Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Uma análise de desempenho de uma abordagem NoSQL no Bussiness Intelligence

Felipe Barreto Fernandes

Monografia apresentada como requisito parcial
para conclusão do Curso de Engenharia da Computação

Orientadora
Prof.a Dr.a Maristela Terto de Holanda

Coorientador
Prof. Dr. Marcio de Carvalho Victorino

Brasília
2017

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Uma análise de desempenho de uma abordagem NoSQL no Bussiness Intelligence

Felipe Barreto Fernandes

Monografia apresentada como requisito parcial
para conclusão do Curso de Engenharia da Computação

Prof.a Dr.a Maristela Terto de Holanda (Orientadora)
CIC/UnB

Prof.a Dr. Aletéia Patrícia Favacho de Araújo
Universidade de Brasília

Prof. Dr. Marcio de Carvalho Victorino
Universidade de Brasília

Prof. Dr. Ricardo Pezzuol Jacobi
Coordenador do Curso de Engenharia da Computação

Brasília, 17 de agosto de 2017

Dedicatória

Aos meus pais e familiares que sempre me ajudaram e incentivaram a atingir meus objetivos.

Agradecimentos

Agradeço a Deus, minha família e amigos que sempre me deram apoio e incentivo para a realização deste trabalho.

Resumo

O BI, também conhecido por *Business Intelligence*(Inteligência do negócio), é o conjunto de tecnologias orientadas a disponibilizar informação e conhecimento em uma organização para a tomada de decisão. Sua implementação inclui um *Data Warehouse*, que é uma ferramenta cuja concepção e administração é voltada a bancos de dados ou um volume de dados. Com o o avanço das tecnologias voltadas a escalabilidade de um banco de dados, o uso de bancos voltados a desempenho que administram uma massiva quantidade de dados vem aumentando e são desenvolvidos para ambientes de baixo custo. Esses bancos de dados são conhecidos como bancos NoSQL. Este trabalho apresenta um estudo de caso para analise de desempenho entre uma nova abordagem de um *Data Warehouse*, onde expõe testes feitos comparativos para avaliar o desempenho da consulta de dados nessa nova abordagem, utilizando o Cassandra, um banco de dados NoSQL, comparando a uma abordagem tradicional em um banco de dado relacional MySQL.

Palavras-chave: banco de dados, nosql, inteligência do negócio, *data warehouse*, cassandra

Abstract

BI, also known as Business Intelligence, is the set of technologies oriented to providing information and knowledge in an organization for decision making. Its implementation includes a Data Warehouse, which is a tool whose design and administration is focused on databases or a volume of data. Through the advancement of technologies focused on the database scalability, the use of performance-oriented databases that manage a massive amount of data has been growing and are designed for low-cost environments. These databases are known as NoSQL. This paper presents a case study for performance analysis between a new approach of a Data Warehouse, where it exposes comparative tests to evaluate the performance of the data query in this new approach, using Cassandra, a NoSQL database comparing to a traditional approach in a MySQL relational database.

Keywords: databases, nosql, business intelligence, data warehouse, cassandra

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

O Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira (INEP) é uma autarquia federal vinculada ao Ministério da Educação (MEC), cuja missão é promover estudos, pesquisas e avaliações sobre o Sistema Educacional Brasileiro com o objetivo de subsidiar a formulação e implementação de políticas públicas na área educacional a partir de parâmetros de qualidade e equidade, bem como produzir informações claras e confiáveis aos gestores, pesquisadores, educadores e público em geral[9]. O portal do INEP também disponibiliza os dados a respeito da instituições de educação superior (IES), que podem ser extraídos via download diretamente site do portal[1].

Um banco de dados NoSQL é um banco de dados que trabalha com uma abordagem diferente do relacional, sendo ele distribuído, rápido, de baixo custo desenvolvido para lidar com grande volume de dados. Os bancos de dados NoSQL usam diversos modelos de dados, incluindo documentos, grafos, chave-valor e colunares. Bancos de dados NoSQL são amplamente reconhecidos pelo desempenho escalável, suporte a replicação de dados, alta disponibilidade e resiliência.

Há vários sistemas gerenciadores de banco de dados (SGBD) NoSQL, e neste documento será demonstrado o uso do SGBD Cassandra. O Cassandra é um projeto *open source* Apache, que nasceu no Facebook e sua construção foi baseado no Dynamo da Amazon[10] e no BigTable do Google[11]. Cassandra é um SGBD NoSQL distribuído para o gerenciamento de grandes quantidades de dados, sendo analisado neste trabalho como proposta para armazenamento dos dados do INEP para analisar o desempenho das consultas destas informações.

A modelagem multidimensional, é a técnica de modelagem de banco de dados para o auxílio as consultas do *Data Warehouse*, nas mais diferentes perspectivas. *Data Warehouse* é um depósito de dados que serve para armazenar informações detalhadas relativas a uma organização, criando e organizando relatórios através de históricos que são depois

[1] `http://portal.inep.gov.br/web/guest/dados`

utilizados pela organização para ajudar a tomar decisões importantes com base nos fatos apresentados. A visão multidimensional permite o uso mais intuitivo para o processamento analítico pelas ferramentas de análise de dados OLAP (On-line Analytical Processing).

Este trabalho tem como objetivo propor uma modelagem dimensional para o Cassandra e comparar relativamente com tecnologias tradicionais, como um banco de dados relacional, com o objetivo de melhorar o desempenho e investigar se as características de um banco NoSQL, como escalabilidade e baixo custo são eficientes para o uso de um modelo multidimensional.

O desafio de propor um modelo dimensional para um banco de dados NoSQL é árduo, pois a modelagem dimensional dos *Data Warehouses* adotada atualmente tem sido usada por anos, para a tomada de decisões em organizações.

Assim, no presente trabalho, é apresentado uma forma de convergir um estudo sobre o modelo dimensional e sobre o banco de dados Cassandra e suas características, propondo um modelo de dados que será utilizado em uma abordagem de Business Intelligence a fim de prover a analisar de dados do INEP.

1.1 Definição do problema

O problema deste trabalho consiste na análise da latência de consulta em *Data Warehouse* NoSQL com uma modelagem proposta no Cassandra, projetada com o intuito de aumentar o desempenho das consultas dos dados. A hipótese é que com essa modelagem no Cassandra, há uma menor latência de consultas, tendo em vista que bancos relacionais gastam muito processamento com *joins* (junções), o que gera uma maior latência de consulta, problema que não é encontrado na modelagem NoSQL, além de poder ser escalado dinamicamente para ter um melhor desempenho.

O uso de banco de dados NoSQL vem sendo usados em grandes empresas que geram uma grande quantidade de dados e que precisam de uma grande performance de consulta de dados. Google, Amazon e Facebook vem quebrando esse paradigma, com o uso e estudo de banco de dados NoSQL, para obter bons resultados. E assim, outras grandes empresas do mercado, também estão seguindo esses passos para obter bons resultados para seus negócios. Como não há grandes estudos desses tipo de banco de dados para uma modelagem tradicionalmente usada no Business Inteligence (OLAP), o contexto desse trabalho é gerar uma abordagem de como os banco de dados NoSQL podem ser eficientes em análises de dados de *Data Warehouses* com um número significante de dados.

1.2 Metodologia

Para fazer tal comparação, adotou-se a tática de fazer um ETL (extração, transformação e carga) de uma fonte de dados para um banco relacional e um ETL para um banco de dados NoSQL, utilizando os dados vindo do INEP[9], e usando uma modelagem dimensional, além de adaptar a modelagem dimensional que é projetada para um DW de banco de dados relacionais para um DW de banco NoSQL, fazendo uma consulta que gere os mesmos resultados, podendo assim comparar a latência das consultas.

1.3 Objetivo

1.3.1 Objetivos Gerais

O objetivo deste trabalho é realizar uma comparação de desempenho do uso de um banco de dados NoSQL Cassandra com um banco relacional MySQL com uma abordagem do modelo dimensional proposta para uso no Cassandra.

1.3.2 Objetivos Específicos

Para alcançar o objetivo geral deste trabalho, os seguintes objetivos específicos foram necessários:

- Especificar um ambiente e aplicações necessárias para instalação dos componentes;
- Construção do *Data Warehouse* usando um modelo dimensional para o MySQL;
- Construção do *Data Warehouse* usando um modelo dimensional para o Cassandra;
- Executar o processo de ETL para o DW relacional;
- Executar o processo de ETL para o DW NoSQL;
- Elaborar consultas e análises idênticas para ambos *Data Warehouses*;
- Fazer um comparação de desempenho entre a abordagem Relacional e a NoSQL;
- Fazer um gráfico de uma análise de uma consulta em uma ferramento OLAP.

1.4 Estrutura do Trabalho

- Capítulo 2: É apresentado todo o referencial Teórico deste trabalho. Os conceitos e fundamentação teórica de Dado, Informação, dados abertos e dados governamentais. Assim como o conceito e teoria sobre Sistemas de Informação, Sistemas de Apoio a

Decisão, *Data Warehouse*, Modelagem Dimensional e Arquitetura e Ciclo de Vida de um DW/BI para a implementação deste trabalho. Também são apresentados os conceitos e fundamentação teórica de Banco de Dados Relacional, Banco de Dados NoSQL e o Cassandra.

- Capítulo 3: Estudo de caso de uma implementação de um *Data Warehouse* Relacional e um *Data Warehouse* NoSQL com o uso de dados abertos.
- Capítulo 4: Resultados obtidos de desempenho dos *Data Warehouses* implementados é apresentado, além de painéis de análises dos dados abertos obtidos.
- Capítulo 5: São realizadas conclusões a respeito do trabalho, explicando pontos importantes discutidos ao longo do trabalho e possíveis trabalhos futuros

Capítulo 2

Referencial Teórico

Neste capítulo, são apresentadas as definições de dados, informações, dados abertos e dados governamentais, é apresentado também uma definição de Sistemas de Informação, Sistemas de Apoio a Decisão, modelagem dimensional, ciclo de vida de uma projeto de um *Data Warehouse*, assim como um abordagem a respeito de banco de dados relacionais e NoSQL, citando as principais características de cada uma delas.

2.1 Dados e Informações

Para este trabalho, é usado um imenso volume de dados que geram informações. Para a definição de um dado, Setzer[12] afirma que é como uma sequência de símbolos quantificados ou quantificáveis. Quantificável é que algo pode ser quantificado e depois reproduzido, sem que se perceba a diferença para com o original. Portanto, um texto é um dado. Conforme Setzer[12] de fato, as letras são símbolos quantificados, já que o alfabeto, sendo um conjunto finito, pode por si só constituir uma base numérica (a base hexadecimal empregada em geral nos computadores usa, além dos 10 dígitos decimais, as letras de A a E). Fotos, figuras, sons gravados e animação também são dados, pois todos podem ser quantificados ao serem introduzidos em um computador, a ponto de se ter eventualmente dificuldade de distinguir a sua reprodução com o original. É muito importante notar-se que, mesmo se incompreensível para o leitor, qualquer texto constitui um dado ou uma sequência de dados. Isso ficará mais claro no próximo item.

Ainda de acordo com Setzer[12], com essa definição, um dado é necessariamente uma entidade matemática e, desta forma, é puramente sintático. Isto significa que os dados podem ser totalmente descritos através de representações formais, estruturais, e ainda podem ser armazenados em um computador e processados por ele. O processamento de dados em um computador limita-se exclusivamente a manipulações estruturais dos

mesmos, e é feito por meio de programas. Estes são sempre funções matemáticas, e portanto também são dados.

E, de acordo com Machado[13], o dado é uma representação, um registro de uma informação e que pode ser registrado, por exemplo em um pelado de papel (uma receita médica), um disco de um computador ou um sinal elétrico. No caso, um registro pode ser processado e originar algo que influencia a vida das pessoas (salvar uma vida de um paciente, tocar um alarme).

Setzer[12] afirma que a informação é uma abstração informal, que vem no pensamento de alguém, representando algo significativo para essa pessoa. Por exemplo, a frase "Paris é uma cidade fascinante" é um exemplo de informação desde que seja lida ou ouvida por alguém, e desde que "Paris" signifique para essa pessoa a capital da França (supondo-se que o autor da frase queria referir-se a essa cidade) e "fascinante" tenha a qualidade usual e intuitiva associada com essa palavra.

Conforme Machado[13], o tratamento das informações dá origem a vários tipos de dados, porém o dado deve registrar apenas os aspectos realmente relevantes da informação, ou seja, o endereço do fabricante do remédio não tem nenhum interesse para um sistema de controle que mantém a vida dos pacientes no hospital. Então, as informações contidas em um sistema de informação, são aquelas necessárias ao objetivo do sistema.

2.2 Dados Abertos

Segundo a Open Definition[14], dados abertos são dados que podem ser livremente utilizados, reutilizados e redistribuídos por qualquer pessoa, que estão sujeitas, no máximo, à exigência de atribuição à fonte original e ao compartilhamento pelas mesmas licenças em que as informações foram apresentadas. Então a abertura dos dados tem como foco evitar qualquer mecanismo de controle e restrição, permitindo que qualquer pessoa possam obter e explorar esses dados de forma livre[15].

Por essa visão, a definição dos abertos segue três normas fundamentais (Open Knowledge Foundation, 2010)[15]:

1. Disponibilidade e acesso: os dados devem estar disponíveis como um todo e sob custo não maior que um custo razoável de reprodução, e preferencialmente devem ser possíveis de ser baixados pela Internet. Os dados devem também estar disponíveis de uma forma conveniente e modificável;

2. Reúso e redistribuição: os dados devem ser fornecidos sob termos que permitam a reutilização e a redistribuição, inclusive a combinação com outros conjuntos de dados;

3. Participação universal: todos devem ser capazes de usar, reutilizar e redistribuir – não deve haver discriminação contra áreas de atuação ou contra pessoas ou grupos. Por exemplo, restrições de uso "não comercial" que impediriam o uso comercial, ou restrições de uso para certos fins (ex.: somente educativos) excluem determinados dados do conceito de "abertos".

