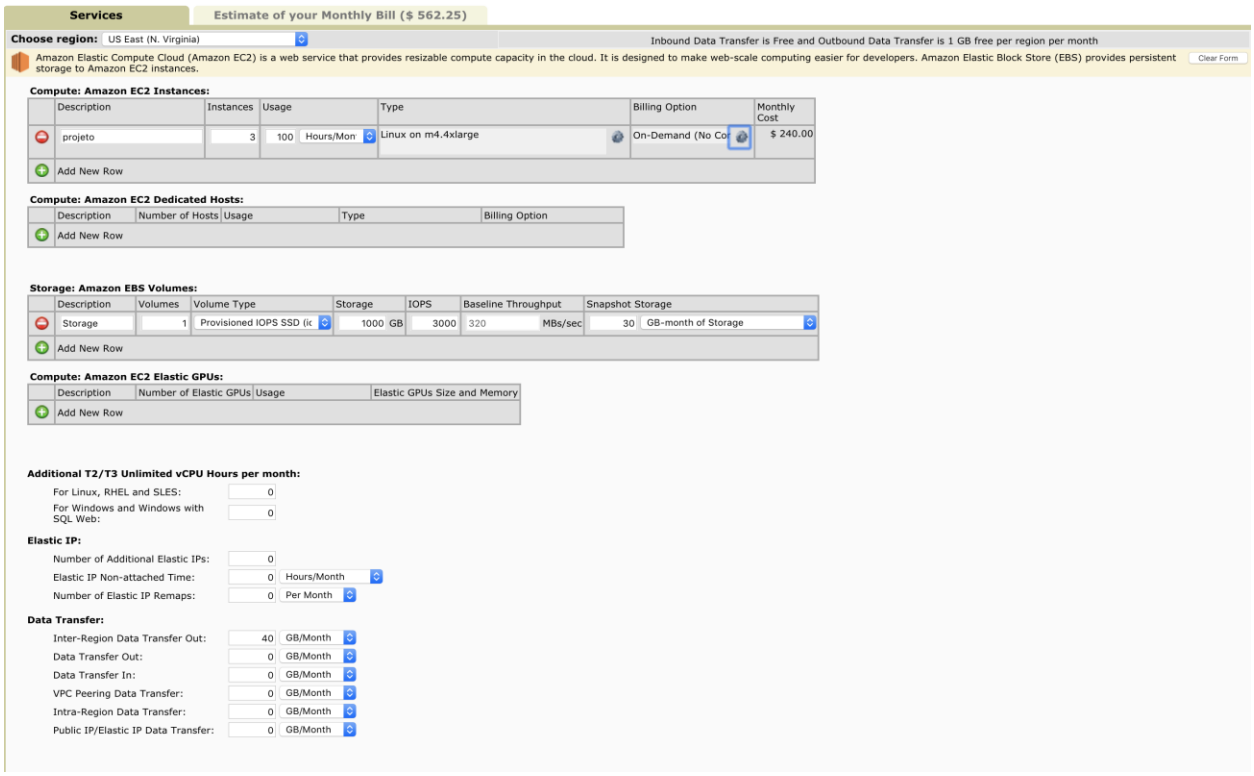

Figura 11 - MySQL utilizando instâncias EC2 na AWS no modo *on-demand.*

Fonte: AWS. Simply Monthly Calculator. 2018.

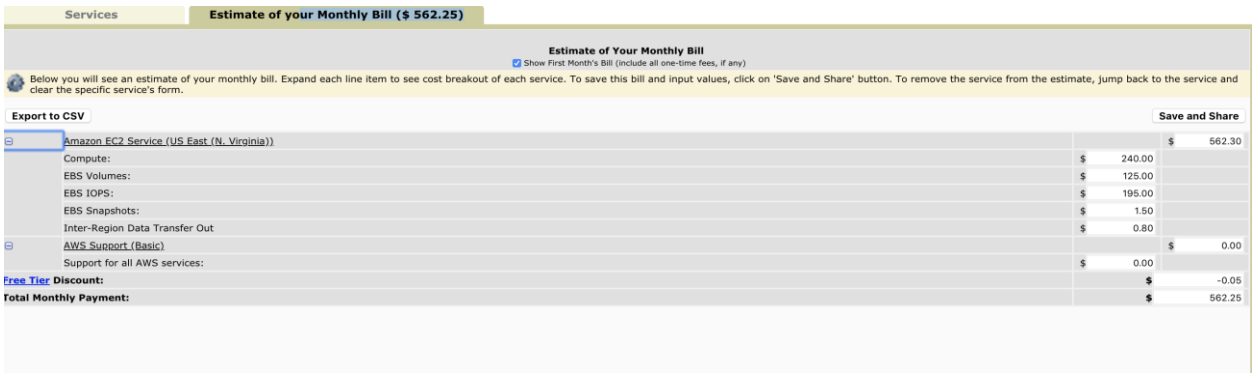

Figura 12 – Cálculo do custo MySQL instância C2 na AWS, no modo *on-demand*.
Fonte: AWS. Simply Monthly Calculator. 2018.

