

Guilherme Lucon Pinto

**UM ESTUDO COMPARATIVO ENTRE BANCO DE DADOS  
RELACIONAL EM DISCO E EM MEMÓRIA**

Trabalho de Conclusão de Curso  
submetido à Universidade Federal de  
Santa Catarina para a obtenção do Grau  
de Bacharel em Tecnologias da  
Informação e Comunicação.

Orientador: Prof. Dr. Alexandre  
Leopoldo Gonçalves.

Araranguá  
2016

Ficha de identificação da obra elaborada pelo autor,  
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Pinto, Guilherme Lucon

Um estudo comparativo entre banco de dados relacional  
em disco e em memória / Guilherme Lucon Pinto ; orientador,  
Alexandre Leopoldo Gonçalves - Araranguá, SC, 2016.  
64 p.

Trabalho de Conclusão de Curso (graduação) -  
Universidade Federal de Santa Catarina, Campus Araranguá.  
Graduação em Tecnologias da Informação e Comunicação.

Inclui referências

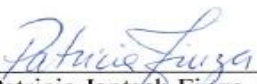
1. Tecnologias da Informação e Comunicação. 2. Banco de  
dados Relacional. 3. Banco de dados em Memória. 4. Sistemas  
de Armazenamento. I. Gonçalves, Alexandre Leopoldo. II.  
Universidade Federal de Santa Catarina. Graduação em  
Tecnologias da Informação e Comunicação. III. Título.

Guilherme Lucon Pinto

## **UM ESTUDO COMPARATIVO ENTRE BANCO DE DADOS RELACIONAL EM DISCO E EM MEMÓRIA**


Esta Monografia foi julgada adequada para obtenção do Título de “Bacharel em Tecnologias da Informação e Comunicação”, e aprovada em sua forma final pelo Curso de Graduação em Tecnologias da Informação e Comunicação.

Araranguá, 07 de Dezembro de 2016.

  
\_\_\_\_\_  
Prof. Patricia Jantsch Fiuza, Dra.  
Coordenadora do Curso

### **Banca Examinadora:**

  
\_\_\_\_\_  
Prof. Alexandre Leopoldo Gonçalves, Dr.  
Orientador  
Universidade Federal de Santa Catarina

  
\_\_\_\_\_  
Prof.<sup>a</sup> Olga Yevseyeva, Dr.<sup>a</sup>  
Universidade Federal de Santa Catarina

  
\_\_\_\_\_  
Prof. Vinicius Faria Culmant Ramos, Dr.  
Universidade Federal de Santa Catarina



Este trabalho é dedicado a todos que direta ou indiretamente contribuíram em minha formação acadêmica.



## **AGRADECIMENTOS**

Agradeço a todos que contribuíram no decorrer desta jornada, em especial:

À minha família pelo apoio fornecido e por nunca medirem esforços para que eu pudesse alcançar meus sonhos.

À minha namorada, Amanda, por estar sempre ao meu lado nos momentos tristes e felizes, me dando apoio, sempre.

Ao Orientador Prof. Dr. Alexandre Leopoldo Gonçalves pelo estímulo e dedicação em minha formação acadêmica.

Aos meus amigos pelo companheirismo e incentivo durante esta jornada.





Um bom começo é a metade.  
- Aristóteles



## RESUMO

No atual cenário da Tecnologia da Informação vem se produzindo e consumindo cada vez mais dados. Estima-se que entre 2010 e 2020 o volume de dados deverá aumentar 40% ao ano. Estes dados são produzidos em alta velocidade e necessitam de tratamento em tempo hábil. Em decorrência do volume e da velocidade, os dados ultrapassam a capacidade de processamento dos bancos de dados convencionais. Apesar de nas últimas décadas a capacidade de processamento ter crescido conforme a projeção da lei de Moore, o desempenho dos sistemas de armazenamento não foi capaz de acompanhar tal evolução. Para aumentar a capacidade de obter e analisar grandes volumes de dados é necessário utilizar métodos de análise adequados que suportem o incremento no volume de dados. Este trabalho possui como objetivo realizar um estudo comparativo, com base em aspectos gerais de desempenho, entre bancos de dados relacionais em disco e em memória. Para tal, utilizou-se um sistema para realizar a carga dos dados nos bancos de dados escolhidos, bem como, foram realizadas consultas para analisar o desempenho em operações de leitura e escrita nos bancos de dados escolhidos. Com base nos resultados obtidos, observa-se que quanto maior a base de dados, maior o impacto causado pelas baixas velocidades de leitura e escrita em disco rígido. A partir disso, acredita-se que as aplicações responsáveis por armazenar e analisar dados serão baseadas em memória principal.

**Palavras-chave:** Banco de Dados Relacional, Banco de Dados em Memória, Sistemas de Armazenamento.



## ABSTRACT

In the current scenario of Information Technology has been producing and consuming more and more data. It is estimated that between 2010 and 2020 the volume of data is expected to increase by 40% per year. These data are produced at high speed and require timely treatment. Due to volume and speed, data exceeds the processing capacity of conventional databases. Although in the last decades the processing capacity has grown according to the projection of Moore's law, the performance of the storage systems was not able to follow this evolution. To increase the capacity to obtain and analyze large volumes of data, it is necessary to use appropriate analysis methods to support the increase in data volume. This work aims to perform a comparative study, based on general aspects of performance, between relational databases in disk and in memory. To do this, a system was used to perform the data loading in the chosen databases, as well as, queries were carried out to analyze the performance in read and write operations in the chosen databases. Based on the results obtained, it is observed that the larger the database the greater the impact caused by the low read-write speeds on the hard disk. From this it is believed that the applications responsible for storing and analyzing data will be based on main memory.

**Keywords:** Relational Databases, In-Memory Databases, Storage Systems.



## LISTA DE FIGURAS

Figura 1: Procedimentos para a criação de um banco, criação de tabelas e de manipulação de dados .....	38
Figura 2: Conceito de banco de dados em memória .....	40
Figura 3: Gráfico de crescimento da plataforma MovieLens.....	45
Figura 4: Modelo lógico do Banco de Dados Relacional.....	47
Figura 5: Diagrama de Sequência .....	49
Figura 6: Conexão com o banco de dados através do SQL Developer .....	50
Figura 7: Carga da estrutura em disco para a memória.....	51





## LISTA DE TABELAS

Tabela 1: Tabela não normalizada .....	33
Tabela 2: Tabela de Projetos .....	34
Tabela 3: Tabela da relação entre Projeto e Empregado .....	34
Tabela 4: Nova tabela para a relação entre Projeto e Empregado .....	35
Tabela 5: Tabela de Empregados .....	35
Tabela 6: Tabela de Categoria .....	36
Tabela 7: Tabela Empregado resultante da 3FN .....	36
Tabela 8: Tempos de população do banco de dados em disco .....	52
Tabela 9: Tempos de população do banco de dados em memória .....	53
Tabela 10: Comparativo entre os tempos de inserção nos bancos de dados .....	54
Tabela 11: Comparativo de seleções entre os bancos de dados .....	55
Tabela 12: Sumarização dos resultados .....	55



## LISTA DE QUADROS

Quadro 1: Lista de bancos de dados em memória (1) .....	41
Quadro 2: Lista de bancos de dados em memória (2) .....	42

## LISTA DE ABREVIATURAS E SIGLAS

ACID – Atomicidade, Consistência, Isolamento e Durabilidade.

API – *Application Programming Interface*.

ASKAP – *Australian Square Kilometer Array Pathfinder*.

BD – Banco de Dados.

DDL - *Data Definition Language*.

DDL – *Data Definition Language*.

DML - *Data Manipulation Language*.

DML – *Data Manipulation Language*.

IMDB – *Internet Movie Database*.

IoT – *Internet of things*.

JDBC – *Java Database Connectivity*.

JSON–*JavaScript Object Notation*.

LHC – *Large Hadron Collider*.

RAM - *Random Access Memory*.

SEQUEL – *Structured English QUery Language*.

SGBD – Sistema de Gerenciamento de Banco de Dados.

SGBDR – Sistema de Gerenciamento de Banco de Dados Relacional.

SQL – *Structured Query Language*.

TBS – *Terminal Business System*.

UFSC – Universidade Federal de Santa Catarina.



## SUMÁRIO

<b>1 INTRODUÇÃO.....</b>	<b>23</b>
1.1 PROBLEMÁTICA .....	25
1.2 OBJETIVOS .....	26
1.2.1 Objetivo Geral .....	26
1.2.2 Objetivos Específicos .....	26
1.3 JUSTIFICATIVA .....	26
1.4 METODOLOGIA.....	27
1.5 ORGANIZAÇÃO DO TEXTO .....	28
<b>2 BANCO DE DADOS.....</b>	<b>29</b>
2.1 INTRODUÇÃO A BANCO DE DADOS .....	29
2.2 BANCO DE DADOS RELACIONAL .....	30
2.2.1 Modelo Relacional .....	31
2.2.2 Transações em banco de dados.....	32
2.2.3 Normalização .....	32
2.2.4 SQL .....	36
2.3 BANCO DE DADOS EM MEMÓRIA .....	40
2.3.1 Bancos em memória.....	41
<b>3 DESENVOLVIMENTO E AVALIAÇÃO DO TRABALHO .....</b>	<b>44</b>
3.1 APRESENTAÇÃO DO CENÁRIO .....	44
3.2 MODELO DE DADOS .....	46
3.2.1 Modelo de dados relacional.....	46
3.3 DETALHAMENTO DO PROTÓTIPO.....	48
3.4 ANÁLISE DOS RESULTADOS .....	51
<b>4 CONSIDERAÇÕES FINAIS .....</b>	<b>57</b>
<b>REFERÊNCIAS.....</b>	<b>59</b>







## 1 INTRODUÇÃO

O contexto atual da Tecnologia da Informação e a sua inserção nos mais variados setores tem promovido uma revolução na forma como as pessoas consomem e produzem dados. Os dados produzidos são provenientes das mais diversas fontes de dados (redes sociais, banco de dados, *intranet*, *smartphones*) e possuem diferentes formatos, podendo ser estruturados ou não estruturados.

Pesquisa realizada em 2014 pela empresa de consultoria em tecnologia, a IDC, aponta que o universo digital deverá crescer 40% ao ano na próxima década, gerando um volume expressivo de dados (IDC, 2014). Também segundo a IDC este aumento no universo digital representaria em 2020 cerca de 40 zetabytes de dados. Tal acréscimo significativo advém principalmente do aumento no uso das tecnologias da informação e comunicação (TICs) pelos diferentes setores da sociedade (GROBELNIK, 2012).