Para publicação de Dados Abertos, foram definidos dez boas práticas, conforme Hyland[16]:

1. **Preparar os Stakeholders:** esta etapa é voltada para a formação dos usuários que irão criar e manter os dados abertos conectados;
2. **Selecionar o Conjunto (Fonte) de Dados:** etapa dedicada a definir que se pretende abrir e conectar a outros dados e disponibilizá-los para reuso;
3. **Modelar os Dados:** com os Stakeholders capacitados e o conjunto de dados definidos, começa-se a etapa de modelagem dos dados conectados. Ou seja, como iremos representar os dados e como eles se relacionam com outros dados e de forma independente de aplicação;
4. **Especificar a Licença:** nesta etapa a equipe/organização, responsável pelos dados que serão abertos deve, definir especificar a licença que será usada;
5. **Nomear boas URIs:** esta etapa é o núcleo dos Dados Abertos Conectados e a definição e uso de boas práticas para URIs é fundamental;
6. **Usar Vocabulários Padrões:** uma das melhores formas de conectar dados é através do reuso de vocabulários conhecidos (ex. Recomendações do W3C);
7. **Converter os Dados:** Uma vez que a estratégia de modelagem, as boas práticas para URIs foram definidas e os vocabulários a serem reusados foram identificados, vem a etapa de converter os dados da fonte original para representação adequada aos Dados Conectados;
8. **Prover acesso aos Dados:** esta etapa define quais serão as formas de acesso que seres humanos e máquinas terão aos dados;
9. **Anunciar novo conjunto de Dados Conectados:** de nada adianta conectar os dados e não anunciar para a sociedade que os mesmos foram disponibilizados. Desta forma, esta etapa cumpre este papel de divulgação do novo Dataset publicado;
10. **Reconhecer o papel social:** esta etapa é definida para que o responsável por publicação dos dados cumpra com o papel de manter os dados publicados ao longo do tempo.

2.3 Dados Governamentais

De acordo com Hiller e Kaltenbock[15] quando falamos hoje sobre dados governamentais, nos referimos a um movimento que foi iniciado pelo "*The Memorandum on Transparency and Open Government*"[17], um memorando assinado pelo presidente dos EUA, Barack Obama logo após sua inauguração em janeiro de 2009. A ideia do *Open Government* é estabelecer uma cooperação moderna entre políticas, administração pública, indústria, pessoa privada e permitindo mais transparência, democracia, participação e colaboração. A parceria *The Open Government* foi lançada no dia 20 de Setembro de 2011 quando os oito governos fundadores (Brasil, Indonésia, México, Noruega, Filipinas, África do Sul, Reino Unido e Estados Unidos) aprovaram uma declaração do *Open Government*, e anunciaram os planos de ação de seus países e deram boas vindas ao compromisso de 38 governos de se juntarem à parceria. Em setembro de 2011, havia 46 países ingressos no *Open Government* em todo o mundo. Alguns dos tópicos mais importantes para o Open Government, são o acesso gratuito à informação e a possibilidade de usar e reutilizar esta informação gratuitamente (por exemplo, dados, conteúdo, etc.).

E de acordo com sítio `http://dados.gov.br/paginas/dados-abertos`[18] os dados abertos também são pautados pelas três leis e oito princípios. O especialista em políticas públicas e ativista dos dados abertos David Eaves propôs as seguintes "leis":

1. Se o dado não pode ser encontrado e indexado na Web, ele não existe;
2. Se não estiver aberto e disponível em formato compreensível por máquina, ele não pode ser reaproveitado; e
3. Se algum dispositivo legal não permitir sua replicação, ele não é útil.

Em 2007, um grupo de trabalho de 30 pessoas reuniu-se na Califórnia, Estados Unidos da América (*The Annotated 8 Principles of Open Government Data*)[19], e chegaram em uma definição dos princípios dos Dados Abertos Governamentais. Chegaram num consenso sobre os seguintes 8 princípios:

1. **Completos.** Todos os dados públicos são disponibilizados. Dados são informações eletronicamente gravadas, incluindo, mas não se limitando a, documentos, bancos de dados, transcrições e gravações audiovisuais. Dados públicos são dados que não estão sujeitos a limitações válidas de privacidade, segurança ou controle de acesso, reguladas por estatutos.
2. **Primários.** Os dados são publicados na forma coletada na fonte, com a mais fina granularidade possível, e não de forma agregada ou transformada.

3. **Atuais.** Os dados são disponibilizados o quão rapidamente seja necessário para preservar o seu valor.
4. **Acessíveis.** Os dados são disponibilizados para o público mais amplo possível e para os propósitos mais variados possíveis.
5. **Processáveis por máquina.** Os dados são razoavelmente estruturados para possibilitar o seu processamento automatizado.
6. **Acesso não discriminatório.** Os dados estão disponíveis a todos, sem que seja necessária identificação ou registro.
7. **Formatos não proprietários.** Os dados estão disponíveis em um formato sobre o qual nenhum ente tenha controle exclusivo.
8. **Livres de licenças.** Os dados não estão sujeitos a regulações de direitos autorais, marcas, patentes ou segredo industrial. Restrições razoáveis de privacidade, segurança e controle de acesso podem ser permitidas na forma regulada por estatutos.

E a partir de 2010, a fundação *Sunlight*[1] adicionou mais 2 princípios:

9. **Permanência.** É a capacidade de achar o dado de acordo com o passar do tempo.
10. **Custo de uso.** Um dos maiores obstáculos ao acesso a informações ostensivamente publicamente disponíveis, é o custo imposto ao público - mesmo quando o custo é mínimo.

2.4 Definição de Sistema e Sistemas de Informação

O conceito de sistema, está oculto no conceito de sistemas de informação, então para melhor entendimento, é necessário compreender o conceito de sistema. O'Brien[20] diz que um sistema pode ser conceituado como um grupo de elementos inter-relacionados ou em interação que formam um todo unificado.

E de acordo com Wakulicz[1], um sistema é um grupo de componentes que estão inter-relacionados e que visam uma meta comum a partir do recebimento de informações produzindo resultados em um processo organizado de transformação.

Um sistema possui três componentes ou funções básicas em interação:

[1]The Sunlight Foundation é uma organização sem fins lucrativos não partidária que defende a abertura governamental a nível mundial e usa tecnologia para tornar o governo melhor explicado para todos.`https://sunlightfoundation.com/contact/`

Figura 2.1: Elementos básicos de um sistema[1]

- ***Inputs***: Captura ou coleta dados que ingressam no sistema para serem processados (dados, instruções).
- **Processamento**: Converte esses dados em uma forma mais significativas, transformando entrada em uma forma mais significativa.
- ***Outputs***: Transfere as informações processadas às pessoas que as utilizarão ou às atividades nas quais serão utilizadas.

Com isso, chegamos a definição de Laudon[2], onde um sistema de informações pode ser definido como um conjunto de componentes inter-relacionados que coletam, processam, armazenam e distribuem informações destinadas a apoiar a tomada de decisão, a coordenação e o controle de uma organização.

Figura 2.2: Os cinco componentes de um sistema de informação[1]

E segundo O'Brien[20], para realizar essas atividades de *inputs*, processamento, armazenamento e *outputs* que convertem recursos de dados em produtos da informação, um sistema de informação depende dos recursos humanos, de hardware, software, dados e redes.

2.5 Classificação dos Sistemas de Informação

A forma de classificação do sistema de informação está relacionado ao tipo de suporte por ele proporcionado a empresa. Os principais tipos de sistema de informação segundo o suporte proporcionado são[2][20]:

- Sistema de suporte inteligente (SSI).

- Sistema de apoio a grupos (GSS).
- Sistema de informação empresarial (EIS).
- Sistema de apoio a decisões (SAD).
- Sistema de automação de escritório (SAE).
- Sistema de administração de conhecimento (KMS).
- Sistema de processamento de transações (SIT).
- Sistema de informação gerencial (SIG).

2.6 Tomada de Decisão e Sistemas de Informação

Uma das principais contribuições dos sistemas de informação tem sido melhorar a tomada de decisão. Segundo Laudon[2] em cada nível de uma organização, há diferentes necessidades de informação para apoiar suas decisões e é responsável por diferentes tipos de decisão. As decisões podem ser classificadas em[2]:

Decisões não estruturadas são aquelas em que o responsável pela tomada de decisão deve usar seu bom senso, sua capacidade de avaliação e seu entendimento e vivencia na definição do problema. Cada uma dessas decisões é inusitada, importante e não rotineira, e não há procedimentos bem compreendidos ou predefinidos para tomá-las.

Decisões estruturadas, ao contrário, são repetitivas e rotineiras, e envolvem procedimentos predefinidos, de modo que não precisam ser tratadas como se fossem novas. Algumas decisões têm características dos dois tipos de decisões, e são chamadas de **decisões semi-estruturadas**, onde apenas parte do problema tem uma resposta clara e precisa. Em geral, decisões estruturadas são mais frequentes nos níveis organizacionais mais baixos, enquanto problemas não estruturados são mais comuns nos níveis mais altos da empresa.

2.6.1 Sistemas de Apoio a Decisão (SAD)

Pode se dizer que os sistemas de apoio à decisão são sistemas de informação baseado em computador que combinam modelos e dados, tentando solucionar problemas semi-estruturados com grande envolvimento por parte do usuário. Ainda pode-se dizer que SAD são sistemas de informações computadorizados que fornecem apoio interativo de informação aos gerentes e profissionais de empresas durante o processo de tomada de decisão[20]. Para Laudon[2], SAD dá apoio a análise de problemas semi-estruturados e não estruturados.

Os SADs são projetados para serem um sistema de consulta *ad-hoc*. Bill Inmon conceitua consulta *ad-hoc* como consultas com acesso casual único e tratamento dos dados segundo parâmetros nunca antes utilizados, geralmente executado de forma iterativa e heurística. Isso tudo nada mais é do que o próprio usuário gerar consultas de acordo com suas necessidades de cruzar as informações de uma forma não vista e com métodos que o levem a descoberta daquilo que procura[21]. São de resposta rápida que são controlados por usuários finais. Assim, são capazes de dar apoio diretamente a todos os tipos específicos de decisões e também para os estilos e necessidades pessoais de tomada de decisão de cada gestos[20]. Dentro dessa visão, podemos dizer que SAD ajudam os gestores a usar, de melhor forma, seus conhecimentos para a tomada de decisão. Os benefícios gerados pelos sistemas de apoio à tomada de decisão na visão de Gordon[22] incluem:

- Um processo de tomada de decisão melhorado, através de um melhor entendimento do negócio;
- Exame de maior número de alternativas para uma decisão;
- A capacidade de implementar análises *ad hoc* ou aleatórias;
- Resposta mais rápida às situações previstas;
- Uma comunicação aprimorada;
- Trabalho de equipe mais eficaz;
- Melhor controle;
- Economia de tempo e de custos.

Componentes do SAD

É composta por um banco de dados, com dados usados para consulta e análise, um aplicativo com modelos, mineração de dados e outras ferramentas analíticas, e uma interface com o usuário.

O **banco de dados SAD** contém os dados, correntes ou históricos, provenientes de uma série de aplicações ou grupos. Pode ser desde um pequeno banco de dados em um pequeno servidor, até um grande data warehouse atualizado de forma contínua pelos principais sistemas organizacionais de processamento de transações. O dados dos banco de dados SAD geralmente são extratos ou cópias de banco de dados de produção[2]. Um banco de dados SAD fornece acesso aos dados internos ou externos relativos às decisões tomadas em períodos anteriores. Os dados de um banco de dados formam um comparativo básico, que é usado para calcular circunstâncias passadas a condições futuras[20].

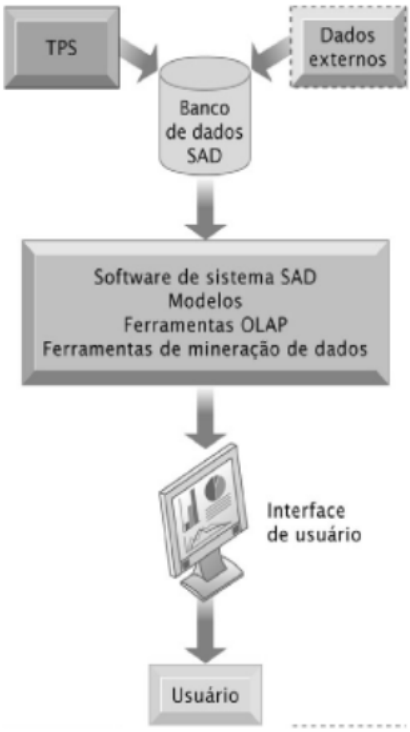

Figura 2.3: Visão geral de um sistema de apoio a decisão(SAD)[2].

O **sistema de software SAD** contém as ferramentas de software empregadas para análise de dados. Pode conter várias ferramentas OLAP, ferramentas de mineração de dados e analíticos que pode ser disponibilizado para o usuário que vai usar o sistema de software SAD[2]. **Modelo** entende-se como um componente do software que consiste em modelos utilizados em rotinas computacionais e analíticas que expressam matematicamente relações entre variáveis[20]. Como exemplo de SAD de referência, temos um sistema que ajuda os administradores de fundos a tomar a decisão sobre quais ações comprar, e certamente, inclui diversos modelos matemáticos que analisam múltiplos aspectos da compra potencial.

2.7 Data Warehouse

Segundo Sen[23], não há uma definição precisa sobre o que é e o que constitui um *Data Warehouse* (Armazém de dados). Então começamos com o conceito de Kimball[3], que afirma que um *Data Warehouse* é o lugar onde as pessoas podem acessar seus dados, e define como a fonte de dados de consulta do empreendimento, onde as pessoas podem acessar as informações que lhe são necessárias.

