Universidade Federal Rural de Pernambuco
Departamento de Estatística e Informática

Análise de desempenho de Banco de Dados Relacionais e Não Relacionais em dados genômicos

Juccelino Rodrigues Alves de Barros

Recife

Julho de 2015

Juccelino Rodrigues Alves de Barros

Análise de desempenho de bancos de dados Relacionais e Não Relacionais em dados genômicos

Orientador: Glauco Gonçalves

Co-orientador: Victor Medeiros

Monografia apresentada ao Curso Bacharelado em Sistemas de Informação da Universidade Federal Rural de Pernambuco, como requisito parcial para obtenção do título de Bacharel em Sistemas de Informação.

Recife

Julho de 2015

À

meus pais, meus irmãos e minha namorada que sempre acreditaram em mim.

Agradecimentos

Primeiramente agradeço à Deus.

Agradeço especialmente à meus pais e irmãos pelo apoio. Ao grupo OS PRIMOS e os demais membros da família. Também agradeço à minha namorada por está comigo a mais de 5 anos, pela paciência nos dias estressantes e pela companhia nos melhores momentos da minha vida.

Aos professores Glauco e Victor, por está sempre presente nos momentos que mais precisei, pela confiança e apoio neste trabalho. Agradeço também à toda equipe do DEINFO, professores, coordenadores e servidores, entre eles, ao servidor Saulo Marques. A todos amigos da UFRPE.

À Carlos Henrique Castelleti e o pessoal do LIKA da UFPE por ter disponibilizado os arquivos para os experimentos e compartilhado os seus conhecimentos.

À Laura Gil Rodrigues por me ajudar no desenvolvimento do abstract.

À TELEIN e a USTORE, empresas que me deram a oportunidade de estagiar e incrementar meu conhecimento pessoal como também profissional. Aos amigos do NTI da UFPE pelo apoio.

Por último, agradeço à todos meus amigos que sempre acreditaram na realização do trabalho.

Resumo

O armazenamento de dados genômicos é um grande desafio hoje, pois o avanço da tecnologia molecular trouxe melhores equipamentos para gerar sequências de DNA maiores e em menor tempo. Em consequência disso, a quantidade de dados genômicos gerados está aumentando, de forma que o sequenciamento de um único organismo pode gerar arquivos com *terabytes* de informações. De forma geral, os processos de manipulação de dados genômicos fazem uso de simples arquivos como o principal meio para armazenamento de tais dados. Contudo, os bancos de dados se apresentam como alternativas para a gerência desses dados por oferecer melhor organização, melhor uso do espaço disponível para armazenamento e bom desempenho, visto que são preparados para otimizar o armazenamento de dados. Além disso, os bancos de dados permitem agregar aos dados brutos do sequenciamento (as próprias cadeias de DNA) meta-informações acerca das sequências de DNA armazenadas. Diante deste cenário, este trabalho apresenta e avalia o desempenho de diferentes estratégias de armazenamento em três bancos de dados pertencente a dois paradigmas diferentes, o MySQL (representante dos bancos de dados Relacionais), o Cassandra e o MongoDB (representantes dos bancos de dados Não-Relacionais). Foi desenvolvida também uma ferramenta de benchmark para automatizar os experimentos. Os resultados demonstraram que os bancos de dados relacionais apresentam limitações quando estão inseridos em um ambiente com grandes massas de dados, como o cenário de dados genômicos.

Palavras-chave: dados genômicos, banco de dados, banco de dados relacional, banco de dados não relacional

Abstract

The genomic data storage is a huge challenge nowadays, once the advancement of molecular technology has brought better equipment to produce larger DNA sequences and in less time. As a result, the amount of genomic data bred has been increasing, so in consequence of that, a single organism can generate files with terabytes of information. In general, genomic data handling processes make use of simple files as the primary means for storing these data. However, the databases show as a good alternative for management of these data by offering better organization, fault tolerance, use of available space for storage and performance, as they are made in order to optimize data storage. Beyond that, the databases allow adding raw sequencing data (their own DNA sequences) meta-information about the stored DNA sequences. Against this background, this project presents and evaluates the performance of different storage strategies in three databases that belongs to two different paradigms, MySQL (which belongs to Relational Database paradigm), Cassandra and MongoDB (which belong to Non-Relational Database paradigm). It has been also developed one benchmark tool in order to automate the experiments. The results show that relational databases have limitations when they live in an environment with large data sets, such as genomic data scenario.

Keywords: genomic data, database, relational database, non-relational database

Sumário

1 Introdução **1**

1.1 Objetivos 4

1.2 Contribuições 5

1.3 Organização do trabalho 5

2 Características dos Bancos de Dados Relacional e Não Relacional **6**

2.1 Bancos de Dados Relacionais 7

2.1.1 Breve histórico 7

2.1.2 Propriedade ACID 8

2.1.3 Modelo Entidade e Relacionamento 9

2.1.4 Normalização 10

2.2 MySQL 10

2.3 Banco de Dados Não Relacionais - NoSQL 12

2.3.1 Breve histórico 13

2.3.2 Propriedade BASE 13

2.3.3 Modelos de Banco de dados NoSQL 14

2.4 Cassandra 15

2.4.1 Protocolo Gossip 17
2.4.2 Exemplos de Aplicações 18
2.5 MongoDB 18
2.5.1 Exemplos de Aplicações 19
2.6 Comparação Qualitativa entre os bancos escolhidos 19

3 Armazenamento de Dados Genômicos 23

3.1 Formatos dos Arquivos 24
3.2 Processamento de dados genômicos 26
3.3 Estratégias de Armazenamento 27
3.3.1 Esquemas dos Bancos de Dados 30

4 Avaliação de Desempenho 33

4.1 Ferramenta de Benchmark - FastaBD 34
4.2 Metodologia de Avaliação 34
4.2.1 Descrição dos Arquivos Fasta Utilizados 36
4.2.2 Método dos Experimentos 37
4.3 Resultados 38
4.3.1 Análise da Inserção 39
4.3.2 Análise da Extração 44
4.3.3 Análise da Consulta 47
4.3.4 Análise da Curva de Consulta 48
4.4 Discussão 50

5 Conclusão e Trabalhos Futuros 52

5.1 Dificuldades Encontradas 53

5.2 Trabalhos Futuros 54

A Tempo de armazenamento dos Arquivos: cabra5.fa, cabra6.fa e cabra7.fa 58

B Resultado dos Experimentos 60

Lista de Tabelas

3.1 Combinação dos números com o par de cadeia de DNA. (Fonte: [4]) 25

4.1 Detalhes dos arquivos *fasta* utilizados neste estudo. (Fonte: o autor) 36

4.2 Propoção do tempo médio de inserção entre o MySQL e os bancos de dados Não Relacionais para SRS 1. (Fonte: o autor) 40

4.3 Propoção do tempo médio de inserção entre o MySQL e os bancos de dados Não Relacionais para SRS 5. (Fonte: o autor) 41

4.4 Propoção do tempo médio de inserção entre o MySQL e os bancos de dados Não Relacionais para SRS 50. (Fonte: o autor) 41

4.5 Variação do tempo médio de inserção entre o MySQL e os bancos de dados Não Relacionais variando o número de SRS. (Fonte: o autor) 41

4.6 Valores estatísticos da Análise da Consulta. (Fonte: o autor) 48

Lista de Figuras

1.1 Queda do custo para sequenciar um genoma. (Fonte: [20]) 2

1.2 Queda do custo para sequenciar um milhão sequências de DNA. (Fonte: [20]) 2

1.3 Crescimento do número de bases nucleicas e das sequências de DNA geradas no GenBank e no WGS. (Fonte: [19]) 3

2.1 Visualização da arquitetura lógica do MySQL (Fonte: [24]) 11

2.2 Estrutura da escrita e leitura no Cassandra. (Fonte: [6]) 17

2.3 Comparação qualitativa entre os três bancos de dados. (Fonte: o autor) . . . 20

3.1 Exemplo de um arquivo fasta. (Fonte: o autor) 24

3.2 Exemplo de um arquivo csfasta. (Fonte: o autor) 25

3.3 arquivo fasta resultante da filtragem do csfasta. (Fonte: o autor) 25

3.4 Diagrama de todo processo de tratamento dos arquivos Fasta. (Fonte: o autor) 26

3.5 Estratégias utilizadas nos trabalhos encontrados. (Fonte: o autor) 28

3.6 Esquema do MySQL. (Fonte: o autor) 30

3.7 Esquemas do Cassandra e do MongoDB. (Fonte: o autor) 31

4.1 Fluxograma do experimento 1. (Fonte: o autor) 37

4.2 Fluxograma do experimento 2. (Fonte: o autor) 38

4.3 Média de Tempo de Inserção - Cabra4.fa. (Fonte: o autor) 40

4.4 Crescimento de inserção de dados - SRS 1. (Fonte: o autor) 43

4.5 Crescimento de inserção de dados - SRS 50. (Fonte: o autor) 43

4.6 Média de Tempo de Extração - Cabra4.fa. (Fonte: o autor) 44

4.7 Média de Tempo de Extração - Cabra5.fa. (Fonte: o autor) 44

4.8 Média de Tempo de Extração - Cabra6.fa. (Fonte: o autor) 45

4.9 Média de Tempo de Extração - Cabra7.fa. (Fonte: o autor) 45

4.10 Crescimento do tempo médio de extração de dados SRS 1. (Fonte: o autor) . 46

4.11 Crescimento do tempo médio de extração de dados SRS 5. (Fonte: o autor) . 47

4.12 Crescimento do tempo médio de extração de dados SRS 50. (Fonte: o autor) 47

4.13 Curva de Consulta. (Fonte: o autor) 49

4.14 Avaliação de desempenho dos três bancos de dados. (Fonte: o autor) 50

A.1 Tempo de armazenamento - Cabra5.fa 58

A.2 Tempo de armazenamento - Cabra6.fa 59

A.3 Tempo de armazenamento - Cabra7.fa 59

Lista de Abreviaturas

ACID = Atomicidade, Consistência, Isolamento e Disponibilidade

BASE = *Basically Avaliable, Soft State, Eventual Consistency*

BLOB = *Binary Large Object*

BSON = *Binary JavaScript Object Notation*

CQL = *Cassandra Query Language*

DER = Diagrama Entidade e Relacionamento

DNA = *Deoxyribonucleic Acid*

JSON = *JavaScript Object Notation*

LTS = *Long Term Support*

MER = Modelo Entidade e Relacionamento

NCBI = *National Center for Biotechnology Information*

NoSQL = *Not Only SQL*

RAM = *Random Access Memory*

SGBD = Sistema Gerenciador de Banco de Dados

SQL = *Structured Query Language*

SRS = *Short Read Sequence*

SSD = *Solid State Drive*

SSL = *Secure Socket Laye*

Capítulo 1

Introdução

A Bioinformática pode ser definida como uma mesclagem de duas grandes áreas, a biologia e a informática. São englobados aspectos da biologia, aquisição, processamento e armazenamento como também a distribuição e análise de dados, combinando com as técnicas de matemática e computação com o objetivo de entender os dados biológicos.

Com o avanço da tecnologia molecular, os dados genômicos são gerados mais rápido e em maior escala (da ordem de *terabytes*). Para ter uma ideia, o projeto Genoma Humano, demorou 10 anos para ser concluído e custou aproximadamente 3 bilhões de dólares gerando aproximadamente 3,5 bilhões de pares de bases. Hoje, é possível sequenciar até 4 bilhões de pares de bases em alguns dias e com um custo bem menor do que o projeto Genoma Humano [27]. As figuras 1.1 e 1.2 ilustram, respectivamente, a queda do custo para sequenciar um genoma e a queda do custo para sequenciar um milhão (*megabase*) de sequências de DNA. Em ambos os gráficos é observado a linha referente a Lei de Moore, que diz que, o poder de processamento dos computadores dobraria a cada 18 meses [29]. Nos dois gráficos, é possível observar que a queda no custo referente ao sequenciamento genético a partir de 2007 acentuou-se, acarretando um maior afastamento da linha da Lei de Moore. Isso quer dizer que, o avanço da tecnologia molecular está trazendo maiores arquivos com baixo custo e consequentemente novos projetos genomas estão surgindo.

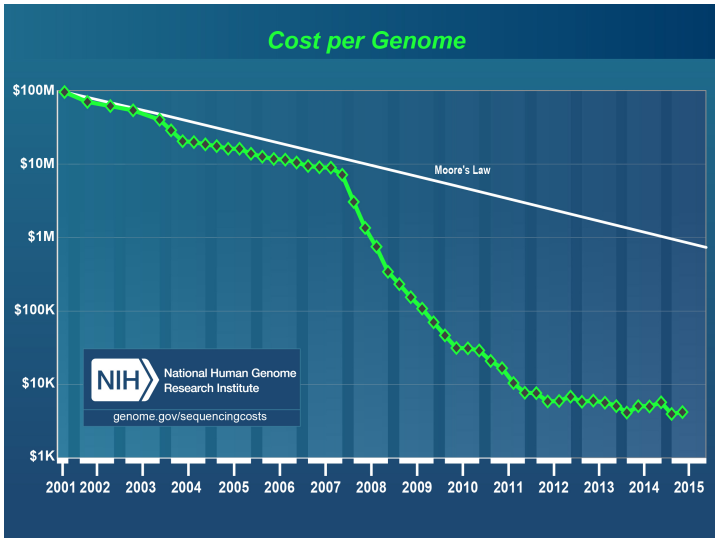

Figura 1.1: Queda do custo para sequenciar um genoma. (Fonte: [20])

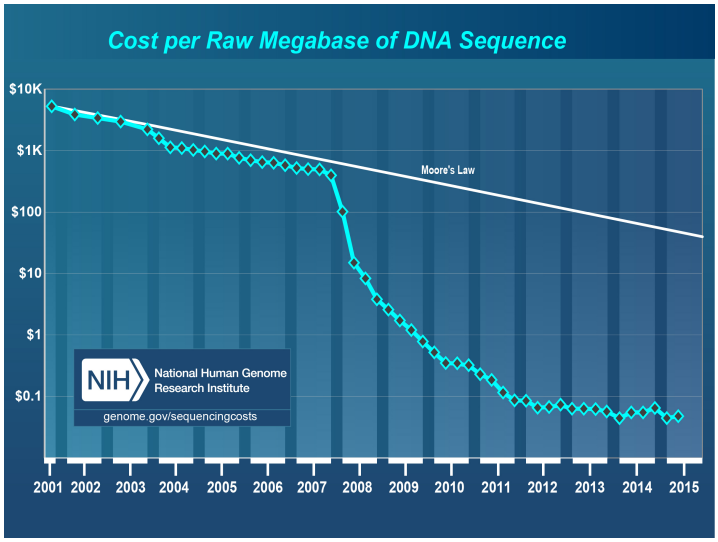

Figura 1.2: Queda do custo para sequenciar um milhão sequências de DNA. (Fonte: [20])

Os dois gráficos representados na figura 1.3 mostram, respectivamente, o crescimento do número de bases nucleicas sequenciadas e o número de sequências de DNA geradas em dois grandes projetos genômicos, o *GenBank*[1] e *WGS*[2]. É possível observar, na linha de ambos os projetos, o número de bases e sequências tendem a aumentar 10 vezes em intervalos de 5 anos. Isso tudo se deve ao fato do avanço dos equipamentos de sequenciamento genético que são desenvolvidos e melhorados com o passar do tempo.

[1]http://www.ncbi.nlm.nih.gov/genbank
[2]http://www.ncbi.nlm.nih.gov/genbank/wgs/

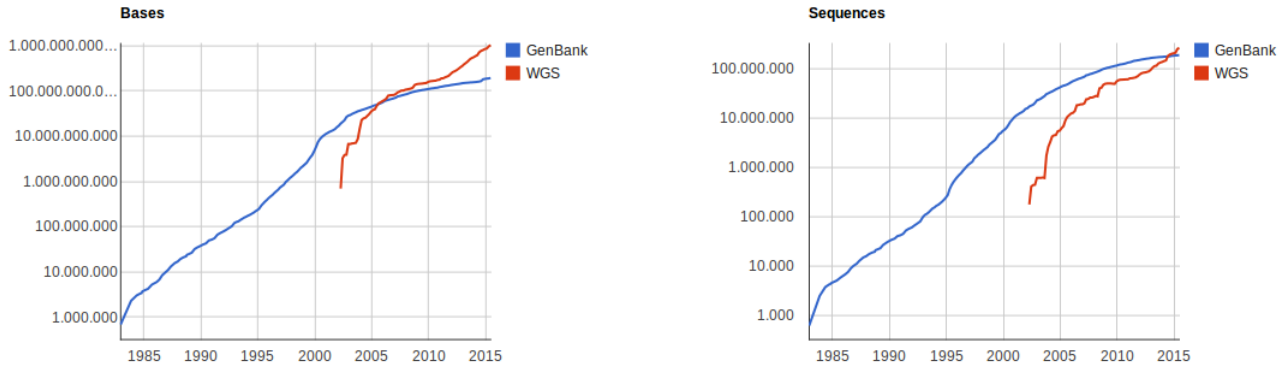

Figura 1.3: Crescimento do número de bases nucleicas e das sequências de DNA geradas no GenBank e no WGS. (Fonte: [19])

Os dados genômicos são gerados através do sequenciamento de um material genético, podendo ser um tecido, um órgão ou partes menores. Quando finalizado o processamento, o sequenciador genético gera arquivos com várias informações, dentre as quais se encontram, as sequências de DNA correspondentes ao material genético e as informações do sequenciador que realizou o procedimento. Esses arquivos tendem a ser grandes, cada um podendo atingir na ordem de alguns *terabytes* e, como consequência disso, faz-se necessário um número elevado de recursos computacionais e de armazenamento para processar e guardar esses dados. O alto custo de armazenamento acaba sendo um grande problema para vários usuários que trabalham nessa área, pois alguns não dominam as melhores tecnologias para gerenciar esses dados.

Existem várias discussões a respeito das melhores tecnologias para o armazenamento e processamento desses arquivos [22, 5]. Atualmente, são utilizadas algumas estratégias para o armazenamento dos dados genômicos, entre elas, encontramos a estratégia que é baseada no armazenamento do arquivo em si em repositórios de arquivos. Os arquivos, em formato texto ou binário, são armazenados em conjunto em um repositório de arquivos e são distribuídos de uma forma que dificulta a recuperação de informações a respeito de cada um deles. Esta abordagem traz muitas desvantagens, visto que, as informações desses arquivos ficam dispersas no repositório, trazendo dificuldades para o usuário encontrar aquilo que realmente deseja. Muitas vezes, nessa abordagem, existe pouca documentação no que diz respeito aos arquivos que estão armazenados.

Outra estratégia de armazenamento encontrada, é a utilização de bancos de dados relacionais. A principal vantagem de utilizar um banco de dados, ao invés de um repositório de arquivos, é a possibilidade de realizar consultas complexas e ter um melhor gerenciamento dos dados que foram armazenados. Além disso, a flexibilidade de acesso junto com a segurança do banco de dados faz com que os usuários não se preocupem em danos causados por acessos indesejado. Contudo, um banco de dados mal projetado, isto é, um banco com pobre organização ou que utilize estratégias inadequadas à gerência de dados genômicos, pode introduzir mais malefícios do que benefícios ao usuário. A escolha certa de um banco de dados pode acarretar em uma economia nos custos de um projeto assim como a satisfação dos usuários.