Neste contexto, surge o termo *Big Data*, com o objetivo de caracterizar o aumento expressivo no volume de informação. Apesar de não existir uma definição clara, vários autores têm contribuído para tal. Dumbill (2012), afirma que o termo é utilizado para descrever extensos volumes de dados que possuem diferentes formatos e que são manipulados em alta velocidade. O'Reilly (2012) enfatiza que o termo *Big Data* representa grandes volumes de dados que excedem a capacidade de processamento dos sistemas convencionais de banco de dados. Para Gartner (2014), extensos volumes de dados, alta velocidade, e dados de diferentes fontes e formatos exigem novas formas de processamento.

Os autores Fan e Bifet (2012) salientam que o conceito de *Big Data* vem recebendo atenção dos pesquisadores, tanto por parte da academia quanto da indústria, e, representa desafios devido à natureza dos dados ser evolutiva e volumosa. Em virtude desta natureza é necessário incluir novas maneiras advindas da tecnologia para analisar, tratar e processar os dados (O'REILLY, 2012). Isso ocorre não somente por conta do volume e velocidade de produção, mas devido aos formatos e esquemas distintos dos dados (FAN; BIFET, 2012; O'REILLY, 2012).

Para a empresa SAS®<sup>1</sup> o termo *Big Data* descreve o imenso volume de dados estruturados e não estruturados que impactam nos negócios e no dia a dia de uma organização. Contudo, não somente em casos envolvendo o mundo dos negócios o *Big Data* está inserido. O conceito de grande volume de dados e a evolução da Internet das Coisas

---

<sup>1</sup> [www.sas.com](http://www.sas.com)

(IoT, do inglês *Internet of Things*) passam a desempenhar um papel cada vez mais relevante na viabilidade de diversas iniciativas voltadas, por exemplo, para cidades e ambientes inteligentes. Isto se torna possível a partir da integração de sensores sem fio em ambiente reais utilizando redes de serviços. A combinação do IoT e *Big Data* representa desafios, mas também oportunidades na oferta de serviços que agreguem valor às organizações e aos usuários. Os desafios se concentram principalmente sobre problemas relacionados a negócios e tecnologias referentes a um ambiente inteligente (HASHEM et al., 2016).

Pode-se citar como outros exemplos o Colisor de Hádrons (LHC, do inglês *Large Hadron Collider*) que anualmente é capaz de gerar uma quantidade próxima a 30 petabytes de dados (CERN, 2016), ou o radiotelescópio ASKAP (*Australian Square Kilometer Array Pathfinder*, em inglês) que transmite a quantidade de 2.8 Gigabytes de dados por segundo (IDC, 2014). Ainda segundo a empresa IBM<sup>2</sup>, 2.5 quintilhões de bytes de dados são criados todos os dias através dos mais variados meios, como por exemplo, em sensores usados para obter informações climáticas, entre outros.

Apesar do poder computacional dos processadores ter crescido nas últimas décadas, conforme a perspectiva da lei de Moore, o desempenho dos sistemas de armazenamento não foi capaz de acompanhar tal evolução (JANUKOWICZ, EASTWOOD, 2012). Com base nessa discrepância e nos pontos principais do conceito de *Big Data*, acredita-se que as aplicações que lidam com esta perspectiva necessitam de uma infraestrutura capaz de ser escalável e paralela (MUKHERJEE et al., 2012). Hashem et al. (2016) ressaltam que o modelo de programação para o processamento de grandes conjuntos de dados com algoritmos paralelos pode ser usado para análise de dados e para obtenção de valor sobre os dados armazenados.

Tendo como objetivo aumentar a capacidade de obter e analisar grandes volumes de dados, é necessário o uso de métodos de análise adequados, sendo a técnica de *clustering* (agrupamento) um dos mais usados (BAHRILL; TIWARI; MALVIYA, 2016). Além dos métodos de *clustering*, também existem *frameworks* para lidar com o contexto atual de grandes volumes de dados, entre eles, o Apache Hadoop® (MUKHERJEE et al., 2012), e o Apache Spark® (BAHRILL; TIWARI; MALVIYA, 2016).

Porém, a busca de informações em meio a grandes quantidades de dados é difícil de ser realizada nos Sistemas de Gerenciamento de Banco

---

<sup>2</sup> [www.ibm.com](http://www.ibm.com)

de Dados (SGBD) atuais devido às entradas e saídas de dados no disco rígido dos servidores e à demora no acesso entre o dispositivo de armazenamento e o servidor de aplicação (PLATTNER, ZEIER, 2011). Com o objetivo de facilitar a busca de informações considerando extensos volumes de dados surge a tecnologia *in-memory*, permitindo carregar e executar todos os dados na memória, reduzindo o tempo de recuperação por determinada informação. De maneira geral, o investimento nesta tecnologia vem se tornando viável com a queda do custo das memórias de computador e passa a ser uma alternativa capaz de lidar adequadamente com o crescente volume dos dados.

## 1.1 PROBLEMÁTICA

Os bancos de dados são definidos como coleções de dados (BEYNON-DAVIES, 2004). O sistema que controla os dados, transações, problemas ou qualquer outro aspecto do banco de dados é denominado sistema de gerenciamento de banco de dados (SGBD). Entre os SGBDs, o modelo relacional é amplamente utilizado no gerenciamento de operações transacionais nas mais diversas aplicações. Ao longo dos anos sua arquitetura tem evoluído e proporcionado diversas ferramentas que facilitam o suporte ao desenvolvimento das mais variadas aplicações. O armazenamento dos dados em SGBDs relacionais é realizado em disco físico no formato de blocos (ELMASRI et al., 2005). Estes blocos são unidades para transferência de dados do disco ou para o disco contendo vários itens de dados.

Com o aumento significativo da quantidade de dados gerados com uso das Tecnologias da Informação e Comunicação (TICs) pelos diferentes setores da sociedade (GROBELNIK, 2012), as tecnologias de bancos de dados relacionais atuais estão se mostrando incapazes de acompanhar essa tendência, pois o volume de dados excede a capacidade de processamento dos sistemas convencionais de banco de dados (O'REILLY, 2012). Mesmo com o aumento da velocidade dos dispositivos de armazenamento o processo de salvamento e recuperação do dado é dispendioso, com um tempo médio de acesso na casa de 8,5ms. Com isso são necessárias novas formas de realizar o processamento dos dados (GARTNER, 2014).

Na busca por novas tecnologias capazes de armazenar, analisar e gerenciar grandes volumes de dados surge os bancos de dados baseados em memória (*in-memory database*). Os bancos de dados em memória

objetivam proporcionar: velocidade, flexibilidade, volume e disponibilidade.

A tecnologia em memória caracteriza-se como um conjunto de aplicações organizacionais capaz de impactar tanto em termos de funcionalidade quanto em redução de custo, principalmente devido ao aumento de desempenho (PLATTNER; ZEIER, 2011).

Sendo assim, tem-se a seguinte pergunta de pesquisa: **“Bancos de dados relacionais em memória são capazes de superar bancos de dados relacionais tradicionais considerando cenários que envolvam a leitura e a escrita de muitos dados?”**.

## 1.2 OBJETIVOS

### 1.2.1 Objetivo Geral

Este trabalho tem como objetivo principal realizar um estudo comparativo entre bancos de dados relacionais tradicionais e em memória considerando aspectos gerais de desempenho.

### 1.2.2 Objetivos Específicos

Com a finalidade de atingir o objetivo geral, têm-se os seguintes objetivos específicos:

- Realizar um levantamento bibliográfico de trabalhos relacionados na área de banco de dados tradicionais e em memória;
- Pesquisar SBGDs capazes de promover suporte a abordagem em memória;
- Desenvolver um ou utilizar um sistema capaz de realizar a carga de uma grande quantidade de dados utilizando a abordagem tradicional e a em memória;
- Analisar e discutir os resultados obtidos.

## 1.3 JUSTIFICATIVA

O aumento crescente dos dados devido aos avanços nas tecnologias da informação e comunicação no que tange a capacidade de processamento, armazenamento e transmissão de dados impõe novos desafios (GROBELNIK, 2012).

Tiwari (2011) aponta que com o crescimento no volume de dados, alguns desafios se tornam evidentes, tais como: a) melhorar a eficiência e o acesso a grandes volumes de dados; b) facilitar a manipulação de grandes conjuntos de dados; e c) flexibilizar a atual estrutura baseada em modelo de dados e metadados utilizada pelos bancos relacionais.

Neste cenário, a área de *Big Data* vem se consolidando como a base para o desenvolvimento de novas tecnologias capazes de manipular grandes conjuntos de dados visando promover subsídios para a tomada de decisão. A área de *Big Data* tem atraído a atenção da academia, indústria e governos ao redor do mundo (JIN et al., 2015).

Neste cenário vem se desenvolvendo a computação em memória, sendo os bancos de dados em memória um elemento importante desta infraestrutura computacional. Com isso, bancos de dados em memória desempenham papel importante na computação em tempo real, por meio da agregação de valor em negócios, uma vez que esta tecnologia viabiliza a diminuição de custos e conduz a novas possibilidades de se trabalhar com os dados (PLATTNER; ZEIER, 2011). Tornam-se, portanto, plataformas capazes de lidar com grandes volumes de dados em memória utilizando conceitos de computação distribuída ou em grade (grid), de modo a facilitar o desenvolvimento de soluções escaláveis aptas a manipular o crescente volume e complexidade dos dados.

## 1.4 METODOLOGIA

A metodologia de desenvolvimento deste trabalho está dividida em 4 etapas:

- Etapa 1: estabelecer uma base de estudo referencial teórico e literário sobre bancos de dados, tendo como foco principal bancos relacionais e em memória;
- Etapa 2: projetar um modelo que possa ser utilizado por um banco de dados relacional e em memória, visando a análise do desempenho dessas duas classes de SGBDs;
- Etapa 3: desenvolver um protótipo ou utilizar um sistema que realize a carga de dados nos bancos de dados escolhidos, criando subsídio para a análise de um cenário;
- Etapa 4: realizar um estudo comparativo de desempenho em operações de escrita e leitura dos modelos de banco utilizados neste trabalho.

## 1.5 ORGANIZAÇÃO DO TEXTO

Este trabalho está dividido em quatro capítulos. No primeiro capítulo é realizada uma breve contextualização do trabalho, a apresentação da problemática e dos objetivos, geral e específicos.

O segundo capítulo é dividido em duas partes. Na primeira é realizado um detalhamento dos bancos de dados relacionais que são amplamente utilizados em aplicações comerciais e de cunho geral. Já o enfoque da segunda parte é voltado a bancos de dados em memória visando superar limitações, principalmente no que se refere ao desempenho em consultas.

O terceiro capítulo apresenta o cenário desenvolvido e utilizado para este trabalho, bem como o estudo comparativo realizado visando responder a pergunta de pesquisa deste trabalho.

Por fim, o quarto capítulo contém as considerações finais e as perspectivas de trabalhos futuros.