Conforme as Figuras 11 e 12, o custo atual *On-demand* instâncias: $ 562,25/Mês.

As Figuras 13 e 14 ilustram o cálculo de custo referente ao banco de dados **MySQL utilizando instâncias EC2 na AWS** no **modo instâncias reservadas.**

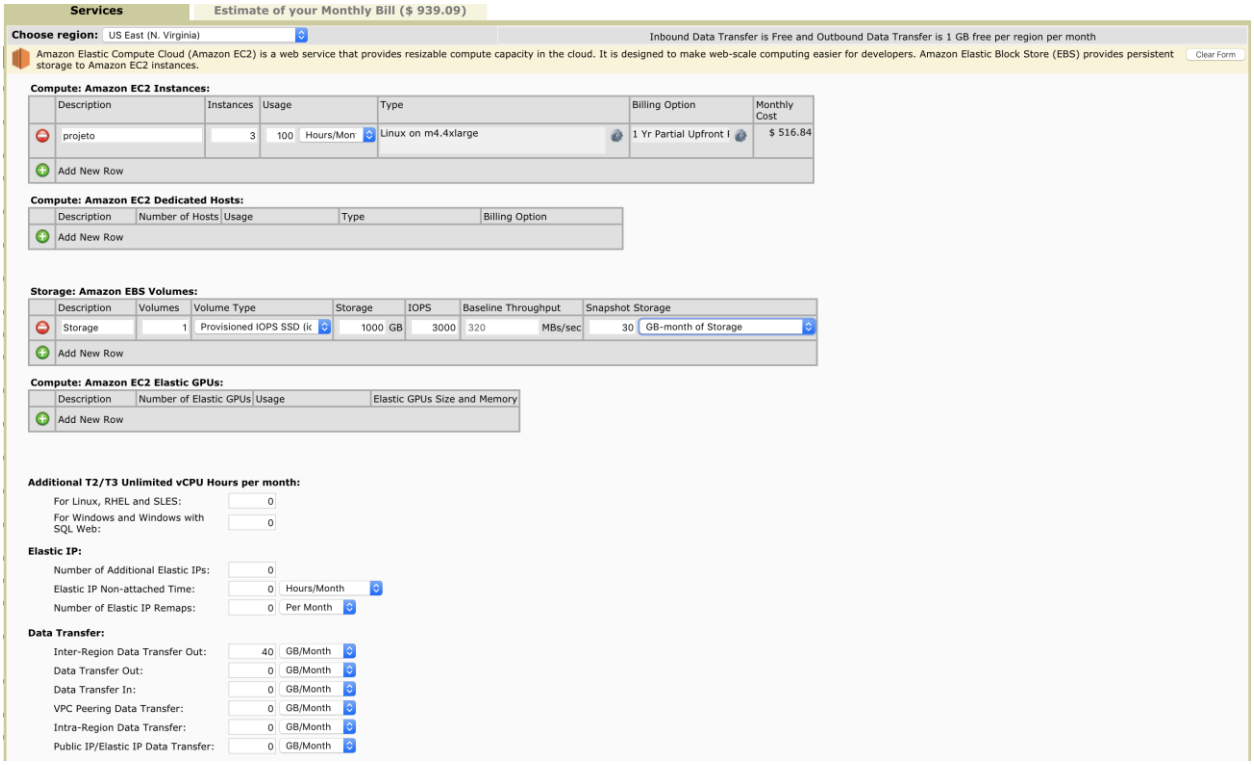

Figura 13 - MySQL utilizando instâncias EC2 na AWS, no modo instâncias reservadas.
Fonte: AWS. Simply Monthly Calculator. 2018.

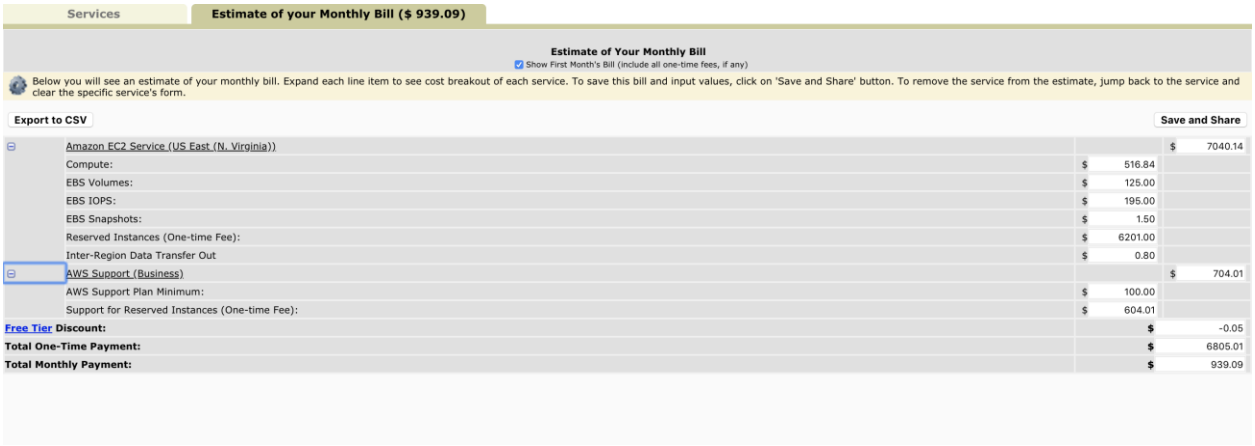

Figura 14 - Cálculo do custo MySQL instância C2 na AWS, no modo instâncias reservadas.
Fonte: AWS. Simply Monthly Calculator. 2018.