É necessário projetar um banco de dados com o foco em desempenho no armazenamento e na extração dos dados genômicos, pois ainda não foi encontrado o modelo ideal para atender a essa demanda. Paralelo ao crescimento dos dados genômicos, surgiram novos conceitos e paradigmas de bancos de dados, entre eles, os bancos de dados Não Relacionais. Os bancos de dados Não Relacionais possuem características de armazenamento distribuído dos dados e boa performance na recuperação destes. O uso dos bancos de dados Não Relacionais mesclado com algumas estratégias de armazenamento de dados genômicos pode trazer grandes contribuições no ambiente científico, podendo ser uma boa alternativa para o gerenciamento desses dados.

1.1 Objetivos

O principal objetivo deste trabalho é avaliar estratégias de armazenamento de dados genômicos em bancos de dados Relacionais e Não Relacionais. Em particular, foi mensurarado o desempenho de operações típicas como inserção, extração e consulta de dados genômicos sob diferentes formas de armazenamento e em três diferentes bancos de dados: MySQL (representante dos bancos de dados Relacionais), Cassandra e MongoDB (representantes dos bancos de dados não-relacionais).

Para a automação dos testes de desempenho e comparação justa entre os diferentes bancos de dados, foi desenvolvida uma ferramenta de *benchmark* que realizou todo o processo, iniciando pela criação do ambiente, em seguida realizando inserção, extração e consulta dos dados.

1.2 Contribuições

Encontrar a opção mais viável, entre as formas de armazenamento sugeridas e os três banco de dados escolhidos, para o armazenamento e extração de dados genômicos. Levando em consideração o desempenho de inserção, extração e consulta de cada banco de dados e o ambiente de produção em que eles estão inseridos.

1.3 Organização do trabalho

A estrutura do trabalho é composta por este capítulo e mais outros 4. O Capítulo 2 apresentará o referencial teórico, com as características dos bancos de dados Relacionais e Não Relacionais e os conceitos dos três bancos de dados utilizados. No final teremos uma comparação, em aspectos qualitativos, em relação a instalação, compatibilidade, usabilidade e escalabilidade. No Capítulo 3 encontraremos as informações a respeito dos dados genômicos e de como são comumente armazenados. Será mostrado o processamento desses arquivos, desde de sua criação, realizado após o sequenciamento, passando pela filtragem e armazenamento. No armazenamento, serão esboçadas as estratégias discutidas durante o trabalho. O Capítulo 4 mostrará a avaliação de desempenho junto com uma breve análise dos resultados obtidos. Por fim, o capítulo 5 apresentará as conclusões, dificuldade encontradas e os trabalhos futuros.

Capítulo 2

Características dos Bancos de Dados Relacional e Não Relacional

Os bancos de dados são coleções de dados relacionados que seguem uma determinada estrutura, de forma que possam ser recuperados quando necessário. Existem vários tipos de bancos de dados, cada um deles com sua particularidade, que podem ser divididos em dois paradigmas: Relacional e Não Relacional. Esses paradigmas possuem características exclusivas, que serão detalhadas nas próximas Seções deste capítulo.

O capítulo está dividido da seguinte forma: a primeira seção descreve os Bancos de Dados Relacionais, onde serão apresentados um breve histórico, a propriedade ACID, o modelo entidade e relacionamento e as regras de normalização. Em seguida, a Seção 2.2 apresenta o banco de dados relacional MySQL com alguns exemplos de aplicações. A Seção 2.3 descreve os Bancos de Dados não Relacionais - NoSQL, apresentando um breve histórico, a propriedade BASE e os modelos de Bancos de Dados NoSQL. Os bancos de dados Cassandra e MongoDB e suas aplicações serão apresentadas nas Seções 2.4 e 2.5, respectivamente. Por último, na Seção 2.6, os três bancos são comparados em aspectos qualitativos como: complexidade de instalação, compatibilidade com o sistema operacional, usabilidade e escalabilidade.

2.1 Bancos de Dados Relacionais

Um banco de dados relacional pode ser definido como uma forma de distribuição de dados de modo que eles tenham alguma relação entre si e que essa relação é representada por meio de tabelas. Tabela é uma estrutura composta por linhas e colunas. As colunas representam os atributos (campos) e as linhas representam os valores. O programa responsável por armazenar os dados e controlar a sua estrutura a nível de hardware é chamado de Sistema Gerenciador de Banco de Dados (SGBD). É esse programa que faz a interface entre os dados inseridos pelo usuário e o servidor onde os dados são armazenados. [6]

Em um modelo relacional é usado o conceito de chave primária e chave estrangeira. A chave primária é um conjunto de um ou mais valores que representa uma linha específica da tabela, ela não pode ser repetida em outra linha. Já a chave estrangeira é a forma de referenciar uma entidade "a" na entidade "b", quando elas possuem um relacionamento.

Os bancos de dados relacionais mais famosos que encontramos são: o Oracle[1], o MySQL[2] e o Postgres[3]. O primeiro possui uma versão gratuita porém é limitada. Já o MySQL e o Postgres são os mais difundidos entre os desenvolvedores por terem uma boa cobertura na versão gratuita. Entre os três citados, o MySQL foi escolhido para ser usado no desenvolvimento do trabalho. A escolha deste banco foi feita por ser um dos mais usados e por ter melhor compatibilidade com o sistema operacional Linux - Ubuntu 14.04, o qual será usado durante todo o desenvolvimento do trabalho. O MySQL será mais detalhado no tópico 2.2.

2.1.1 Breve histórico

Em junho de 1970 - Edgar Frank Codd - publicou o artigo *Relational Model of Data for Large Shared Data Banks*. Este artigo tratava sobre o uso de cálculo e álgebra relacional para permitir que usuários não técnicos armazenassem e recuperassem grande quantidade de informações. Codd visionava um sistema onde o usuário seria capaz de acessar as informações através de comandos em inglês, onde as informações estariam armazenadas em tabelas. Devido à natureza técnica deste artigo e a relativa complicação matemática, o significado

[1] http://www.oracle.com/br/index.html
[2] https://www.mysql.com/
[3] http://www.postgresql.org/

e proposição do artigo não foram prontamente realizados. Entretando ele levou a IBM a montar um grupo de pesquisa conhecido como *System R.*[23]

O objetivo do projeto *System R* era criar um sistema de banco de dados relacional o qual eventualmente se tornaria um produto. Os primeiros protótipos foram utilizados por muitas organizações, tais como o MIT Sloan School of Management (uma escola renomada de negócios norte-americana). Novas versões foram testadas em empresas de aviação para rastreamento do manufaturamento de estoque. O *System R* evoluiu para SQL/DS o qual posteriormente tornou-se o DB2. A linguagem criada pelo grupo de pesquisa foi a Structured Query Language (SQL), inicialmente chamada de SEQUEL. Esta linguagem tornou-se um padrão na indústria para banco de dados relacionais e hoje é um padrão ISO.

2.1.2 Propriedade ACID

Para atender as características dos banco de dados relacionais, é necessário que os SGBDs atendam uma série de requisitos, dentre eles estão: controle de concorrência, segurança, recuperação de falhas, gerenciamento dos mecanismos de armazenamento de dados, controle das restrições de integridade do banco de dados e gerenciamento de transações. Para executar essas transações, o SGBD deve respeitar algumas propriedades a fim de garantir o funcionamento correto do sistema e a respectiva consistência dos dados [17]. Estas propriedades são chamadas de propriedade ACID (atomicidade, consistência, isolamento e durabilidade) e são definidas a seguir:

- **Atomicidade:** todas as operações da transação são executadas, ou seja, a transação é executada por completo ou nada é executado.
- **Consistência:** No final de uma transação, o banco de dados deve permanecer em um estado consistente, ou seja, deve satisfazer as condições de consistência e restrições de integridade previamente assumidas.
- **Isolamento:** Caso duas transações estejam sendo executadas ao mesmo tempo seus efeitos devem ser isolados uma das outras.
- **Durabilidade:** Uma vez que uma transação ocorreu com sucesso, seu efeito não poderá mais ser desfeito, mesmo em caso de falha.

2.1.3 Modelo Entidade e Relacionamento

Durante a fase inicial de um projeto de sistemas de informações é ideal criar um esquema conceitual para o banco de dados, utilizando um modelo de dados de alto nível. Esse modelo é chamado de Modelo Entidade e Relacionamento (também chamado Modelo ER, ou simplesmente MER). O MER é um dos principais modelos que representa uma visão de como os dados serão organizados no banco de dados [11].

O MER, desenvolvido em 1976 por Peter P. Chen, descreve os dados como Entidades, Relacionamentos e atributos. Uma entidade pode ser entendida como um objeto do mundo real com uma existência independente. Pode ser um objeto físico (pessoa, carro e etc) ou um objeto conceitual (curso, emprego e etc). Cada entidade tem um ou vários atributos, ou seja, propriedades particulares que a descrevem, por exemplo: pessoa (nome, cpf, data de nascimento). Um relacionamento é como as entidades estão envolvidas entre si. Existem 3 fases no modelo ER são elas: conceitual, lógico e físico [13], detalhadas a seguir:

- **Conceitual:** modelo que representa fielmente o negócio em questão, demonstrando características do ambiente observado. Esse modelo é independente de qualquer SGBD e procura focar mais em como será a lógica do sistema em si.
- **Lógico:** os modelos lógicos são os modelos em que os objetos, suas características e relacionamentos têm sua representação de acordo com as regras de implementação.
- **Físico:** elaborado a partir do Modelo lógico levando em consideração os limites impostos pelo SGBD e pelos requisitos não funcionais dos programas que acessam os dados.

Esses modelos são representados em forma de diagramas, chamados de Diagrama Entidade Relacionamento (DER). O diagrama facilita a comunicação entre os integrantes da fábrica de software, pois oferece uma linguagem comum utilizada pelos analistas e os desenvolvedores [21].

2.1.4 Normalização

A normalização trata de um conjunto de regras que devem ser seguidas para que uma tabela seja considerada bem projetada. Existem várias formas normais, sendo cada uma delas um conjunto de regras diferentes [6]. As três primeiras formas normais são fundamentais para garantir um mínimo de redundância para um bom modelo de dados [9]. Essas 3 formas serão detalhadas a seguir.

- **Primeira Forma Normal (1FN):** uma tabela encontra-se na primera forma normal quando não contém tabelas aninhadas dentro dela, ou seja, informações não diretamente relacionadas guardadas de maneira redundante dentro da tabela.
- **Segunda Forma Normal (2FN):** uma tabela está na segunda forma normal quando, além de estar na primeira forma normal, não apresenta dependências parciais. Uma coluna não pode depender de apenas parte da chave primária.
- **Terceira Forma Normal (3FN):** a terceira forma normal acontece quando a tabela está na segunda forma normal e todo atributo chave não for dependente de outro atributo que não pertença a chave primária.

2.2 MySQL

O MySQL é um dos principais banco de dados relacionais de código aberto. É utilizado por 9 dos 10 principais sites do mundo, assim como por milhares de aplicativos corporativos baseados na web. O MySQL foi desenvolvido e otimizado para aplicativos web, tornou-se a plataforma preferida dos desenvolvedores web e o banco de dados padrão para aplicativos web [1].

Segundo o *DB-Engines* [1] MySQL está entre os principais bancos de dados utilizados no mercado. É usado por Facebook, Twitter, LinkedIn, Yahoo!, Amazon Web Services. É um dos bancos mais populares entre as startups. Facebook, Twitter, Wikipedia, Youtube e Ticketmaster são exemplos de grandes sites que empregam o MySQL em alguns de seus serviços [2].

[1] http://db-engines.com/en/

A arquitetura do MySQL é composta por três camadas: conexão com o cliente, análise do SQL e execução da instrução SQL. A figura ilustrada em 2.1 é a visualização da arquitetura lógica [24]. Na primeira camada está o serviço de conexão com o cliente, por onde as aplicações se conectam com o banco de dados. É por essa camada que é feita a autenticação dos usuários, é necessária boa segurança pois é a única camada que outras aplicações acessam o MySQL. Na segunda camada acontece a análise dos comandos SQL, (*query*). O objetivo da análise das *queries* é verificar a possibilidade de otimizá-la, essa análise é feita no bloco *Parser*. É nessa camada que são construídas as *procedures*, *triggers* e *views*. A *Query Cache* é responsável por armazenar os últimos comandos, em comandos de consultas, esse bloco salva os úlimos resultados, ou seja, caso uma consulta no primeiro momento foi realizada em 10 segundos, por exemplo, ela será bem mais rápida quando executada pela segunda vez, se executada logo após a primeira vez. A camada *Storage Engines* é responsável por armazenar e recuperar todos os dados. Essa camada não analisa os comandos SQL, ela apenas responde as requisições que vem das camadas acima.

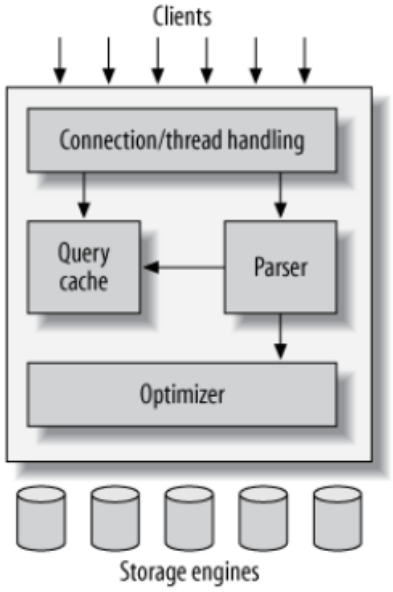

Figura 2.1: Visualização da arquitetura lógica do MySQL (Fonte: [24])

2.3 Banco de Dados Não Relacionais - NoSQL

Um dos motivos para o surgimento de novos paradigmas e tecnologias de armazenamento de dados foi atender a grande demanda de volume de dados gerados por aplicações, principalmente aplicações WEB. Os bancos de dados não relacionais surgiram com o objetivo de atender essa demanda, com foco em gerenciamento de grandes volumes de dados, semi-estruturados ou não estruturados, que necessitam de alta disponibilidade e escalabilidade [17].

Os bancos de dados não relacionais não apresentam todas as características ACID. Os bancos de dados orientado a objetos e o NoSQL são exemplos de bancos que se encaixam nesse contexto [6]. NoSQL (*Not Only SQL*) é um conjunto de conceitos que permitem o processamento de dados de forma rápida e eficiente com o foco em performance [8]. É uma alternativa para modelar dados sem se preocupar com os padrões rígidos proposto pelo modelo relacional.

O banco de dados NoSQL tem uma estrutura distribuída e tolerante a falhas que se baseia na redundância de dados em vários servidores. Em consequência disso, o sistema pode ser escalado facilmente agregrando mais servidores, e assim a falha de um deles pode ser tolerada. Bancos NoSQL são projetados para trabalhar com uma grande quantidade de dados distribuídos. Para isso algumas características são bem comuns nesses bancos [17]:

- **Escalabilidade Horizontal:** consistem em cada aplicação ser capaz de aumentar o número de nós, ou seja, máquinas sem que afete o sistema negativamente.
- **Ausência de esquema ou esquema flexível:** a ausência de esquema facilita a escalabilidade e também contribui para o aumento da disponibilidade. Porém não dá garantias quanto a integridade dos dados
- **Suporte nativo a replicação:** permitir a replicação de forma nativa, diminui o tempo gasto para recuperar informações.

Essas características fizeram com que os bancos de dados NoSQL ficassem populares tanto na indústria quanto na academia quando se trata de armazenar grandes volumes de dados. A utilização de bancos NoSQL se dá principalmente, devido ao aumento gradativo

de informações a serem armazenadas, no qual o desempenho é prejudicado e o tempo de resposta se torna um fator preocupante.

2.3.1 Breve histórico

O termo NoSQL foi usado pela primeira vez em 1998 para citar um banco de dados relacional open-source que omitia o uso de SQL, o Strozzi NoSQL, criado por Carlo Strozzi [18]. O nome veio pelo fato que o banco de dados utilizava shell script como linguagem de busca, não o SQL. Hoje, o uso do termo não se refere ao banco desenvolvido por Strozzi. Em 2009, na conferência, "NoSQL Meetup", que foi organizada por Johan Oskardson, o criador do Last.fm [28], o termo foi utilizado novamente, mas agora referenciando os bancos de dados não relacionais.

Na conferência ficou claro que o uso do Amazon's Dynamo e do Google Bigtable, precursores do movimento NoSQL, estava crescendo. Hoje existem vários tipos de bancos de dados NoSQL diferentes, que podem ser dos seguintes modelos: chave-valor, orientado a colunas, orientado a documentos e orientado a grafos. Cada um desses modelos serão detalhados no tópico 2.3.3

2.3.2 Propriedade BASE

Diferente dos bancos de dados relacionais, os bancos NoSQL não se baseiam na propriedade ACID e sim na propriedade BASE. BASE (*Basically Avaliable, Soft State, Eventual Consistency*) significa basicamente disponível, estado leve e consistente em momento indeterminado. A consistência eventual é uma característica dos bancos NoSQL relacionada ao fato da consistência nem sempre ser mantida entre os diversos pontos de distribuição de dados. Essa característica é baseada no teorema CAP (*Consistency, Availability e Partition tolerence*) que diz que em um dado momento, só é possível garantir duas das três propriedades entre a consistência, disponibilidade e a tolerância à partição [17]. Basicamente, a consistência diz respeito à ordem de execução das requisições. Disponibilidade é a propriedade de um sistema responder a todas as requisições que chegam a um nó. A tolerância a partição é a propriedade de um sistema continuar funcionando mesmo quando um problema ocorre na rede dividindo o sistema em uma ou mais partições. Baseado nesses conceitos a

propriedade BASE indica que se deve planejar um sistema de forma a tolerar inconsistência temporárias quando se quer priorizar a disponibilidade [10].

A disponibilidade da propriedade BASE é garantida tolerando falhas parciais no sistema, sem que o sistema todo falhe. Por exemplo, se um banco de dados está particionado em cinco nós e um deles falha, apenas os clientes que acessam aquele nó serão prejudicados, pois o sistema como todo não irá interromper seu funcionamento [25].

2.3.3 Modelos de Banco de dados NoSQL

Atualmente é possível encontrar diversos modelos de bancos de dados NoSQL. Entre os modelos de dados mais importantes, podemos encontrar os modelos de chave-valor, orientado a coluna, orientado a documentos e orientado a grafos. Cada modelo possui vantagens e desvantagens dependendo de onde for aplicado. É importante lembrar que cada modelo de dados tem formas diferentes de armazenamento e consulta. A seguir será detalhado cada modelo de dados.