## 2 BANCO DE DADOS

Neste capítulo será realizada uma explanação sobre a área de pesquisa do trabalho. Contextualizar-se-á a evolução histórica envolvendo a área de banco de dados.

### 2.1 INTRODUÇÃO A BANCO DE DADOS

Ao fazer o uso de um cartão de crédito ou gerar uma transação *online* de qualquer tipo, como uma compra em um *webshop* ou até mesmo o simples fato de acessar determinado site, ocorre uma interação com algum banco de dados (WADE; CHAMBERLIN, 2012).

A tecnologia de Banco de Dados, a partir de 1970, acabou se tornando a base para a criação de muitas aplicações, tanto de agências governamentais como da indústria em geral. Isso só foi alcançado porque essa tecnologia permitiu que os usuários criassem aplicações *online/desktop* de uma maneira mais rápida e com uma melhor relação entre custo e benefício (GRAD; BERGIN, 2009).

No entanto, como qualquer tecnologia, com o passar dos anos ela foi evoluindo e hoje existem diversos modelos em uso, como por exemplo:

- Relacional: o modelo de banco de dados relacional é uma coleção de múltiplos conjuntos de dados organizados em tabelas (linhas/colunas), possuindo uma relação bem definida entre as mesmas. A linguagem SQL (*Structured Query Language*) é usada por oferecer uma interface facilitada para interação com o banco (WADE; CHAMBERLIN, 2012).

- Em memória: o modelo de banco de dados em memória é um sistema de gerenciamento de banco de dados que, ao contrário dos modelos convencionais que utilizam a memória secundária do sistema, estes utilizam a memória principal para armazenar os dados. Sendo assim, esse tipo de modelo é usado principalmente em aplicações onde o tempo de resposta é um fator crucial (MOLKA; CASALE, 2015).

- Orientado a Documentos: No modelo de banco de dados orientado a documento, as informações são armazenadas no formato chave-valor em arquivos JSON, ou similares. Todo documento tem uma

chave única, que serve como meio para identificá-lo por toda a base de dados.

- Orientado a Objetos: Este modelo é um sistema de gerenciamento de banco de dados que suporta a modelagem e criação de dados como objetos. Estes podem ser referenciados de maneira similar ao que acontece com objetos usados na programação orientada a objetos (KIM, 1990).

Nas próximas seções serão apresentadas as tecnologias de banco de dados relacionais e banco de dados em memória, foco deste trabalho.

## 2.2 BANCO DE DADOS RELACIONAL

A história do surgimento dos Sistemas de Gerenciamento de Banco de Dados Relacionais (SGBDRs) segundo Wade e Chamberlin (2012), pode ser considerada uma das mais bem sucedidas e com impacto mais direto na vida das pessoas. E isso só aconteceu graças ao trabalho de Edgar Frank Codd no começo dos anos 70, quando passou a fazer parte do Laboratório de Pesquisas da IBM® em San José.

No entanto, Darwen (2012) diz que, antes da existência desse modelo a IBM® já trabalhava com um modelo de Sistema de Gerenciamento de Banco de Dados (SGBD) parecido, só que mais limitado, chamado de “*Terminal Business System*” (TBS). O TBS continha métodos de acesso que poderiam ser processados tanto sequencialmente, como pelo uso de um par chave-valor ou através de ponteiros.

Este modelo fornecia alguns programas utilitários de uso geral, escritos na linguagem própria do TBS, para manutenção e geração de relatórios. O utilitário de manutenção permitia por parte dos usuários a adição, alteração e exclusão de registros de arquivos. Já o de geração de relatórios, como o nome já diz, permitia a geração de relatório a partir de determinado arquivo com a opção de escolha de registros a serem apresentados, e cálculos a serem feitos como totais e subtotais e outros itens (DARWEN, 2012).

Entretanto, o que diferenciava este modelo dos que seriam criados eram as suas limitações:

- Somente um arquivo poderia ser acessado por vez;
- Os registros no arquivo tinham que estar todos no mesmo formato;



- Esses registros não podiam apontar para outros registros, nem mesmo em registros contidos no mesmo arquivo.

No início dos anos 1970, enquanto Codd trabalhava no laboratório de pesquisas da IBM® em San Jose, ele escreveu um artigo que propunha um novo foco para o modo como a informação era armazenada em computadores chamado de “*A Relational Model of Data for Large Shared Databanks*”, o qual foi batizado de “Modelo de Dados Relacional”, por ter sido baseado na teoria matemática das relações. Também descreveu as chamadas “Formas Normais”, que teriam como objetivo eliminar a redundância e evitar inconsistência no conteúdo armazenado no banco de dados (WADE; CHAMBERLIN, 2012).

Contudo, de acordo com Wade e Chamberlin (2012) houve uma demora, em torno de dez anos, para que houvesse uma aceitação em larga escala dos sistemas comerciais de bancos de dados. A ideia principal do modelo relacional projetado por Codd foi baseada no chamado “Princípio da Informação”, o qual diz que: “Toda informação contida no banco de dados deve ser representada única e somente uma vez, através de valores contidos em colunas dentro das linhas das tabelas”.

Através da implementação dessa ideia, acabou tornando-se possível um aumento de produtividade dos programadores, pois ao abstrair algumas estruturas físicas de dados, como índices de conteúdo, a criação de ponteiros e outras estruturas necessárias para obtenção de um melhor desempenho, o programador passava então a necessitar de um modelo de dados composto de valores em tabelas (WADE; CHAMBERLIN, 2012).

### **2.2.1 Modelo Relacional**

O conceito de Modelo Relacional foi criado por Edgar Frank Codd em 1970, sendo descrito no artigo “*Relational Model of Data for Large Shared Data Banks*”, usando como base o conceito de relação matemática, da lógica e da teoria de conjuntos. Pode ser considerado o sucessor do modelo hierárquico.

Segundo Strawn e Strawn (2016), o modelo relacional fornece uma separação clara entre como a representação física e lógica da informação é representada. Com os dados sendo projetados como uma tabela e ficando a cargo do sistema de gerenciamento de banco de dados determinar uma maneira de como armazenar essas informações. Para suportar o modelo relacional outros elementos tornam-se importantes,

entre eles o esquema de controle de transações, a normalização dos dados e as linguagens de definição e consulta de dados.

### **2.2.2 Transações em banco de dados**

O gerenciamento de dados tem um papel crítico em qualquer organização. Através da utilização de um SGBD é possível obter um ambiente propício para o armazenamento e recuperação de dados de uma maneira eficiente e econômica (SUMATHI; ESAKKIRAJAN, 2007).

Ainda segundo Sumathi e Esakkirajan (2007), dentro desse ambiente é possível que um usuário faça um conjunto de instruções de leitura e escrita no banco de dados. Os quais são denominados de transações.

É necessário que um SGBD garanta a execução de maneira adequada de determinada transação, ou seja, ela deve ser efetuada de maneira completa ou caso aconteça algum problema, nenhuma parte dela deve ser aceita (SILBERSCHATZ et al., 2006).

Segundo Silverschatz et al. (2006) e Sumathi e Esakkirajan (2007), para que a integridade dos dados seja garantida é necessário que as transações mantenham as propriedades conhecidas como ACID:

- Atomicidade: Todas ou nenhuma as operações contidas em uma transação são refletidas corretamente no banco de dados;
- Consistência: A execução de uma transação de maneira isolada (sem a interferência de outra transação) mantém a consistência do banco de dados;
- Isolamento: Ainda que seja possível que várias transações executem de maneira simultânea, duas transações quaisquer  $T_i$  e  $T_j$  não conhecimento o que cada uma está processando;
- Durabilidade: Após o término de uma transação com sucesso, todas as informações modificadas por esta devem ser persistidas no banco de dados.

### **2.2.3 Normalização**

Normalização é o processo de reorganização de dados que tem como objetivo a eliminação de redundâncias que possam existir e, fazer com que os dados que tenham relação sejam armazenados em conjunto. Esse processo é realizado através das chamadas formas normais, que

podem ser divididas em diversos casos, sendo a primeira, segunda e terceira formas normais as principais (HEUSER, 2009).

**Forma Não Normal (FNN):** Uma tabela está na forma não normal, ou não normalizada, quando não está na primeira, na segunda ou na terceira forma. Isto significa que a mesma possui pelo menos dois valores numa mesma linha e em uma mesma coluna. São também chamadas de tabelas aninhadas ou campos multivalorados. A Tabela 1 apresenta um exemplo em que a coluna **Emp** pode ser subdividida em outras tabelas. Neste sentido, diz-se que a coluna **Emp** possui uma tabela aninhada, ou que esta, não se caracteriza por conter somente valores atômicos.

Tabela 1: Tabela não normalizada

CodProj	Tipo	Descrição	Emp					
			CodEmp	Nome	Cat	Sal	DataIni	Temp Al
LSC001	Novo Desenv.	Sistema de Estoque	2146	João	A1	4	1/11/91	24
			3145	Silvio	A2	4	2/19/91	24
			6126	José	B1	9	3/19/92	18
			1214	Carlos	A2	4	4/10/92	18
			8191	Mario	A1	4	1/11/92	12
PAG02	Manut.	Sistema de RH	8191	Mario	A1	4	1/05/93	12
			4112	João	A2	4	4/01/91	24
			6126	José	B1	9	1/11/92	12

Fonte: Heuser (2009).

**Primeira Forma Normal (1FN):** Para que se atinja esta forma normal deve-se garantir que todos os domínios básicos contenham somente valores atômicos (não pode haver grupos repetidos), tão pouco deve existir tabelas aninhadas. O processo objetiva criar uma nova relação interligada por uma chave primária (campo único e responsável por identificar a tabela) e estrangeira (referência a uma chave primária de outra tabela). Desta forma, as tabelas 2 e 3 são o resultado da aplicação da 1FN, na tabela não normalizada. Como pode ser observado na tabela de relacionamento entre Projeto e Empregado (Tabela 3) esta possui uma ligação com a tabela de projetos (Tabela 2) através do **CodProj**.

Tabela 2: Tabela de Projetos

CodProj	Tipo	Descrição
LSC001	Novo Desenv.	Sistema de Estoque
PAG02	Manutenção	Sistema de RH

Fonte: Heuser (2009).

Tabela 3: Tabela da relação entre Projeto e Empregado

CodProj	CodEmp	Nome	Cat	Sal	DataIni	TempAl
LSC001	2146	João	A1	4	1/11/91	24
LSC001	3145	Silvio	A2	4	2/10/91	24
LSC001	6126	José	B1	9	3/10/92	18
LSC001	1214	Carlos	A2	4	4/10/92	18
LSC001	8191	Mário	A1	4	1/11/92	12
PAG02	8191	Mário	A1	4	1/05/93	12
PAG02	4112	João	A2	4	4/01/91	24
PAG02	6126	José	B1	9	1/11/92	12

Fonte: Heuser (2009).