Conforme as Figuras 13 e 14, o custo atual Instâncias reservadas (1 ano, parcial *upfront*) é: $ 939,09/Mês.

c) Simulação de custos de um banco de dados MongoDB em instâncias EC2 (AWS, 2018):

- Tipos de instâncias: 3 x Linux c3.4xlarge
- Multi-AZ
- Storage: 1Tb (provisioned IOPS – 3000 IOPS, one for each instance)
- Backup space: 1000Gb/month
- Intra-Region Data Transfer: 40Gb/Month
- Com suporte Business.

As Figuras 15 e 16 ilustram o cálculo de custo referente ao banco de dados **MongoDB utilizando instâncias EC2 na AWS** no **modo *on-demand***.

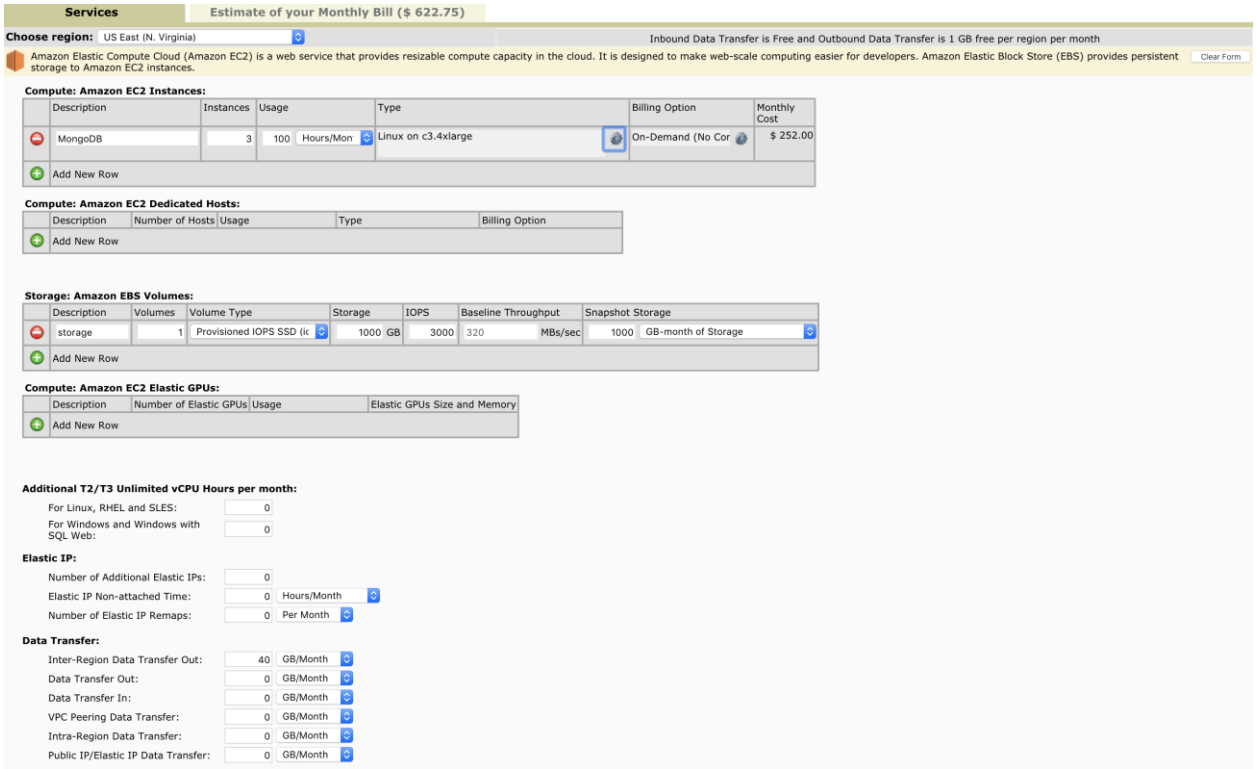

Figura 15 - MongoDB utilizando instâncias EC2 na AWS no modo *on-demand.*
Fonte: AWS. Simply Monthly Calculator. 2018.