No modelo **chave-valor**, o sistema armazena dados estruturados como pares de chaves e valores. Uma chave é um identificador para diversos valores, que podem ser expressos por índices *hash*. Dessa forma, é o modelo de estrutura mais simples. Inserções de dados e consultas nesse modelo são realizadas intrinsecamente sobre as chaves [7]. A desvantagem deste modelo é que não é possível realizar consultas mais complexas [17]. Exemplos de banco de dados que seguem o modelo chave-valor são: Redis[1], Riak[2] e o Dynamo[3], este último foi criado pela Amazon e foi usado como base para o desenvolvimento do Cassandra. No **Orientado a Coluna** o armazenamento dos dados são em colunas de uma tabela. Porém, diferentemente do modelo relacional, essas tabelas não possuem relacionamento e são armazenadas separadamente. Portanto, cada coluna é exclusivamente independente de cada tabela. Além disso, as colunas possuem índices padrões e forma de compressão dos dados para melhorar o processamento de consultas e o armazenamento [7]. Aqui existe o conceito de *Column Family*, que é usado com o intuito de agrupar colunas que armazenam os mesmo tipos de dados. Alguns exemplos de bancos deste tipo são o Hbase[4] e o Cassandra[5],

[1]http://redis.io/
[2]http://basho.com/products/
[3]http://aws.amazon.com/pt/documentation/dynamodb/
[4]http://hbase.apache.org/
[5]http://cassandra.apache.org/

que será detalhado na Seção 2.4. Já no **Orientado a documentos**, o armazenamento dos dados são em coleções de documentos. Um documento é um objeto com um identificador único e um conjunto de campos, que podem ser textos, listas ou outros documentos. Neste modelo temos um conjunto de documentos em que cada documento contém um conjunto de campos (chaves) e o valor deste campo. O modelo não depende de um esquema rígido, ou seja, não existe uma estrutura fixa como nos bancos relacionais. Esta flexibilidade é uma das grandes vantagens deste modelo. Exemplos de bancos deste tipo são CouchDB[6] e MongoDB [7]. Este último será detalhado na Seção 2.5. O modelo **Orientado a grafos** possui 3 componentes básicos: os nós (vértices dos grafos), os relacionamentos (as arestas), e os atributos. Neste caso o banco de dados pode ser visto como um conjunto de grafos rotulado e direcionado. A vantagem de utilização do modelo baseado em grafos é permitir a execução rápida de consultas complexas. Exemplos de bancos: Neo4j[8] e AllegroGraph[9].

Não existe um modelo melhor que o outro, tudo depende de como será projetada a aplicação. Por exemplo, para manipulação de dados estatísticos, frequentemente escritos mas raramente lidos, pode ser usado um banco de dados do tipo chave e valor, o Redis, ou um banco orientado a documentos, o MongoDB. Caso queira uma aplicação com alta disponibilidade, onde a minimização da inatividade é fundamental, recomenda-se utilizar um banco de dados orientado a coluna, o Cassandra por exemplo. Para aplicações que necessitam de alto desempenho de consultas e com muitas junções, o ideal é usar um banco de dados orientado a grafos. Fica claro conhecer a importância de cada um desses modelos, pois dependendo da aplicação em questão pode-se obter bons resultados ou não.

2.4 Cassandra

O Cassandra é um banco de dados distribuído massivamente escalável, criado para armazenar uma grande quantidade de dados espalhados por vários servidores e, mesmo assim, oferecer alta viabilidade de acesso a dados consistentes [3]. Avinash Lakshman e Prashant Malik foram os seus criadores. Seu lançamento ocorreu em 2008 como um projeto open source. Em 2009 foi adotado pela Apache Software Foundation.

[6]http://couchdb.apache.org/
[7]https://www.mongodb.org/
[8]http://neo4j.com/
[9]http://franz.com/agraph/allegrograph/

O Cassandra foi baseado em outros 2 bancos NoSQL, o Dynamo da Amazon e o BigTable da Google. Foi baseado na arquitetura do Dynamo enquanto o modelo de dados foi baseado no BigTable. Mesmo que seu modelo de dados seja voltado a coluna, o Cassandra permite a consulta como no modelo chave-valor, podendo ser considerado um modelo híbrido. As principais característica do Cassandra são: distribuído, descentralizado, escalável, altamente disponível, tolerante a falhas e alta perfomance.

O foco principal do Cassandra é no desempenho das consultas, desta forma, não existe uma estrutura de dados que estabelece os relacionamentos entre uma tabela e outra. Os principais elementos do Cassandra são: Cluster, Nó, Data Center, Keyspaces, as famílias de colunas, as tabelas, colunas e linhas.

O *Cluster* é um grupo de nós ou apenas um nó onde se armazena os dados, enquanto que o nó é uma instância física do Cassandra, ou seja, uma máquina que esteja executando o Cassandra. O Data Center é um grupo de nós que pertencem ao mesmo cluster, não é necessariamente um data center físico. O *Keyspace* é o agrupamento de dados, similares ao banco de dados relacional. Porém o *Keyspace* possui informações como o fator de replicação e a estratégia de armazenamento. As famílias de colunas ou tabelas são agrupamento de colunas ordenadas por nome que é pesquisada por linha. Coluna é a menor unidade para armazenar dados, sendo composta pelos campos: nome, valor e um campo que registra o momento de alguma alteração do dado. Esse último campo é importante, pois é uma estratégia que o Cassandra utiliza para gerenciar a consistência desse dado a ser consultado. Uma coluna em uma família de coluna deve ter, pelo menos, em cada um do seus campos uma chave primária, chamada de row key. A linha, diferente das de um banco de dados relacional, é um conjunto de colunas que possuem a mesma chave primária. Outra diferença é a forma de armazenamento da linha, no relacional já se aloca espaço para todas colunas de uma linha mesmo que seu valor seja NULL. Por outro lado o Cassandra só aloca espaço para uma linha quando já existem colunas presentes, causando um menor esforço computacional.

O Cassandra possui uma linguagem de consulta própria chamada CQL (Cassandra Query Language), muito semelhante ao SQL. A diferença entre as duas linguagem é que a CQL é mais simplificada e não suporta alguns recursos como junções *(joins)* e agrupamentos *(group by)*.

A escrita no Cassandra passa por algumas etapas. Quando uma escrita ocorre, o Cassandra armazena os dados na memória RAM, em uma estrutura chamada *memtable*, enquanto a instrução de escrita é gravada em outra estrutura no disco chamada *commitlog*. O *commitlog* é importante para que em caso de falha no hardware não percam os registros que serão inseridos. Quando a *memtable* fica cheia acontece o processo de *flush*, os dados são descarregados para outra estrutura em disco chamada de *SSTable*. Após o *flush*, o *commitlog* é apagado. Para um dado ser escrito de forma rápida no banco, é necessário diminuir a memória da *memtable*, pois assim os dados são escritos mais rápidos no *commitlog*, mas isto diminui a perfomance da consulta. A figura 2.2 mostra a Estrutura da escrita e leitura do banco.

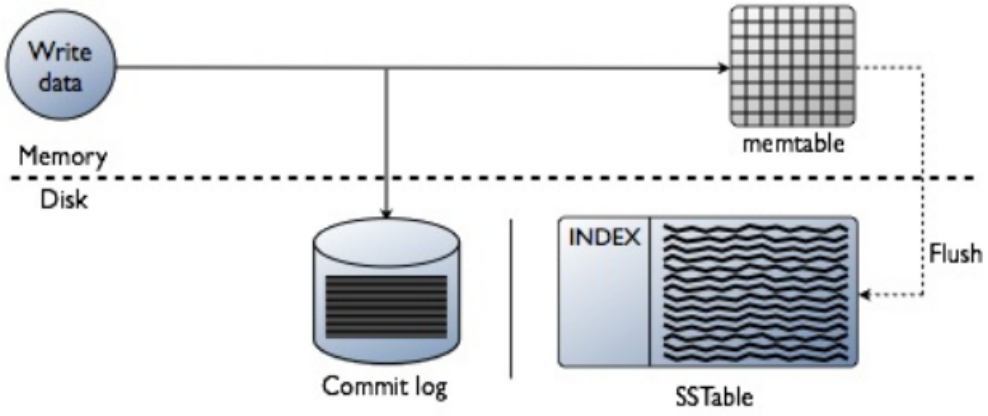

Figura 2.2: Estrutura da escrita e leitura no Cassandra. (Fonte: [6])

Uma instrução de remoção de dados não os apaga imediatamente. Após o recebimento da instrução o dado fica marcado como apagado na *SSTable*. Um processo de compactação executa a instrução e agrupa os fragmentos de linhas para otimizar a pesquisa.

2.4.1 Protocolo Gossip

O Protocolo *Gossip* (fofoca) é usado para fazer a comunicação entre os nós de um *cluster* do Cassandra. O protocolo funciona da seguinte maneira: quando um nó "A" se comunica com o nó "B" no *cluster*, o nó "A" além de passar suas próprias informações ele repassa também as informações dos outros nós que ele se comunicou anteriormente. Isso faz com que o nó "B" não precise se comunicar diretamente com os outros nós do *cluster* [12].

O Cassandra usa o protocolo *Gossip* para descobrir a localização e o estado em que os nós do *cluster* se encontram. Como os nós estão em constante comunicação, ao utilizar o *Gossip* ocorre uma redução nos logs do sistema como também no consumo dos recursos da rede. Outros fatores importantes ao utilizar o protocolo *Gossip* são que os nós não precisam saber como está a distribuição dos demais no *cluster* e as informações trocadas entre eles estão sempre atualizadas.

2.4.2 Exemplos de Aplicações

O Cassandra é utilizado no Facebook para otimização do sistema de busca. Também é utilizado para dar suporte à replicação, detecção de falhas e armazenamento de cache.

O Twitter utiliza o Cassandra para armazenar resultados de data mining realizados sobre a base de usuários, resultados de *trend topics*, *@toptweets* e análises em tempo real em larga escala. A utilização do Cassandra trouxe vantagens tanto na implementação da modelagem dos dados relacionados *tweets, timeline*, entre outros, como no desempenho com relação aos campos de busca de usuários ou por palavras-chaves. Também aumentou a disponibilidade dos serviços. Outras grandes empresas também utilizam o Cassandra, como Netflix, Ebay e Cisco, além de orgãos governamentais como a NASA [17].

2.5 MongoDB

O MongoDB é um banco de dados NoSQL orientado a documentos e de código livre. O desenvolvimento do MongoDB começou em 2007 e a primeira versão foi lançada em 2009, foi implementado em C++ [15]. Hoje é um banco de dados referência para o modelo orientado a documentos.

Um registro em MongoDB é um documento. Documentos no MongoDB são objetos JSON (JavaScript Object Notation). Os valores dos campos podem incluir outros documentos, matrizes e matrizes de documento, além de estruturas de dados mais simples como textos, datas, números inteiros e pontos flutuantes.

MongoDB armazena documentos no disco no formato BSON serializado. BSON é uma representação binária de documentos JSON, embora contenha mais tipos de dados que

JSON. O Tamanho máximo de um BSON é de 16MB. O modelo de dados do MongoDB é bastante simples de compreender. O Banco de dados armazena um conjunto de coleções. A Coleção armazena um conjunto de documentos. O Documento é um conjunto de campos. O Campo é uma chave do objeto que foi inserido, enquanto que o valor pode ser um texto, outro documento, um inteiro, um ponto flutuante, um binário, um campo data e outros.

O MongoDB possui uma linguagem de consulta completamente diferente do CQL e do SQL. É uma linguagem mais simples, necessitando de poucas linhas para executar a instrução.

2.5.1 Exemplos de Aplicações

Entre as principais organizações que utilizam o MongoDB podemos listar: *The Weather Chanel* (canal que mostra as previsões do tempo em todo o mundo), a Forbes (Revista de Negócios e economia americana) e a *Otto Shopping Online*. Outras organizações que utilizam o MongoDB podem ser encontrado em [16].

2.6 Comparação Qualitativa entre os bancos escolhidos

Diante de tudo que foi visto a respeito dos três bancos de dados que serão usados durante o trabalho, esse tópico apresenta uma comparação qualitativa entre o MySQL, Cassandra e MongoDB com o foco em: Instalação, Compatibilidade, Usabilidade e Escalabilidade. Essas comparações são baseadas no uso dos bancos de dados no sistema operacional Ubuntu 14.04 - 64bits.

Na instalação são avaliados a disponibilidade de tutoriais e documentação que podem ser encontrados e o grau de dificuldade para realizar a instalação. Será feita uma análise no número de sistemas operacionais que esses bancos de dados tem compatibilidade, levando em consideração também como é feita a instalação nos principais sistemas operacionais. A usabilidade pode ser entendida como o grau de facilidade que o sistema oferece para o usuário realizar uma determinada ação. A escalabilidade é a forma de monitorar e criar soluções para o crescimento do sistema sem afetar o desempenho, em banco de dados, pode ser dividida em duas formas: vertical e horizontal. Na vertical o crescimento é realizado aumentando os

recursos computacionais do servidor como memória RAM, espaço em disco e processadores. Já a escalabilidade horizontal é a adição de novas máquinas no sistema, com o objetivo de distribuir o processamento entre elas.

A figura 2.3 resume a comparação dos três bancos de dados considerando as características escolhidas. O número máximo de estrela para a melhor avaliação é três.

Fator	MySQL	Cassandra	MongoDB
Instalação	★★★	★	★★
Compatibilidade	★★★	★★	★
Usabilidade	★★	★★	★★★
Escalabilidade	★	★★★	★★

Figura 2.3: Comparação qualitativa entre os três bancos de dados. (Fonte: o autor)

O MySQL possui uma instalação fácil, bastando algumas poucas linhas de comando no terminal no caso do sistema operacional Ubuntu 14.04-LTS. Oferece ferramentas robustas e fáceis de usar. Por ser um dos bancos mais populares na comunidade dos desenvolvedores, possui um bom suporte às linguagens de programação e é compatível com a maioria dos sistemas operacionais. Apesar de ter formas de escalar o MySQL horizontalmente, o banco de dados tem por padrão a escalabilidade vertical, ou seja, aumentar os recursos do servidor (memória, espaço em disco e processador).

No Cassandra, a vantagem principal é a sua escalabilidade, este banco de dados foi projetado para ser escalado horizontalmente. Possui bom suporte às principais linguagens de programação. O Cassandra é compatível com Windows, Debian e Ubuntu, para os demais sistemas operacionais tem a opção de baixar o pacote compactado. Para usuários que conhecem o SQL, entender o CQL não é um problema. Tem uma ferramenta bem útil, chamada de Opscenter, usada para gerenciar tanto os dados quanto os nós do *cluster*. A instalação do Cassandra, quando comparada ao MySQL e MongoDB, é a mais complicada. É necessário que o usuário tenha bons conhecimentos do sistema operacional que está usando, no linux principalmente. Por ser um banco de dados recente, o Cassandra possui poucas ferramentas de gerenciamento e também possui pouco material de suporte ao uso e instalação disponível.

A instalação do MongoDB é bastante simples, possui uma boa documentação no site oficial. Apesar da linguagem de consulta ser completamente diferente da SQL e CQL, ela é mais simples e oferece uma baixa curva de aprendizado. O MongoDB também tem a escalabilidade horizontal, diferente do Cassandra, na escalabilidade do MongoDB existe distribuição de tarefas diferentes por cada máquina no *Cluster*. Tem um ótimo suporte às principais linguagens de programação. A compatibilidade do MongoDB para Ubuntu acontece somente para as versões LTS. No caso do MAC OS e no Solaris a distribuição não possui encriptação SSL.

A escolha desses três bancos de dados foram motivadas pela compatibilidade com o sistema operacional Ubuntu 14.04 LTS e suporte a linguagem de programação Java. A escolha do MySQL se deve ao fato de que é um dos bancos de dados relacionais mais utilizado e o preferido dos desenvolvedores. O MySQL oferece um ótimo suporte as linguagens de programação, entre elas encontramos a linguagem de programação Java, boas documentações e tutoriais são facilmente encontrados na internet. Outras vantagens de utilizar o MySQL são boa usabilidade e muitas ferramentas para o gerenciamento do banco. Grandes organizações utilizam o MySQL, então podemos supor que o MySQL tem características para atender a demanda dos dados genômicos.

A escolha dos dois bancos de dados Não Relacional foi motivada por eles apresentarem diferentes estratégias de gerenciamento de dados. O Cassandra foi escolhido pelo fato de ser um banco de dados altamente disponível e por ser projetado para atender a grande demanda de dados. A exemplo das organizações que utilizam esse banco de dados, como o Twitter e Facebook, o Cassandra é usado principalmente nos campos de buscas dessas redes sociais. Como foi um banco de dados que se originou através de outros dois bancos, o Dynamo da Amazon e o BigTable da Google, ambos utilizados para atender grande massa de dados, pode-se concluir que o Cassandra tem condições de suprir a necessidade de armazenar o conteúdo dos arquivos genômicos deste trabalho. Igual ao MySQL, o Cassandra oferece bom suporte a linguagem de programação Java. Outro fator importante na escolha deste banco é que o Cassandra é distribuído e escalável horizontalmente, os dados são distribuídos entre várias instâncias desse mesmo banco de dados que estejam executando em outras máquinas.

O MongoDB foi escolhido, assim como o Cassandra, por ser um banco de dados projetado para atender uma grande demanda de dados. Como o MongoDB armazena os registros no formato JSON, torna-se fácil gerenciar os dados através de uma linguagem de programação. Apesar de não ter uma linguagem de consulta similar ao SQL e o CQL, utilizados no MySQL e no Cassandra, respectivamente, a linguagem de consulta do MongoDB é simples e não apresenta maiores dificuldades para compreensão. Outro motivo da escolha desse banco foi por ele ser projetado para ser livre de esquema, ou seja, ao trabalhar com o MongoDB, o usuário não se preocupa em como os dados tem que ser inseridos, pois ele permite armazenar dados pertencentes a mesma tabela (que no caso do MongoDB chama-se Coleção) com números de atributos (campos) diferentes. Esta característica oference um esquema dinâmico e livre de regras de inserção e extração.

Capítulo 3

Armazenamento de Dados Genômicos

Os dados genômicos são gerados através do sequenciamento de amostras de algum material genético, podendo ser um tecido, órgão ou partes menores. Ao fim do sequenciamento são gerados arquivos brutos que contém diversas informações, dentre elas: a cadeia de DNA, os identificadores referentes a cada cadeia de DNA e as informações do sequenciador que realizou o procedimento. Os prováveis valores encontrados durante o sequenciamento de DNA são as 4 bases de nucleotídeos: Adenina (A), Guanina(G), Citosina (C), Timina(T) e um valor que representa o indefinido, que é utilizado quando o sequenciador não consegue identificar.

Após isso, o arquivo passa por processos que limpam informações indesejadas, como por exemplo o nucleotídeo que o sequenciador não conseguiu identificar e as cadeias pequenas de DNA. Normalmente, depois desse processo, os arquivos filtrados são armazenados em repositórios compartilhados. Uma outra alternativa, é o uso de bancos de dados para melhorar a organização e a consulta das sequências de DNA. Esta alternativa tem sido investigada na academia [6, 14, 22], já que os bancos de dados oferecem melhor organização dos dados, podendo agregar meta-informações das sequências de DNA encontradas nesses arquivos.