**Segunda Forma Normal (2FN):** Além da tabela se encontrar na primeira forma, é preciso garantir que não existam dependências parciais, ou seja, cada coluna não chave depender de parte da chave primária. Caso existam valores repetidos para vários registros, os mesmos devem ser transferidos para uma nova tabela, sendo vinculados por uma chave estrangeira.

Considerando a tabela de relacionamento nota-se que a mesma possui campos, como **Nome**, **Cat** e **Sal** que se caracterizam como informações referentes somente ao empregado. Neste sentido, ao se aplicar a 2FN as colunas que dependem de parte da chave e a(s) coluna(s) da chave primária que promovem a dependência parcial são

transformadas em uma nova tabela (Tabela 5). Na tabela de relacionamento (Tabela 4) a(s) coluna(s) que promove(m) a dependência parcial, neste caso a coluna **CodEmp**, é(são) transformada(s) em chave estrangeira criando um relacionamento entre as tabelas.

Tabela 4: Nova tabela para a relação entre Projeto e Empregado

CodProj	CodEmp	DataIni	TempAl
LSC001	2146	1/11/91	24
LSC001	3145	2/10/91	24
LSC001	6126	3/10/92	18
LSC001	1214	4/10/92	18
LSC001	8191	1/11/92	12
PAG02	8191	1/05/93	12
PAG02	4112	4/01/91	24
PAG02	6126	1/11/92	12

Fonte: Heuser (2009).

Tabela 5: Tabela de Empregados

CodEmp	Nome	Cat	Sal
2146	João	A1	4
3145	Silvio	A2	4
6126	José	B1	9
1214	Carlos	A2	4
8191	Mário	A1	4
8191	Mário	A1	4
4112	João	A2	4
6126	José	B1	9

Fonte: Hauser (2009).

**Terceira Forma Normal (3FN):** A tabela, para que seja considerada na terceira forma normal, deve obrigatoriamente estar na segunda forma normal. O processo de transformação de 2FN para 3FN indica que todas as colunas dependentes indiretamente da chave primária da tabela devam ser movidas para uma nova tabela ou removidas.

Exemplo disso é a coluna **Sal** (Tabela 5) que depende indiretamente da coluna **CodEmp** através da coluna **Cat**. Ou seja, se diz que existe uma dependência transitiva. Nesta situação, é gerada uma nova tabela com a coluna transitiva e a outra coluna que dela depende diretamente produzindo a tabela Categoria (Tabela 5). A tabela

**Empregado** gerada mantém a coluna **Cat** que referencia a chave primária na tabela de **Categoria** (Tabela 5).

Tabela 6: Tabela de Categoria

Cat	Sal
A1	4
A2	4
B1	9

Fonte: Hauser (2009).

Tabela 7: Tabela Empregado resultante da 3FN

NumEmp	Nome	Cat
2146	João	A1
3145	Silvio	A2
6126	José	B1
1214	Carlos	A2
8191	Mário	A1
8191	Mário	A1
4112	João	A2
6126	José	B1

Fonte: Hauser (2009).

## 2.2.4 SQL

Segundo Elmasri e Navathe (2011) um dos fatores que foram responsáveis por alavancar os bancos de dados relacionais ao patamar de sucesso alcançado, foi a criação da linguagem SQL. O motivo pela qual a SQL ter sido amplamente estabelecida como padrão, era o fato de facilitar a migração entre diferentes estruturas de SGBD's relacionais.

Em um primeiro momento, SQL era chamada SEQUEL (*Structured English QUERy Language*) tendo sido desenvolvida para funcionar como interface para um banco de dados relacional experimental, chamado de SYSTEM R, da IBM®. Hoje, SQL é a abreviação de *Structured Query Language* (Linguagem de Consulta Estruturada) (ELMASRI; NAVATHE, 2010).

A linguagem SQL possui dois grupos principais de instruções que podem ser classificados como: DDL - *Data Definition Language* (Linguagem de Definição de Dados) e DML - *Data Manipulation*

*Language* (Linguagem de Manipulação de Dados) (RAMAKRISHNAN; GEHRKE, 2008), (SILBERSCHATZ et al., 2006).

Para Elmasri e Navathe (2011), as instruções (ou comandos) DDL são utilizadas para a definição da estrutura do modelo físico do banco de dados, através de três comandos básicos:

- **CREATE**: responsável pela criação de estruturas/objetos do banco, tais como, as tabelas;
- **ALTER**: responsável por modificar determinada estrutura/objeto já existente. Por exemplo, no caso de uma tabela, possibilita adicionar/remover colunas e até mesmo mudar o tipo de dados de uma coluna;
- **DROP**: diferente do comando “ALTER”, o comando “DROP” é responsável pela remoção de determinada estrutura/objeto por completo, independente se esta estrutura/objeto possui ou não conteúdo.

Já as instruções DML, como o nome indica, são responsáveis pela manipulação de dados do banco, através de inserções, atualizações e remoções, como também, para execuções de consultas. Utiliza-se para tal de quatro comandos básicos:

- **INSERT**: responsável pela inserção de um ou mais registros nas tabelas do banco;
- **UPDATE**: através deste comando é possível atualizar o valor de uma ou mais linhas em uma tabela. A determinação de filtros ou não indica quais linhas devem ser atualizadas;
- **DELETE**: possibilita a remoção de uma ou mais linhas de determinada tabela através da especificação ou não de alguma condição. A condição geral “\*” exclui todas as linhas de determinada tabela desde que estas não estejam relacionadas a outras tabelas;
- **SELECT**: comando usado para extrair dados específicos de uma ou mais tabelas (é o comando mais utilizado na linguagem SQL).

No caso dos comandos DELETE e UPDATE existe ainda a condição de propagação chamada de CASCADE. Para o DELETE, caso determinada linha seja excluída todas as linhas relacionadas serão excluídas em cascata. O mesmo vale para o comando UPDATE.

Pode-se observar na Figura 1 um conjunto de operações onde os dois tipos de instruções SQL são executadas:

Figura 1: Procedimentos para a criação de um banco, criação de tabelas e de manipulação de dados

```

CREATE DATABASE TCC;

CREATE TABLE Tcc (
Id_tcc integer PRIMARY KEY,
titulo varchar(255),
nota integer,
notas integer
);

CREATE TABLE Aluno (
Id_aluno integer PRIMARY KEY,
nome varchar(255),
Id_tcc integer,
FOREIGN KEY(Id_tcc) REFERENCES Tcc (Id_tcc)
);

ALTER TABLE Tcc DROP COLUMN notas;

INSERT into Tcc (Id_tcc, titulo, nota) values (1, 'ab', 10);
INSERT into Tcc (Id_tcc, titulo, nota) values (2, 'bc', 7);
INSERT into Tcc (Id_tcc, titulo, nota) values (3, 'cd', 9);
INSERT into Aluno (Id_aluno, nome, Id_tcc) values (1, 'Teste Aluno', 1);

UPDATE aluno SET nome = 'Joao' where Id_aluno = '1';

```

Fonte: Autor.

Pode-se dividir a Figura 1 em dois blocos. No primeiro bloco constam os comandos DDL, responsáveis pela criação da estrutura do banco de dados e criação de algumas tabelas. Neste caso, o primeiro comando é responsável pela criação do próprio banco no SGBD, o segundo e terceiro irão criar novas tabelas no banco criado pelo comando anterior e, o quarto comando irá retirar uma coluna de uma tabela.

O segundo bloco é o bloco com os comandos DML, que são responsáveis pela manipulação de dados no banco. Neste bloco, os quatro primeiros comandos realizam uma inserção de dados nas duas tabelas criadas anteriormente e, o último comando efetua uma atualização de um valor já inserido na tabela.

Por fim, o último comando DML que não consta na imagem é o comando “SELECT”, responsável pela obtenção dos dados contidos no banco através das chamadas consultas. Este também possibilita realizar a



junção de várias tabelas para gerar resultados mais específicos. A seguir é apresentada a estrutura geral do comando:

**SELECT** \* | colunas **FROM** tabela(s) **WHERE** restrições **GROUP** by coluna(s) **ORDER BY** coluna(s) **HAVING** condição;

Onde:

- **SELECT**: Indica o conjunto de colunas que será projetado, ou seja, será retornado com resultado da consulta executada. O retorno pode ser composto por todas as colunas das tabelas envolvidas (opção “\*”) ou somente um conjunto das colunas;
- **FROM**: indica para o banco de dados, em qual(is) tabela(s) deve buscar as informações. Caso exista mais de uma tabela a condição de junção entre as mesmas pode ser definida nesta parte ou como uma restrição na cláusula **WHERE**;
- **tabela(s)**: indica o nome da(s) tabela(s) pertencente(s) ao banco de dados e que farão parte da consulta como requisito para se obter as informações desejadas. Na Figura 1 são exemplos as tabelas **Tcc** e **Aluno**;
- **WHERE**: indica qual(is) restrição(ões) será(ão) aplicada(s) na recuperação dos dados. A(s) restrição(ões) pode(m) ser de igualdade (=), intervalo (>, >=, <, <=, **BETWEEN**), negação (**NOT**) ou de junção;
- **GROUP BY**: utilizado sempre que na projeção sejam utilizadas funções de agregação (**SUM**, **AVG**, **MIN** e **MAX**) juntamente com colunas que não sofram a aplicação destas funções. Desta forma, se indica como ocorrerá determinada agregação dos dados;
- **ORDER BY**: permite a ordenação do conjunto resultante com base nas colunas projetadas e em determinada ordem. Uma coluna em particular pode ser ordenada de maneira crescente (**ASC**, opção padrão) ou decrescente (**DESC**);
- **HAVING**: uma vez que se tenha utilizado funções de agregação na projeção da consulta é possível filtrar o conjunto resultante aplicando uma restrição (filtro) sobre a coluna calculada a partir da agregação.

## 2.3 BANCO DE DADOS EM MEMÓRIA

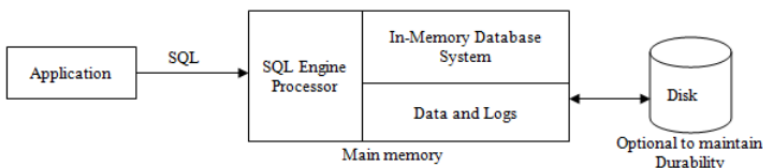
Segundo Wang et al. (2015) a capacidade das memórias RAM (*Random Access Memory*) vem aumentando consideravelmente enquanto que o preço vem decaindo. Tal fato possibilita armazenar completamente bancos de dados cada vez maiores na memória principal do sistema. Desta forma, o conceito de banco de dados em memória se torna cada vez mais real e uma tendência. Afirmações similares podem ser encontradas no início da década de 90 através do trabalho de Molina (1992).