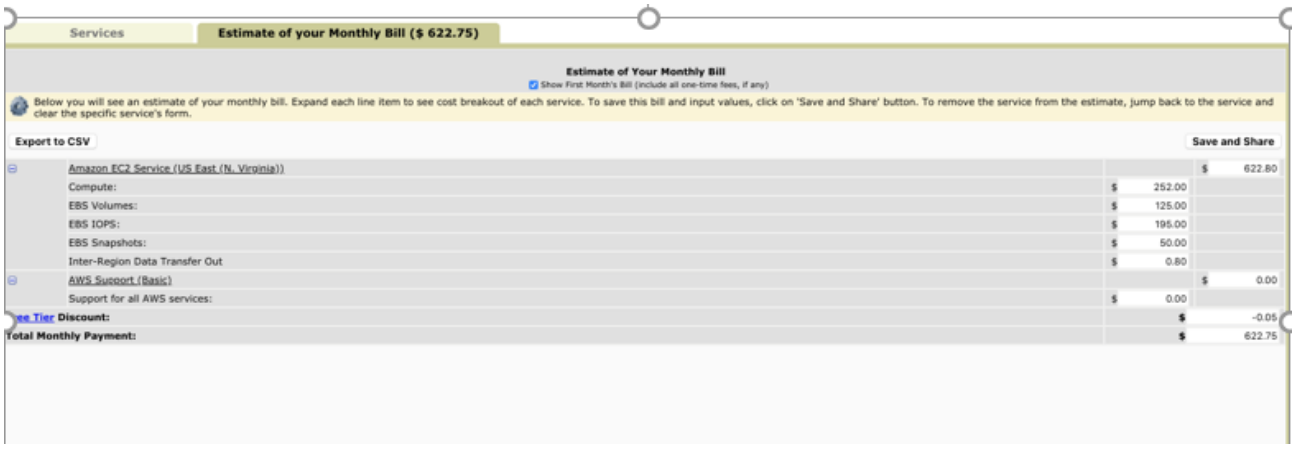

Figura 16 - Cálculo do custo MongoDB utilizando instâncias EC2 na AWS no modo *on-demand*.
Fonte: AWS. Simply Monthly Calculator. 2018.

Conforme as Figuras 15 e 16, o custo atual *On-demand* instâncias é:

$ 622,75/Mês.

As Figuras 17 e 18 ilustram o cálculo de custo referente ao banco de dados **MongoDB** utilizando **instâncias EC2 na AWS** no **modo instâncias reservadas.**

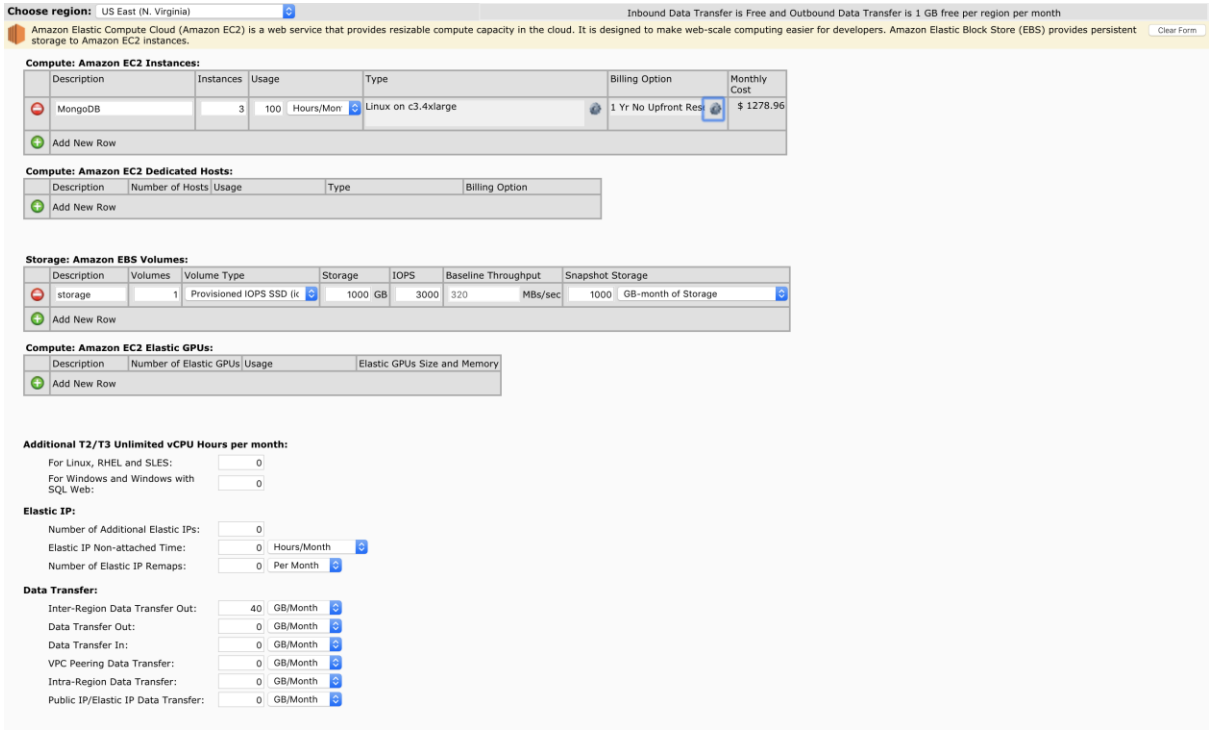

Figura 17 - MongoDB utilizando instâncias EC2 na AWS no modo instâncias reservadas.
Fonte: AWS. Simply Monthly Calculator. 2018.