Esse capítulo apresenta algumas estratégias que serão utilizadas para o armazenamento dos dados genômicos, bem como uma descrição desses arquivos. A seção a seguir apresenta os formatos desses dados. Em seguida, será mostrado como é realizado o processamento dos dados. Também serão mostradas as estratégias de armazenamento levantadas durante o desenvolvimento do trabalho. Por fim, serão mostrados os esquemas utilizados nos três bancos de dados.

3.1 Formatos dos Arquivos

Existem vários formatos de arquivos para dados genômicos. Tais formatos podem ser encontrados em grandes bancos de dados online como o *National Center for Biotechnology Information* (NCBI) [1] e são utilizados na maioria das aplicações genômicas. Entre esses formatos estão os *fastq, csfasta* e o *fasta*.

O *fastq* é um formato que armazena tanto a sequência de DNA quanto a qualidade de cada sequência ambos codificados em ASCII. Já o formato *fasta* é a forma mais simples de representar um sequenciamento, sendo dividido em duas partes: identificador de sequência, que é representado após o sinal de "maior que", e a própria sequência de DNA, que vem na próxima linha após o seu identificador. O arquivo *fasta* é encontrado com o sufixo ".fasta" ou ".fa". A figura 3.1 ilustra um exemplo de um arquivo fasta, onde a primeira e a terceira linha são os identificadores de sequência e a segunda e a quarta são as sequências de DNA propriamente.

```
>746_81_147_F3
CCACTACGTTCCACGGCACACTACTAAGTCCGTGTGACACACACACACAC
>746_81_203_F3
CCACTACGTTCCACGGCACACTACTAAGTCCGTGTGACACAGTGTGTGTG
```

Figura 3.1: Exemplo de um arquivo fasta. (Fonte: o autor)

A representação dos arquivos *fasta* podem ser definidas em blocos de *Short Read Sequence* (SRS). Uma SRS é representado como uma tupla que combina o identificador de sequência com a sequência de DNA referente a esse identificador. Na figura acima, tem-se duas SRSs, a primeira é o par das duas primeiras linhas e a segunda SRS corresponde às duas últimas.

O *csfasta* é um arquivo proprietário e é gerado logo após o término do sequenciamento, pelo próprio sequenciador. Além de informações complementares no cabeçalho do arquivo, o formato *csfasta* armazena o identificador de sequência e a cadeia de DNA referente, que representa os nucleotídeos através de números. Um exemplo de uma parte do arquivo *csfasta* é ilustrado na figura 3.2

[1] http://www.ncbi.nlm.nih.gov/

```
>sequence
T3122013131
```

Figura 3.2: Exemplo de um arquivo csfasta. (Fonte: o autor)

Esses números se referem às cores que o sequenciador gera quando está processando o material genético. Geralmente, os números estão no intervalo de 0 a 3 (onde azul=0, verde=1, amarelo=2 e vermelho=3) [4]. Cada um desses números se referem a um possível par de DNA, como mostra a tabela 3.1.

AA	CC	GG	TT	0	azul
AC	CA	GT	TG	1	verde
AG	CT	GA	TC	2	amarelo
AT	CG	GC	TA	3	vermelho

Tabela 3.1: Combinação dos números com o par de cadeia de DNA. (Fonte: [4])

Para realizar a conversão do arquivo *csfasta* para o arquivo *fasta* é necessário conhecer as possíveis combinações de cada número. Levando em consideração os números da figura 3.2, o primeiro passo a fazer é realizar a combinação do nucleotídeo "T", que é o nucleotídeo padrão para iniciar o sequenciamento, com o número 3. Observando a tabela 3.1 vemos que essa combinação resulta no nucleotídeo "A". Esse procedimento é feito a cada novo nucleotídeo encontrado. A tradução da sequência na figura 3.2 para o arquivo *fasta* é mostrada na figura 3.3.

```
>sequence
ACTCCATGCA
```

Figura 3.3: arquivo fasta resultante da filtragem do csfasta. (Fonte: o autor)

3.2 Processamento de dados genômicos

Após realizado o sequenciamento do material genético, o sequenciador gera um conjunto de arquivos texto que indicam a qualidade, as informações sobre o sequenciador e a própria sequência de DNA. O aquivo que possui as sequências de todo o conteúdo, na maioria das vezes, se encontra no formato *csfasta*. Após esse procedimento, a sequência encontrada pode ser tratada e utilizada das mais variadas formas, de acordo com o estudo em questão.

Segundo Röhm e Blakeley [22], após o sequenciamento, os arquivos de dados genômicos, na maioria das vezes, são armazenados em repositórios de arquivos. O problema dessa estratégia, conforme apontam os autores, é que as informações encontradas a respeito desses arquivos não são suficientes, pois os meta-dados desses arquivos ou não são armazenados ou não são facilmente localizáveis. Assim, o uso do banco de dados pode trazer muitas vantagens, comparado ao armazenamento em repositório de arquivos. Podemos citar as seguintes vantagens: melhor organização dos dados, melhor uso do espaço disponível, ou seja, o próprio banco de dados conta com recursos de compressão de dados a fim de melhorar o espaço para o armazenamento e a capacidade de agregar meta-informações das sequências de DNA encontradas nesses arquivos, indexando essas informações com os arquivos referentes. Além disso, os bancos de dados fornecem controle de acesso aos usuários e segurança dos dados armazenados.

O diagrama da Figura 3.4 apresenta como um banco de dados pode ser integrado ao *pipeline* de processamento de dados genômicos realizado. As etapas deste *pipeline* envolvem desde a filtragem dos arquivos *csfasta* até o processamento dos arquivos *fasta* utilizados por meio de ferramentas de análise das sequências de DNA, e passa pelas etapas referentes à inserção e extração dos dados no banco.

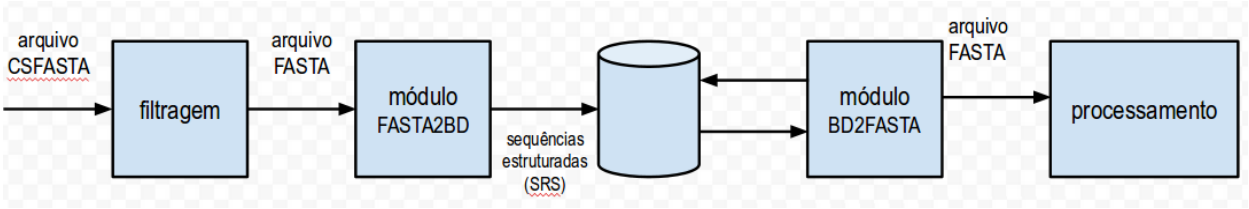

Figura 3.4: Diagrama de todo processo de tratamento dos arquivos Fasta. (Fonte: o autor)

A filtragem do arquivo *csfasta* para o *fasta* é feita seguindo os passos mostrados anteriormente e utilizando software de tradução implementado em *python*. Além disto, a filtragem consiste também na remoção de sequências de baixa qualidade e na remoção de sequências com características indesejadas. Após a criação dos arquivos *fasta*, realiza-se um tratamento que faz a leitura do conteúdo do arquivo gerado e transforma as SRSs encontradas por meio do módulo Fasta2BD. As estratégias de armazenamento destes dados no banco serão discutidas em detalhes na Seção 3.3. O módulo BD2Fasta realiza extrações de todo o conteúdo de um arquivo e também pode realizar consultas por uma SRS, de modo a localizar sequências específicas de DNA. Os dados obtidos na extração dos conteúdos são armazenados novamente no formato *fasta*. O armazenamento dos dados extraídos do banco de dados em arquivos deste tipo é essencial, já que as ferramentas de análise das sequências de DNA, utilizadas na fase de processamento, comumente reconhecem apenas os arquivos desse formato.

3.3 Estratégias de Armazenamento

Sabendo da necessidade de utilizar um banco de dados, foram planejadas várias estratégias com o objetivo de saber qual a melhor forma de armazenar os dados genômicos. Durante o planejamento foram encontrados alguns trabalhos referentes ao gerenciamento desses dados utilizando bancos de dados.

Os trabalhos encontrados discutem qual a melhor forma de armazenamento dos dados genômicos. A dissertação de mestrado de Ruben Cruz Huacarpuma [14] propõe o desenvolvimento de um modelo de dados para um pipeline de sequenciamento de alto desempenho, utilizando o MySQL. O modelo de dados tinha informações do sequenciador, da espécie a que o material genético pertence, a qualidade do sequenciamento e o armazenameto do próprio arquivo, no formato *fastq*. O arquivo é armazenado em campos do tipo BLOB, estrutura de dados que normalmente é utilizada para armazenar arquivos de multimídia e binários. Já no trabalho de Rodrigo Aniceto e Rene Xavier [6], foi utilizado o Cassandra para fazer a análise dos mesmos arquivos utilizados no trabalho de Ruben Cruz Huacarpuma [14] para realizar uma comparação, porém os dados não foram armazenados da mesma forma que o trabalho anterior. O foco deste trabalho era realizar uma comparação em relação ao tempo de armazenamento e de extração desses dados. Este trabalho propôs um modelo que armazenava o

conteúdo do arquivo organizados em tabelas com várias colunas e os campos armazenavam o conteúdo dentro do arquivo. Ambos utilizavam o sistema operacional linux. O trabalho de Röhm e Blakeley [22] utilizou uma abordagem diferente para o armazenamento dos dados genômicos, utilizando o SQL Server 2008, um banco de dados relacional e normalizado. A abordagem consistia em criar um modelo de dados híbrido que tinha informações dos metadados dos arquivos e o próprio arquivo, armazenado em BLOB. Essa combinação apresentou bons resultados comparado ao armazenamento do arquivo em repositórios.

Neste trabalho, identificou-se quatro aspectos que devem ser considerados no projeto de um banco de dados genômico: número de colunas da tabela de armazenamento do arquivo; armazenamento das SRSs em estruturas BLOB ou em vários registros; tamanho do conjunto de SRSs por registro; e criação de uma tabela única ou de várias tabelas por arquivo. A figura 3.5 ilustra as estratégias que foram utilizadas nos trabalhos encontrados como também a estratégia utilizada neste trabalho.

Estratégia	Rohm e Blakeley, 2009 <SQL Server 2008>	Huacarpuma, 2012 <MySQL>	Aniceto e Xavier, 2014 <Cassandra>	Este trabalho
Número de colunas por Tabela	1	1	10	1
Armazenamento SRSs em BLOB ou em registros	BLOB	BLOB	Registros	Registros
Tamanho do conjunto da SRS por registro	Não se aplica	Não se aplica	5 (50 por linha da tabela)	1, 5 e 50
Número de tabela por arquivo	1 tabela para todos arquivos	1 tabela para todos arquivos	1 tabela por arquivo	1 tabela por arquivo

Figura 3.5: Estratégias utilizadas nos trabalhos encontrados. (Fonte: o autor)

Os trabalhos de Röhm e Blakeley [22] e de Huacarpuma [14] utilizaram bancos de dados Relacionais e utilizou a estratégia de armazenamento dos arquivos em BLOB em uma única tabela para todos os arquivos. Enquanto o trabalho de Aniceto e Xavier [6] utilizou um banco de dados Não Relacional onde era armazenado o conteúdo do arquivo em uma tabela com 10 colunas para cada arquivo com 50 registro por linha da tabela. Neste trabalho foram utilizados três bancos de dados, dois deles Não Relacional e o outro Relacional, onde foi

armazenado o conteúdo do arquivo em uma tabela específica para cada arquivo. O tamanho do conjunto da SRS por registro da tabela foi variado em três valores diferentes 1, 5 e 50.

A abordagem de utilizar várias colunas na tabela para armazenar os dados genômicos tem como objetivo de propor uma boa organização do conteúdo dos arquivos dentro do banco de dados, melhorando assim os tempos de inserção e extração. Esta abordagem foi analisada por Rodrigo Aniceto e Rene Xavier em [6] e apresentou bons resultados de armazenamento quando comparado ao de Ruben Cruz Huacarpuma [14]. Porém, não tiveram bons resultados nos tempos de extração dos arquivos. O impacto de utilizar a estrutura BLOB para armazenamento dos dados genômicos é ter um melhor controle dos arquivos quando comparado a utilização de repositórios de arquivos. Esta estratégia foi estudada no trabalho de Röhm e Blakeley em [22], eles chegaram a conclusão de que armazenar os dados genômicos em bancos de dados relacionais é melhor do que armazenar em repositório de arquivos, pois além de ter a possibilidade de inserir informações do conteúdo dos arquivos no banco de dados, eles diminuiram a utilização dos recursos de armazenamento.

Definir um número fixo de SRSs armazenada por linha da tabela é uma abordagem que foi pouco analisada em outros trabalhos. Nesta abordagem é possível observar o comportamento dos bancos de dados quando estão sendo armazenadas grandes cadeias de caracteres por campo da tabela, assim pode-se analisar se o tempo de inserção e de extração sofrem alguma influência. O armazenamento de uma SRS única por linha da tabela tem como consequência um número maior de registros inseridos no banco de dados, mas é a única forma que permite realizar consultas por identificadores de sequência específicos, visto que, quando inseridos em conjunto, os valores dos identificadores e das sequências de DNA serão concatenados de acordo com o tamanho definido de cada conjunto. Por outro lado, a vantagem de armazenar um conjunto com múltiplas SRSs por linha da tabela é que o número de registros inseridos será menor quanto maior for esse conjunto.

3.3.1 Esquemas dos Bancos de Dados

Como se pode observar há diversos aspectos que devem guiar o projeto de um banco de dados genômico, contudo, o restrito escopo deste trabalho permitiu que sejam avaliados apenas um destes aspectos, a saber: o tamanho do conjunto de SRSs. Considerando isto, esta seção apresenta os esquemas dos Bancos de Dados utilizados neste trabalho e discute quais aspectos guiaram o projeto destes esquemas.

Os esquemas dos bancos de dados foram projetados de acordo com os conceitos vistos no capítulo 2. O esquema do MySQL possui duas tabelas: *fasta_info* e *fasta_collect*. A primeira tabela armazena as informações dos arquivos existentes no banco de dados, como por exemplo, nome, número de linhas, tamanho em *megabyte* e um comentário do arquivo. Já a tabela *fasta_collect* armazena o conteúdo do arquivo, ou seja, as SRSs. Existem pelo menos 3 campos: identificador de sequência, sequência de DNA e a linha referente à localização do identificador de sequência no arquivo *fasta*. Os demais campos da tabela *fasta_collect* são a chave primária e a chave estrangeira referente à tabela *fasta_info*. O relacionamento entre essas tabelas segue os conceitos das três primeiras regras de normalização. O relacionamento em questão é do tipo 1 para N, ou seja, 1 arquivo possui N SRSs, assim como N SRSs pertencem a 1 arquivo. Ambas as tabelas têm como chave primária um campo chamado "id"que é incrementado automaticamente a cada registro inserido no banco de dados. A figura 3.6 ilustra o esquema do MySQL.

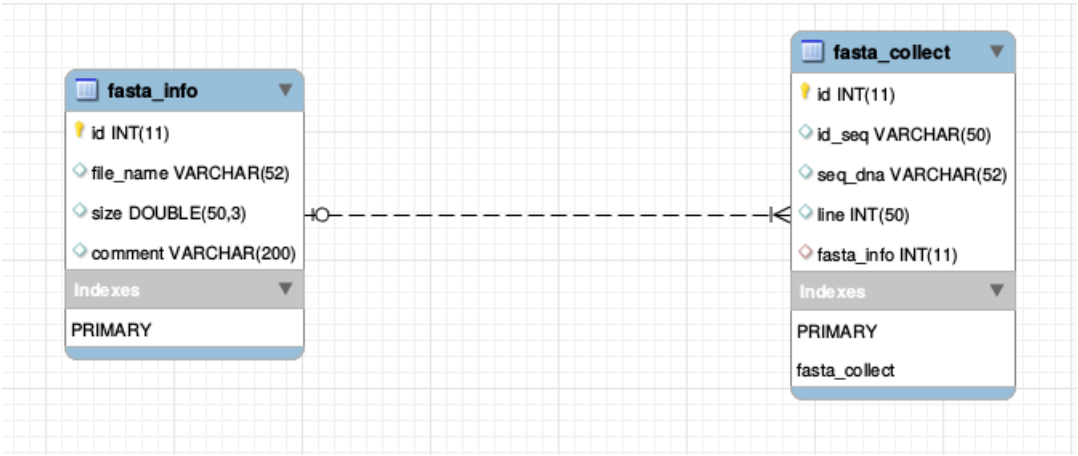

Figura 3.6: Esquema do MySQL. (Fonte: o autor)

Enquanto o MySQL tem um esquema bem definido e estático, o MongoDB e o Cassandra possuem um esquema totalmente dinâmico. A estratégia de usar esquema dinâmico é justificada pelo fato de que os dois bancos de dados NoSQL têm como característica uma modelagem livre de esquema. Os esquemas desses bancos de dados são representados da seguinte maneira: uma tabela da mesma estrutura da tabela *fasta_info* encontrada no esquema do MySQL e a cada arquivo inserido é criada uma nova tabela com o mesmo nome desse arquivo e possuindo os mesmos campos que a tabela *fasta_collect* com a exceção da chave estrangeira e do identificador auto incrementado. O Cassandra não permite a criação de tabela com "."então esse caracter é trocado por três "_"(*underlines*).

Então os esquemas desses dois bancos de dados Não Relacionais possuem o mesmo número de tabelas. Como os esquemas do MongoDB e do Cassandra não realizam o incremento automático dos campos de chave primária, o identificador de sequência tornou-se a chave primária para as tabelas criadas dinamicamente, enquanto que na tabela de índices, a *fasta_info*, o campo que tem a chave primária é o *file_name*. A figura 3.7 ilustra os esquemas do Cassandra do MongoDB. A diferença para o MongoDB é o nome das tabelas (coleções) criadas dinamicamente, elas têm o mesmo nome do arquivo.