Inmon[21] diz que um *Data Warehouse* é um banco de dados que armazena dados sobre as operações da empresa (vendas, compras, etc) extraídos de uma fonte única ou múltipla e, transforma-os em informações úteis, oferecendo um enfoque histórico, para permitir um suporte efetivo à decisão. Uma filosofia de *Data Warehouse* pode prover múltiplas visões da informação para um espectro de usuários. O poder deste conceito é que possibilita aos usuários acesso a dados de fontes não relacionadas para a procura de respostas a questões de negócios, ou seja, o *Data Warehouse* permite que os usuários prevejam informações relevantes de dados antes independentes.

Sen[23] afirma que os *Data Warehouses* são construídos no interesse de suporte à decisão de negócios e contêm dados históricos sumarizados e consolidados provenientes de registros individuais de bancos de dados operacionais.

A informação é um bem valioso da empresa. Decisões precisam ser tomadas rápida e corretamente, usando todo a informação disponível. Os sistemas convencionais de informações são projetados para gerar e armazenar informações de operações transacionais, o que torna os dados vagos e sem valor para apoio ao processo de tomada de decisões das organizações. Para a tomada de decisão estratégica, os dados tem que ser, também, baseadas em fatos históricos que foram armazenados pelos diversos sistemas de informação utilizados pelas organizações. No *Data Warehouse* possuímos uma fonte única de informações com dados consolidados para a resposta de perguntas para a tomada de decisão estratégica[3][21].

Vantagens do uso de um Data Warehouse

Segundo Sen[23] e Inmon[21], há algumas vantagens no uso de um Data Warehouse:

- **Qualidade de dados:** consulta em dados de maior qualidade o que garante a consistência, precisão;
- **Acesso rápido:** Concede ao usuário, fazer uma consulta centralizada dos dados, recuperando os dados e eliminando o trabalho de busca desses dados em vários sistemas;
- **Facilidade de uso:** Com o uso de ferramentas específicas, há uma facilidade realização de consultas e recuperação de dados, trabalhando com interfaces gráficas e comandos predefinidos, o que torna a análise das informações armazenadas no *Data Warehouse* uma tarefa mais intuitiva para os usuários finais;
- **Separação das operações de decisão das operações de produção:** Como os dados do *Data Warehouse* ficam separados dos dados dos sistemas operacionais, mesmo sendo continuamente atualizados, os usuários finais podem fazer estudos

nestes dados sem interferir nos sistemas OLTP(*Online Transaction Processing* ou Processamento de Transações em Tempo Real), que são a fonte dos dados do *Data Warehouse*.

- **Valores quantitativos:** Mostra um retrospecto realista da evolução dos dados, com medidas quantitativas que permitem a comparação e a análise em períodos distintos.

- **Segurança:** Os usuários do *Data Warehouse* não acessam diretamente as base de dados dos sistemas OLTP, aumentando a segurança destas informações.

E um *Data Warehouse* possui as seguintes características[21][24]:

- **Orientado por assunto:** O *Data Warehouse* é montado por assunto, sejam eles referentes aos temas de maior interesse das organizações. A orientação por assunto é uma característica importante, pois toda a modelagem do DW é orientada a partir dos principais assuntos da Organização. Por exemplo uma empresa de arrecadação de impostos, seus principais assuntos são os cadastros de contribuintes, impostos a recolher, onde haverá a análise mais importante para o empresa.

- **Integrado:** As inconsistências são removidas em nomenclatura e conflito de informação, isto é os dados são limpos. Os dados fontes de sistemas OLTP são modificados e convertidos para um estado uniforme de modo a permitir a carga no *Data Warehouse*.

- **Histórico:** O *Data Warehouse* geralmente armazenam informações por um período válido de tempo de 5 a 10 anos, enquanto os sistemas operacionais armazenam dados históricos limitados a um horizonte de tempo de 60 a 90 dias[21].

  Os dados operacionais devem sempre apresentar valores atualizados e precisos no momento em que forem acessados pelo usuários. Já o *Data Warehouse* armazena uma grande quantidade de dados históricos formados por um conjunto definido no tempo, sendo os dados capturados em momento.

  Em um DW, haverá uma dimensão cuja estrutura registrará especificamente o elemento tempo, que identificará o tempo exato do registro ou fato acontecido, de acordo com a granularidade do DW. Nos sistemas OLTP, essa é uma característica opcional.

- **Não Volátil:** Em um Data Warehouse, os dados depois de serem integrados, são carregados e armazenados no banco de dados, ficando disponíveis para os usuários realizarem apenas consultas e geração de relatórios que permitam a tomada de

decisão. Isso mostra uma característica de somente leitura para os usuários finais de um banco de dados de um DW.

A especificação de um ambiente de suporte a decisão, que é vinculado a um DW, é diferente da especificação de um ambiente operacional de uma empresa. Claramente, os sistemas do ambiente operacional são especificados para executarem funções da organização, como realizar um cadastro, ou uma venda[21]. Esses sistemas que apoiam os usuários em suas funções do dia a dia são chamados OLTP (*On Line Transaction Processing*), e seu objetivo é executar o maior número de transações e com o melhor desempenho possível.

Em geral, os sistemas OLTP são limitados em relação a quantidade de relatórios e consultas, devido as restrições imposta por seu modelo de dados. Para suprir essas restrições, são adotados os sistemas OLAP (*On Line Analytical Processing*), que permitem que usuários de alto nível, como gerentes e analistas de negócio, tenha uma maior facilidade de acesso aos dados da organização, proporcionando uma visão multidimensional desses dados[21].

2.7.1 Modelagem Dimensional

Modelagem dimensional é a técnica de projeto lógico que permitia tornar os bancos de dados fáceis e compreensíveis, e é usada para um *Data Warehouse*, cujo o principal objetivo é apresentar os dados em uma arquitetura padrão e intuitiva, que permite acesso de alto performance[3].

O modelo dimensional surgiu para atender sistemas de processamento analíticos, com consultas para planejamento tático e estratégico de uma organização, que atende um pequeno número de usuários que realizam consultas de relatórios pré-definidos e consultas *ad-hoc*.

A composição do modelo dimensional segue por dois tipos de tabelas [3][23]:

- **Fato:** Uma tabela fato usualmente é uma coisa sobre a qual não se conhece antecipadamente, é uma observação da realidade[3]. As tabelas fato são centrais e, geralmente, armazenam grande quantidade de dados (de gigabytes a terabytes), dependendo diretamente da granularidade adotada. Possuem chaves primárias compostas e contêm as medições numéricas do negócio denominados fatos [23].

  Os melhores fatos e os mais úteis são numéricos e caracterizam-se por serem continuamente valorados (diferente a cada medida) e aditivos (os valores modificam-se a cada combinação de atributos das tabelas dimensão[3].

- **Dimensão:** É uma tabela que alimenta a tabela fato e seus componentes descrevem as características de uma coisa tangível. Normalmente possuem uma chave primária

simples e campos denominados atributos, e armazenam um quantidade pequena de dados, em relação a tabela fato, e contêm os dados descritivos do negócio[3].

A cada chave primária da tabela dimensão corresponderá exatamente a uma chave estrangeira na tabela fato, permitindo a ligação entre ambas.

2.7.2 Esquema Estrela versus Cubo OLAP

Modelos dimensionais implementados em sistemas de banco de dados relacionais, são referidos como esquema estrela (*star schema*) por causa da sua semelhança com uma estrutura de uma estrela. Modelos dimensionais implementados em ambientes de banco de dados multidimensional são referidos como cubos OLAP[3], como ilustrado na Figura 2.4.

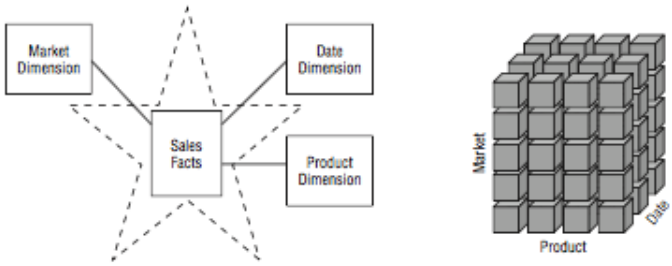

Figura 2.4: Visão de um Start schema versus cubo OLAP[3].

Quando os dados estão carregados em um cubo OLAP, está armazenado e indexado usando formatos e técnicas e formatos que são designados para dados dimensionais. Sistemas OLAP fornecem uma visão multidimensional dos dados, os dados são vistos pelo usuário como um cubo multidimensional onde cada célula contém um valor ou uma medida[21], tendo como um exemplo de uma venda de um mercado, ilustrado na Figura 2.4.

E segundo Kimball[3], as principais funções oferecidas pelos sistemas OLAP são:

Drill-down: Para o usuário ter uma visão mais detalhada do conjunto de dados. Por exemplo quando se está vendo a situação de vendar de uma rede de supermercados por uma Região brasileira, e passa a ver por um Estado brasileiro, aumentando a granularidade;

Roll-up: Consiste na operação inversa do *drill-down*, ou seja, apresenta os dados cada vez mais agrupados, diminuindo a granularidade;

Pivoting: Para o usuário adicionar ou rearranjar as dimensões das tabelas. Por exemplo quando se enxerga os dados do número de vendas da rede de supermercado pela data, e passa a ver de acordo com a região (lugar) da venda.

Slide-dice: É a função que fixa uma informação de dimensão do cubo, reduzindo dimensões de apresentação de dados. Por exemplo, quando se enxerga a informação por região, data e produto, e passa a ver essa informação somente por região e data.

2.7.3 Processo de ETL

O processo de extração, transformação e carga (em inglês, *extaction, transformation, and load* - ETL), no ambiente DW/BI, consiste de uma área de trabalho, com estruturas de dados instanciadas e um conjunto de processos. O processo de ETL é tudo entre os sistemas operacionais fontes e o DW/BI[3].

Extração é o primeiro passo no processo de obtenção dos dados dentro do ambiente do DW. **Extração** significa ler e entender as fontes de dados e copiar todos os dados necessários para o sistema de ETL para uma posterior manipulação[3].

Depois do ponto de extração, há varias **transformações** em potencial (limpar, eliminar, combinar, padronizar, validar, consolidar, agregar e sumarizar), combinando dados de múltiplas fontes, e retirando os dados duplicados. O sistema ETL adiciona valor aos dados com essas tarefas de limpeza e conformidade, alterando os dados e aprimorando-o[3].

O passo final do processo ETL é a estruturação física e o **carregamento** de dados nos modelos dimensionais alvo da área de apresentação. Como a missão principal do sistema ETL é entregar as tabelas de dimensão e fato na etapa final do processo, esses subsistemas de **carregamento** são críticos.

2.8 Implementação e Ciclo de Vida DW/BI Kimball

A abordagem de Ciclo de vida do Kimball de construção de um *Data Warehouse* é ilustrado na Figura 2.5, onde é mostrado o diagrama de sequência, concorrência e dependência de tarefas para a construção de um DW/BI[3].

2.8.1 Planejamento do Projeto

De acordo com Kimball[3], o ciclo de vida inicia com o planejamento do projeto. Nesta fase, trata da definição e do escopo do projeto do DW, incluindo uma avaliação do investimento e suas justificativas. Especificam-se os recursos, a capacidade necessária no preenchimento das vagas do projeto, juntamente com as atribuições, duração e sequenciamento. No planejamento, identifica-se todo o trabalho associado e as partes envolvidas ao ciclo de negócios.

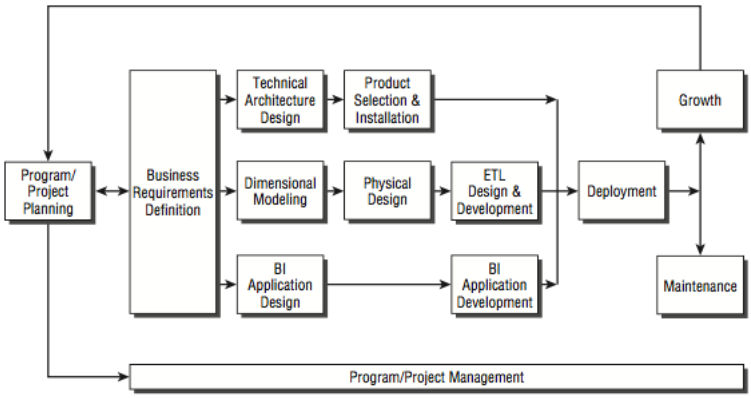

Figura 2.5: Diagrama de ciclo de vida Kimball[3].

2.8.2 Definição dos Requisitos de Negócio

O sucesso de um DW está diretamente ligado ao entendimento das necessidades e exigências dos usuários finais. Os projetistas devem entender os fatores-chave que conduzem o negócio, para determinar, efetivamente, os requisitos necessários de negócio, considerando-os dentro do projeto do DW.

2.8.3 Modelagem Dimensional

Projetar modelos de dados para suportar as análises de negócios requer abordagens diferentes das usadas para o projeto de bancos de dados operacionais relacionais. Deve-se iniciar com a construção de uma matriz, que representa os processos-chave de negócios e suas dimensionalidades. A partir desta matriz, faz-se um detalhamento da análise de dados relevantes, juntando com os requisitos de negócios levantados anteriormente, e desenvolve-se um modelo dimensional. Este modelo identifica a granularidade das tabelas, as dimensões associadas, atributos e fatos.

2.8.4 Projeto Físico e Configuração do Ambiente

O projeto físico tem seu foco na definição das estruturas físicas necessárias para suportar o seu projeto lógico. Definição de padronização de nomes, meio-físico, índices preliminares e estratégias de particionamento são determinadas nesta fase.

2.8.5 Desenvolvimento e Projeto do ETL

O desenvolvimento e projeto do ETL é a fase da especificação e documentação dos três passos principais, que são extração, transformação e carga. São necessários o projeto e

desenvolvimento de duas plataformas: uma para a população inicial do DW e outra para as cargas regulares e incrementais.