Esse conceito se diferencia do modelo convencional por manter os dados na memória principal e não em disco, o que por sua vez acaba gerando uma melhora nos tempos de respostas das transações quando comparadas com a sua contraparte convencional baseada em disco.

Para Wang, Zhong e Kun (2015), outra característica que diferencia os bancos em memória dos convencionais é a falta de operações de “Entrada e Saída” (E/S), objetivando a eliminação desse possível gargalo. Pois, em um sistema baseado em disco onde a informação é acessada através de um leitor que é movimentado por um braço mecânico, não é possível garantir o desempenho necessário para operações mais complexas. Outros autores que concordam com essa afirmação, são Gupta, Verma e Verma (2013) ao ressaltarem que ao armazenar as informações na memória principal, eliminando as operações de E/S, a velocidade de leitura e escrita é melhorada consideravelmente.

Na Figura 2 é apresentado um conceito básico do funcionamento de um banco de dados em memória:

Figura 2: Conceito de banco de dados em memória



Fonte: Gupta, Verma e Verma (2013).

A Figura 2 pode ser dividida em três blocos principais, sendo que no primeiro é representada uma aplicação que faz uso do banco de dados. No segundo bloco é representada a memória principal do sistema, onde é armazenado o próprio banco, os dados e os registros de transações. É também a unidade responsável pelo processamento das consultas em SQL.

O último corresponde a um bloco opcional que representa um disco e/ou nós de processamento responsáveis por armazenar registros de operações, para que em caso de alguma falha mais graves seja possível restaurar as informações do banco.

### 2.3.1 Bancos em memória

Nos últimos anos, o mercado de banco de dados tem sido incrementado com uma variedade de soluções em memória (tanto versões proprietárias como *open-source*) de grandes nomes da indústria mundial, como a IBM®, a Oracle®, a SAP®, entre outros. Entretanto, por mais que todos suportem o padrão SQL e tenham a capacidade de manter o banco completamente em memória, eles apresentam recursos variados entre si (GUPTA; VERMA; VERMA, 2013).

Nos Quadro 1 e Quadro 2 são expostos alguns exemplos de bancos de dados em memória disponíveis atualmente.

Quadro 1: Lista de bancos de dados em memória (1)

IMDB Name	SQL 92 Standard	Concurrency	Data Size	Storage Type	System Architecture	Availability	Scalability
eXtremeDB	SQL-89	multi-version concurrency control(MVCC) 0.7 times/ms	Billion	No Translation	Embedded /Shared memory	Two phase synchronous mode	One pair of N memory mirroring
Oracle TimesTen	√	2.5 ms read 10 ms write Transaction isolation Locks,MVCC	Billion	Relational model based on row	C/S, Embedded	Transaction logging and database checkpoint	X/Open XA Specification Java Transaction API
Altiibase	√	MVCC	Billion	Multiple data formats	C/S	Copy, Switch, Write Ahead Logging	Star copy 1:32
SQLite	√	The entire database file for read / write lock	Million	Typelessness Five kinds of genetic types	Embedded	Lasting ACID	None
Berkeley DB	√	Support thousands of concurrent threads	Maximum 256TB	Any type of key / value pairs	Embedded	Two lock technology and write-ahead logging policy	None
FastDB	Object-Oriented	Concurrent read., not write concurrent	Million	Each row is considered an object instance	Embedded single	Memory data backup to disk file	None
H2	√	Concurrent read MVCC	Million	UUID and other 21 kinds of data formats	C/S Embedded	Database mirroring mechanism	Built-in Clustering / Replication

Fonte: Wang, Zhang e Kun (2015).

Quadro 2: Lista de bancos de dados em memória (2)

MEMCACHED	×	Multi-threaded locking mechanism to ensure the mutex of data update operation	Million	Key-Value Single Item maximum 1M	C/S Two-stage hash	Support data mutual replication backup	C/S, memcached doesn't share information distributed algor. require the client to complete the routing
Redis	×	No support, but support for transaction processing	Billion	Key-Value	C/S	Snapshotting Append-only file	Master-slave synchronization (e.g. Sina weibo)
MONGO DB	Compatible	per-database-level locking ACID support will reduce performance	Billion	Similarly json format, content is a document storage type	Cluster structure	Master – slave, master - master mode and data replication between servers	Support for level database cluster, and can dynamically add
HBASE	×	Map-reduce mode, no support the ACID	Billion	Column stores Big table	Distributed	Hadoop HDFS	Support for ultra-large-scale clusters
TOKYO CABINET /TYRANT	×	concurrency in multi-thread	Billion	Key-Value can be binary or strings	C/S	Dual-active main and auxiliary mode, support for long-lasting record	C/S, support for HTTP
LEVEL DB	√	Share Nothing. All stored procedures (affairs) are the global order, to avoid the use of locks	Billion	SkipList Key-Value	Distributed	Snapshot, logging record	Support for ultra-large-scale clusters
Aerospike	√	multi-key transaction	Billion	Key-Value Sets/Records	C/S, Cloud	Partition Tolerant Mode multi-server replication	cluster-aware
VOLTDDB	subset	Single-threaded parallel processing mode to ensure data consistency and avoid the use of locks	Billion	structured or unstructured data	Hash partition database	Snapshot, logging record	Support for clusters
MySQL+han	√	Gather request while	Billion	Key-Value	Support	Same with MySQL	Same with MySQL

Fonte: Wang, Zhang e Kun (2015).

Com base nessa lista é possível criar uma divisão entre os bancos e seus diferentes focos:

### Foco comercial:

- TimesTen: é um sistema de banco de dados em memória, proprietário da Oracle®, baseado no modelo relacional. Suas principais características são: suporte as propriedades ACID e um tempo de resposta quase instantâneo necessário para aplicações em tempo real, como aplicações de telecomunicações (ORACLE, 2016).

- eXtremeDB: é um sistema em memória, desenvolvido pela McObject®. Tem como principais características o suporte as propriedades ACID e a capacidade de ser usado em sistemas embarcados por utilizar pouquíssima memória (McObject, 2016).

### **Foco em sistemas embarcados:**

- SQLite: é um sistema em memória que tem como foco, dispositivos que contam com poucos recursos de espaço e memória, como por exemplo, *smartphones*. Foi desenvolvido por Dwayne Richard Hipp e tem como principais características ser um projeto *open source* (código aberto) e a não necessidade de um servidor fora do aparelho que o utiliza (OWENS, 2006).

- Berkeley DB: é um sistema baseado no conceito de chave-valor, desenvolvido pela Oracle®. Uma das características é a possibilidade de uso de uma API baseado no conceito chave-valor ao invés da linguagem SQL para o armazenamento de dados (ORACLE, 2016).

### **Foco em novas tecnologias:**

- MongoDB: é um sistema baseado no conceito de NoSQL, desenvolvido pela empresa de publicidade DoubleClick®. O modelo utilizado é o de orientação a documentos. É um projeto *open source* e utiliza a linguagem própria para realizar as operações de manutenção e consulta de dados ao invés da SQL (MONGODB, 2016).

- VoltDB: é um sistema baseado no conceito de NewSQL, desenvolvido por Mike Stonebraker com suporte as propriedades ACID. VoltDB é um projeto *open source* e tem o foco voltado a aplicações que necessitam de um baixo tempo de resposta entre transações, como por exemplo, aplicações financeiras e de telecomunicações (VOLTDB, 2016).

### 3 DESENVOLVIMENTO E AVALIAÇÃO DO TRABALHO

A partir da problemática apresentada, este trabalho tem como proposta a realização de um estudo comparativo entre os conceitos de Banco de Dados Relacional Tradicional (em disco) e Banco de Dados em Memória, tendo como foco os aspectos gerais de desempenho dessas duas classes.

Devido aos bancos de dados escolhidos serem baseados no modelo relacional, para que uma comparação e obtenção de dados quantitativos sejam possíveis, é proposto neste capítulo um único modelo de representação de dados, onde os testes serão realizados.

A aplicação desenvolvida realiza a carga dos dados extraídos de arquivos em formato “.csv” (cada um dos conteúdo é separado por “;”), representando informações sobre filmes contidos no Banco de Dados de Filmes da Internet (IMDB, do inglês *Internet Movie Database*). Esta base de dados é utilizada para gerar um *ranking* dos filmes por meio de avaliações atribuídas por usuários. Através da obtenção das informações foi realizada uma carga em ambos os bancos com o intuito de realizar um estudo comparativo avaliando-se o desempenho nos aspectos principais.

Para este estudo foi utilizado um computador pessoal com processador Intel® Core i5-6400 2.7GHz com Turbo Boost de 3.3GHz; 16GB de memória DDR4; disco (HDD) SATA 6Gb/s, com 7200 rpm, com 500GB de capacidade de armazenamento, Western Digital Caviar Blue; e Sistema Operacional Windows 10® 64-bits.

#### 3.1 APRESENTAÇÃO DO CENÁRIO

O aumento na quantidade de dados gerados e a necessidade que algumas aplicações têm de analisar esses dados em tempo real estão entre as principais demandas para bancos de dados em memória. Neste sentido optou-se pela plataforma de recomendação de filmes MovieLens®, que disponibiliza conjuntos de dados para serem utilizados em pesquisas, como no caso da pesquisa do autor (PERALTA, 2007). Esta plataforma utiliza como base as informações contidas no banco do IMDB.

A plataforma MovieLens foi inicialmente elaborada pelo Departamento de Ciência da Computação e Engenharia da Universidade de Minnesota em 1997. Funciona como um sistema de filtragem colaborativa colhendo as preferências dos usuários e os agrupando através de gostos similares de modo que estas informações possam ser utilizadas na realização de sugestões (PERALTA, 2007).

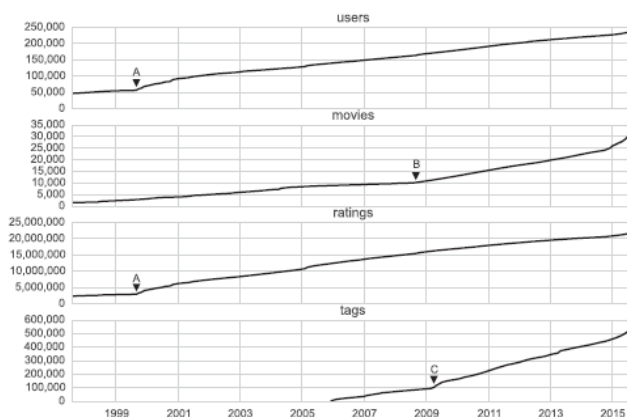
Segundo Rashid et al. (2002) sistemas de recomendação são recursos imprescindíveis para usuários que buscam uma maneira inteligente de obterem informações, ajudando os mesmos a tomarem decisões em cenários complexos de informação.