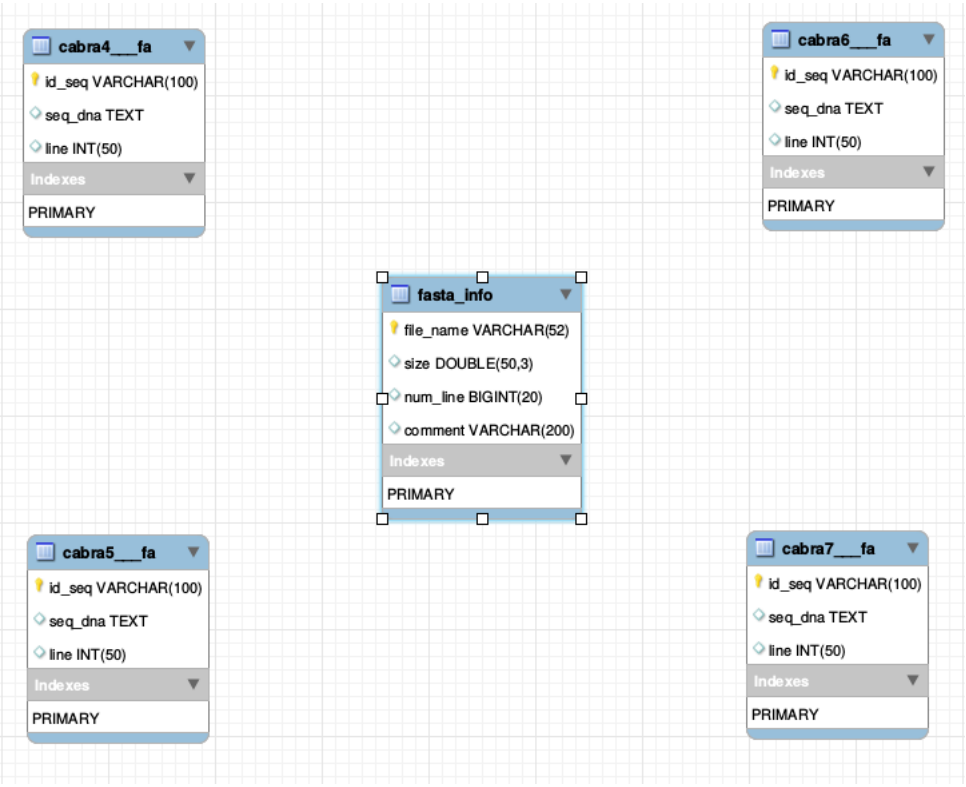

Figura 3.7: Esquemas do Cassandra e do MongoDB. (Fonte: o autor)

No MySQL, quando o número de SRSs por linha da tabela for maior que 1, é necessário trocar a estrutura de dados dos campos *seq_dna* e *id_seq* da tabela *fasta_collect* para *TEXT*, pois o tamanho do conteúdo a ser inserido será muito maior do que o suportado pelo *VARCHAR*. Já para o MongoDB e o Cassandra, não é necessário ter essas mudanças, pois os campos do tipo texto nesses bancos de dados suportam os valores que serão inseridos neste trabalho.

Capítulo 4

Avaliação de Desempenho

Este capítulo descreve e apresenta os resultados de experimentos que avaliam o desempenho de uma das estratégias de armazenamento discutidas no capítulo 3 quando implementadas nos bancos de dados MySQL, Cassandra e MongoDB. Os experimentos foram todos automatizados por meio de uma ferramenta de benchmark desenvolvida ao longo deste trabalho (Seção 4.1) e que permitiu medir os tempos individuais de inserção, extração e consulta aos bancos a partir de diferentes arquivos de dados genômicos. Toda a avaliação foi conduzida utilizando máquinas virtuais, que permitiram oferecer ambientes equivalentes a cada um dos bancos de dados escolhidos. Mais detalhes sobre o ambiente de testes será descrito na Seção 4.2. Os resultados obtidos são apresentados na Seção 4.3 e uma discussão geral sobre as observações é apresentada na Seção 4.4.

4.1 Ferramenta de Benchmark - FastaBD

Durante o decorrer do trabalho foi desenvolvida uma ferramenta de benchmark chamada FastaBD. O principal objetivo dessa ferramenta é automatizar o processo de criação do ambiente e medição dos tempos de inserção, consulta e extração. No capítulo 03 apresentamos o processo de tratamento dos arquivos *fasta* (conferir a Figura 3.4) em que figuravam dois módulos chamados Fasta2BD e BD2Fasta. A ferramenta de benchmark - FastaBD - realiza justamente as funções desses dois módulos.

A ferramenta FastaBD foi desenvolvida em Java, versão 7. É executada na máquina principal, não em máquinas virtuais. Como pré-requisito para utilizar essa ferramenta, é necessário um computador com memória RAM mínima de 4 *gigabytes* e um processador com quatro ou mais núcleos. A ferramenta possui um arquivo de configuração, chamado "configExperiment.properties", que permite a escolha do banco e os requisitos para iniciar a conexão, assim como, permite escolher os demais parâmetros do experimento.

Para iniciar a FastaBD, é necessário a passagem de dois parâmetros. O primeiro é o diretório onde se encontram os arquivos *fasta* e o segundo é o número de SRS que serão inseridas por cada linha da tabela no banco de dados. Esses valores podem ser inseridos também no arquivo de configuração, mas é recomendável inserir através dos parâmetros. O uso dessa ferramenta trouxe muitas vantagens, dentre elas a automação dos experimentos, como também, a flexibilidade de inserir configurações de outros bancos de dados para realizar essa análise.

4.2 Metodologia de Avaliação

Cada experimento nesta avaliação consiste em três etapas executadas nesta ordem: inserção, consulta e extração. Na inserção, as SRSs são lidas do arquivo sequencialmente e tratadas de acordo com o parâmetro do número de SRSs por linha para ser enviada ao banco de dados. Os arquivos foram inseridos na ordem que estavam distribuídos no diretório e o tempo de inserção individual de cada arquivo foi registrado. Na etapa da consulta, realiza-se a recuperação de 5 SRSs de cada arquivo a partir de seu identificador e o tempo desta consulta é medido. Por fim, a extração consiste no processo inverso da inserção, onde são obtidas todas as SRSs referentes a cada um dos arquivos para sua remontagem. Os tempos

de extração de cada arquivo são observados e registrados.

Dois fatores foram utilizados nestes experimentos: número de SRSs por linha e o banco de dados escolhido. Para o número de SRSs por linha foram utlizados três níveis: 1, 5 e 50 SRSs por linha da tabela, enquanto que os bancos de dados escolhidos foram o MySQL, Cassandra e MongoDB, conforme justificado na Seção 3.3. O objetivo de inserir conjuntos de SRSs diferentes é analisar o impacto no tempo de inserção e extração dos bancos de dados, uma vez que, quanto maior o conjunto da SRS por linha, menor o número de registros inseridos no banco de dados. Entretanto, deve-se observar que as cadeias de caracteres inseridas em cada registro são maiores, o que permite observar o compromisso existente entre estes dois aspectos em cada banco de dados. Para cada combinação dos níveis dos fatores escolhidos (9 combinações) foram obtidas 5 amostras a afim de obter resultados estatisticamente significativos e permitir conclusões confiáveis.

Para os níveis de SRSs por linha 5 e 50 ,a etapa de consulta não foi realizada, visto que o armazenamento de múltiplas SRSs em um único registro impossibilita a realização da consulta por somente um identificador de SRS, já que elas estão concatenadas de acordo com o tamanho do conjunto. Por isso, a etapa de consulta e o experimento da curva de consulta foram realizados somente quando o número de SRS por linha na tabela é 1.

Além deste experimento que busca observar o comportamento dos diferentes bancos quando sujeitos às diferentes formas de armazenamento, realizou-se um segundo experimento para observar o comportamento da consulta aos bancos de dados à medida que o número de registros aumenta. Este experimento, denominado curva de consulta, realiza consultas aleatórias de identificadores de sequências de DNA no banco enquanto lotes de novas SRSs são armazenadas no banco. Ao longo do experimento foi medido o tempo médio de consulta das sequências de DNA. Detalhes sobre este experimento são apresentados na Seção 4.3.4.

O ambiente de teste foi criado em um Notebook com 8 GB de memória RAM, processador intel i7 e 120 GB de SSD. É importante lembrar que o *Solid State Drive* (SSD) possui leitura e escrita mais rápida do que os HDs convencionais, chegando a ser até 100 vezes mais rápidos [26], resultando em melhores tempos em todos os experimentos quando comparados com um HD convencional. A velocidade da rede local do experimento é de 100Mbps.

Os bancos de dados MySQL e MongoDB foram instalados em máquinas virtuais com 4 GB de memória RAM e 35 GB de espaço em disco. Para o Cassandra foram utilizadas

duas máquinas virtuais com 2GB de memória RAM e 35 GB de espaço em disco para criar o *cluster*. A escolha de criar um *cluster* no Cassandra era avaliar a escalabilidade horizontal, característica forte nos modelos Não Relacionais. Todas as máquinas virtuais utilizavam o sistema operacional Ubuntu 14.04 - LTS e tinham o mesmo processador do Notebook. A aplicação cliente, FastaBD, era executada no próprio Notebook.

4.2.1 Descrição dos Arquivos Fasta Utilizados

Os arquivos *fasta* foram obtidos junto ao Laboratório de Imunopatologia Keizo Asami (LIKA) [1] e foram obtidos a partir da filtragem de arquivos *csfasta* gerados no sequenciamento do material genético de caprinos. No total, foram utilizados quatro arquivos *fasta*, com tamanhos variados, chegando ao total de 2.3 *gigabytes* e 69.848.142 linhas. A escolha desse conjunto de arquivos foi feita com o objetivo de analisar o comportamento dos três bancos de dados com arquivos de tamanhos variados. A tabela 4.1 mostra as informações detalhadas de cada arquivo *fasta*.

Arquivo	**Origem da Amostra**	**Tamanho em MB**	**Número de SRS**
cabra4.fa	próstata	763	11.756.047
cabra5.fa	glândula mamária	238	3.802.481
cabra6.fa	ovário	31	473.886
cabra7.fa	hipófise	1331,2	18.891.657
-	-	total aproximadamente 2.3 GB	34.924.071

Tabela 4.1: Detalhes dos arquivos *fasta* utilizados neste estudo. (Fonte: o autor)

Cada cadeia de DNA nos arquivos *fasta* deste estudo possui em torno de 50 caracteres que podem ser: Adenina (A), Guanina(G), Citosina (C), Timina(T) ou Desconhecido (N). Os identificadores de sequência não podem ser repetidos no mesmo arquivo, mas podem existir identificadores iguais em arquivos diferentes. No final da inserção dos arquivos com o número de SRS por linha de tabela igual 1, por exemplo, eram inseridos 34.924.071 registros no banco de dados. Para saber a quantidade de registros inseridas no banco de dados é realizado o seguinte cálculo:

total de SRSs/quantidade de SRSs por registro

[1] https://www.ufpe.br/lika/

4.2.2 Método dos Experimentos

Foram realizados um total de dois experimentos. O primeiro foi dividido em três etapas: inserção, extração e consulta de 5 identificadores de sequência. A figura 4.1 ilustra o fluxograma do experimento 1.

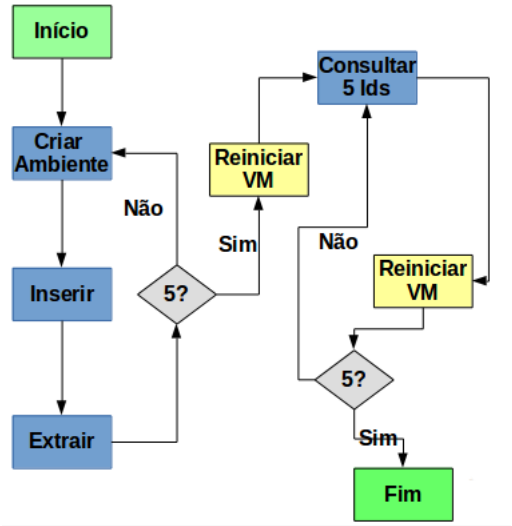

Figura 4.1: Fluxograma do experimento 1. (Fonte: o autor)

De acordo com o fluxograma do experimento 1, eram realizadas cinco amostras de cada etapa do experimento. A criação do ambiente, a inserção e extração eram realizadas seguindo um único fluxo, sem interrupções. Essas três etapas eram realizadas automaticamente através do uso da ferramenta FastaBD. Ao final da quinta repetição, as máquinas virtuais eram reiniciadas, limpando o *cache* dos bancos de dados. Depois, eram realizadas cinco consultas de identificadores de sequência. No final das consultas, as máquinas virtuais também eram reiniciadas devido ao *cache* dos bancos de dados. As consultas também eram realizadas pela ferramenta FastaBD. Importante lembrar que ao final da inserção e extração de cada arquivo, o tempo de realização dessas operações eram registrados, assim como o tempo de consulta de cada identificador de sequência. As consultas foram realizadas apenas quando o número de SRS por linha da tabela era apenas igual a 1.

No segundo experimento, chamado de Curva de Consulta, eram realizadas inserções de valores fixos e ao final de cada inserção eram realizadas consultas de cinco identificadores de sequências diferentes, onde o tempo de consulta de cada um deles era registrado. A figura 4.2 ilustra o fluxograma do experimento 2.

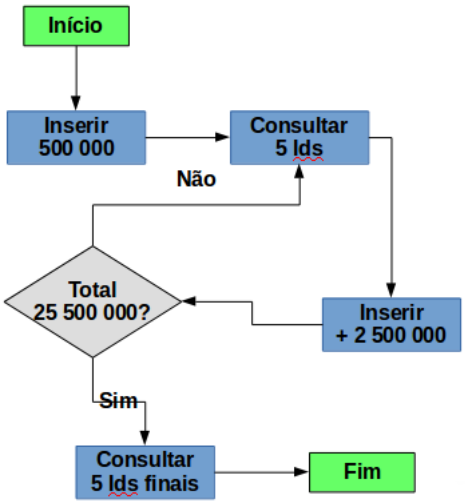

Figura 4.2: Fluxograma do experimento 2. (Fonte: o autor)

O fluxograma acima mostra que ao final da consulta de cinco identificadores de sequências diferentes, era armazenadas mais um arquivo com 2 500 000 SRSs, até quando o total de SRSs inseridas no banco de dados fosse igual a 25 500 000. Após atingir esse valor, eram consultados mais 5 identificadores de sequência diferentes. Para cada ciclo, novos identificadores de sequências eram consultados, assim não precisou reiniciar as máquinas virtuais, como no caso do experimento 1. Da mesma forma que a etapa de consulta do experimento 1, a curva de consulta foi realizada apenas quando o número de SRS por linha da tabela era igual a 1.

4.3 Resultados

Os resultados obtidos estão divididos em subseções, onde cada subseção compreende uma das etapas do experimento. No apêndice deste trabalho são encontradas todas as estatísticas obtidas nos experimentos. Os tempos encontrados nos gráficos estão em segundos e os tamanhos dos arquivos foram mensurados na ordem de *megabytes*. O intervalo de confiança para a média, considerando 95% de confiança, foram calculados e, na maioria dos casos, os intervalos não se sobrepõe, mostrando que os valores das médias são estatisticamente diferentes. Por isso, os intervalos de confiança não serão mostrados nos gráficos, mas os casos em que não for possível definir a diferença estatística serão pontuados no texto. Todos os valores foram aproximados até a segunda casa decimal.

4.3.1 Análise da Inserção

A Figura 4.3 mostra o gráfico da média do tempo de inserção do arquivo cabra4.fa em cada um dos bancos de dados com a variação do número de SRSs por linha. Como os resultados ficaram proporcionalmente parecidos nos demais arquivos, são apresentados apenas os valores considerando o arquivo cabra4.fa. Os resultados para os demais arquivos apresentaram comportamento parecido e por isso são apresentados apenas no apêndice deste trabalho.

Pode-se observar que o MySQL possui o mais baixo tempo de inserção tanto quando a SRS é armazenada sozinha como também em conjuntos de 5. Para o caso do conjunto com 50 SRSs quando comparando os bancos de dados MySQL e MongoDB observa-se que os tempos não são estatisticamente diferentes entre si (considerando 95% de confiança nos intervalos de confiança para a média), contudo são estatisticamente menores do que os do Cassandra, que é o banco de dados que possui o tempo de inserção mais elevado em todas as situações. Neste mesmo caso, conjunto de SRSs é igual a 50, o MongoDB diminui drasticamente a diferença do tempo para o MySQL. Nos dois maiores arquivos, o MongoDB chega a ter uma pequena vantagem em relação ao MySQL, concluindo que para essa situação, o MongoDB e o MySQL, estatisticamente, possuem o mesmo tempo de inserção e não apresentam vantagem entre eles. Logo, podemos deduzir que, caso estas reduções se mantenham para arquivos maiores que os utilizados no trabalho e com o conjunto de SRS maior que 50, o MongoDB se apresenta bons resultados no quesito inserção.

Foi observado também que a proporção do tempo médio de inserção entre o banco de dados relacional e os bancos de dados não-relacionais são independentes do tamanho do arquivo, conforme mostram as três tabelas a seguir (4.2, 4.3 e 4.4) que apresentam esta proporção para os diferentes tamanhos do conjunto de SRSs escolhidos.

Na tabela 4.2 pode-se observar que os tempos médios de inserções do Cassandra chegam a ser mais que o dobro do tempo do MySQL. Enquanto o MongoDB apresenta valores mais baixos que o Cassandra. Isto mostra que o tamanho do arquivo não influencia o valor da proporção.

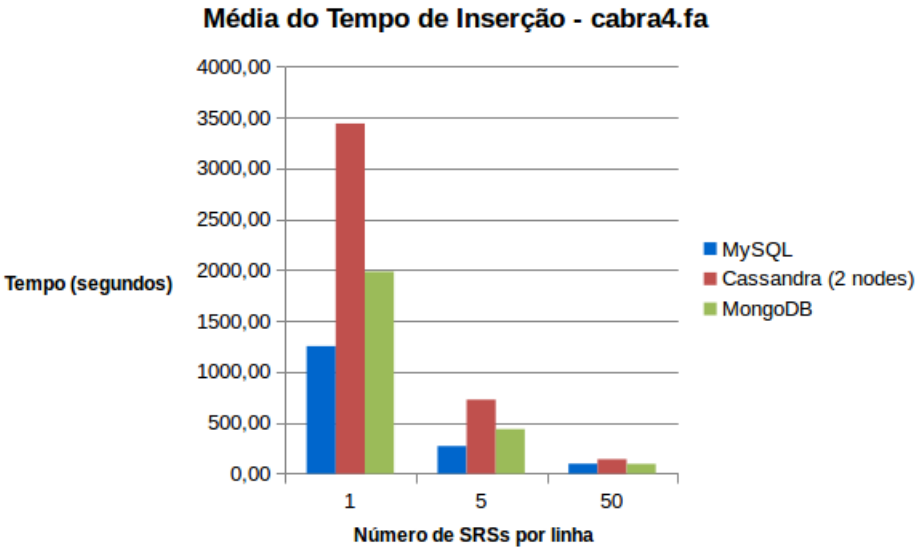

Figura 4.3: Média de Tempo de Inserção - Cabra4.fa. (Fonte: o autor)

	MYSQL			
	SRS 1			
	cabra4.fa	cabra5.fa	cabra6.fa	cabra7.fa
Cassandra	2,75	2,80	2,86	2,79
MongoDB	1,58	1,62	1,63	1,62

Tabela 4.2: Propoção do tempo médio de inserção entre o MySQL e os bancos de dados Não Relacionais para SRS 1. (Fonte: o autor)

Nas Tabelas 4.3 e 4.4, novamente não há maiores variações nos valores observados nos dois bancos. É importante observar a queda do valor da proporção na comparação dos bancos com o MySQL quando a SRS é 50, enquanto para as SRS 1 e 5 os valores permanceram próximos (em torno de 1,60 para o MongoDB e 2,75 para o Cassandra). Este dado reforça a observação de que conjuntos maiores de SRSs devem levar a maiores reduções no tempo de inserção dos bancos de dados não-relacionais quando comparados aos banco de dados relacional.