2.8.6 Projeto e Arquitetura Técnica

O ambiente de DW requer uma integração de inúmeras tecnologias. Para isso, devem-se considerar três fatores, que são os requisitos de negócios, o atual ambiente físico e o planejamento estratégico técnico definido pela empresa, para estabelecer o projeto de arquitetura técnica do DW.

2.8.7 Instalação e Seleção de Produtos

Nesta fase, são especificados, avaliados e selecionados os componentes de arquitetura, como a plataforma de hardware, sistema de administração de banco de dados, ferramentas de transição e de acesso aos dados. Após, os produtos são instalados e testados, para garantir uma perfeita integração com o ambiente do DW.

2.8.8 Projeto de Aplicação BI

As especificações definem os modelos de relatórios, os parâmetros operados pelos usuários, e os cálculos necessários. Essas especificações asseguram que o grupo de desenvolvimento e os usuários de negócios tenham um entendimento comum sobre as aplicações que serão entregues.

2.8.9 Implementação da Aplicação BI

O desenvolvimento de aplicações para usuários finais inclui ferramentas de configuração de metadados, e a construção de ferramentas para geração de relatórios.

2.8.10 Implantação

A implantação representa a convergência entre tecnologia, dados, e aplicações de acesso na área de trabalho. Um planejamento é exigido para assegurar que tudo seja ajustado corretamente. Deve ser estabelecido apoio ao usuário e estratégias de realimentação antes que qualquer usuário tenha acesso ao *Data Warehouse*.

2.8.11 Manutenção e crescimento

Devem ser estabelecidos processos de prioridades para tratar esta demanda de modificações, para se obter uma evolução e crescimento. Depois que as prioridades do projeto são

identificadas, volta-se ao início do ciclo de vida, para melhor o que já foi estabelecido no ambiente do DW, com o foco nas novas necessidades.

2.8.12 Gerencia do projeto

A gerencia do projeto assegura que as atividades do ciclo de vida dimensional permaneçam sincronizadas. Essas atividades focam no monitoramento do estado do projeto, rastreamento dos resultados e controle de alterações.

2.9 Sistema Gerenciador de Banco de Dados

Um banco de dados é uma coleção de dados persistentes utilizada pelos sistemas de aplicação de uma determinada organização[24]. Em outras palavras, pode se dizer que banco de dados é um local onde são armazenados dados necessários das atividades de uma organização, sendo este repositório a fonte de dados para as aplicações atuais e as que vierem a existir.

Para Elmasri e Nathavi[25], um banco de dados é uma representação de um "minimundo", ou universo de discurso, e cada mudança nele é refletida no banco de dados. Uma coleção aleatória de dados não pode ser chamada corretamente de banco de dados. Um banco de dados é projetado, construído e populado com dados para uma finalidade específica. Assim um banco de dados é um conjunto organizado de dados relacionados, criado com determinado objetivo e que atende uma comunidade de usuários.

De acordo com Date[24], o banco de dados pode ser visto como o equivalente eletrônico de um armário de arquivamento. Então como para o armário temos o gerenciamento manual, e para sistemas computacionais, temo um sistema gerenciador de banco de dados (SGBD). Então um SGBD, é um *software* que permite aos usuários criar e manter um banco de dados, de forma em que é facilitado os processos de definição, que implica especificar os tipos de dados e as estruturas e as restrições para os dados armazenados, de construção, que é o processo de armazenar os dados em algum lugar apropriado controlado pelo SGBD, de manipulação, que inclui algumas funções como consulta a dados armazenado, atualização do banco de dados para refletir as mudanças no minimundo e gerar os relatórios de dados, e de compartilhamento, que permite múltiplos usuários e programas acessar o banco de forma concorrente, de bancos de dados entre vários usuários e aplicações[25].

Entre os motivos de usar um SGBD, de acordo com Date[24], um SGBD proporciona um controle centralizado de seus dados. E são considerada algumas vantagens no seu uso[25][24]:

1. Controle de redundância
2. Maior impedimento de inconsistência,
3. Os dados podem ser compartilhados,
4. Padronização de dados podem ser reforçados, para comunicação com outros aplicações,
5. Podem ser aplicadas restrições de segurança, para controle de acesso,
6. Manutenção de integridade,
7. Necessidades conflitantes podem ser balanceadas, usando mais recursos onde há maior necessidade,
8. Backup e restauração de banco de dados,
9. Representação de relacionamento complexos entre dados,
10. Armazenamento de estruturas para o processamento eficiente de consultas.

2.10 Banco de Dados Relacional

Um banco de dados relacional é um banco de dados que usa uma estrutura de dados baseado em um modelo relacional. O modelo relacional foi desenhado pelo cientista de pesquisa e matemático da IBM, o Dr. E.F.Codd[26], quando publicou um documento de referência[27] que defini o modelo relacional e formas não processuais de consulta de dados no modelo relacional, e assim os bancos de dados relacionais nasceram.

2.10.1 Estrutura de um Modelo Relacional

Um banco de dados relacional consistem em uma coleção de tabelas, no qual cada tabela tem um nome único. De acordo com o artigo[27], foi incluído os seguintes conceitos que se aplicam aos sistemas de gerenciamento de banco de dados para bancos de dados relacionais.

A relação é a única estrutura de dados utilizada no modelo de dados relacionais para representar tanto as entidades quanto as relações entre elas. Linhas da relação são referidas como tuplas da relação e as colunas são seus atributos. Cada atributo da coluna é extraído do conjunto de valores conhecido como domínio. O domínio de um atributo contém o conjunto de valores que o atributo pode assumir[28].

Na Figura 2.6, temos uma tabela, de exemplo, que tem quatro colunas de cabeçalho: *ID, name, dept_name* e *salary.* Cada linha da tabela então é uma informação a respeito

ID	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

Figura 2.6: Tabela de Instrutores[4]

de um instrutor, composta pelo ID, nome, nome do departamento, e cada instrutor é identificado pelo valor da coluna ID. Assim, no modelo relacional, o termo relação é usada para referenciar uma tabela, enquanto o termo tupla é usado para referenciar uma linha e o termo atributo, se refere a uma coluna da tabela. Para cada atributo de uma relação, há um conjunto de valores permitidos, o que é chamado de domínio do atributo[4].

2.10.2 Chave

O relacionamento entre tuplas de relações é formado por um atributo, que chamamos de chave. Devemos ter uma maneira de especificar como as tuplas dentro de uma dada relação são distinguidas. Isso é expresso em termos de seus atributos, ou seja, o valor de um atributo de uma tupla, tem que ser um valor unicamente identificável, em outras palavras, duas tuplas não podem ter o mesmo valor para seus atributos[4], e esse atributo ou conjunto de atributos pode ser dito ser a chave primária[24].

Chaves primárias tem que ser escolhidas com cuidado.Para exemplificar, na Figura 2.6, podemos notar que o nome de uma pessoa não é o suficiente, pois pode haver duas ou mais pessoas com o mesmo nome, então a chave primária candidata da relação instrutor é o atributo ID, que é um código gerado e no qual é unicamente identificável.

Uma relação, supondo r1, pode incluir entre seus atributos a chave primária de um outra suposta relação r2. Esse atributo é chamado de chave estrangeira de r1, referenciando r2.

2.10.3 Restrições de atributo, chave e integridade referencial

Podemos especificar restrições de atributo, chave e integridade referencial como parte da criação de relações que normalmente é feito com acréscimo de algumas cláusulas[25]:

- ***not null***: usada para especificar atributos no qual o valor não pode ser nulo;
- ***default***: define um valor padrão para dado atributo. Deste modo, ao se criar uma nova linha na tabela, esse valor será automaticamente atribuído a coluna caso nenhum valor seja explicitamente informado. Se a cláusula *default* for omitida, o valor padrão é *null* para aqueles atributos que não contenham a restrição *not null*;
- ***check***: empregada para limitar os valores de um atributo. Assim, toda linha da tabela deve satisfazer a condição especificada na cláusula;
- ***primary key***: usada para especificar a chave primária de uma relação;
- ***unique***: aplicada na especificação da chave única de uma relação;
- ***foreign key***: utilizada para especificar a chave estrangeira de uma relação, e com essa cláusula também impõe integridade referencial ao banco de dados;
- ***constraint***: opcionalmente utilizada para nomear uma restrição. Quando uma restrição é nomeada, ela fica passível a exclusão ou substituição, caso necessário.

2.10.4 Forma Normal

As formas normais foram propostas por Codd[27] em 1970, como uma sequência para alcançar o estado desejável de relações que chegam na 3º Forma Normal, passando pelos estados intermediários da 1º Forma Normal e 2º Forma Normal. Como veremos, a 2º Forma Normal e a 3º Forma Normal abordam problemas diferentes. Porém, por razões históricas, é habito seguir essa sequência, é presumido que quando uma relação está na 3º Forma normal, já satisfez a 2º e a 1º Forma Normal[25].

As três propostas de Forma Normal de Codd[27][25]:

- **1º Forma Normal:** É considerada parte da definição formal de uma relação no modelo relacional básico. Estabelece-se que o domínio de um atributo só deva incluir os valores atômicos (simples, indivisíveis), e que o valor de qualquer atributo em uma tupla deve ter um único valor no domínio daquele atributo.
- **2º Forma Normal:** É baseada no conceito de dependência funcional total. Uma relação está na 2º se, e somente se, estiver na 1º e cada atributo não-chave for

dependente da chave primária inteira, isto é, cada atributo não-chave não poderá ser dependente de apenas parte da chave.

- **3º Forma Normal:** Está baseada no conceito de dependência transitiva. Uma relação R está na 3º se ela estiver na 2º e cada atributo não-chave de R não possuir dependência transitiva, para cada chave candidata de R. Todos os atributos dessa tabela devem ser independentes uns dos outros, ao mesmo tempo que devem ser dependentes exclusivamente da chave primária da tabela.

2.10.5 Transação

Uma transação é uma coleção de atividades que envolvem mudanças na base de dados, todas as quais devem ser executadas com êxito se as alterações devem ser persistidas no banco de dados e nenhuma delas pode ser comprometida se uma ou mais das atividades falharem. Podem ser consideradas as transações do tipo *write*(escrita no banco) e *read*(leitura no banco). Normalmente, tal conjunto de atividades é representado por uma sequencia de comandos relacionais[26]. Ainda de acordo com Codd, o início da sequencia é sinalizado por um comando como *BEGIN* ou *BEGIN TRANSACTION*. Sua terminação é sinalizada por um comando como *END* ou *COMMIT* ou, se for necessário abortar a transação, *ABORT*.

As transações devem possuir algumas propriedades, chamadas propriedades ACID, e elas devem ser impostas pelo controle de concorrência e métodos de restauração do SGBD. As propriedades ACID são as seguintes[4][25]:

- **Atomicidade**: Uma transação é uma unidade atômica de processamento, sendo ela executada em sua totalidade ou não é executada de forma alguma. Em exemplo de uma transação bancária, Suponha que, imediatamente antes da execução de uma transação T, os valores das contas A e B sejam $ 1000 e $ 2000, respectivamente. Suponha agora que, durante a execução da transação T, ocorra uma falha que impeça T de completar sua execução com êxito. Além disso, suponha que a falha ocorreu após a operação *write* (A) mas antes da operação *write* (B). Nesse caso, os valores das contas A e B refletidas no banco de dados são $ 950 e $ 2000. O sistema destruiu $ 50 como resultado desta falha. Em particular, observamos que a soma A + B não é mais preservada.

- **Consistência**: O requisito de consistência aqui é que a soma de A e B não sejam alteradas pela execução da transação. Sem o requisito de consistência, em um exemplo de uma transação bancária, dinheiro poderia ser criado ou destruído pela

transação. Pode-se verificar facilmente testando que, se o banco de dados é consistente antes da execução da transação, o banco de dados permanece consistente após a execução da transação.

- **Isolamento**: Uma transação será ser executada isoladamente das demais transações, ou seja, a execução de uma transação não deve sofrer interferência de quaisquer outras transações concorrentes. Por exemplo, como vimos anteriormente, o banco de dados é temporariamente inconsistente enquanto a transação para transferir fundos de A para B está sendo executada, com o total deduzido escrito para A e o total aumentado ainda a ser gravado em B. Se uma segunda transação em execução concorrente lê A e B neste ponto intermédio e calcula A + B, irá observar um valor inconsistente. Além disso, se essa segunda transação executa atualizações em A e B com base nos valores inconsistentes que ele ler, o banco de dados pode ser deixado em um estado inconsistente mesmo após ambas as transações concluídas.

- **Durabilidade**: A propriedade de durabilidade garante que, uma vez que uma transação foi concluída com êxito, todas as atualizações que ele realizou no banco de dados persistem, mesmo se houver uma falha do sistema após a conclusão da transação. No caso de a execução da transação ter sido efetuada com êxito e o utilizador que iniciou a transação tiver sido notificado de que a transferência de fundos foi efetuada, nenhuma falha do sistema possa resultar numa perda de dados correspondentes a esta transação.

2.11 Banco de Dados NoSQL

Banco de dados NoSQL, também conhecido como banco de dados *Not only* SQL ou não-relacional, foi especificamente introduzido para lidar com o aumento dos tipos de dados, acesso a dados e necessidades de disponibilidade de dados provocados pelo boom do acesso a internet[29].

No entanto, quando as aplicações de internet e as empresas começaram a explodir durante o final dos anos 90 e início dos anos 2000, as aplicações passaram de servir milhares de funcionários internos dentro das empresas para ter milhões de usuários na internet pública. Para essas aplicações, desempenho e disponibilidade foram primordiais. O novo problema de alta disponibilidade em grande escala levou empresas como Google, Facebook e Amazon a criar novas tecnologias. Felizmente, eles documentaram seus esforços, e

publicaram *white papers*[2], e sua tecnologia *open source* para a comunidade da internet continuar a evolução dessas tecnologias. No final dos anos 2000, surgiram várias novas tecnologias de banco de dados não relacional e NoSQL foi o nome que descreveu todas elas[29].