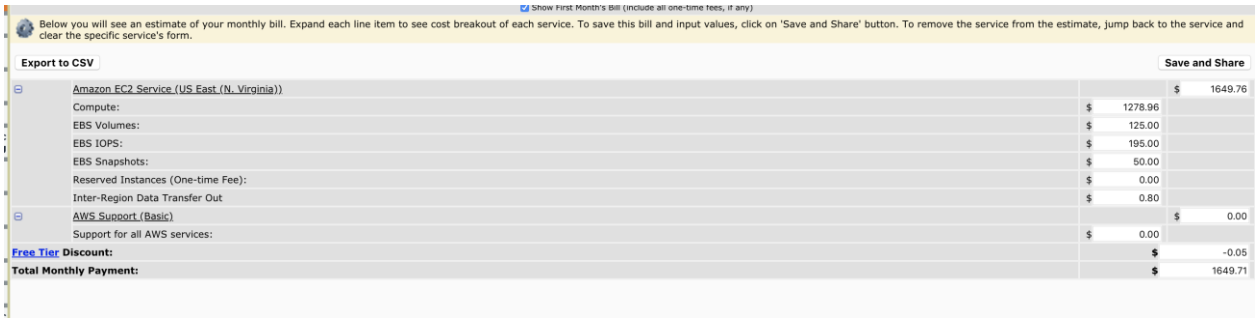

Figura 18 – Cálculo do custo para o MongoDB utilizando instâncias EC2 na AWS, no modo instâncias reservadas.
Fonte: AWS. Simply Monthly Calculator. 2018.

Conforme as Figuras 17 e 18, o custo atual Instâncias reservadas (1 ano, parcial *upfront*) é:

$ 1649,71/Mês.

Não há decisão certa ou errada quanto à escolha de uma solução de banco de dados na *cloud*. O que irá definir a melhor escolha é o que melhor atende as particularidades de cada negócio. Ambas as tecnologias de banco de dados MySQL e MongoDB são excelentes, porém cada uma com um foco. O custo pode variar dependendo do quão robusta a infraestrutura tenha que ser. Geralmente o fator decisivo para a escolha do uso de um banco de dados na *cloud* está atrelado aos custos dos recursos das instâncias e seu suporte do que no licenciamento, pois a há uma infinidade de opções e formas de uso de banco de dados na *cloud*. A AWS fornece a opção de orquestração da infraestrutura baseado no gosto do cliente ou baseado em seus serviços, como no caso da Amazon RDS. Além disso, a AWS oferece o AWS *Marketplace*, onde empresas oferecem infraestruturas pré-configuradas para uso na cloud da AWS.

O custo é definitivamente um fator chave na escolha/migração para a *cloud*, especialmente quando a empresa tem uma quantidade considerável de recursos envolvidos para manter o negócio. Uma das vantagens de se usar uma infraestrutura na *cloud*, é a possibilidade de ter uma infraestrutura elástica, onde em determinados horários poderá diminuir a quantidade dos recursos utilizados e consequentemente, diminuir o custo mensal do negócio.

5.4 A Segurança da hospedagem de Banco de Dados na *Cloud*

A segurança da informação está cada vez mais em ascensão no meio digital. E em se tratando de informações que estão armazenadas em bancos de dados, não é diferente. Empresas têm armazenado durante anos, ou em alguns casos durante décadas os dados correspondentes às informações que são sensíveis para o negócio. Estas informações, muitas vezes, dirigem o valor do negócio no mercado.

As empresas estão cada vez mais migrando suas estruturas de TI para *cloud*, e isso por causa de vários motivos, como por exemplo custos e segurança. No entanto, a decisão de migrar um banco de dados para cloud com todas as informações do negócio pode gerar um desconforto para muitas empresas, pois ainda há a dúvida se, na *cloud*, as informações estariam tão seguras quanto em seus *datacenters*.

As *clouds* têm evoluído de forma satisfatória na questão da segurança da informação em seus datacenters, facilitando o uso e a migração. Entretanto, seus servidores são hospedados em todo o mundo, e por causa disto, pode ser um empecilho no processo de migração, pois os dados poderão estar hospedados em um país estrangeiro com leis vigentes diferentes do país da origem do cliente. No final, a tomada de decisão de migrar para a *cloud* ou não sempre estará ligada à estratégia e à representatividade do negócio.

Em se tratando da migração de bancos de dados, seja relacional ou não, e de forma técnica, as *clouds* têm ofertado várias opções de migração e segurança na hospedagem, com criptografia dos dados em sua estrutura. A AWS fornece o processo de criptografia das instâncias do banco de dados e *snapshots* do Amazon RDS MySQL. Os dados que serão criptografados incluem um armazenamento subjacente de um banco de dados. Além disso, oferece uma plataforma com *backups* automatizados, réplicas de leitura, etc. As instâncias criptografadas dos bancos de dados do Amazon RDS usam o algoritmo de criptografia AES-256, padrão da indústria para criptografar seus dados em servidores que hospedam instâncias de banco de dados. O Amazon RDS lida muito bem com a autenticação do acesso e decodificação dos seus dados de forma transparente com o mínimo de impacto sobre o negócio. Este mesmo conceito pode ser aplicado ao banco de dados MongoDB. No entanto, vale lembrar que a projeção da estrutura do banco é

fundamental para que o processo de segurança seja conduzido de forma natural a fornecer segurança aos dados que serão armazenados na AWS.