	MYSQL			
	SRS 5			
	cabra4.fa	cabra5.fa	cabra6.fa	cabra7.fa
Cassandra	2,68	2,68	2,77	2,71
MongoDB	1,62	1,64	1,62	1,62

Tabela 4.3: Propoção do tempo médio de inserção entre o MySQL e os bancos de dados Não Relacionais para SRS 5. (Fonte: o autor)

	MYSQL			
	SRS 50			
	cabra4.fa	cabra5.fa	cabra6.fa	cabra7.fa
Cassandra	1,45	1,49	1,49	1,43
MongoDB	0,98	1,00	1,01	0,97

Tabela 4.4: Propoção do tempo médio de inserção entre o MySQL e os bancos de dados Não Relacionais para SRS 50. (Fonte: o autor)

Independente do arquivo que será armazenado, a diferença do tempo de inserção entre o MySQL e os bancos de dados Não Relacionais se manteve na mesma proporção. A tabela 4.5 mostra esses valores. Os valores encontrados são o resultado da diferença entre a maior proporção e a menor proporção variando o número de SRS. Quanto mais esses valores estiverem próximos de zero mais estável é a proporção do tempo de inserção entre o MySQL e os bancos de dados Não Relacionais para diferentes tamanhos de arquivos.

	Diferença da maior pela menor proporção		
	MYSQL		
	SRS 1	SRS 5	SRS 50
Cassandra	0,11	-0,09	0,07
MongoDB	0,05	0,02	0,03

Tabela 4.5: Variação do tempo médio de inserção entre o MySQL e os bancos de dados Não Relacionais variando o número de SRS. (Fonte: o autor)

A Figura 4.4 ilustra o crescimento do tempo de inserção pelo tamanho dos arquivos com SRS única. Observa-se que o crescimento é linear. O gráfico mostra que a linha do MySQL é a que mais se aproxima do eixo X, ou seja, apresenta os melhores tempos de inserção, diferente do Cassandra que possui o maior tempo de inserção. A partir da regressão linear sobre os dados podemos observar que há uma tendência de aproximação entre os valores do tempo de inserção. Através dos valores do coeficiente ângular da reta, é possível ver o quanto essas retas ficam próximas uma das outras com o aumento do número de SRS por linha da tabela. Os valores dos coeficientes ângular calculados para o Cassandra, MySQL e o MongoDB no caso da SRS 1 foram 4,14, 1,48 e 2,40, respectivamente. Quando realizado o experimento para SRS igual a 5 esses valores diminuiram para 0,87, 0,32 e 0,52, na mesma ordem anterior. Enquanto para a SRS 50 estes coeficientes foram 0,17, 0,11 e 0,11. Mostrando que a diferença diminui drasticamente. Se projetarmos essas retas para valores maiores, nessa situação, podemos concluir que os bancos de dados Não Relacionais podem alcançar tempos de inserção próximos ao tempo de inserção do MySQL. Os dois gráficos ilustrados nas figuras 4.4 e 4.5 mostram essa aproximação no caso da SRS igual a 1 e a 50 respectivamente. No gráfico 4.5, os valores do eixo y são bem menores quando comparados aos valores do gráfico 4.4.

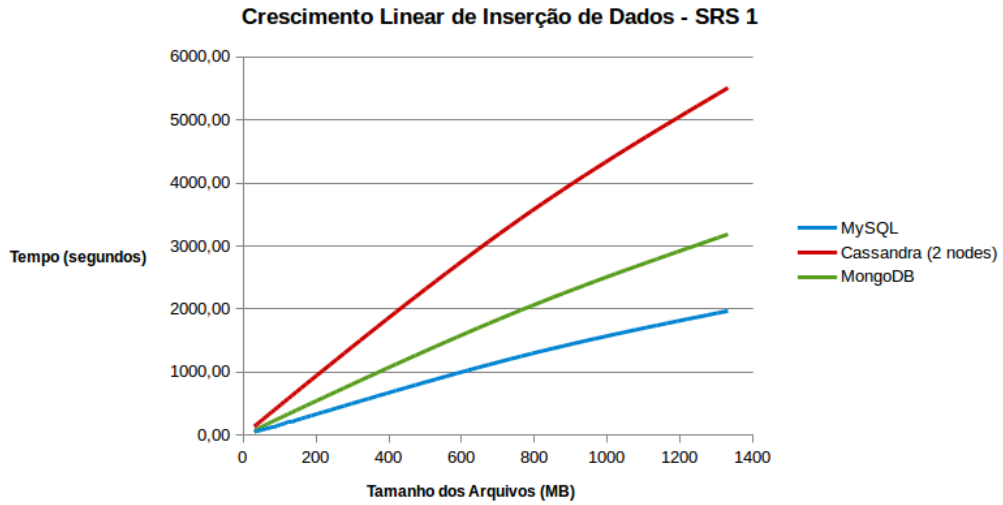

Figura 4.4: Crescimento de inserção de dados - SRS 1. (Fonte: o autor)

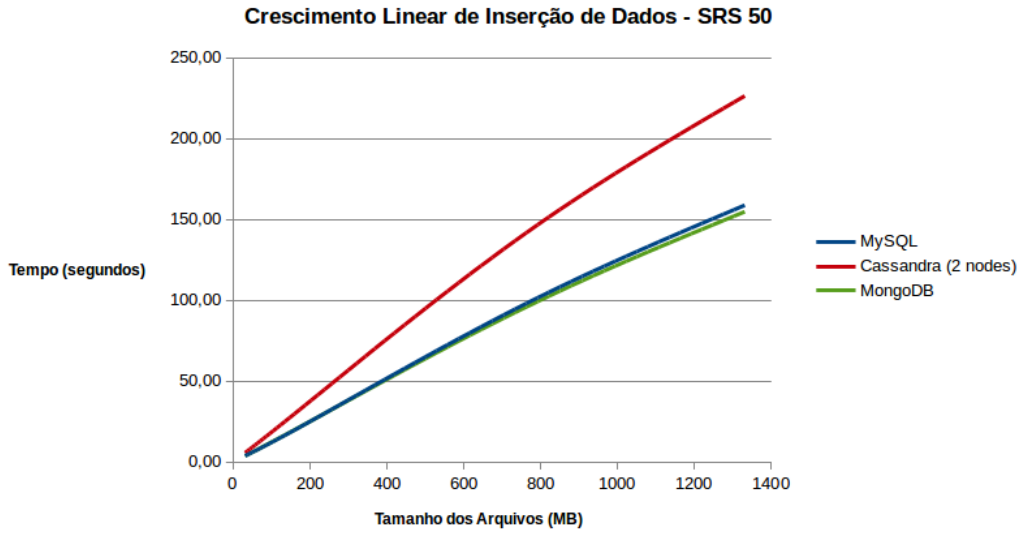

Figura 4.5: Crescimento de inserção de dados - SRS 50. (Fonte: o autor)

4.3.2 Análise da Extração

As figuras 4.6, 4.7, 4.8 e 4.9 mostram os gráficos da média do tempo de extração dos quatro arquivos em cada um dos bancos de dados com a variação do tamanho do conjunto de SRSs por linha da tabela.

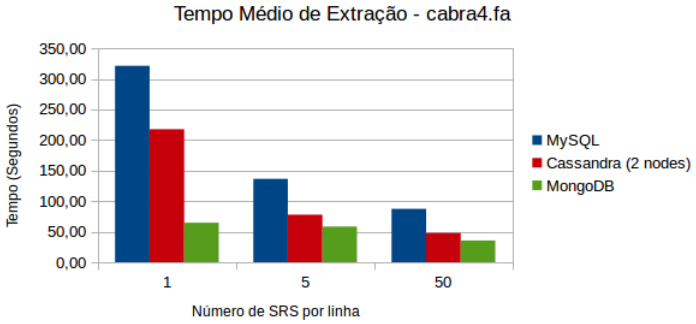

Figura 4.6: Média de Tempo de Extração - Cabra4.fa. (Fonte: o autor)

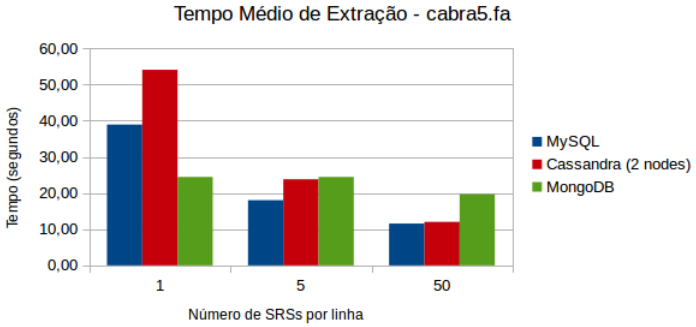

Figura 4.7: Média de Tempo de Extração - Cabra5.fa. (Fonte: o autor)

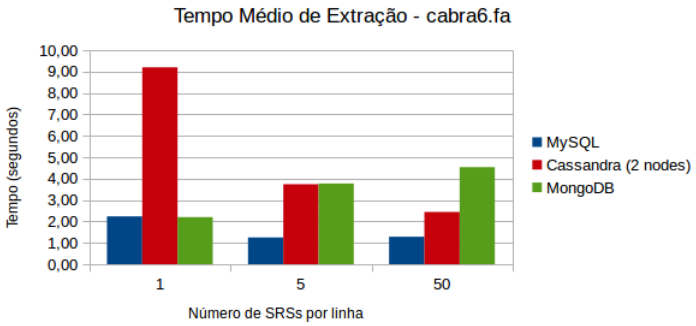

Figura 4.8: Média de Tempo de Extração - Cabra6.fa. (Fonte: o autor)

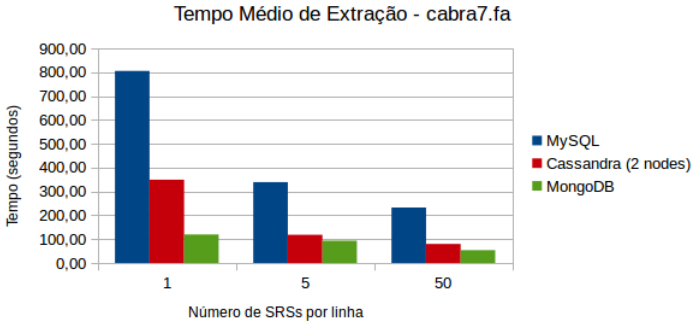

Figura 4.9: Média de Tempo de Extração - Cabra7.fa. (Fonte: o autor)

O MongoDB possui o melhor tempo na maioria das situações, excetuando nos arquivos cabra5.fa e cabra6.fa, que são os dois menores. No cabra5.fa, o MongoDB tem melhor tempo de extração apenas quando o número de SRSs por linha é igual a 1, porém quando esse número é igual a 5 o tempo médio de extração do Cassandra não é estatisticamente diferente ao do MongoDB, enquanto o MySQL tem o melhor tempo de todos. Quando o número de SRSs por linha é igual a 50, o MongoDB apresenta o pior tempo, enquanto os tempos do Cassandra e do MySQL estão estatisticamente iguais.

Já no cabra6.fa, temos um resultado diferente, onde o MySQL tem o melhor tempo médio de extração quando o número de SRSs é igual a 5 e a 50 e ficando estatisticamente igual ao MongoDB quando este número é igual a 1. O MongoDB apresenta um crescimento no tempo médio de extração independente do número de SRSs por linha, chegando a ficar estatisticamente igual ao do Cassandra quando o número de SRSs é igual a 5.

Nos dois maiores arquivos cabra4.fa e cabra7.fa, o MongoDB obteve os melhores tempos independente do número de SRSs. Enquanto o MySQL teve o pior tempo médio

de extração. O pior tempo de extração do MySQL é justificado pelo fato das consultas serem paginadas para grandes números de registros, ou seja, quando o número de registros de um arquivo específico inseridos no banco de dados era maior que 500 000, as consultas eram divididas em blocos de 500 000, valor escolhido baseado no número de linhas do arquivo cabra6.fa. As consultas foram realizadas dessa forma, pois o MySQL consome muitos recursos computacionais para realizar essas consultas e retornar os resultados. Tentativas de retornar o conteúdo de um grande número de registros de uma única vez não puderam ser realizadas devido à limitações do ambiente de teste. A análise do consumo de memória RAM e CPU na extração não-paginada constatou que a CPU atingia o limite máximo de uso e a memória RAM atingia aproximadamente 50% de uso, tornando impossível fazer a extração de uma vez só independente do número de registros inseridos.

As Figuras 4.10, 4.11 e 4.12 mostram que a linha de crescimento do tempo médio de extração do MySQL tende a se afastar das linhas dos bancos de dados Não Relacionais a cada vez que a quantidade de *megabytes* inseridos aumenta. Enquanto o Cassandra e o MongoDB permanecem com valores próximos. O tempo médio de extração do MySQL é maior quanto menor for o número de SRSs, consequência do menor número de registros inseridos no banco de dados e da paginação das consultas. É observado também que na extração de pequenos arquivos, o MySQL apresenta os melhores tempos, mas quando a quantidade de *megabytes* fica em torno de 300 acontece a inversão dos tempos, ou seja, os bancos de dados Não Relacionais apresentam os melhores tempos quando a quantidade de dados aumenta.

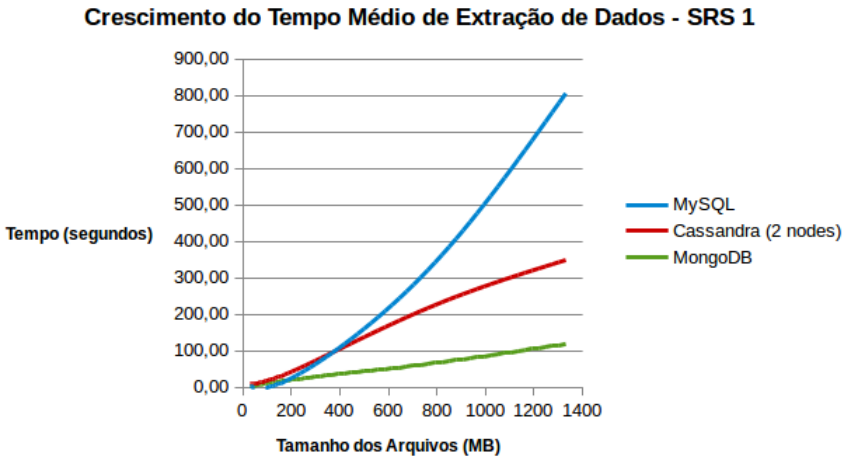

Figura 4.10: Crescimento do tempo médio de extração de dados SRS 1. (Fonte: o autor)

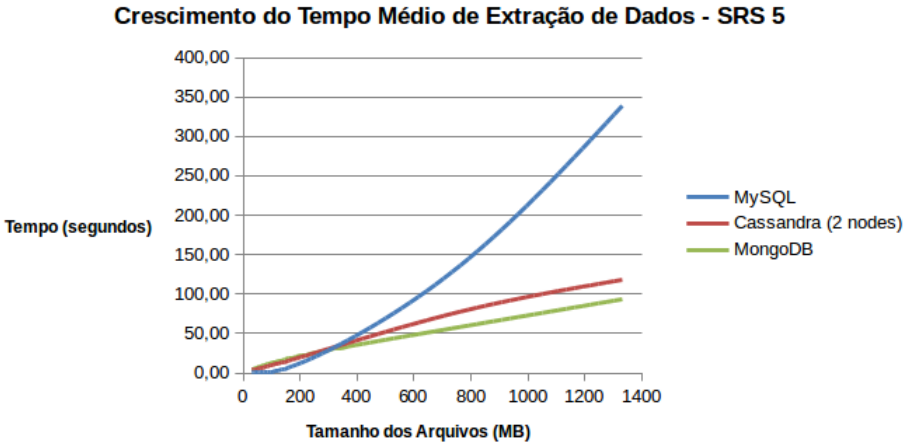

Figura 4.11: Crescimento do tempo médio de extração de dados SRS 5. (Fonte: o autor)

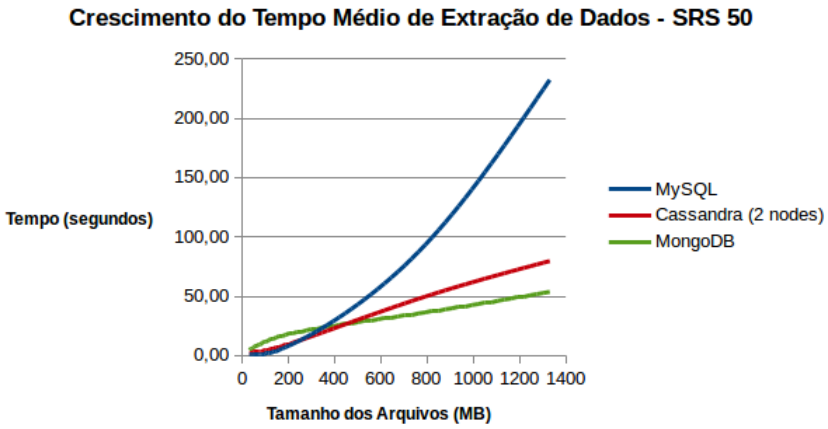

Figura 4.12: Crescimento do tempo médio de extração de dados SRS 50. (Fonte: o autor)

4.3.3 Análise da Consulta

As consultas foram realizadas com 5 identificadores de sequências escolhidos aleatoriamente, porém esses mesmos identificadores eram novamente consultados na próxima amostra. Para evitar que o tempo das consultas não fosse influenciado pelo *cache* existente em alguns bancos de dados, toda vez que era realizado uma amostra as máquinas virtuais eram reiniciadas e em seguida era novamente realizado o experimento. Lembrando que esse experimento foi realizado somente quando era inserido 1 SRS por linha da tabela. A tabela 4.6 mostra os valores estatísticos do experimento.

-	**MySQL**	**Cassandra**	**MongoDB**
Média	28,98	0,3	28,72
Mediana	29,13	0,28	28,68
Desvio Padrão	1,47	0,11	0,13
Mínimo	25,53	0,22	28,59
Máximo	31,69	0,71	29,18

Tabela 4.6: Valores estatísticos da Análise da Consulta. (Fonte: o autor)

A partir da tabela observa-se que o Cassandra possui o menor tempo de consulta em relação ao MySQL e ao MongoDB. Esta observação foi confirmada por meio de testes t de *Student unicaudal*, com 95% de confiança, para cada par de bancos de dados. As consultas realizadas no Cassandra chegam a ser quase 100 vezes mais rápidas do que os demais bancos de dados. O MongoDB e o MySQL são estatisticamente equivalentes. A diferença entre os valores máximo e mínimo no MySQL é a maior entre os três bancos de dados e também tem o maior desvio padrão, isso quer dizer que, os valores estão mais distantes da média. O MongoDB apresenta mais estabilidade nos valores, uma vez que, a diferença entre os valores máximo e mínimo é próxima de 0. Esta estabilidade também é encontrada no Cassandra, tornando-se o banco de dados com os menores tempos médios de consulta e variabilidade. Para confirmar a diferença entre as variâncias dos bancos de dados, utilizamos o teste F para duas amostra com 95% de confiança. E os resultados mostraram que a variância do MySQL é estatisticamente diferente dos demais bancos de dados.