2.11.1 Tipos de bancos de dados NoSQL

NoSQL é simplesmente o termo usado para descrever uma família de bancos de dados que são todos não relacionais. Embora as tecnologias, tipos de dados e casos de uso variam entre eles, geralmente existem quatro tipos de bancos de dados NoSQL[30][29]:

- **Banco de dados chave-valor**: Esses bancos de dados vinculam chaves a valores, como uma tabela de *hash* na programação de computadores. Uma boa analogia para um armazenamento de valor-chave é um sistema de arquivos. O caminho atua como a chave e o conteúdo age como o arquivo. Muitas vezes não há campos para atualizar, em vez disso, o valor inteiro diferente da chave deve ser atualizado se as alterações devem ser feitas. Essa simplicidade se equilibra bem, mas no entanto, pode limitar a complexidade das consultas e outros recursos avançados disponíveis em armazenamentos de valor-chave. Exemplos de armazenamentos de valor-chave puro incluem Amazon Dynamo, MemcacheD, REDIS e Riak.

- **Banco de dados orientado a coluna**: A ideia básica por trás de bancos de dados orientados a colunas é que os dados não são um conjunto de dados com seus atributos armazenados em uma unidade como em bancos de dados SQL(orientados por linhas), mas um atributo de um conjunto de conjuntos de dados é armazenado em uma unidade. Bases de dados relacionais armazenam todos os dados na linha de uma tabela particular juntos no disco, o que torna a recuperação de uma linha particular rápida. As bases de dados da família de colunas geralmente serializam todos os valores de uma determinada coluna em conjunto no disco, o que torna a recuperação de dados agregados em um atributo específico rapidamente. Isto é valioso em cenários de *data warehousing* e análise onde você pode executar consultas de intervalo em um valor específico. Exemplos de armazenamento de colunas incluem Hbase e Cassandra.

- **Banco de dados orientado a grafo**: As bases de dados orientadas por grafos originaram-se da teoria do grafos, onde você tem vértices e arestas que conectam os

[2]White papers: é um documento oficial publicado por um governo ou uma organização internacional, a fim de servir de informe ou guia sobre algum problema e como enfrentá-lo.

vértices. Em bancos de dados, os vértices são entidades como pessoas e as arestas são relações entre entidades. Essas relações são semelhantes às relações no RDBMS. O Twitter usa o FlockDB, o banco de dados orientado a grafos. Exemplos desses bancos de dados incluem Neo4j e Sesame.

- **Banco de dados orientado a documento**: Os bancos de dados orientados a documentos usam documentos de diferentes tipos como conjuntos de dados. Os tipos de documento são estruturados formato de dados legível humanos como XML ou JSON. Bases de dados orientadas a documentos podem ser interpretadas como casos particulares de uma base de dados de chave-valor. O banco de dados conhece o formato do documento e o interpreta, e esta é a maior diferença para o armazenamento de chave-valor. Os bancos de dados de armazenamento de documentos estão livres de esquema, e em um esquema de banco de dados livre você não tem que predefinir como os dados estão organizados. Para o armazenamento, isso não parece ser um problema, porque os documentos não dependem uns dos outros. Quando os documentos têm algum tipo de estrutura ou esquema é chamado semi-estruturado. Exemplos desses bancos incluem CouchDB, MongoDB e RavenDB.

2.11.2 Características de um banco de dados NoSQL

Os vários bancos de dados NoSQL disponíveis hoje em dia são um pouco diferentes uns dos outros, mas existem ligações comuns que os unem: flexibilidade, escalabilidade, disponibilidade, operações de baixo custo e capacidades especiais[29][31].

- **Flexibilidade**: A flexibilidade de esquema e estruturas de dados intuitivas são características chave que atraem desenvolvedores que trabalham em ciclos de desenvolvimento ágeis, e a maioria dos bancos de dados NoSQL usam essa flexibilidade. Atualizações de esquema e seu tempo de inatividade associado geralmente não são necessários, ao contrário em um banco de dados relacional. A flexibilidade do NoSQL é popular por suportar práticas de desenvolvimento ágil, eliminando a complexidade das alterações no esquema do banco de dados para suportar mudanças nas bases de código.

- **Escalabilidade**:Escala refere-se tanto ao tamanho dos dados como aos usuários simultâneos que atuam sobre esses dados. Bases de dados NoSQL são tipicamente mais especializados para vários casos de uso envolvendo escala e pode ser muito

mais simples para o desenvolvimento de funções de aplicativos relacionados do que bancos de dados relacionais. Muitos bancos de dados NoSQL são construídos para escalar horizontalmente e espalhar seus dados através de um *cluster* de servidores mais facilmente do que suas contrapartes relacional. O desempenho do banco de dados relacional perde desempenho quando as consultas executam JOINs(junções) entre relações, enquanto que um banco de dados NoSQL pode evitar JOINs completamente e manter alto desempenho.

- **Disponibilidade**:O *downtime* (tempo de inatividade) do banco de dados resulta em perda de receita, perda de clientes e usuários frustrados. Uma característica, de alguns bancos de dados NoSQL possuem novas ou diferentes arquiteturas de replicação que podem tornar diferentes tipos de bancos de dados NoSQL mais disponíveis para escritas, além de leituras. Isso significa que, se um ou mais servidores de banco de dados, ou *nós*, tiver uma queda de servidor, os outros nós do sistema poderão continuar as operações sem perda de dados.

- **Operações de baixo custo**: As raízes dos bancos de dados NoSQL é *open source*, o que tornou um ponto de entrada a um incentivo de baixo custo. É comum ouvir dos usuários de bancos NoSQL, que os custos são reduzidos significativamente em relação aos bancos de dados existentes, enquanto continuam recebendo o mesmo ou melhor desempenho e funcionalidade. Historicamente, grandes bancos de dados relacionais funcionam em máquinas caras e mainframes. A natureza de sistemas distribuídos dos bancos de dados NoSQL significa que eles podem ser implantados e operados em *clusters* de servidores de baixo custo ou em arquiteturas em nuvem.

- **Capacidades especiais**: Nem todos os bancos de dados NoSQL tem funções especificas iguais, podendo cada tipo de banco NoSQL usar suas características para melhorar determinada função. Para incentivar e adicionar integração aos seus usuários, muitos bancos NoSQL incluem vários recursos especializados. Exemplos incluem recursos específicos de indexação e consulta para dados geoespaciais, indexação de texto completo integrada para pesquisa, replicação automática de dados e recursos de sincronização e APIs e *frameworks* compatíveis com aplicativos, o que facilita o trabalho dos desenvolvedores.

Atualmente, o banco de dados que você escolher para sua próxima aplicação importa agora mais do que nunca. Felizmente, você não tem que escolher apenas um. Espera-se

que as aplicações de hoje funcionem sem parar e que, de forma contínua, possam gerir quantidades crescentes de dados multi-estruturados de forma contínua. Isso tem causado NoSQL para crescer a partir de uma promessa para uma consideração séria para cada banco de dados, de pequenas lojas até grandes empresas[31].

Embora os bancos de dados NoSQL compartilhem algumas qualidades comuns, tais como ser não-relacionais e geralmente fáceis de escalonar, existem muitos identificadores exclusivos a considerar ao decidir sobre a solução correta do NoSQL para suas necessidades de dados exclusivas. O banco de dados NoSQL correto pode atuar como uma alternativa viável aos bancos de dados relacionais ou pode ser utilizado de forma complementar, juntamente com os sistemas existentes[31][32].

O requisito para a entrega de dados e eficiente é o grande objetivo, deve-se considerar a tecnologia NoSQL ao planejar qualquer projeto de desenvolvimento de aplicativos que envolva escala, grande volume de dados, alto demanda por processamento de dados, variedades diferentes de dados ou grandes quantidades de potenciais usuários[29].

2.12 Banco de Dados Cassandra

O Apache Cassandra é um banco de dados NoSQL de código aberto amplamente escalável. O Cassandra foi projetado para gerenciar grandes quantidades de dados estruturados, semi-estruturados e não estruturados em vários data centers e na nuvem. Este banco de dados oferece disponibilidade contínua, escalabilidade linear e simplicidade operacional em muitos servidores de *commodities* sem um único ponto de falha, juntamente com um poderoso modelo de dados dinâmico projetado para máxima flexibilidade e tempos de resposta rápidos[33].

2.12.1 Cluster Cassandra

Um *cluster* Cassandra é um conjunto de nós que se comunicam via protocolo *peer-to-peer*, onde todos os nós tem o mesmo peso. Segue a terminologia comum utilizada para o *cluster* Cassandra[6]:

- **Nó:** Um servidor Cassandra entre vários servidores que compõe um *cluster*;
- **Rack:** Um conjunto lógico de nós;
- **Data Center:** Um conjunto logico de *racks*;
- **Cluster:** O conjunto completo de nós que mapeiam para um único anel completo.

A arquitetura do banco de dados Cassandra provê alta disponibilidade, computação distribuída e satisfaz requisitos de recuperação desastrosa. Um *cluster* Cassandra, se

projetado corretamente, pode lidar com falhas de nó, falhas de *rack* e até falhas de *data centers*[6].

A arquitetura do Cassandra permite que qualquer usuário autorizado se conecte a qualquer nó em qualquer data center e utilize a linguagem CQL para a consulta dos dados. Para facilidade de uso, o CQL utiliza uma sintaxe semelhante ao SQL com algumas particularidades, onde não permite *joins* ou filtragem de dados que não contenha a *partition key* e/ou a *clustering key*.

2.12.2 Particionamento

No Cassandra, o particionamento dos dados visa implementar a escalabilidade, o balanceamento de dados. Uma das principais características de design para o banco Cassandra é a capacidade de escalar de forma incremental. Isso requer, a capacidade de particionar dinamicamente os dados através do conjunto de nós no *cluster*. O Cassandra particiona dados através do *cluster* usando hash consistente da chave chamada *partition key* além de uma *clustering key* que ordena os dados no determinado nó[11]. No *hash* consistente, o intervalo de saída de uma função *hash* é tratado como um espaço circular fixo ou anél (ou seja, o maior valor de *hash* envolve o menor valor de *hash*). Cada nó no sistema é atribuído a um valor aleatório dentro desse espaço que representa sua posição no anel. Assim, cada nó torna-se responsável pela região no anel entre ele e seu nó predecessor no anel, como mostrado na Figura 2.7.

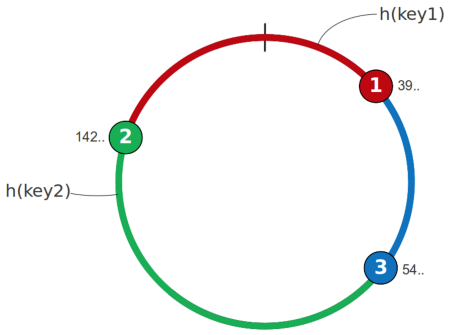

Figura 2.7: *Hashing* de particionamento[5].

2.12.3 Coordenador

Cada registro identificado por uma chave é atribuído a um nó, fazendo um *hash* da *partition key* do registro para produzir sua posição no anel e, em seguida, caminhar o anel no sentido horário para encontrar o primeiro nó com uma posição maior do que a posição do registro. Este nó é considerado o coordenador dessa chave.

O coordenador é um nó no *cluster* que recebe uma requisição de leitura/escrita do cliente e coordena com outros nós para executar a operação, fazendo a leitura de onde os dados estiverem distribuídos.

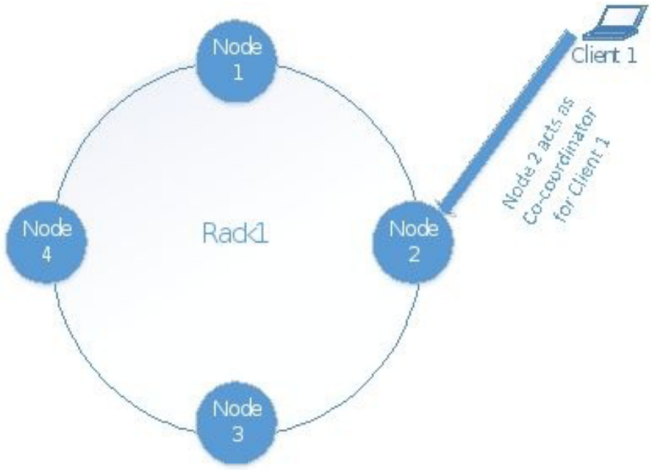

Figura 2.8: Requisição de cliente no Casssandra[6].

Se observarmos na Figura 2.8, vemos que o nó 2 age como o coordenador do *cluster* para o Cliente 1.

2.12.4 Keyspaces

Um *cluster* é um contêiner para *keyspaces*. Uma *keyspace* é o recipiente mais externo para os dados no Cassandra, correspondente a uma base de dados em um banco relacional[11]. Como em um banco de dados relacional, uma *keyspace* tem um nome e um conjunto de atributos que definem o seu comportamento.

No Cassandra, os atributos básicos que podem definir a *keyspaces* são[6][11]:

- **Fator de Replicação**

O Fator de Replicação define em quantos nós cada escrita deve ser escrita, o que faz o *cluster* Cassandra não ter um ponto único de falha. O fator de replicação é definido no nível da *keyspace*, exatamente quando é criada e pode ser alterada depois usando um comando de alteração da *keyspace*.

- **Estratégia de Replicação**

A Estratégia de Replicação, é a forma como as réplicas serão colocadas no Cluster. Existem duas estratégias disponíveis no Cassandra[6]:

1. ***SimpleStrategy*:** Onde há somente um Data Center. Se o *cluster* Cassandra está em múltiplos data centers então a estratégia *NetworkTopologyStrategy* deve ser usada;

2. ***NetworkTopologyStrategy*:** É uma estratégia mais flexível pois o *cluster* Cassandra pode abranger vários data centers. Por exemplo, o *cluster* Cassandra pode ser espalhado pelo Brasil, Europa e Estados Unidos. Os usuários em cada território pode acessar o *cluster* Cassandra pelo data center mais próximo a ele, além de ajudar no caso de falha no data center.