CONCLUSÃO

Neste trabalho foi apresentado dois tipos de bancos de dados principais: o SQL (relacional) e o NoSQL (não-relacional). As diferenças entre eles estão enraizadas na maneira como são projetadas, nos tipos de dados que suportam e em como são armazenados.

No primeiro Capítulo foi apresentado o Banco de Dados Relacional, conceituando-o, além de apresentar os tipos de dados que correspondem a este tipo de *data base*.

No Capítulo 2 foi apresentado um Banco de Dados NoSQL, suas características, e os tipos de armazenamento utilizados para este *database*, além de apresentar as vantagens e desvantagens deste tipo de banco de dados.

No Capítulo 3, foi demonstrado uma comparação entre os dois bancos de dados, SQL e NoSQL, além de apresentar as razões para usar cada um dos bancos de dados. A implantação em nuvem também foi abordada neste capítulo, conceituando DBaaS, os benefícios da migração para a *cloud* bem como foi demonstrado o funcionamento do banco de dados em Nuvem.

O Capítulo 4 trata da migração para a *cloud*, discorrendo sobre as dificuldades de migrar os bancos de dados MySQL e MongoDB. Também apresenta uma análise técnica de ambos os bancos, as simulações de custo de manutenção na cloud para os bancos abordados, além de demonstrar a segurança no processo de migração para a Nuvem.

Bancos de dados relacionais são entidades relacionalmente estruturadas, geralmente representando um objeto do mundo real, por exemplo, uma pessoa ou detalhes do carrinho de compras. Os bancos de dados não relacionais são estruturados e distribuídos em documentos, armazenando informações em uma hierarquia semelhante a uma pasta, que armazena os dados em um formato não estruturado.

Na linguagem cotidiana, são chamamos de SQL e NoSQL, o que reflete o fato de que os bancos de dados NoSQL não são escritos apenas na linguagem de consulta estruturada (SQL). Porém, não se trata apena de uma questão de SQL vs. NoSQL. Em vez disso, é SQL e NoSQL, cada vez mais sendo integrados uns aos outros. Como exemplo, Microsoft, Oracle e Teradata, já vendem alguma forma de

integração do Hadoop para conectar a análise baseada em SQL ao mundo dos *bigdata* não estruturados.

O banco de dados relacional SQL é a escolha certa para qualquer empresa que se beneficiará de sua estrutura predefinida e definirá esquemas. Por exemplo, aplicativos que exigem transações em várias linhas – como sistemas contábeis ou sistemas que monitoram inventário – ou que são executados em sistemas legados, irão prosperar com estrutura do MySQL.

O NoSQL, por outro lado, é uma boa opção para empresas que possuem crescimento rápido ou bancos de dados sem definições de esquema claras. Mais especificamente, nos casos em que não se pode definir um esquema para o banco de dados, se estiver desnormalizando esquemas de dados ou se o esquema continuar a mudar – como acontece com aplicativos móveis, análises em tempo real, sistemas de gerenciamento de conteúdo, etc. neste caso, o MongoDB pode ser uma ótima escolha.

A nuvem permite que se implante mais rapidamente, se reduza os custos e se gaste mais tempo nas principais competências de TI, apenas para citar alguns benefícios. É definitivamente uma virada de jogo. As plataformas que disponibilizam o armazenamento em nuvem permitem que as organizações integrem, processem e preparem dados para análises na nuvem. Com essas plataformas, toda empresa pode ter conectividade imediata com uma variedade de armazenamentos de dados e um rico conjunto de componentes de transformação de dados prontos para uso.

Conforme o objetivo geral apresentado como proposta para este estudo, entendemos que o mesmo foi atingido, tendo sido apresentado o conceito de migração para a *cloud*, as vantagens e desvantagens para manter o banco de dados na *cloud*, além das dificuldades da migração propriamente dita.

REFERÊNCIAS BIBLIOGRÁFICAS

ALTARADE, Mohammad. **The Definitive Guide to NoSQL Databases**. 2018. Disponível em: <https://www.toptal.com/database/the-definitive-guide-to-nosql-databases.> Acesso em: 16 nov. 2018.

AMAZON WEB SERVICES. 2018. **Armazenamento em Nuvem**. Disponível em: <https://aws.amazon.com/>. Acesso em: 05 set. 2018.

AWS. **Simply Monthly Calculator**. 2018. Disponível em: <http://calculator.s3.amazonaws.com/index.htmlhttp://calculator.s3.amazonaws.com/index.html>. Acesso em: 16 nov. 2018.

AZURE. **Microsoft Azure**. 2016. Disponível em: <http://www.microsoft.com/>. Acessado em: 01 set. 2018.

BARROS, L. E. B. **Banco de Dados em Nuvem**. 2008. Disponível em: <http://pesquompile.wikidot.com/>. Acesso em: 07 set. 2018.