Partindo da ideia de facilitar a busca dos usuários sobre quais seriam os filmes mais indicados para si, a plataforma MovieLens utiliza uma técnica de filtragem colaborativa para fazer uma análise comparativa, através de sua ampla base de dados onde uma grande quantidade de informações sobre filmes é armazenada. São armazenados no banco de dados os títulos, avaliações aferidas pelos usuários da plataforma, gêneros e rótulos dos filmes (RASHID et al., 2002).

Segundo Harper e Konstan (2015), as informações contidas nos diferentes conjuntos de dados (100K, 1M, 10M, 20M e 24M), que são representados primariamente pela quantidade de avaliações (*ratings*) contida nos mesmos, além de serem utilizadas como base para o sistema de recomendação são usadas em larga escala como fonte de pesquisas tanto em instituições educacionais como na indústria.

Ao longo dos 17 anos, a plataforma, assim como seus conjuntos de dados, tem tido um crescimento estável, como pode ser visto na Figura 3, onde é possível verificar o aumento no número de usuários, filmes, avaliações e rótulos.

Figura 3: Gráfico de crescimento da plataforma MovieLens



Fonte: Harper e Konstan (2015).

Também é possível observar três eventos (A, B e C) distintos que ocorreram e tiveram impactos positivos para o crescimento:

- Evento A: houve uma aceleração do crescimento de usuários e avaliações devido a uma exposição da plataforma na mídia especializada;
- Evento B: a partir do evento A, houve uma ascensão no número de filmes devido a opção de adicionar novos títulos ter sido disponibilizada ao público;
- Evento C: o aumento no número de rótulos ocorreu em virtude do lançamento da propriedade de expressões de rótulos para os usuários, fazendo com que os usuários pudessem usar rótulos próprios.

Para a realização deste trabalho, o conjunto de dados escolhido foi o chamado “ML-Latest” sendo constituído por 24.000.000 avaliações, 67.0000 rótulos aplicados por usuários, 40.000 títulos de filmes e 260.000 usuários.

## 3.2 MODELO DE DADOS

Essa seção detalha o modelo de dados proposto com base nas informações representadas pelo conjunto de dados escolhido, visando fornecer suporte ao estudo comparativo dos conceitos relacional tradicional (em disco) e relacional em memória.

### 3.2.1 Modelo de dados relacional

O modelo de dados proposto é composto por um conjunto de tabelas relacionadas que representam determinado domínio de maneira estruturada. O modelo de dados desenvolvido e considerado para o presente trabalho possui as seguintes relações:

- Cadastro de Usuário: estabelece uma única informação com a função de promover um identificador de usuários;
- Cadastro de Filmes: contém um identificador, um campo para o título do filme e outro campo para os possíveis gêneros;
- Cadastro de Relacionamento de Avaliações entre Usuário e Filme: composto por dois campos que fazem referência aos

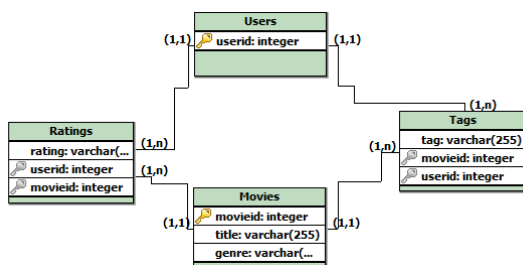


identificadores de usuário e filme e um campo para a indicação de avaliações.

- Cadastro de Rótulos: contém dois campos que fazem referência aos identificadores de usuário e filme e também um campo de texto para os rótulos.

De acordo com o modelo proposto, o banco de dados relacional foi projetado para comportar as informações requeridas conforme apresentado na Figura 4.

Figura 4: Modelo lógico do Banco de Dados Relacional



Fonte: Autor.

O cadastro de usuários é realizado na tabela *Users*, o qual é identificado apenas por um campo inteiro (*userId*) e, por se tratar de algo anônimo representa o usuário por um valor aleatório.

O cadastro de filmes é feito na tabela *Movies*, composta por um identificador sequencial (*movieId*) e dois campos descritivos, sendo um o próprio título, e o segundo a descrição do gênero (*genres*).

O relacionamento entre filmes e usuários é composto pela tabela de Avaliações (*Ratings*) que consiste de três campos inteiros, o primeiro (*rating*) é referente as notas (de 0 a 5) que usuários podem atribuir aos filmes e os outros dois (*userId* e *movieId*) fazem referência as suas respectivas tabelas, *Movies* e *Users*.

Por fim, tem-se a tabela de rótulos (*Tags*), composta por três campos, sendo dois destes inteiros (*userId* e *movieId*) e o terceiro campo textual, que tem como função armazenar os rótulos criados pelos usuários.

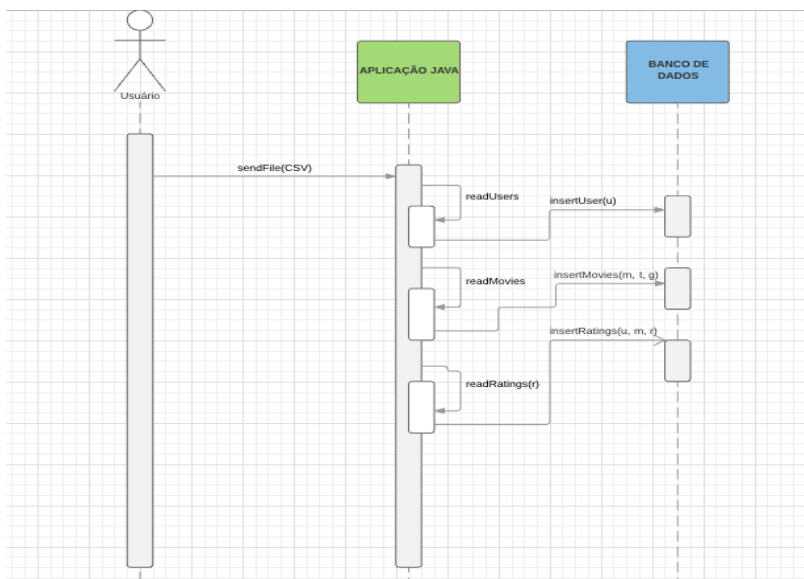
### 3.3 DETALHAMENTO DO PROTÓTIPO

Para o desenvolvimento deste protótipo foi utilizada a aplicação *SQL Developer*®, pois esta tem como objetivo facilitar o gerenciamento de bancos de dados, tanto tradicional como em memória, desenvolvidos pela Oracle®. O banco de dados Oracle XE® versão 11g foi utilizado para suportar o estudo do modelo tradicional (em disco), já para o modelo em memória o banco escolhido foi o TimesTen® versão 11.2.

Considerando que a fonte dos dados utilizada neste trabalho é composta por uma série de arquivos CSVs, os mesmos foram analisados para que pudessem ser replicados através da aplicação *SQL Developer*® encarregada de fazer a carga de dados no banco de dados tradicional. Por se tratarem de arquivos CSV, eles não necessitam estar de acordo com as formas normais e contém campos multi-valorados, o que pode dificultar uma simples leitura e transferência dos dados para o banco.

Na população dos bancos de dados tradicional (em disco) e em memória não foi utilizada a mesma metodologia. No primeiro utilizou-se o processo de carga a partir dos arquivos CSV utilizando uma aplicação Java®. Para o banco em memória, foi utilizada a característica de cópia da estrutura em disco para a memória da ferramenta *SQL Developer*®. Na Figura 5 é demonstrada a sequência realizada para a leitura dos arquivos da base e para fazer a carga dos mesmos no banco tradicional.

Figura 5: Diagrama de Sequência

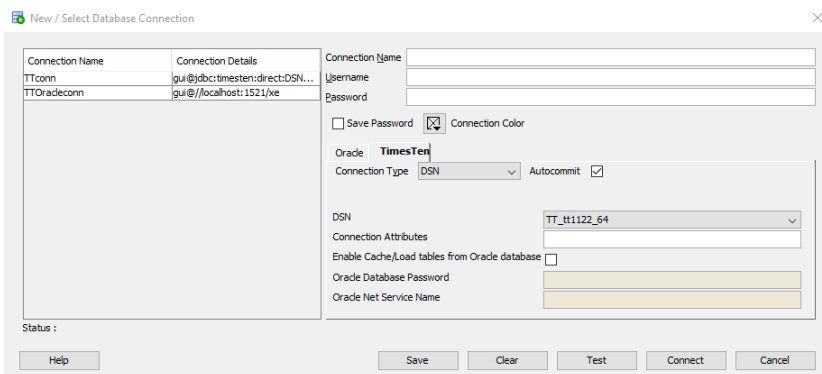


Fonte: Autor.

No diagrama apresentado na Figura 5, o usuário envia os arquivos para a aplicação a qual faz a leitura dos usuários, filmes, rótulos e das avaliações. Depois é feita a inserção dessas informações no banco, onde após esse processo é criada a relação entre as tabelas.

Devido a ambos os bancos serem relacionais foi possível utilizar um método de carga diferenciado no *TimesTen*, onde a aplicação *SQL Developer®* se conecta com o banco tradicional e realiza a carga de toda estrutura já criada em disco para a memória. Na Figura 6 e Figura 7, é possível verificar como esse processo é feito.

Figura 6: Conexão com o banco de dados através do SQL Developer

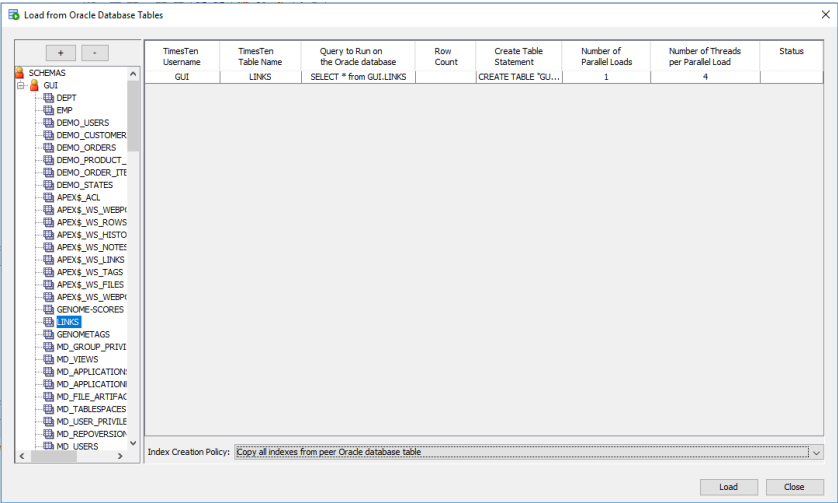


Fonte: Autor.

A Figura 6 apresenta a janela de conexões, sendo possível dividi-la em 2 grandes blocos. Na esquerda ficam as conexões já criadas com uma sumarização das informações das mesmas, como: nome da conexão, usuário, endereço do servidor, tipo de conexão entre outros dados. Na direita ficam as opções para alterar uma conexão existente ou criar uma nova, como: usuário e senha, tipo de banco a ser conectado, tipo de conexão (local, cliente/servidor).