• **Família de coluna**

No mesmo caminho em que um banco de dados é um contêiner para tabelas, uma *keyspace* é um contêiner para a lista de um ou mais família de colunas. Uma família de colunas é aproximadamente uma tabela fazendo uma analogia ao modelo relacional, e é um contêiner de registros. Cada registro contem colunas ordenadas e representa a estrutura de seus dados.

2.12.5 Nível de consistência

O nível de consistência determina quantos nós devem reconhecer as requisições de leitura/escrita, o que é feito a nível de consulta.

A maioria dos banco de dados NoSQL são construídos na "consistência eventual", o que significa que os dados podem ser consistente em todos os nós do *cluster*, mas talvez pode não ser consistente em um certo ponto particular do tempo.

Cassandra suporta consistência ajustável, e isso é implementado usando um parâmetro de nível de consistência para consultas de leitura e escrita. A seguir há os níveis de consistência disponíveis no Cassandra, da versão 2.2 em diante[6]:

1. ***All*:** As leituras e escritas tem que ser feitas em todos os nós réplicas, para a operação ter sucesso. É o nível mais alto de consistência;

2. ***Quorum*:** As leituras e escritas tem que ser feitas em um número Quorum de nós réplicas. O número pode ser calculado pela fórmula Quorum = int(Fator de replicação / 2) + 1;

3. ***Local_Quorum*:** Usado quando a estratégia de replicação foi definida para *NetworkTopologyStrategy* e o *cluster* abrange múltiplos data centers. É como se fosse o *Quorum* em cada *data center*.

4. ***ONE*:** É o nível de consistência mais fraco. As leituras e escritas devem ser feitas apenas em um nó do conjunto de nós réplicas;

5. **N (Dois, Três...):** N deve ser menor ou igual o fator de replicação. É similar ao *ONE*, e significa que as leituras e escritas deve ser feitas no número N de nós do conjunto de nós replicas para a operação ter sucesso;

6. ***ANY*:** Provê a menor latência para a escrita. Se todos os nós com o fato de replicação definido, então as mudanças podem ser escrita para qualquer nó do *cluster*. As mudanças serão então propagadas para o conjunto de nós réplicas quando eles se tornarem disponíveis, e este nível de consistência é somente para escritas.

Capítulo 3

Estudo de Caso e Implementação

Neste capítulo, é feita uma apresentação do Instituto Nacional de Estudos e Pesquisa Anísio Teixeira (INEP), explicando o papel da instituição no ensino brasileiro. É exposto o modelo de dados usado no banco de dados relacional MySQL, e o modelo de dados NoSQL proposto usado no Cassandra. Também é abordado as etapas necessárias para a implementação e configuração do ambiente MySQL e Cassandra, além das aplicações usadas para resolver os problemas surgidos deste trabalho.

3.1 Estudo de Caso e Implementação

De acordo com o próprio sítio institucional, o INEP foi criado, por lei, no dia 13 de janeiro de 1937, sendo chamado inicialmente de Instituto Nacional de Pedagogia[9]. No ano seguinte, o órgão iniciou seus trabalhos de fato, com a publicação do Decreto-Lei no 580, regulamentando a organização e a estrutura da Instituição e modificando sua denominação para Instituto Nacional de Estudos Pedagógicos.

Segundo o Decreto-Lei, cabia ao INEP[9]:

- Organizar a documentação relativa à história e ao estado atual das doutrinas e técnicas pedagógicas;
- Manter intercâmbio com instituições do País e do estrangeiro;
- Promover inquéritos e pesquisas; prestar assistência técnica aos serviços estaduais, municipais e particulares de educação, ministrando-lhes, mediante consulta ou independentemente dela, esclarecimentos e soluções sobre problemas pedagógicos;
- Divulgar os seus trabalhos;
- Participar da orientação e seleção profissional dos funcionários públicos da União.

Nas décadas anteriores à sua criação, algumas tentativas de sistematizar os conhecimentos educacionais e propor melhorias ao ensino já haviam sido articuladas, sem conseguirem, no entanto, alcançar a continuidade desejada.

Em 1952, assumiu a direção do Instituto o professor Anísio Teixeira, que passou a dar maior ênfase ao trabalho de pesquisa. Seu objetivo era estabelecer centros de pesquisa, que foi concretizada com a criação do Centro Brasileiro de Pesquisas Educacionais (CBPE). Em 1972, o INEP virou um órgão autônomo, sendo denominado o Instituto Nacional de Estudos e Pesquisas Educacionais, com o objetivo de levantar a situação educacional do País.

Após o período de dificuldades pelas quais passou no início do governo Collor, quando quase foi extinto, o INEP iniciou um outro processo de reestruturação e redefinição de sua missão, centrado em dois objetivos[9]:

- Reorientação das políticas de apoio a pesquisas educacionais, buscando melhorar sua performance no cumprimento das funções de suporte à tomada de decisões nesta área;
- Reforço do processo de disseminação de informações educacionais, incorporando novas estratégias de modalidades de produção e difusão de conhecimentos e informações.

No início dos anos 90, o INEP atuou como um financiador de trabalhos acadêmicos voltados para a educação. A partir de 1995, aconteceu o processo de reestruturação do órgão. Com a reorganização do setor responsável pelos levantamentos estatísticos, pretendia-se que as informações educacionais pudessem, de fato, orientar a formulação de políticas do Ministério da Educação. Nos últimos anos, o Instituto reorganizou o sistema de levantamentos estatísticos e teve como eixo central de atividades as avaliações em praticamente todos os níveis educacionais.

3.1.1 Censo da Educação Superior do INEP

O INEP é uma autarquia federal vinculada ao Ministério da Educação (MEC). Sua missão é subsidiar a formulação de políticas educacionais dos diferentes níveis de governo com intuito de contribuir para o desenvolvimento econômico e social do país.

O foco deste trabalho é a análise do desempenho em banco de dados relacionais e NoSQL dos dados de um Censo da Educação Superior, produzidas pelo INEP com o objetivo de promover uma conduta profissional adequada, a manutenção da qualidade das estatísticas e a melhoria contínua dos métodos e processos para a sua produção, tratamento, análise e disseminação.

Para este trabalho foram usados os dados do Censo da Educação Superior dos anos de 2014 e 2015, que são os últimos anos disponíveis, onde foi utilizado dados públicos disponibilizados pelo sítio do INEP.

Os microdados do Censo da Educação Superior encontram-se compactado e é composto pelos dados, contendo relações IES, Cursos, Docentes, Alunos e Locais de Oferta. Contém também o Leia-me, Manual do usuário, Filtros Censo Educação Superior, além de 2 anexos:

ANEXO I Contém, em formato XLS, com os seguintes Dicionários das Variáveis do Censo da Educação Superior:

- TABELA DE IES
- TABELA DE CURSO
- TABELA DE DOCENTE
- TABELA DE ALUNO
- TABELA DE LOCAL DE OFERTA
- TABELA CONTENDO O NOME DO PAÍS DE ORIGEM OU NATURALIZAÇÃO

ANEXO II Contém, em formato PDF, os seguintes questionários do Censo da Educação Superior:

- MÓDULO IES
- MÓDULO CURSO MÓDULO DOCENTE MÓDULO ALUNO

3.2 Implementação

Foram elaboradas as etapas de construção do projeto para a análise dos dados do INEP do Ensino Superior Brasileiro, adaptadas do cilo de vida de construção de um DW/BI proposto por Kimball, para um banco Relacional, de acordo com a Seção 2.8. E também, algumas etapas foram adaptadas para uma abordagem de um *Data Warehouse* NoSQL usando Cassandra, onde os dados serão persistidos para a consultas e análises.

Como o foco do trabalho é uma comparação de desempenho da consulta de dados entre um DW do tipo relacional e um do tipo NoSQL, usando MySQL e Cassandra, respectivamente, algumas destas etapas do ciclo de vida foram baseadas em um artigo, "Análise de Dados Abertos Sobre o Ensino Superior Brasileiro"[7].

Este trabalho foi feito em duas grandes etapas, sendo a primeira um Ciclo de vida de Kimall para o DW usando banco de dados relacional, e a segunda etapa, usando um Ciclo de vida de Kimball adaptado para o DW de banco de dados NoSQL.

Planejamento do Projeto

Para este trabalho foi considerado o planejamento do projeto para uma análise de dados públicos do Ensino Superior Brasileiro, o levantamento da fonte dos dados, e também uma arquitetura para implementação do projeto do DW de banco de dados relacional, assim como um projeto do DW de banco de dados NoSQL, projetando um BI para construção de painéis e análises em uma ferramenta OLAP.

Definição dos Requisitos de Negócio

Para este trabalho não foi definido os requisitos do negócio, pois seria necessário entrevistas com a equipe de analistas do cliente para coleta e documentação dos requisitos. Foram consideram alguns requisitos básicos para o desenvolvimento deste trabalho, obter a análise de consultas a respeito do INEP em dois DWs e comparar o desempenho entre ambos usando um ambiente de baixo custo.

Modelagem Dimensional

Para este trabalho foi usado o mesmo modelo dimensional do artigo "Análise de Dados Abertos Sobre o Ensino Superior Brasileiro"[7] para a modelagem dimensional definida para o banco de dados relacional MySQL, mostrado na Figura 3.1. O modelo de dados é composto pelas dimensões Aluno, Curso, Instituição de Ensino Superior, Geografia e Tempo e o pelo fato Acompanhamento Aluno.

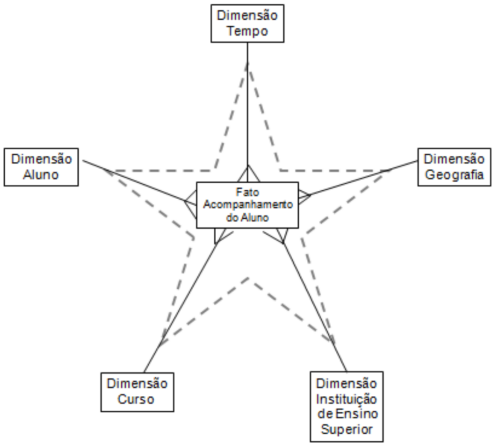

Figura 3.1: Modelagem Dimensional usado no banco de dados Relacional[7].

Para a definição do modelo de dados para a abordagem NoSQL do projeto, foi feita uma adaptação do modelo dimensional de Kimball[3], tentando unir as características de ambas as partes, tanta do modelo dimensional, quanto de uma modelagem NoSQL.

Para chegar nessa modelagem foi pensado em usar as principais características de uma modelagem para o Cassandra, e tentando explorar os principais pontos de se usar um banco de dados NoSQL para persistência dos dados, que não se preocupa com a quantidade e tamanho dos registros e da tabela, não se preocupando também com a redundância dos dados, cujo o modelo proposto é totalmente não normalizado, e onde os dados podem ser espalhados por um *cluster* usando o particionamento do Cassandra.

Foi projetado um modelo, já havendo uma junção de todas as tabelas da dimensões com a tabela fato do esquema estrela, chegando assim em uma única tabela, que vamos nomear neste trabalho de tabela estrela ou *star table*, ilustrada na Figura 3.2.

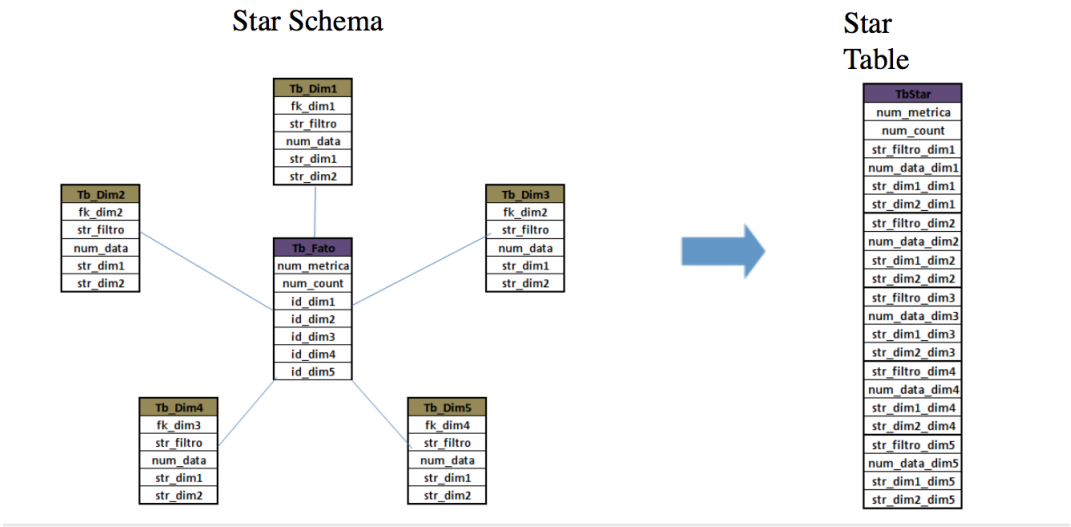

Figura 3.2: Modelagem Dimensional genérica adaptada para ser usado no banco de dados NoSQL Cassandra.

Então para o modelo de dados para o Cassandra deste trabalho, temos as dimensões Aluno, Curso, Instituição de Ensino Superior, Geografia e Tempo e a fato Acompanhamento Aluno todas unidas em uma única família de colunas chamada Tabela Estrela Acompanhamento Aluno. Para armazenar essa família de coluna foi usado uma *keyspace* chamada DW_INEP, com fator de replicação 1, pois não há preocupação com alta disponibilidade e tolerância a falha, com a *SimpleStrategy* como estratégia de replicação, pois há somente um *data center* e com o nível de consistência ONE, pois fazer leitura em somente um nó já satisfaz o ambiente de teste proposto.