4.3.4 Análise da Curva de Consulta

O experimento da curva de consulta tem como objetivo observar o tempo de consulta médio após inserir quantidades determinadas de registros no banco de dados. Para realizar o experimento, arquivos *fasta* anteriores foram particionados em novos arquivos *fasta*, com número de linhas bem definido. Baseado no segundo fluxograma ilustrado na figura 4.2, o experimento consiste em realizar consultas de cinco identificadores de sequência ao final de cada inserção. Novos identificadores de sequência eram consultados ao final de cada inserção, ou seja, eles não eram repetidos. Este experimento foi realizado com o número de SRS igual a 1.

O gráfico na Figura 4.13 mostra o tempo médio de consulta pela quantidade de registros inseridos em cada banco de dados.

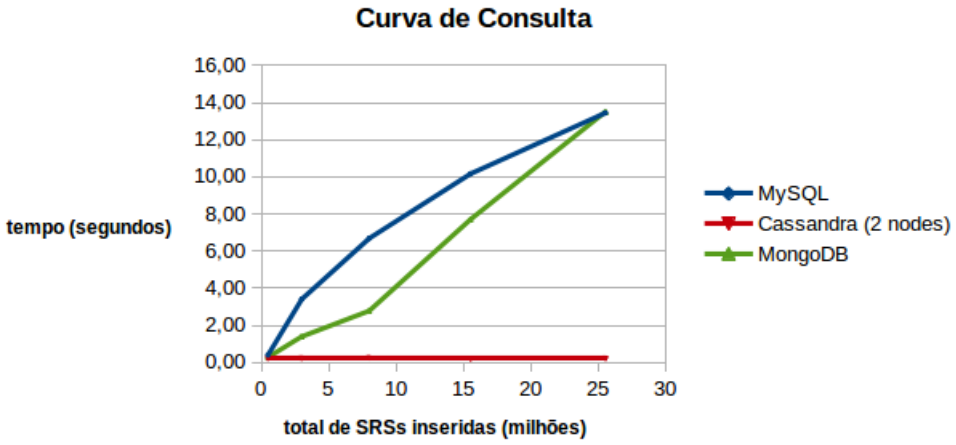

Figura 4.13: Curva de Consulta. (Fonte: o autor)

É possível observar que o Cassandra é o banco de dados com o melhor desempenho do experimento na maioria das situações, excetuando apenas quando são realizadas as consultas em um universo com 500 000 SRSs. Nesse caso, os desempenhos dos três bancos de dados não são diferentes estatisticamente. Porém a partir do momento que são inseridos novos registros, o MySQL e o MongoDB apresentam um crescimento linear no tempo médio de consulta, já o Cassandra não apresenta muita variação. O MySQL tem um crescimento constante no tempo médio de consulta. O MongoDB é o que tem o segundo melhor desempenho no experimento, mas apresenta um maior crescimento após novas inserções, chegando a ficar estatisticamente igual ao MySQL quando inseridos os últimos 10 000 000 de SRSs no banco de dados.

4.4 Discussão

-	Inserção	Extração	Consulta
MySQL	★★★	★	★
MongoDB	★★★	★★★	★
Cassandra	★	★★	★★★

Figura 4.14: Avaliação de desempenho dos três bancos de dados. (Fonte: o autor)

Visualizando a figura 4.14, podemos concluir que na inserção o MySQL e o MongoDB possuem os melhores tempos, já o Cassandra possui o pior. No quesito extração, os bancos de dados Não Relacionais possuem melhores tempos do que o MySQL, entre os bancos de dados Não Relacionais, o MongoDB tem o melhor tempo de extração. Por último, o Cassandra tem o melhor tempo de consulta entre os bancos de dados avaliados, enquanto o MySQL e o MongoDB possuem valores estatisticamente iguais.

Diante dos resultados obtidos podemos concluir que a estratégia de armazenar múltiplas SRSs por linha da tabela, em todos os bancos de dados, permite obter melhor desempenho quando comparada com a estratégia de uma única SRS por linha da tabela. Deve-se observar contudo, que o uso desta estratégia depende da necessidade do usuário do banco de dados genômicos. Já que, ao se utilizar múltiplos SRSs, perder-se-á funcionalidades importantes, como a consulta específica e a possibilidade de filtrar cadeias de DNA por meio das linguagens de consulta disponíveis nos bancos de dados. Assim, o pipeline de processamento e tratamento dos dados é que vai definir qual a melhor abordagem para atender todos os requisitos.

O MySQL não apresentou bons resultados em atender a grande demanda de dados. Na inserção, apesar de aparecer com bons resultados em algumas situações, o MySQL não apresenta boa perspectiva se projetarmos a possibilidade de utilizar mais dados genômicos, enquanto que os bancos de dados Não Relacionais possuem as melhores perspectivas para maiores massas de dados, visto que as curvas de crescimento do tempo foram menores se comparado ao do MySQL. Na extração, o MySQL apresenta o melhor resultado quando era extraída poucas linhas do banco de dados. Porém, quando extraída grandes resultados, o MySQL deixou a desejar, pois era necessário muito recurso computacional para extrair o

conteúdo de uma única vez, ocasionando a paginação das consultas. Já os bancos de dados NoSQL apresentaram bons resultados e não precisando de muito recurso computacional para extrair grandes massas de dados.

O Cassandra teve um ótimo desempenho no experimento da consulta, chegando a ser 100 vezes melhor que os outros dois bancos de dados. O MySQL e o MongoDB tiveram comportamento parecido, mas o banco de dados Relacional apresentou maior instabilidade nesse experimento, apresentando um maior desvio padrão. Na curva de consulta, percebemos que o crescimento do tempo de consulta quando maior era maior o número de registros inseridos no Cassandra foi mínimo, no MySQL tivemos um crescimento constante e o MongoDB apresentou um crescimento muito maior que os demais quando o número de registros era maior que 5 milhões.

Analisando o resultado de outros trabalhos, percebemos que o comportamento dos bancos de dados relacionais e não-relacionais foram diferentes no quesito de inserção e extração. Em [6], o banco de dados relacional teve um tempo de inserção pior que o Cassandra, enquanto o tempo de extração foi melhor. Apesar de utilizarem outras estratégias e arquivos diferentes, os resultados foram bem diferentes comparado ao deste trabalho. Enquanto em [6] utilizou a estratégia de armazenar os arquivos em várias colunas da tabela, onde cada tabela tinha o conteúdo de somente um arquivo, neste trabalho foram utilizadas estratégias de variar o tamanho do conteúdo a ser inserido por cada linha da tabela, com poucas colunas. Aqui também foi abordada a estratégia de armazenar o conteúdo dos arquivo em tabelas diferentes (uma tabela para cada arquivo) como também foram armazenados todos os conteúdos em uma única tabela, neste último caso foi utilizado o banco de dados MySQL enquanto a no primeiro caso foram utilizados dois bancos de dados Não Relacional.

Capítulo 5

Conclusão e Trabalhos Futuros

O rápido crescimento dos arquivos genômicos e a modernização dos equipamentos que geram esses dados possibilitaram o surgimento novos projetos genoma. Surgindo também a necessidade de encontar a melhor estratégia e tecnologia de armazenamento para gerenciar esses dados. O problema de armazenar os dados genômicos ainda não foi resolvido. Porém, com o desenvolvimento de novos modelos de bancos de dados, criados com o objetivo de atender a grande demanda de dados, esse problema está cada vez mais perto de ser solucionado.

Com o objetivo de encontrar as melhores estratégias e o melhor banco de dados entre os três utilizados, neste trabalho foi possível perceber que a estratégia de armazenar várias SRSs por linha da tabela, torna-se viável em todos os três bancos de dados. Os bancos de dados Não Relacionais apresentaram grande melhoria no tempo de inserção, mesmo que em algumas vezes esse tempo ficou pior do que o do MySQL, foi possivel perceber que para projeções futuras, o Cassandra e o MongoDB provavelmente apresentará bons resultados na inserção. Na extração, o banco de dados relacional deixa a desejar, visto que ele necessita de muito recurso para extrair todo conteúdo. Enquanto os NoSQLs tiveram ótimos tempos e não precisaram consumir muito recurso computacional. Já na realização de consultas, fica claro o uso do Cassandra, pois o tempo de consulta chega a ser 100 vezes menor e não apresenta grandes variações na medida que o número de registros aumenta quando comparado aos demais bancos de dados, que apresentam um crescimento linear no tempo de consulta.

Neste trabalho, encontramos novas formas de armazenar os dados genômicos utilizando três bancos de dados diferentes. Dois deles pertencem a um paradigma recente, que tem novas estratégias de gerenciamento dos dados e uma escalabilidade diferente, proporcionando a inserção de novas máquinas à medida que a quantidade de dados armazenados aumenta. A utilização dos bancos de dados Não Relacionais, pode ser uma boa alternativa para o armazenamento de dados genômicos, visto que, eles apresentam bons comportamentos quando inseridos em um ambiente com muitos dados e que está em crescimento. A utilização do banco de dados relacional neste trabalho trouxe a conclusão que eles não apresentam bons resultados quando estão inseridos neste mesmo ambiente.

Com as estratégias de armazenamento criadas neste trabalho, podemos concluir que a sua utilização em bancos de dados Não Relacionais traz melhorias nos tempos de inserção e extração dos dados genômicos. Utilizando os recursos que esses bancos de dados nos oferecem, como a modelagem livre de esquema e a escalabilidade horizontal, temos capacidade de implementar grandes sistemas de armazenamentos de dados genômicos com eficiência nas operações básicas de um banco de dados: inserção, extração e consulta.

5.1 Dificuldades Encontradas

No começo, os experimentos foram realizados no laboratório de ensino da UFRPE. Cada computador realizava seu papel separadamente, alguns hospedavam as máquinas virtuais e outros executavam a ferramenta FastaBD. Porém, a dificuldade do acesso aos laboratórios (era necessário que um professor estivesse presente) junto com alguns problemas de implementação encontrados na ferramenta no início dos experimentos, fizeram com que não fosse possível completar todos os experimentos planejados. Por isso, foi necessário refazê-los todos no Notebook.

O próprio ambiente de teste no Notebook limitou os experimentos possíveis, já que os experimentos foram executados em máquinas virtuais com poucos recursos computacionais, sendo que algumas tinham apenas 2GB de memória RAM, estando longe das características computacionais dos servidores atuais. Além disso, o uso de máquinas virtuais nos experimentos tem outras desvantagens, posto que elas dependem do hardware da máquina que as hospeda, o qual compartilham com as demais máquinas virtuais e com a ferramenta de benchmark FastaBD. Para experimentos mais adequados, faz-se necessário o uso de compu-

tadores mais robustos e a instalação dos bancos de dados diretamente no servidor, evitando o uso de máquina virtuais.

5.2 Trabalhos Futuros

Como trabalhos futuros, pretende-se avaliar o impacto que as demais estratégias de armazenamento sugeridas no contexto deste trabalho têm sobre o desempenho das operações de inserção, extração e consulta em bancos de dados NoSQL. Assim como explorar ainda mais o aumento do número da SRS por linha da tabela, visto que tivemos bons resultados nos quesitos de inserção e extração. Como muitos desses bancos de dados possuem escalabilidade horizontal, pretende-se ainda analisar o impacto do uso de mais nós no *cluster* do Cassandra, assim como a criação de um *cluster* para o MongoDB.

Com o desenvolvimento da ferramenta FastaBD, temos a possibilidade de avaliar outros bancos de dados NoSQL, inclusive de modelos diferentes. Então, planeja-se avaliar outros bancos de dados NoSQL com o objetivo de estudar o seu comportamento e verificar a viabilidade do uso destes para armazenar dados genômicos.

Referências Bibliográficas

[1] 10 motivos para escolher o mysql para aplicativos web. Technical report, Oracle, 2011.

[2] Dez principais motivos para escolher o mysql para a próxima geração de aplicações web. Technical report, Oracle, 2014.

[3] P Malik A Lakshman. Cassandra: a decentralized structured storage system. *ACM SIGOPS Operating Systems Review*, 2010.

[4] Istvan Albert. Transforming and manipulating color space reads. Disponível em =https://www.biostars.org/p/43855/. Acessado em 28/05/2015.

[5] Rodrigo Aniceto, Rene Xavier, Maristela Holanda, Maria Emilia Walter, and Sergio Lifschitz. Genomic data persistency on a NoSQL database system. In *Bioinformatics and Biomedicine (BIBM), 2014 IEEE International Conference on*, pages 8–14. IEEE, 2014.

[6] Rodrigo Cardoso Aniceto and Renê Freire Xavier. *Um estudo sobre a utilização do banco de dados NoSQL cassandra em dados biológicos*. Monograph, 2014.

[7] Anderson Chaves Carniel$ˆ1$, Aried de Aguiar Sá, Vinícius Henrique Porto Brisighello, Marcela Xavier, Renato Bueno Ribeiro, Ricardo Rodrigues Ciferri, and Cristina Dutra de Aguiar Ciferri. Query processing over data warehouse using relational databases and nosql. 2012.

[8] Ann Kelly Dan McCreary. *Making Sense of NoSQL: A guide for managers and the rest of us*. Manning Publications Company, 2013.

[9] C.J. Date. *Introdução a sistemas de bancos de dados*. Campus, 2004.

[10] Mauricio De Diana and Marco Aurélio Gerosa. Nosql na web 2.0: Um estudo comparativo de bancos não-relacionais para armazenamento de dados na web 2.0. 2010.

[11] Ramez Elmasri, Shamkant B Navathe, Marília Guimarães Pinheiro, and Luis Ricardo de Figueiredo. *Sistemas de banco de dados*. Pearson Addison Wesley, São Paulo, 2008.

[12] DataStax Enterprise. Internode communications (gossip) — datastax cassandra 2.0 documentation. Disponível em =http://docs.datastax.com/en/cassandra/2.0/cassandra/architecture/architectureGossipAbout_c.html. Acessado em 15/06/2015.

[13] Adail Tiago Lima Faleiro. Modelo entidade e relacionamento. Disponível em =http://www.devmedia.com.br/modelo-entidade-relacionamento-mer/19821. Acessado em 23/05/2015.

[14] Ruben Cruz Huacarpuma. Modelo de dados para um Pipeline de seqüenciamento de alto desempenho transcritômico. Master's thesis, Universidade de Brasília, 2012.

[15] Mongo Inc. Mongodb 3.0 manual. Disponível em =http://docs.mongodb.org/manual/. Acessado em 20/05/2015.

[16] Mongo Inc. Our customers — mongodb. Disponível em =https://www.mongodb.com/who-uses-mongodb. Acessado em 13/06/2015.

[17] Bernadette Farias Lóscio, Hélio Rodrigues de OLIVEIRA, and Jonas César de Sousa PONTES. NoSQL no desenvolvimento de aplicações Web colaborativas. *VIII SIMPÓSIO BRASILEIRO DE SISTEMAS COLABORATIVOS, Paraty, RJ: SBC*, 2011.

[18] Pramodkumar J. Sadalage Martin J. Fowler. *Nosql distilled a brief guide to the emerging world of polyglot persistence*. Addison-Wesley Professional, 2012.

[19] NCBI. Genbank release notes. Disponível em =http://www.ncbi.nlm.nih.gov/genbank/statistics/. Acessado em 23/07/2015.

[20] NIH. Dna sequecing cost. Disponível em =http://www.genome.gov/sequencingcosts/. Acessado em 23/07/2015.

[21] Joel Rodrigues. Modelo entidade e relacionamento e diagrama entidade e relacionamento. Disponível em =http://www.devmedia.com.br/modelo-entidade-relacionamento-mer-e-diagrama-entidade-relacionamento-der/14332. Acessado em 20/05/2015.

[22] Uwe Röhm and José A. Blakeley. Data Management for High-Throughput Genomics. In *CIDR*, 2009.

[23] Andre Rodrigo Sanches. Banco de dados. Disponível em =http://www.ime.usp.br/ andrers/aulas/bd2005-1/aula3.html. Acessado em 12/05/2015.

[24] Baron Schwartz, Peter Zaitsev, and Vadim Tkachenko. *High performance MySQL*. O'Reilly, Beijing ; Cambridge [Mass.], 3rd ed edition, 2012.

[25] Bruno Eduardo Soares and Clodis Boscarioli. Modelo de Banco de Dados Colunar: Características, Aplicações e Exemplos de Sistemas. *Escola Regional de BDs, Camobiu. IX ERBD*, 2013.

[26] OCZ Storage Solutions. Ssd vs hdd why solid state drives are better hard drives. Disponível em =http://ocz.com/consumer/ssd-guide/ssd-vs-hdd. Acessado em 02/07/2015.

[27] Illumina Genome Analyzer Specification. Ga specsheet. Disponível em =http://www.ucl.ac.uk/cancer/supportservices/SupportDocs/GA_SpecSheet.pdf. Acessado em 03/07/2015.

[28] Christof Strauch, Ultra-Large Scale Sites, and Walter Kriha. NoSQL databases. *Lecture Notes, Stuttgart Media University*, 2011.

[29] TecMundo. O que é a lei de moore. Disponível em =http://www.tecmundo.com.br/curiosidade/701-o-que-e-a-lei-de-moore-.htm. Acessado em 28/07/2015.