BATISTA, F. et al. **MongoDB: Banco de dados orientado a documento**. 2012. Disponível em: <http://mongodb.machinario.com/>. Acesso em: 07 set. 2018.

BSON. **Binary JSON**. 2012. Disponível em: <https://www.mongodb.com/> Acesso em: 12 set. 2018.

BUYYA, R.; YEO C. S.; VENUGOPAL, S. **Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utili-ties.** Grid Computing and Distributed Systems (GRIDS) Laboratory Department of Computer Science and Software Engineering. Australia: The University of Melbourne. 2008.

CASSANDRA. **Cassandra**. 2016. Disponível em: http://cassandra.apache.org/>. Acesso em: 03 set. 2018.

CATTELL, Rick. **Scalable SQL and NoSQL Data Stores**. Originally published in 2010, last revised December 2011.

CHANG, F.; DEAN, J.; GHEMAWAT, S.; HSIEH, W. C.; WALLACH, D. A.; BURROWS, M.; CHANDRA, T.; FIKES, A.; GRUBER, R. E. **Bigtable: a distribu-ted storage system for structured data**. In Proceedings of OSDI. 2006. Disponível em: < https://static.googleusercontent.com/media/research.google.com/pt-BR//archive/bigtable-osdi06.pdf>. Acesso em 17 nov. 2018.

CLARENCE, J. M. Tauro; ARAVINDH, S; SHEEHARSHA, A. B. **Comparative Study of the New Generation, Agile, Scalable, High Performance NOSQL Database**, International Journal of Computer Applications, V. 48, N.20. 2012. Disponível em: <https://pdfs.semanticscholar.org/1206/f9d1c02db2d5ef573c30e0982092c201c6c1.pdf>. Acesso em: 03 set. 2018.

CULTURAMIX. Disponível em: <http://www.culturamix.com>. Acesso em: 15 nov. 2018.

CUNHA, T. M. de A. **Escalabilidade de Sistemas com Banco de Dados NoSQL: um Estudo de Caso Comparativo com MongoDB e MySQL**. Trabalho de Conclusão de Curso (Ciência da Computação). Salvador: Centro Universitário Estácio. 2011.

DAS, C.; MOHAN, G.; ROY, R.; BHATTACHARYA, S. **Quo vadis, SaaS a system dynamics model based enquiry into the SaaS industry**. Information Management and Engineering (ICIME), The 2nd IEEE International Conference. p.732-737. 2010. Disponível em: <https://ieeexplore.ieee.org/document/5477966>. Acesso em: 03 set. 2018.

DATE, C. J. **Introdução a Sistemas de Bancos de Dados.** 8ª ed. Rio de Janeiro: Elsevier. 2004.

DEVMEDIA. **Repositório de Dados Relacional ou NoSQL?** 2018. Disponível em: <https://www.devmedia.com.br/> Acesso em: 15 set. 2018.

ELMASRI, R; NAVATHE, S, B. **Sistemas de Bancos de dados**. 6ª ed. São Paulo: Addison Wesley. 2011.

EVANS, E. **NoSQL**. 2009. Disponível em: < http://blog.symlink.com/> Acesso em: 22 set. 2018.

GUNTER, D.; et al. **Performance Evaluation of a MongoDB and Hadoop Platform for Scientific Data Analysis**. In: ScienceCloud'13, New York, NY, USA. 2013. Disponível em: < http://datasys.cs.iit.edu/events/ScienceCloud2013/p02.pdf>. Acesso em: 20 set. 2018.

GYORODI, C.; et al. **A Comparative Study: MongoDB vs. MySQL**. In: 13th International Conference on Engineering of Modern Electric Systems (EMES). 2015. <https://www.researchgate.net/publication/278302676_A_Comparative_Study_MongoDB_vs_MySQL>. Acesso em: 20 set. 2018.

HECHT, Robin; JABLONSKI, Stefan. **NoSQL Evaluation A Use Case Oriented Survey.** In: International Conference on Cloud and Service Computing. 2011. Disponivel em: <https://www.researchgate.net/publication/254048347_NoSQL_evaluation_A_use_case_ oriented_survey>. Acesso em: 20 set. 2018.

JSON. **Introducing JSON**. 2018. Disponível em: <http://www.json.org/>. Acesso em: 17 set. 2018.

KLINE, K. **SQL in a nutshell**, 3rd ed. O'Reilly Media. 2008.

LAZZARETTI, T. A. **Integração de Banco de Dados e Modelos de Simulação de Culturas Para Estimar o Impacto de Mudanças do Clima no Ren-dimento de Grãos e na Severidade da Giberela em Trigo**. Tese (Doutorado em Fitopatologia) -

Universidade de Passo Fundo, Passo Fundo. 2013. Disponível em: <http://tede.upf.br/jspui/handle/tede/434>. Acesso em: 17 set. 2018.

LEAVITT, N. (2010). **Will NoSQL Databases Live Up to Their Promise?** Published by the IEEE Computer Society, 2010. p. 12–14. Disponível em: <http://leavcom.com/pdf/NoSQL.pdf>. Acesso em: 17 set. 2018.

LENNON, J. **MongoDB**. 2013. Disponível em: <http://alfavillecode.blogspot.com.br/2013/01/MongoDB.html>. Acesso em: 06 mai, 2018.