Foi usado o atributo UF da dimensão geografia como a *partition key*, pois como visto anteriormente na Seção 2.12.2, escolhendo o valor UF como *partition key*, supostamente

há uma distribuição dos dados que atenda um bom balanceamento de carga. Como o atributo UF tem um conjunto de 27 diferentes valores que se pode assumir e a *partition key* é o atributo selecionado para distribuir os registros entre os nós via *hashing*, então assim, pode se ter um bom balanceamento de carga no *cluster*.

Para a consulta de dados no esquema estrela no *Data Warehouse* relacional, é usado a linguagem de consulta MySQL, que é o padrão de consulta nas tabelas de dados em modelo relacional neste trabalho.

Já para a consulta de dados na tabela estrela do *Data Warehouse* NoSQL, não é possível usar a linguagem padrão de consulta de dados do Cassandra, já que essa linguagem não oferece a liberdade de filtragem da consulta sem algumas restrições. A linguagem do Cassandra não permite fazer uma consulta filtrada sem conter a *partition key*, ou seja, toda consulta filtrada no Cassandra tem que haver no mínimo a *partition key* como visto na Seção 2.12.1.

Pelo fato de um projeto de BI/DW, necessitar de consultas *ad-hoc*, que são não pré-definidas, não há a possibilidade de usar a linguagem padrão do Cassandra, mesmo neste trabalho, no qual são usadas 5 consultas pré definidas na Seção 3.2.1. Não é possível definir uma *partition key* que atenda todas essas consultas, pois nem todas as consultas usam a *partition key* UF, não podendo assim cobrir todos os filtros de consulta e não podendo também prever as consultas que o cliente usará.

Para solucionar este problema, foi usado o Presto, um motor de consulta que permite o uso da linguagem SQL na consulta de dados persistentes no Cassandra sem restrições, que é especificado na Seção 3.3.

Projeto Físico e Configuração do Ambiente

O projeto físico foi idealizado totalmente para ser trabalhado em máquinas virtuais, ligados por uma rede local. O uso de máquina virtuais, tem como objetivo facilitar a implementação para este trabalho. O modelo de dados foi implementado em 4 tipos de ambientes, 3 deles formado por *clusters* Cassandra, de 1, 2 e 3 nós(máquinas) e um ambiente de uma máquina para a consulta dos dados no MySQL. Também foram usadas para todos os *clusters* Cassandra, um ambiente Presto para a consulta dos dados, que também foram utilizados *clusters* de 1, 2 e 3 nós, além de um nó para a coordenação do *cluster* Presto como é visto na Figura 3.3.

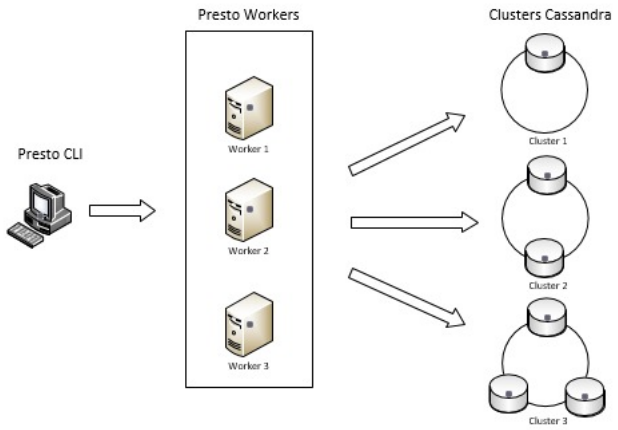

Figura 3.3: Ilustração de ambiente físico de um *cluster* presto com 3 *workers* acessando os *clusters* Cassandra

Para todas as máquinas desse ambiente, foi usado o sistema operacional Ubuntu 16.04.2 LTS. Foi usado a versão 3.10 do banco de dados Cassandra[33], a versão 0.179 para o motor de consulta Presto[8] e a versão 5.7 do MySQL.

Para todas as máquinas virtuais que usam Cassandra e o MySQL, a configuração de hardware foi a mesma, para manter uma uniformidade de hardware para não afetar nos resultados, usando um núcleo do processador Intel Core i7-5500U 2.4 GHz Quad-Core CPU, e 3GB de Ram DDR3 e um HD de 50 gigas para o armazenamento dos dados.

Para todas as máquinas virtuais que usam Presto, tamém com uniformidade de configuração de hardware, a configuração usada foi, um núcleo do processador Intel Core i7-720QM 1.6GHz Quad-Core CPU, e 3GB de Ram DDR3. Foi usado um HD de 8 gigas para cada máquina Presto, pois como ele trabalha com processamento em memória, não há necessidade de grande armazenamento.

Desenvolvimento e Projeto do ETL

O desenvolvimento e projeto do ETL é a fase da especificação e documentação dos três passos principais, que são extração, transformação e carga. Para este trabalho, foram especificados dois projetos de ETL, um de obtenção de dados públicos do INEP, com um processo de ETL para o banco de dados relacional MySQL, e um segundo, com um processo de ETL para o banco de dados NoSQL Cassandra. A Figura 3.4 demonstra o projeto de um ETL.

Projeto e Arquitetura Técnica

Para este trabalho o Projeto e Arquitetura Técnica foi estabelecido com os recursos disponíveis. Para a simplificação do projeto, foi todo pensando no uso de máquina virtuais

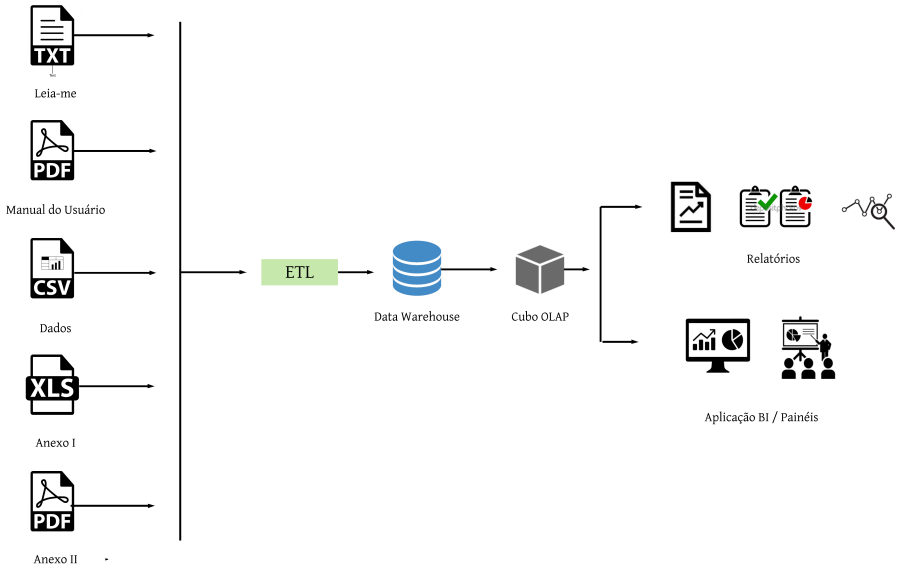

Figura 3.4: Modelo de processo de ETL implementado para a geração de relatórios e painéis

para a facilidade, e uso de ferramentas de software livre ou de licença livre.

Instalação e Seleção de Produtos

Para este trabalho foi selecionado algumas ferramentas, todos do tipo software livre que podem ser baixado e usado gratuitamente:

- **Pentaho Data-Integration:** Software para construção dos processos de ETL para a DW que usa banco de dados relacional MySQL;
- **Talend Open Studio for Big Data:** Software para construção dos processos de ETL para a DW que usa banco de dados NoSQL Cassandra;
- **MySQL:** Sistema de gerenciamento de banco de dados para inserção e consulta dos dados para o DW de abordagem relacional;
- **Cassandra:** Sistema de gerenciamento de banco de dados para inserção e consulta dos dados para o DW de abordagem NoSQL;
- **PrestoDB:** Mecanismo de consulta SQL distribuído de código aberto otimizado para baixa latência, análise de dados e consultas ad-hoc;

- **Oracle VirtualBox:** Um virtualizador de máquinas, usado para a instalação de nós em maquinas virtuais para a execução;
- **Tableau:** Software para a construção de análises e gráficos dos painéis.

3.2.1 Projeto de Aplicação BI

Para o foco deste trabalho, de análise de desempenho de consulta dos dados nas distintas abordagem do DW, foram definidas as seguintes consultas para análise, baseado no artigo de Análise de Dados do Ensino Superior Brasileiro[7]:

- **Consulta 1:** Quantidade de alunos matriculados no ensino superior por curso;
- **Consulta 2:** Quantidade de alunos matriculados no ensino superior por faixa etária e cor/raça;
- **Consulta 3:** Quantidade de alunos matriculados no ensino superior por faixa etária e sexo;
- **Consulta 4:** Quantidade de alunos matriculados no ensino superior por região geográfica;
- **Consulta 5:** Quantidade de alunos matriculados no ensino superior por sexo e Unidade da Federação.

3.2.2 Implementação da aplicação BI

Para este trabalho foi selecionado uma ferramenta de análise de dados OLAP Tableau, a configuração de metadados considerando as 5 principais análises para a geração dos relatórios.

3.2.3 Implantação

Para este trabalho não foi necessário a etapa de implantação, pois não houve uma equipe para treinamento antes da liberação do acesso ao DW.

3.2.4 Manutenção e crescimento

Para este trabalho não foram necessários a etapa de manutenção e crescimento, pois não há continuidade no projeto.

3.2.5 Gerencia do projeto

Para este trabalho, foram dividido por partes para haver um controle. A gerencia do projeto foi dividida em:

- Aquisição da fonte de dados;
- Decisão do modelo para ambos DWs (Relacional e NoSQL);
- ETL para ambos DWs;
- Comparação de desempenho;
- Montagem dos painéis e gráficos de análise, usando a ferramenta OLAP.

3.3 PrestoDB

PrestoDB ou Presto é um mecanismo de consulta SQL distribuído de código aberto para executar consultas analíticas interativas em fontes de dados de todos os tamanhos que variam de *gigabytes* a *petabytes*[8]. Para este trabalho foi usado em média 5 *gigabytes* de registros já no banco de dados Cassandra.

O Presto foi projetado e escrito para análises interativas e aborda a velocidade dos *Data Warehouses* comerciais, enquanto pode ser escalável para atender consultas de dados de organizações de uma magnitude como o Facebook.

3.3.1 Descrição

A principal característica do Presto, é permitir a consulta de dados onde está armazenado, incluindo Hive, Cassandra, bancos de dados relacionais ou até mesmo armazenamentos de dados proprietários. Uma única consulta Presto pode combinar dados de várias fontes, permitindo análises em toda a organização usando uma única linguagem de consulta padrão SQL, além de enfrentar barreiras de bancos que limitam as consultas de dados, como no banco Cassandra que só permite a linguagem CQL[8].

É um sistema distribuído que é executado em um conjunto de máquinas (*cluster* Presto). Na Figura 3.5 é apresentado o fluxo de trabalho do Presto. Uma instalação completa inclui um coordenador e vários presto *workers*. As consultas são enviadas de um cliente, como o Presto CLI para o coordenador. O coordenador analisa e planeja a execução da consulta, distribuindo o processamento para os trabalhadores, podendo trabalhar com o processamento em paralelo.

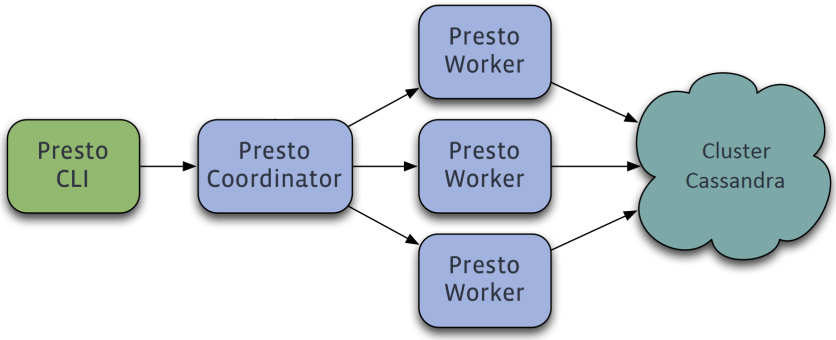

Figura 3.5: Arquitetura exemplo Presto DB para consulta no Cassandra[8].

Capítulo 4

Resultados

Neste capítulo são exibidos os resultados obtidos para inserção do dados utilizando a ferramenta de ETL open talend big data e os resultados de consulta das 5 consultas pré definidas no tópico da aplicação BI no capítulo 5 em diferentes tipos de configuração de sistema diferente. Além de uma comparação entre os resultados, e a comparação dos resultados obtidos entre o banco de dados relacional MySQL, com o banco de dados NoSQL Cassandra.

4.1 Resultados da Carga de dados

A tabela 4.1 mostra o resultado da inserção da carga total dos dados em número de registros por segundo, que atinge em todos os casos aproximadamente 22 milhões de registros. Os *clusters* são especificados na Seção 3.2.

A carga de dados foi realizada uma vez, e sua estimativa de linhas por segundo foi extraída da própria ferramenta.

Cluster	Média linhas/seg
Cluster 1	5900
Cluster 2	5650
Cluster 3	4300

Tabela 4.1: Tabela de média de linhas por segundo na inserção dos dados nos *clusters*

Como visto na tabela, há uma variação de linhas por segundo para a inserção dos dados, no qual acontece pela distribuição de dados que é definido pelo *partition key*, onde define o balanceamento de cada *cluster* de acordo com os dados inseridos.

4.2 Resultados da Consulta dos dados Cassandra

Para a consulta dos dados, foram usados diferentes tipos de ambientes em relação ao número de nós *workers*, o motor de consulta Presto, para uma melhor análise do comportamento da consulta, usando 1, 2 e 3 nós. Todas as consultas foram feitas nos 3 *clusters* Cassandra para posterior comparação dos resultados.

Para toda a consulta de cada uma das perguntas em cada *cluster* foram registrados 3 tempos e tirado uma média entre esses 3 resultados, para obter um tempo final com mais consistência de cada consulta.