Apêndice A

Tempo de armazenamento dos Arquivos: cabra5.fa, cabra6.fa e cabra7.fa

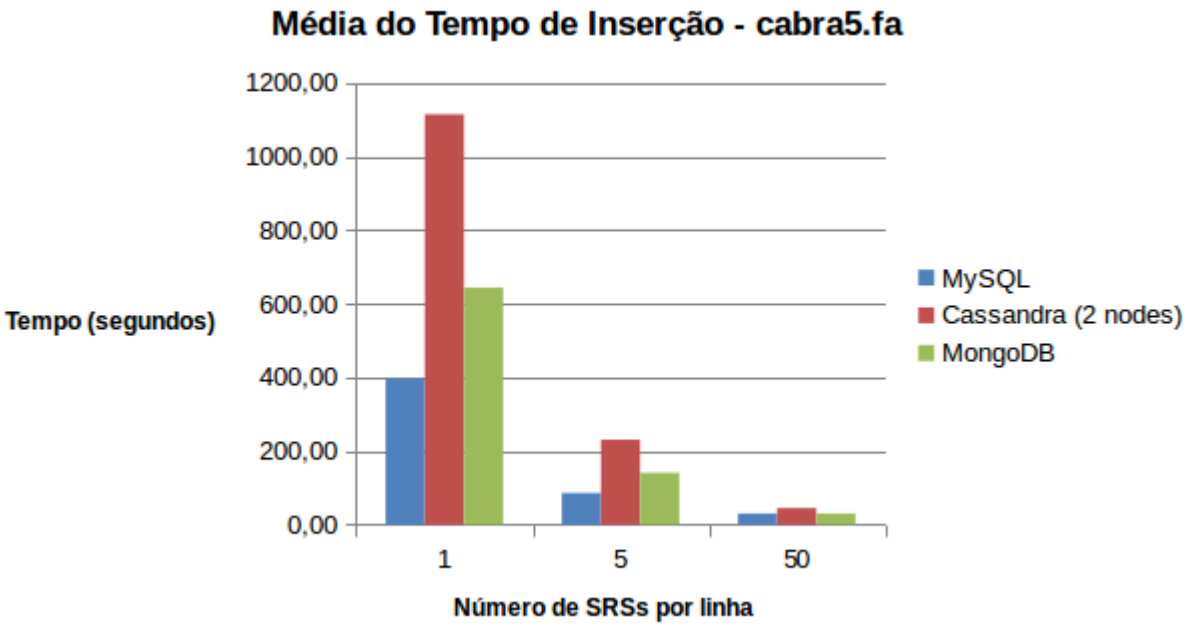

Figura A.1: Tempo de armazenamento - Cabra5.fa

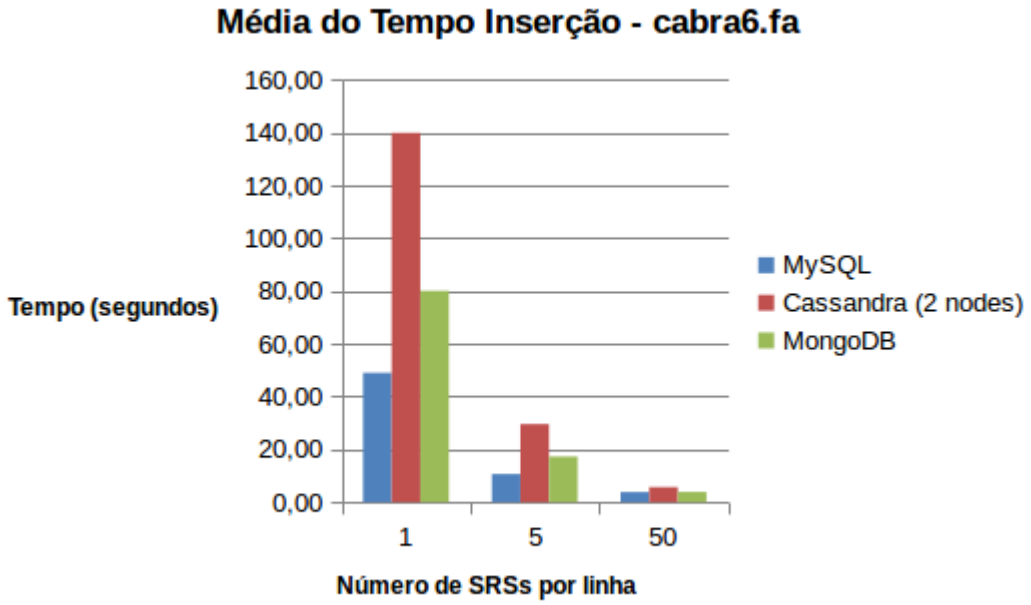

Figura A.2: Tempo de armazenamento - Cabra6.fa

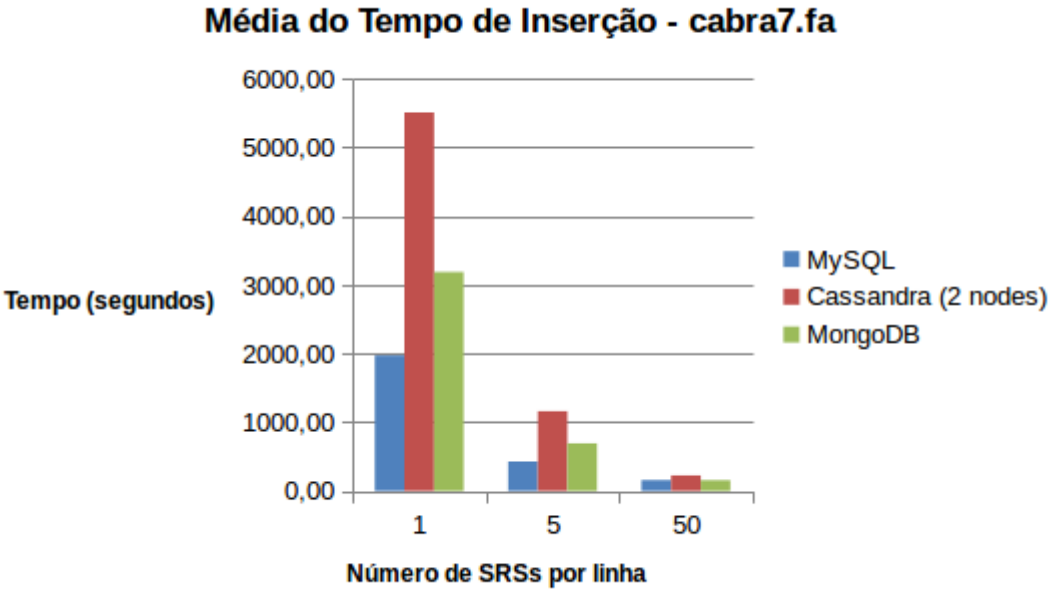

Figura A.3: Tempo de armazenamento - Cabra7.fa

Apêndice B

Resultado dos Experimentos

Inserção-Cabra4

# SRR/linha	MySQL	Cassandra (2 nodes)	MongoDB	INSERÇÃO CABRA 4		MySQL	Cassandra (2 nodes)	MongoDB
1	1271,88	3400,96	2005,43		Média	1251,89	3439,17	1984,19
	1205,45	3462,17	2012,8		Mediana	1219,03	3452,59	1976,77
	1219,03	3452,59	1976,77		Desv Pad	59,10	36,59	24,24
	1216,06	3400,15	1955,1		Mínimo	1205,45	3400,15	1955,1
	1347,02	3479,97	1970,83		Máximo	1347,02	3479,97	2012,8
5	279,71	726,09	440,23		Média	271,012	727,25	437,89
	267,36	713,7	449,6		Mediana	267,36	716,06	436,63
	265,58	711,44	431,56		Desv Pad	6,67	23,98	7,52
	265,75	716,06	431,41		Mínimo	265,58	711,44	431,41
	276,66	768,96	436,63		Máximo	279,71	768,96	449,6
50	101,32	143,38	97,98		Média	98,10	141,84	95,93
	96,69	139,59	94,85		Mediana	96,69	142,9	95,48
	96,63	139,44	95,48		Desv Pad	2,19	2,16	1,25
	96,39	143,91	96,17		Mínimo	96,39	139,44	94,85
	99,45	142,9	95,16		Máximo	101,32	143,91	97,98

Inserção-Cabra5

# SRR/linha	MySQL	Cassandra (2 nodes)	MongoDB	INSERÇÃO CABRA 5		MySQL	Cassandra (2 nodes)	MongoDB
1	408,81	1112,87	648,25		Média	397,63	1114,52	643,74
	394,28	1121,06	655,75		Mediana	394,44	1112,87	643,23
	391,95	1127,8	641,5		Desv Pad	6,70	9,84	9,48
	394,44	1104,91	629,97		Mínimo	391,95	1104,91	629,97
	398,68	1105,97	643,23		Máximo	408,81	1127,8	655,75
5	86,27	230,9	140,07		Média	86,02	230,69	140,72
	85,55	230,71	149,96		Mediana	86,19	230,9	138,94
	85,69	228,12	138,94		Desv Pad	0,38	1,54	5,37
	86,19	231,7	136,02		Mínimo	85,55	228,12	136,02
	86,42	232,04	138,61		Máximo	86,42	232,04	149,96
50	30,35	45,13	30,01		Média	30,13	44,91	30,06
	30,01	44,2	29,79		Mediana	30,06	45,13	30,01
	30,06	43,9	30,43		Desv Pad	0,18	0,88	0,25
	29,93	46,12	29,9		Mínimo	29,93	43,9	29,79
	30,28	45,2	30,15		Máximo	30,35	46,12	30,43

Inserção-Cabra6

# SRR/linha	MySQL	Cassandra (2 nodes)	MongoDB	INSERÇÃO CABRA 6		MySQL	Cassandra (2 nodes)	MongoDB
1	48,64	139,64	80,96		Média	48,98	139,95	80,08
	48,84	140,77	81,28		Mediana	48,84	140,29	80,22
	49,41	140,64	79,98		Desv Pad	0,35	0,96	1,30
	49,31	138,43	77,95		Mínimo	48,64	138,43	77,95
	48,71	140,29	80,22		Máximo	49,41	140,77	81,28
5	10,64	32,83	17,49		Média	10,67	29,598	17,28
	11,11	28,82	17,23		Mediana	10,64	28,87	17,23
	10,51	28,98	17,14		Desv Pad	0,28	1,82	0,17
	10,36	28,87	17,1		Mínimo	10,36	28,49	17,1
	10,72	28,49	17,43		Máximo	11,11	32,83	17,49
50	3,81	5,71	4		Média	3,82	5,70	3,84
	3,93	5,68	3,78		Mediana	3,81	5,71	3,79
	3,86	5,6	3,64		Desv Pad	0,08	0,08	0,16
	3,78	5,71	3,79		Mínimo	3,71	5,6	3,64
	3,71	5,82	4,01		Máximo	3,93	5,82	4,01

Inserção-Cabra7

# SRR/linha	MySQL	Cassandra (2 nodes)	MongoDB	INSERÇÃO CABRA 7		MySQL	Cassandra (2 nodes)	MongoDB
1	2007,2	5471,92	3194,68		Média	1971,65	5509,16	3187,38
	1962,08	5565,8	3226,23		Mediana	1962,08	5496,96	3194,68
	1960,11	5541,68	3214,53		Desv Pad	22,32	42,95	36,34
	1978,57	5469,44	3138,6		Mínimo	1950,28	5469,44	3138,6
	1950,28	5496,96	3162,84		Máximo	2007,2	5565,8	3226,23
5	428,18	1203,44	693,86		Média	428,58	1159,74	692,28
	429,18	1137,54	691,75		Mediana	428,18	1146,2	691,75
	426,99	1146,2	693,38		Desv Pad	1,93	26,58	1,26
	426,95	1145,56	691,03		Mínimo	426,95	1137,54	691,03
	431,62	1165,96	691,36		Máximo	431,62	1203,44	693,86
50	162,98	226,04	155,34		Média	158,84	226,43	154,83
	157,1	227,67	153,76		Mediana	157,1	226,04	155
	156,76	229,2	154,9		Desv Pad	2,76	1,99	0,62
	160,42	224,66	155		Mínimo	156,76	224,6	153,76
	156,95	224,6	155,15		Máximo	162,98	229,2	155,34

Extração-Cabra4

# SRR/linha	MySQL	Cassandra (2 nodes)	MongoDB	EXTRAÇÃO CABRA 4		MySQL	Cassandra (2 nodes)	MongoDB
1	318,15	225,79	100,15		Média	321,32	217,58	64,61
	320,24	224,6	79,75		Mediana	321,57	214,3	79,75
	323,24	212,73	30,73		Desv Pad	2,20	7,10	32,26
	323,4	214,3	82,5		Mínimo	318,15	210,48	29,92
	321,57	210,48	29,92		Máximo	323,4	225,79	100,15
5	136,03	73,47	51,52		Média	136,32	77,71	58,32
	135,49	77,91	67,73		Mediana	136,05	77,91	54,59
	136,05	80,23	54,59		Desv Pad	0,81	3,06	7,08
	137,66	80,91	53,84		Mínimo	135,49	73,47	51,52
	136,35	76,02	63,93		Máximo	137,66	80,91	67,73
50	84,86	43,2	30,27		Média	87,34	47,81	35,48
	87,26	48,16	46,03		Mediana	87,26	46,57	32,32
	86,58	44,29	32,32		Desv Pad	1,75	5,40	6,59
	89,05	56,82	31,01		Mínimo	84,86	43,2	30,27
	88,96	46,57	37,76		Máximo	89,05	56,82	46,03

Extração-Cabra5

# SRR/linha	MySQL	Cassandra (2 nodes)	MongoDB	EXTRAÇÃO CABRA 5		MySQL	Cassandra (2 nodes)	MongoDB
1	36,74	56,28	24,61		Média	38,92	54,04	24,43
	39,87	54,86	40,37		Mediana	39,83	54,09	24,6
	39,93	52,76	24,6		Desv Pad	1,41	1,63	10,26
	38,25	52,22	20,41		Mínimo	36,74	52,22	12,16
	39,83	54,09	12,16		Máximo	39,93	56,28	40,37
5	17,81	22,66	22,79		Média	18,01	23,78	24,42
	17,68	24,2	22,79		Mediana	17,68	23,6	22,79
	17,33	23,6	27,09		Desv Pad	0,987	0,82	3,05
	17,47	23,57	21,21		Mínimo	17,33	22,66	21,21
	19,74	24,86	28,22		Máximo	19,74	24,86	28,22
50	10,98	13,58	15,5		Média	11,55	12,02	19,67
	11,16	12	24,08		Mediana	11,3	12	18,21
	12,9	10,37	18,21		Desv Pad	0,77	1,22	3,62
	11,4	11,44	17,74		Mínimo	10,98	10,37	15,5
	11,3	12,7	22,8		Máximo	12,9	13,58	24,08

Extração-Cabra6

# SRR/linha	MySQL	Cassandra (2 nodes)	MongoDB	EXTRAÇÃO CABRA 6		MySQL	Cassandra (2 nodes)	MongoDB
1	2,33	10,48	2,98		Média	2,24	9,20	2,20
	2,18	8,24	1,84		Mediana	2,22	9,03	1,84
	2,22	8,95	1,64		Desv Pad	0,06	0,82	0,90
	2,24	9,32	1,23		Mínimo	2,18	8,24	1,23
	2,21	9,03	3,32		Máximo	2,33	10,48	3,32
5	1,17	6,14	8,5		Média	1,25	3,74	3,77
	1,41	3,91	2,47		Mediana	1,22	2,89	2,47
	1,21	2,86	4,3		Desv Pad	0,09	1,42	2,87
	1,22	2,89	2,39		Mínimo	1,17	2,86	1,2
	1,23	2,88	1,2		Máximo	1,41	6,14	8,5
50	1,77	2,07	6,5		Média	1,28	2,44	4,53
	1,08	1,66	2,62		Mediana	1,14	2,1	5,51
	1,14	2,1	5,51		Desv Pad	0,28	1,01	2,32
	1,13	2,16	1,51		Mínimo	1,08	1,66	1,51
	1,3	4,21	6,53		Máximo	1,77	4,21	6,53

Extração-Cabra7

# SRR/linha	MySQL	Cassandra (2 nodes)	MongoDB	EXTRAÇÃO CABRA 7		MySQL	Cassandra (2 nodes)	MongoDB
1	796,92	343,56	131,81		Média	805,71	349,01	119,49
	800,41	355,32	139,44		Mediana	808,67	348,6	131,81
	812,08	349,2	138,45		Desv Pad	6,66	4,19	23,76
	810,48	348,35	89,67		Mínimo	796,92	343,56	89,67
	808,67	348,6	98,06		Máximo	812,08	355,32	139,44
5	341,04	122,52	90,7		Média	338,86	118,04	93,44
	337,07	116,99	75,24		Mediana	338,49	116,99	90,7
	337,39	113,51	113,68		Desv Pad	1,76	3,51	16,96
	338,49	120,46	79,62		Mínimo	337,07	113,51	75,24
	340,3	116,73	107,95		Máximo	341,04	122,52	113,68
50	229,71	62,19	24,558		Média	232,44	79,65	53,66
	233,74	87,96	54,08		Mediana	233,04	82,46	54,08
	233,34	85,93	79,15		Desv Pad	1,61	10,26	19,55
	232,35	82,46	58,88		Mínimo	229,71	62,19	24,558
	233,04	79,73	51,65		Máximo	233,74	87,96	79,15

Consulta

# SRR/linha	MySQL	Cassandra (2 nodes)	MongoDB	CONSULTA POR ID DE SEQUENCIA		MySQL	Cassandra (2 nodes)	MongoDB
1	27,91	0,76	28,78		Média	28,98	0,31	28,72
	27,81	0,36	28,65		Mediana	29,13	0,28	28,68
	25,61	0,32	28,59		Desv Pad	1,47	0,11	0,13
	25,53	0,31	28,62		Mínimo	25,53	0,22	28,59
	27,82	0,31	28,63		Máximo	31,69	0,76	29,18
	28,57	0,32	28,82					
	28,28	0,28	28,62					
	28,37	0,28	28,67					
	28,59	0,27	28,69				IDs Consultados	
	28,89	0,28	28,7				>1303_40_1460_F3	
	30,28	0,27	29,18				>1303_42_1190_F3	
	31,69	0,26	28,66				>1303_37_58_F3	
	29,42	0,3	28,7				>1303_43_50_F3	
	29,2	0,3	28,68				>1303_38_874_F3	
	29,13	0,24	28,68					
	29,96	0,4	28,85					
	31,04	0,32	28,61					
	29,13	0,25	28,67					
	28,66	0,25	28,68					
	29,17	0,22	28,67					
	30,22	0,4	28,98					
	31,66	0,32	28,77					
	29,16	0,25	28,71					
	29,3	0,25	28,75					
	29,17	0,22	28,7					

Curva de consulta

Qtde de registros	# SRR/linha	MySQL	Cassandra (2 nodes)	MongoDB	CURVA DE CONSULTA		MySQL	Cassandra (2 nodes)	MongoDB
1 milhão	1	0,42	0,26	0,3		Média	0,40	0,22	0,28
		0,4	0,19	0,28		Mediana	0,4	0,2	0,28
		0,41	0,2	0,28		Desv Pad	0,01	0,03	0,01
		0,39	0,25	0,28		Mínimo	0,39	0,19	0,28
		0,4	0,2	0,28		Máximo	0,42	0,26	0,3
5 milhões	1	3,46	0,36	1,45		Média	3,41	0,22	1,40
		3,41	0,18	1,37		Mediana	3,41	0,19	1,38
		3,38	0,19	1,38		Desv Pad	0,04	0,08	0,03
		3,42	0,2	1,37		Mínimo	3,37	0,18	1,37
		3,37	0,18	1,41		Máximo	3,46	0,36	1,45
10 milhões	1	6,72	0,23	2,95		Média	6,68	0,24	2,78
		6,64	0,18	2,73		Mediana	6,68	0,21	2,75
		6,68	0,39	2,74		Desv Pad	0,051	0,09	0,09
		6,63	0,19	2,75		Mínimo	6,63	0,18	2,73
		6,75	0,21	2,75		Máximo	6,75	0,39	2,95
15 milhões	1	10,29	0,21	10,32		Média	10,16	0,20	7,71
		10,17	0,19	9,6		Mediana	10,2	0,19	6,95
		10,2	0,2	6,16		Desv Pad	0,13	0,01	2,13
		10,2	0,19	6,95		Mínimo	9,95	0,19	5,51
		9,95	0,19	5,51		Máximo	10,29	0,21	10,32
20 milhões	1	13,3	0,28	13,86		Média	13,44	0,21	13,51
		13,52	0,2	13,31		Mediana	13,5	0,2	13,59
		13,24	0,19	13,59		Desv Pad	0,17	0,04	0,42
		13,5	0,2	13,9		Mínimo	13,24	0,19	12,88
		13,66	0,2	12,88		Máximo	13,66	0,28	13,9