LI, Xiang; MA, Zhiyi; CHEN Hongjie. **QODM: A Query-Oriented Data Modeling Approach for NoSQL Databases**. In: IEEE Workshop on Advanced Research and Technology in Industry Applications (WARTIA). 2014. Disponível em: < https://www.researchgate.net/publication/286562080_QODM_A_query-oriented_data_modeling_approach_for_NoSQL_databases>. Acesso em: 06 abr. 2018.

MARCÁRIO, Carla Geovanna do N.; BALDO, Stefano Monteiro; **O Modelo Relacional**. Instituto de Computação – Universidade Estadual de Campinas. Campinas, São Paulo, 2005. Disponível em: < http://www.ic.unicamp.br/~geovane/mo410-091/Ch03-RM-Resumo.pdf>. Acesso em: 06 mai, 2018.

MONGODB. 2018. Disponível em: <http://mongodb.com/>. Acesso em: 16 set. 2018.

MONGODB. **Documents**. 2018. Disponível em: <https://docs.mongodb.com/manual/reference/ operator/query>. Acesso em: 15 nov. 2018.

MONGODB-MYSQL. 2018. Disponível em: < https://www.mongodb.com/compare/ mongodb-mysql>. Acesso em: 15 nov. 2018.

MONGODB-SUPPORT. 2018. Disponível em: < https://docs.mongodb.com/manual/ support >. Acesso em: 15 nov. 2018.

MYSQL. **Tipo de Dados**. 2018. Disponível em: <https://www.mysql.com/> Acesso em: 09 set. 2018.

_____. **Documents**. 2018. Disponível em: <https://dev.mysql.com/doc/refman/8.0/en/select.html>. Acesso em: 15 nov. 2018.

ORACLE. 2018. Disponível em: <https://www.oracle.com/support/lifetime-support/>. Acesso em: 15 nov. 2018.

PINHEIRO, D. R. S., SOUZA, D. S., VASCONCELOS, R. O., SILVA, F. S. **Comparativo entre Banco de Dados Orientado a Objetos e Banco de Dados Objeto Relacional**. 2009.

POLITOWSKI, C.; MARAN, V. **Comparação de Performance entre PostgreSQL e MongoDB.**. In: X Esc. Reg. de Banco de Dados (ERBD), São Fran-cisco do Sul, SC. 2014. Disponível em: < https://www.researchgate.net/>. Acesso em: 26 set. 2018.

POPESCU, A. **Presentation: NoSQL at CodeMash - An Interesting NoSQL categorization**. 2015. Disponível em <http://nosql.mypopescu.com/>. Acesso em: 12 set. 2018.

POSTGRESQL. 2017. Disponível em: <http://postgresql.com/>. Acesso em: 10 set. 2018.

SANDALAGE, P; FOWLER, M. **NoSQL Essentials**. 1ª ed. São Paulo: Novatec. 2013.

SHREINER, Geomar A.; DUARTE; Denio; MELLO, Ronaldo dos S. **SQLtoKeyNoSQL: A Layer for Relational to Key-based NoSQL Database Mapping**. In: International Conference on Information Inte-gration and Webbased Applications & Services, Brussels, Belgium. 2015. Disponível em: <https://dl.acm.org/citation.cfm?id=2837224>. Acesso em: 15 set. 2018.

SILBERSCHATZ, A.; KORTH, H. F.; SUDARSHAN, S. (2006). **Sistema de banco de dados**, Rio de Janeiro: Editora Elsevier. [on-line]. 2006. Disponível em: <https://www.ebah.com.br/content/ABAAAfRigAH/banco-dados-silberschatz>.Acesso em: 18 abr, 2018.

SLIDEPLAYER. Disponível em: <https://slideplayer.com.br/>. Acesso em: 15 nov. 2018.

SRIVASTAVA, P. P.; GOYAL, S.; KUMAR, A. **Analysis of Vari-ous NoSql Database**. In: Dept. of Computer Science and Engineering Mody University of Science and Technology Rajasthan, India. 2015. Disponível em: < https://ieeexplore.ieee.org/abstract/document/7380523>. Acesso em: 12 set. 2018.

SUISSA, J. **Introdução ao NoSQL. NoSQL Br.** 2010. Disponível em: <http://www.sorojet.com.br/>. Acesso em: 17 set. 2018.

VOGELS, W. (2009). **Eventually consistent. Commun.** ACM, 52(1). 2009. p. 40–44. Disponivel em: <https://cacm.acm.org/>. Acesso em: 21 set. 2018.

TAMANE, SharvarI. **Non-Relational Databases in Big Data**. In: Italian Conference on Theoretical Computer Science. Udaipur, India. 2016. <https://dl.acm.org/citation.cfm?id=2905194>. Acesso em: 17 set. 2018.

WINMANM, Joe. **Cloundonomics: The business value of clound computing**. New Jersey: John Wiley & Sons, INC. 2012.

XPLENTY. **Arquitetura em Nuvem**. Disponível em: <https://www.xplenty.com/>. Acesso em: 24 set. 2018.