No caso do banco TimesTen® e devido as características deste trabalho, também é necessário informar os dados do modelo em memória e os dados do banco tradicional.

Figura 7: Carga da estrutura em disco para a memória



Fonte: Autor.

Já a Figura 7 apresenta a janela onde consta a estrutura do banco em disco (todas as tabelas criadas), sendo possível escolher uma ou mais tabelas a serem carregadas. Também é possível verificar os comandos que serão utilizados para recuperar os dados e para criar a nova tabela em memória, sendo também possível o carregamento de estruturas de índices previamente criadas.

3.4 ANÁLISE DOS RESULTADOS

Para a avaliação do desempenho nos modelos acima citados e para o cenário proposto foram carregadas aproximadamente 25 milhões de entradas (divididas em usuários, avaliações, rótulos e filmes). A carga das avaliações foi dividida em três cargas distintas (6, 12 e 24 milhões de linhas) e o tempo de cada uma mensurado individualmente.

Para a análise de resultados a coleta de dados foi dividida em três etapas para cada uma das execuções: a) tempo gasto para a carga de dados em cada banco; b) tempo gasto para a execução de consultas; e c) tempo total decorrido.

Em cada uma das etapas de envolvendo consultas foram coletados os dados referentes às três tabelas de avaliações, sendo calculado também consultas com e sem índice (onde aplicável).

Nas cargas realizadas no modelo tradicional representadas na Tabela 8, a qual apresenta a carga no banco em disco, é possível visualizar que devido a natureza do banco os tempos de cada inserção se mantém relativamente proporcional as demais quando comparados com a quantidade de dados.

Tabela 8: Tempos de população do banco de dados em disco

<b>Tradicional (em disco)</b>			
<b>Etapa</b>	<b>Tempo (hh:mm:ss)</b>		<b>Quantidade de registros</b>
<b>1ª Etapa (25%) (6 Milhões de Avaliações)</b>	Inserir Usuários:	00:00:02s	260,000
	Inserir Filmes:	00:00:01s	40,000
	Inserir Rótulos:	00:00:07s	670,000
	Inserir Avaliações:	00:01:15s	6,000,000
<b>2ª Etapa (50%) (12 Milhões de Avaliações)</b>	Inserir Usuários:	00:00:02s	260,000
	Inserir Filmes:	00:00:01s	40,000
	Inserir Rótulos:	00:00:07s	670,000
	Inserir Avaliações:	00:02:28s	12,000,000
<b>3ª Etapa (100%) (24 Milhões de Avaliações)</b>	Inserir Usuários:	00:00:02s	260,000
	Inserir Filmes:	00:00:01s	40,000
	Inserir Rótulos:	00:00:07s	670,000
	Inserir Avaliações:	00:07:20s	24,000,000

Fonte: Autor.

Nas cargas realizadas no banco de dados em memória (Tabela 9) é possível observar que os tempos não apresentaram uma melhora considerável e, em alguns casos existe até uma demora maior quando comparado com a sua contra parte em disco. Isso deve ocorrer devido ao método de carga escolhido, onde é necessário ler as informações contidas em disco e replicá-las na memória.

Tabela 9: Tempos de população do banco de dados em memória

Em memória			
Etapa	Tempo (hh:mm:ss)		Quantidade de registros
<b>1ª Etapa (25%) (6 Milhões de Avaliações)</b>	Inserir Usuários:	00:00:03s	260,000
	Inserir Filmes:	00:00:01s	40,000
	Inserir Rótulos: Inserir Avaliações:	00:00:08s 00:01:10s	670,000 6,000,000
<b>2ª Etapa (50%) (12 Milhões de Avaliações)</b>	Inserir Usuários:	00:00:03s	260,000
	Inserir Filmes:	00:00:01s	40,000
	Inserir Rótulos: Inserir Avaliações:	00:00:08s 00:02:17s	670,000 12,000,000
<b>3ª Etapa (100%) (24 Milhões de Avaliações)</b>	Inserir Usuários:	00:00:03s	260,000
	Inserir Filmes:	00:00:01s	40,000
	Inserir Rótulos: Inserir Avaliações:	00:00:08s 00:07:03s	670,000 24,000,000

Fonte: Autor.

Na tabela 10 é realizada a comparação entre as diferenças de desempenho em tempo de execução, baseado nos dados coletados nas Tabela 8 e 9.

Através da comparação de tempo realizada, pode-se observar que a diferença para inserir os dados nas tabelas do modelo (Filmes, Usuários, Rótulos e Avaliações) não são expressivas. Em uma primeira análise é possível observar que não existem motivos para a mudança de um banco em disco para um baseado em memória. Testes adicionais são requeridos, por exemplo, o desenvolvimento de uma aplicação para realizar a carga ao invés de utilizar o *SQL Developer®*.

No entanto, quando a quantidade de dados passa a casa dos milhões, o desempenho do banco em memória acaba se evidenciando, chegando a criar uma diferença de até 17 segundos no caso da inserção de 24 milhões de entradas.

Tabela 10: Comparativo entre os tempos de inserção nos bancos de dados

Comparativo				
Etapa	Inserção	Tempo em Disco	Tempo em Memória	Diferença (Memória – Disco)
<b>1ª Etapa (25%) (6 Milhões de Avaliações)</b>	Usuários:	00:00:02s	00:00:03s	+00:00:01s
	Filmes:	00:00:01s	00:00:01s	00:00:00s
	Rótulos:	00:00:07s	00:00:08s	+00:00:01s
	Avaliações:	00:01:15s	00:01:10s	-00:00:05s
<b>2ª Etapa (50%) (12 Milhões de Avaliações)</b>	Usuários:	00:00:02s	00:00:03s	+00:00:01s
	Filmes:	00:00:01s	00:00:01s	00:00:00s
	Rótulos:	00:00:07s	00:00:08s	+00:00:01s
	Avaliações:	00:02:28s	00:02:17s	-00:00:11s
<b>3ª Etapa (100%) (24 Milhões de Avaliações)</b>	Usuários:	00:00:02s	00:00:03s	+00:00:01s
	Filmes:	00:00:01s	00:00:01s	00:00:00s
	Rótulos:	00:00:07s	00:00:08s	+00:00:01s
	Avaliações:	00:07:20s	00:07:03s	-00:00:17s

Fonte: Autor.

Para a avaliação do desempenho de busca dos bancos foram desenvolvidas quatro seleções, sendo duas com foco no modelo com o maior número de linhas (utilizando todas as quatro tabelas do modelo proposto), uma considerada intermediária (utilizando as tabelas *users*, *movies* e *ratings*) e uma considerando o modelo com o menor número de linhas (com a utilização das tabelas *movies* e *ratings*, somente). A Tabela 11 sumariza os resultados.



Tabela 11: Comparativo de seleções entre os bancos de dados

<b>Comparativo de Seleções</b>			
<b>Seleção</b>	<b>Tempo em disco</b>	<b>Tempo em Memória</b>	<b>Diferença (Memória – Disco)</b>
<b>1ª Seleção (25%) (6 Milhões de Avaliações)</b>	00:00:32s	00:00:18s	-00:00:14s
<b>2ª Seleção (50%) (12 Milhões de Avaliações)</b>	00:01:02s	00:00:42s	-00:00:20s
<b>3ª Seleção (100%) (24 Milhões de Avaliações)</b>	00:01:57s	00:01:12s	-00:00:45s
<b>4ª Seleção (100%) (24 Milhões de Avaliações)</b>	00:03:00s	00:01:45s	-00:01:15s

Fonte: Autor.

Através da comparação dos tempos das seleções, é possível notar que diferentemente das inserções onde o método utilizado claramente influenciou os resultados, este não foi um fator de impacto neste caso. O tempo resultante das buscas devido à diferença dos ambientes de armazenamento dos dados do banco em memória são visivelmente melhores que o banco em disco, possibilitando finalizar as consultas com picos de até um minuto e quinze segundos de diferença, uma diferença de mais de 40%.

Por fim, a Tabela 12 realiza uma sumarização de todos os resultados obtidos para que seja possível fazer uma análise do desempenho geral de ambos os bancos.

Tabela 12: Sumarização dos resultados

<b>Sumarização dos Resultados</b>			
<b>Etapas</b>	<b>Tempo em disco</b>	<b>Tempo em Memória</b>	<b>Diferença Total (Memória – Disco)</b>
<b>1ª Etapa (25%) (6 Milhões de Avaliações)</b>	00:01:57s	00:01:40s	-00:00:17s
<b>2ª Etapa (50%) (12 Milhões de Avaliações)</b>	00:03:40s	00:03:11s	-00:00:29s

<b>3ª Etapa (100%) (24 Milhões de Avaliações)</b>	00:12:27s	00:10:12s	-00:02:15s
---	-----------	-----------	------------

Fonte: Autor.

Como a tabela de sumarização demonstra, é possível observar que entre os dois modelos propostos, quando a quantidade de dados não é muito grande a diferença no tempo total não é tão expressiva. No entanto, com o aumento no número de entradas torna-se nítido o gargalo gerado pelo disco, como é apresentado na última linha desta tabela. Sendo assim, pode-se afirmar que devido ao avanço da tecnologia de memória a classe de bancos de dados em memória tende a fornecer um ferramental importante para o cenário de aumento expressivo nos dados.

## 4 CONSIDERAÇÕES FINAIS

O objetivo geral deste trabalho foi realizar um estudo comparativo entre as abordagens de Banco de Dados Relacional Tradicional (em disco) e o modelo Relacional em Memória, considerando aspectos gerais de desempenho no que tange a população e consulta de dados.

A partir do levantamento bibliográfico na área de Banco de Dados, buscou-se entender o domínio e o contexto do trabalho de forma que a partir deste conhecimento fosse possível implementar uma aplicação capaz de realizar inserções dos dados para atender o objetivo proposto.

Este estudo foi aplicado sobre a base de dados do serviço de recomendação de filmes MovieLens®. Tal base é composta por informações contidas na base de Base de Dados de Filmes na Internet (IMDb). Com o objetivo de simular um cenário real, para que a comparação dos bancos pudesse ser realizada, foi efetuada uma avaliação de desempenho. Foram executados estudos com o maior conjunto de dados possível, cerca de 25 milhões de linhas, divididos em três partes, uma com todos os dados, outra com cerca de 13 milhões de linhas e a última com cerca de 7 milhões de linhas.

Considerando a implementação da aplicação de carga foi possível analisar a inserção e coleta de resultados a partir do estabelecimento de parâmetros comparativos sobre o desempenho apresentado nos dois conceitos de armazenamentos de dados escolhidos.

O modelo baseado no conceito de armazenamento em disco apresentou um desempenho similar em comparação ao modelo baseado em memória quando a carga de dados contida no banco não tinha um tamanho expressivo. No entanto, deve-se salientar que essa similaridade de resultados com um volume menor de dados ocorreu devido ao método utilizado para popular o modelo em memória. A carga foi realizada as estruturas já presentes em disco para a memória, o que fez com que a população do banco em memória fosse limitado pela velocidade do disco.

O modelo baseado no conceito de armazenamento em memória apresentou uma melhora considerável em todas as três etapas do processo avaliativo, em decorrência do modelo não sofrer com o gargalo gerado pela baixa velocidade de leitura e escrita de um disco, por manter toda a base de dados na memória principal. Mesmo mantendo toda a base em memória, este modelo consegue manter as propriedades ACID, por permite salvar periodicamente em disco *logs* com o estado do banco, para que no caso de alguma falha mais grave seja possível a recuperação do banco ao estado anterior consistente.

De modo geral, pode-se concluir que o modelo orientado a memória apresenta desempenho similar ao modelo tradicional quando a carga de dados é pequena. Entretanto, a medida que aumenta a quantidade de entradas na base, o banco em memória começa a se distanciar do modelo tradicional, devido às limitações do disco e, por conseguir manter os dados no mesmo local (memória) onde são realizadas todas as operações, tanto de escrita quanto de leitura. Este fator diminui consideravelmente o tempo de busca de dados para execução de consultas.

Durante o desenvolvimento deste trabalho e pensando em perspectivas futuras de pesquisas, foram vislumbrados alguns possíveis trabalhos. Entre as possibilidades encontram-se a continuidade deste trabalho no que tange o aumento da escalabilidade computacional, com o objetivo de verificar como esses modelos se comportam com a criação de uma estrutura em *cluster* ou *grid* computacional.

Outra possibilidade está na expansão da análise realizada neste trabalho comparando a abordagem relacional tradicional e em memória, com alguma nova abordagem como o NewSQL, que mescla alta disponibilidade e escalabilidade com os requisitos transacionais e de consistência dos dados.

## REFERÊNCIAS

- GRAD, B.; BERGIN, T. J. History of Database Management Systems. IEEE Ann. Hist. Comput., vol. 31, no. 4, p. 3 - 5, 2009.
- BAHRILL, Neha; TIWARI, Aruna; MALVIYA, Aayushi. Fuzzy Based Clustering Algorithms to Handle Big Data with Implementation on Apache Spark. In: IEEE second international conference on big data computing service and applications, p. 95 – 104, 2016.
- BEYNON-DAVIES, P. Database Systems: Palgrave Macmillan Limited. (Macmillan computer science series). ISBN 9781403916013. 2004.
- CERN. Computing: Experiments at CERN generate colossal amounts of data. The Data Centre stores it, and sends it around the world for analysis. Disponível em: <<http://home.cern/about/computing>>.
- DUMBILL, Edd. What is big data?: An introduction to the big data landscape... 2012. Disponível em: <<https://www.oreilly.com/ideas/what-is-big-data>>.
- ELMASRI, Ramez; NAVATHE, Shamkant B. Sistema de Banco de Dados. São Paulo: Pearson Addison Wesley. 2011.
- ELMASRI, Ramez; NAVATHE, Shamkant B. Sistemas de Banco de Dados. 5. ed. Tradução de Marília G. Pinheiro; Cláudio C. Canhette; Glenda C. V. Melo; Claudia V. Amadeu; Rinaldo M. de Moraes. São Paulo: Editora PEARSON. 2005.
- FAN, Wei; BIFET, Albert. Mining Big Data: Current Status, and Forecast to the Future. SIGKDD Explorations, China, v. 2, n. 14, p.1-5, mar. 2012. Acesso em: 15 set. 2016.
- GARTNER. Big Data. IT Glossary [20--]. Disponível em <http://www.gartner.com/it-glossary/big-data/>. Acesso em: 01 out. 2016.
- GRAVES, Steve. In-Memory Database Systems. Acm Digital Library, [s.i], p.1-10, set. 2002.
- GROBELNIK, Marko. Big Data Tutorial. Kalamaki: Jožef Stefan Institute, 2012. Color. Disponível em: [http://videlectures.net/eswc2012\\_grobelnik\\_big\\_data/](http://videlectures.net/eswc2012_grobelnik_big_data/). Acesso em: 22 set. 2016.

GARCIA-MOLINA, H.; SALEM, K. Main Memory Database Systems : An Overview, IEEE Trans. Knowl. Data Eng., vol. 4, no. 6, pp. 509–516, 1992.

HARPER, F. Maxwell; KONSTAN, Joseph A.. The MovieLens Datasets. Acm Transactions On Interactive Intelligent Systems, [s.l.], v. 5, n. 4, p.1-19, 22 dez. 2015. Association for Computing Machinery (ACM). <http://dx.doi.org/10.1145/2827872>

HASHEM, Ibrahim Abaker Targio et al. The role of big data in smart city. International Journal of Information Management, v. 36, n. 5, p. 748-758, 2016.

HECHT, R., JABLONSKI, S; NoSQL evaluation: A use case oriented survey. International Conference on Cloud and Service Computing (CSC), p. 336-341, IEEE, 2011.

HEUSER, C. A.. Projeto de Banco de Dados. 4 ed. Porto Alegre: Bookman, 1998.

IBM. Você realmente sabe o que é Big Data? 2012. Disponível em: <[https://www.ibm.com/developerworks/community/blogs/ctaurion/entry/voce\\_realmente\\_sabe\\_o\\_que\\_e\\_big\\_data?lang=en](https://www.ibm.com/developerworks/community/blogs/ctaurion/entry/voce_realmente_sabe_o_que_e_big_data?lang=en)>. Acesso em: 01 set. 2016.

IBM. What is Big Data? Disponível em: <<https://www-01.ibm.com/software/data/bigdata/what-is-big-data.html>>.

IDC. The Digital Universe of Opportunities: Rich Data and the Increasing Value of the Internet of Things. 2014. Disponível em: <<http://www.emc.com/leadership/digital-universe/2014iview/executive-summary.htm>>. Acesso em: 01 out. 2016.

JANUKOWICZ, J.; EASTWOOD, M. Storage Acceleration Solving IO Performance Gap Problem. IDC Analyst Connection, IDC, v. 1359, p. 1–4, set. 2012.

JIN, Xiaolong; WAH, Benjamin W.; CHENG, Xueqi; WANG, Yuanzhuo. Significance and Challenges of Big Data Research. Big Data Research, n.2, p. 59-64. 2015.

LANEY, Doug. 3D Data Management: Controlling Data Volume, Velocity, and Variety. 2001. Disponível em: <<http://blogs.gartner.com/doug-laney/files/2012/01/ad949-3D-Data-Management-Controlling-Data-Volume-Velocity-and-Variety.pdf>>.

GUPTA, M. Kumar; VERMA, V.; VERMA, M. S. In-Memory Database Systems -A Paradigm Shift, Int. J. Eng. Trends Technol., vol. 6, no. 6, 2013.

MCOBJECT – extremeDB in-memory database systems. Disponível em: <<http://www.mcobject.com/extremedbfamily.shtml>>.

MOLKA, Karsten; CASALE, Giuliano. Experiments or Simulation? A Characterization of Evaluation Methods for In-Memory Databases. Acm Sigmod Record:, South Kensington, v. 44, n. 0, p.1-9, ago. 2015.

MONGODB. Introduction to MongoDB. Disponível em: <<https://docs.mongodb.com/manual/introduction/>>

MUKHERJEE, Anirban et al. Shared Disk Big Data Analytics with Apache Hadoop. In: HIGH PERFORMANCE COMPUTING, 19., 2012, Pune. Artigo. Pune: IEEE, 2012. p. 1 - 6.

ORACLE. Oracle Berkeley DB 12c. Disponível em: <<http://www.oracle.com/technetwork/database/database-technologies/berkeleydb/overview/index.html>>.

ORACLE. Oracle TimesTen In-Memory Database Overview. Disponível em: <<http://www.oracle.com/technetwork/database/database-technologies/timesten/overview/timesten-imdb-086887.html>>.

O'REILLY, Media. Big Data Now. Sebastopol: O'reilly Media, 2012. 119 p.

OWENS, Michael, The Definitive Guide to SQLite. Estados Unidos, Apress Inc., 2006.

PERALTA, Verónica. Extraction and Integration of MovieLens and IMDb Data. Versailles: Laboratoire Prism, 2007.

PLATTNER, Hasso; ZEIER, Alexander; In-Memory Data Management: An Inflection Point for Enterprise Application. ISBN 978-3-642-19362-0 e-ISBN 9783-642-19363-7 – DOI 10.1007/978-3-642-19363-7. Springer Heidelberg Dordrecht London New York. Berlin. Springer-Verlag, 2011.

RASHID, Al Mamunur et al. Getting to know you: Learning new user preferences in recommender systems. Proceedings Of The 7th International Conference On Intelligent User Interfaces - Iui '02, [s.l.], p.127-134, 2002. Association for Computing Machinery (ACM). <http://dx.doi.org/10.1145/502716.502737>.

‘SAS COMPANY. Big Data: What it is and why it matters. Disponível em: <[http://www.sas.com/en\\_us/insights/big-data/what-is-big-data.html](http://www.sas.com/en_us/insights/big-data/what-is-big-data.html)>.

SILBERSCHATZ, Abraham; KORTH, Henry F.; SUDARSHAN, S.. Sistema de Banco de Dados. 5. ed. Rio de Janeiro: Elsevier, 2006.

STRAWN, George; STRAWN, Candace. Relational Databases: Codd, Stonebraker, and Ellison. It Professional, [s.l.], v. 18, n. 2, p.63-65, mar. 2016. Institute of Electrical and Electronics Engineers (IEEE). <http://dx.doi.org/10.1109/mitp.2016.25>.

SUMATHI, S.; ESAKKIRAJAN, S. Fundamental of Relational Database Management Systems. Studies in Computational Intelligence, v. 47, Springer, 2007.

TIWARI, Shashank. Professional NoSQL. John Wiley & Sons, 2011.

VOLTDDB. What Is VoltDB? Disponível em: <<https://www.voltddb.com/overview>>.

KIM, W. Object-Oriented Databases: Definition and Research Directions, IEEE Trans. Knowl. Data Eng., vol. 2, no. 3, pp. 327–341, 1990.

WADE, B.; CHAMBERLIN, D.; IBM Relational Database Systems: The Early Years. IEEE Annals of the History of Computing, v. 34, n. 4, p. 38-48, 2012.

WANG, Y. et al. The performance survey of in memory database, Proc. Int. Conf. Parallel Distrib. Syst. - ICPADS, vol. 2016–Janua, pp. 815–820, 2016.