Nas Tabelas 4.2, 4.3 e 4.4, é exibido o resultado das 5 consultas, vistas na seção 3.2.1, de cada *cluster*, usando de 1 a 3 nós Presto Workers.

Consulta\Num. Workers	1	2	3
Consulta 1	05:09	05:42	05:52
Consulta 2	05:40	06:29	06:32
Consulta 3	05:40	06:13	06:18
Consulta 4	05:27	06:00	06:02
Consulta 5	05:26	06:01	05:56

Tabela 4.2: Resultados da consulta no Cluster 1

Consulta\Num. Workers	1	2	3
Consulta 1	04:22	04:22	04:23
Consulta 2	04:48	04:56	04:49
Consulta 3	04:21	05:12	04:41
Consulta 4	04:27	04:27	04:28
Consulta 5	04:19	04:17	04:26

Tabela 4.3: Resultados da consulta no Cluster 2

Consulta\Num. Workers	1	2	3
Consulta 1	04:21	04:14	04:13
Consulta 2	04:51	04:36	04:36
Consulta 3	04:29	04:15	04:22
Consulta 4	04:30	04:15	04:16
Consulta 5	04:22	04:12	04:06

Tabela 4.4: Resultados da consulta no Cluster 3

Como pode ser observado nas Tabelas 4.2, 4.3 e 4.4, quanto maior o número de nós Cassandra, menor o tempo de consultado dos dados. Portanto, também é notado que há uma melhor combinação em número de nós Cassandra e número de nós Presto *worker* para obter resultados de menor latência na consulta dos dados. Na tabela 4.2 é notado que

com um nó Presto *Worker* as consultas tiveram um menor tempo para todas as consultas. Para uma melhor visão dessas latências de consultas, observamos as Figuras 4.1, 4.2 e 4.3.

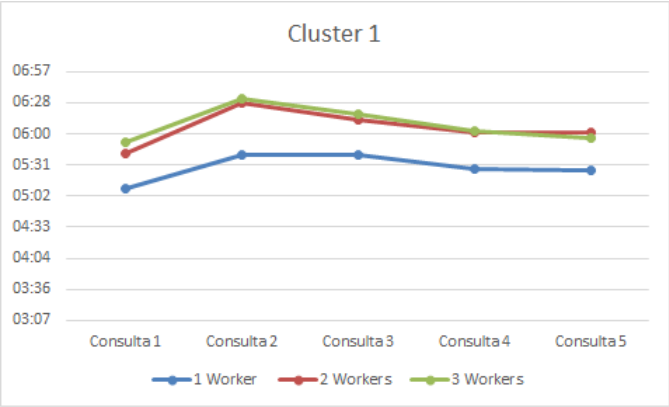

Figura 4.1: Gráfico referente a Tabela 4.2

Pela Figura 4.1, é visível que com 1 *worker*, obtemos o melhor resultado, isso pode ser explicado pelo fato de somente um nó Cassandra ficar sobrecarregado, gerando um gargalo na consulta, pois quanto maior o número de *workers* Presto consultando dados no Cassandra, maior a concorrendo de requisições nesse nó, além de ter um maior tempo de planejamento da consulta pelo nó Coordenador e convergência dos dados de cada *worker*, o que demanda tempo.

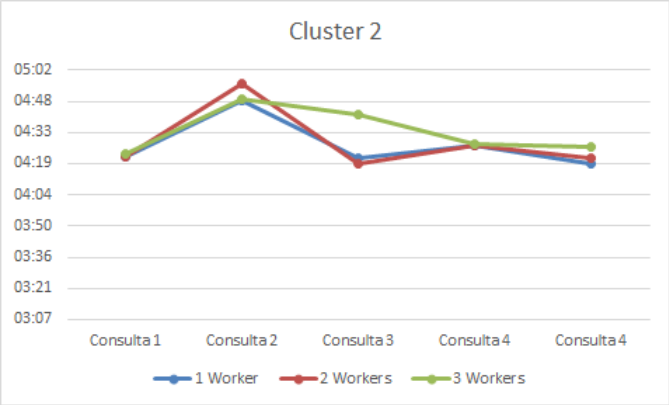

Figura 4.2: Gráfico referente a Tabela 4.3

Na Figura 4.2, temos uma diferença menor entre as 3 curvas, mas mesmo assim tendo resultado melhores com um menor número de *workers*, resultado explicado pelo balanceamento entre nós Presto fazendo requisições a nós Cassandra, onde o gargalo da consulta não fica tão visível.

Como pode ser observado na Figura 4.3, 1 *worker* teve a pior curva de resultados, mostrando que o gargalo passou a ser a quantidade insuficiente de requisições que so-

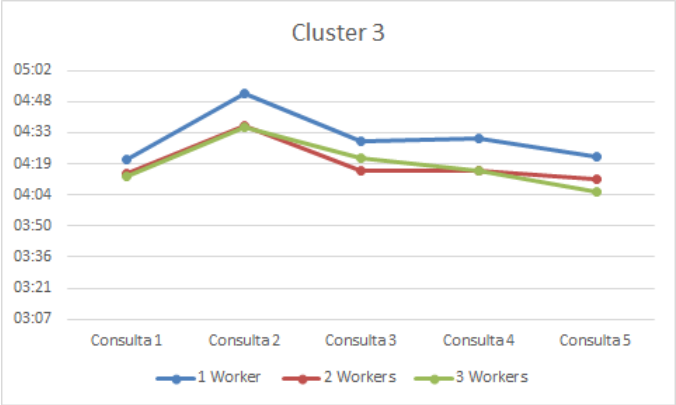

Figura 4.3: Gráfico referente a Tabela 4.4

mente 1 *worker* pode processar para 3 nós Cassandra, e para 2 e 3 *workers* obtiveram praticamente os mesmos resultados.

4.3 Resultados da Consulta dos dados MySQL

Para a consulta de dados no MySQL, foi usada a consulta diretamente no console do sistema operacional, usando o próprio software de consulta do SGBD, e para todo o resultado, foram tirado a média de 3 consultas, para se obter um tempo final com maior consistência.

Consulta	Tempo
Consulta 1	04:39
Consulta 2	05:12
Consulta 3	05:00
Consulta 4	05:11
Consulta 5	05:05

Tabela 4.5: Resultados da consulta no MySQL

4.4 Comparação da Consulta dos dados

A Tabela 4.4 expõe os a média das 5 consultas como resultado obtidos em cada abordagem. Foi considerado para os *clusters* Cassandra, a melhor resultado média, supondo que estaríamos usando o melhor equilíbrio entre nós Cassandras e nós Prestos, dos dados das Tabelas 4.2, 4.3 e 4.4.

A Tabela 4.7 mostra a eficiência de consulta entre o modelo de dados dimensional implementado no MySQL, e a consulta de dados no modelo dimensional proposto para o

	MySQL	Cluster 1	Cluster 2	Cluster 3
Tempo	05:01	05:28	04:27	04:18

Tabela 4.6: Resultados da consulta nos *clusters* Cassandra

banco de dados Cassandra. Somente o *cluster* 1 obteve resultados piores que o modelo dimensional proposto. A Figura 4.4 apresenta o gráfico comparativo entre os *cluster* 1, 2 e 3, e o banco MySQL.

	Consulta de dados		
Modelo Relacional	05:01	5:01	5:01
	Cluster 1	Cluster 2	Cluster 3
Modelo NoSQL	5:28	4:27	4:18
Eficiência	-8,9%	11,2%	14,3%

Tabela 4.7: Tabela de eficiência entre MySQL e os *clusters* Cassandra

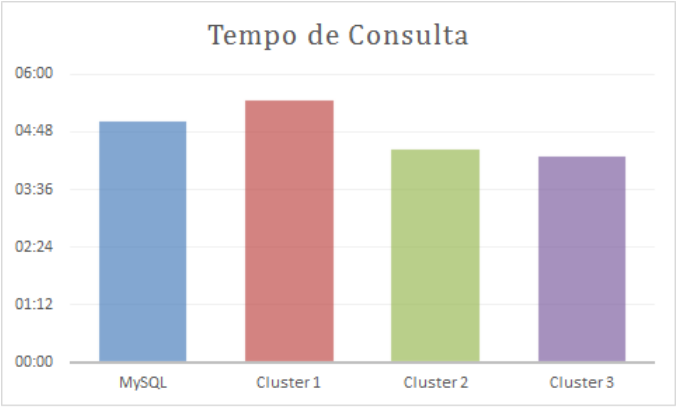

Figura 4.4: Gráfico de tempo de consulta

Nota-se que há um maior aumento de desempenho na consulta dos dados do cluster 1 para o cluster 2, de 20,1%, do que do cluster 2 para o cluster 3, de 3,1%. Isso acontece, pois a partir do cluster 2, como há mais de um nó no anel Cassandra, ele passa a balancear os registros via *hash* da *partition key*, e algum nós acabam tendo uma maior porcentagem de carga dos dados(por exemplo, o estado de SP tem a maior população, e portanto o maior número de alunos e registros), como pode ser visto na Tabela 4.8, e com isso, gerando um gargalo dos dados em um nó que estiver desbalanceado. Portanto usando outra *partition key*, gerariam novos resultados.

Cluster\% Carga	Nó 1	Nó 2	Nó 3
Cluster 1	100%	-	-
Cluster 2	39,47%	60,43%	-
Cluster 3	23,01%	50,74%	26,25%

Tabela 4.8: Tabela de balanceamento de carga.

A consistência da *keyspace* escolhida para o banco também foi ONE, e o fator de replicação usado foi 1, que é somente uma cópia do registro no *cluster*. Esses parâmetros podem ser alterados, gerando diferentes níveis de consistências, replicação dos dados e, consequentemente, novos resultados.

4.5 Painéis de análise de dados

Conforme o que foi visto na Seção 3.2.1 em que foram definidas as perguntas estratégicas para realização de consultas para um Sistema de Apoio a Decisão, as respostas foram obtidas através das consultas da tabela estrela para a geração dos seguintes relatórios, conforme imagens dos Painéis realizadas na ferramenta de análise de dados OLAP Tableau:

Consulta 1 - Quantidade de alunos matriculados no ensino superior por curso

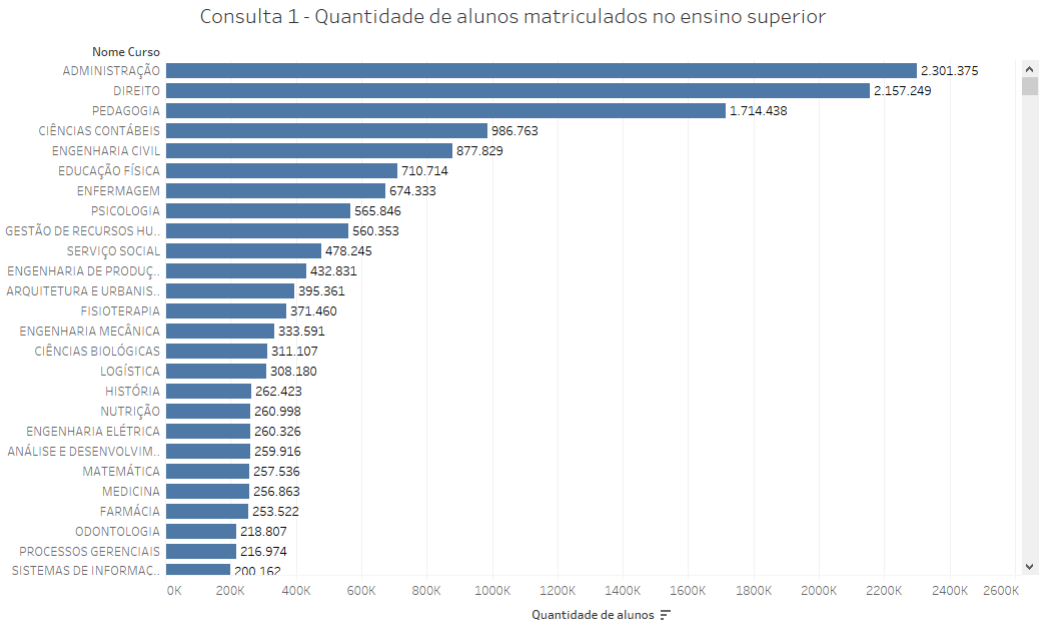

Figura 4.5: Gráfico da Consulta 1

Consulta 2 - Quantidade de alunos matriculados no ensino superior por faixa etária e cor/raça

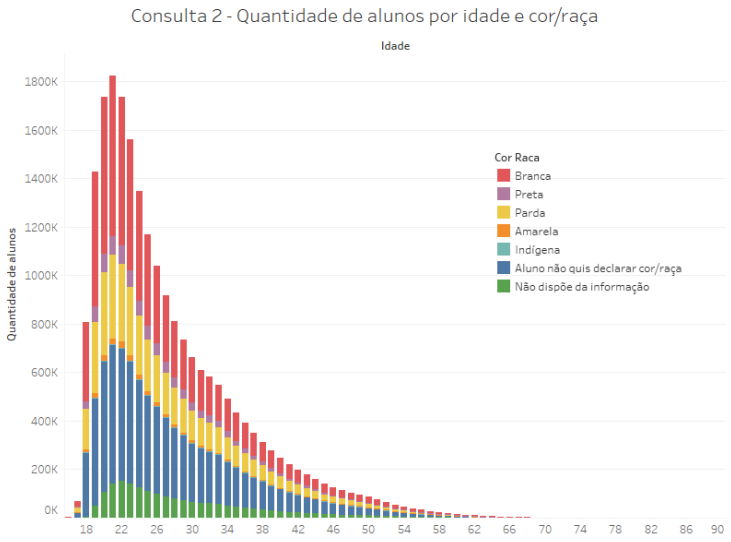

Figura 4.6: Gráfico da Consulta 2

Consulta 3 - Quantidade de alunos matriculados no ensino superior por faixa etária e sexo

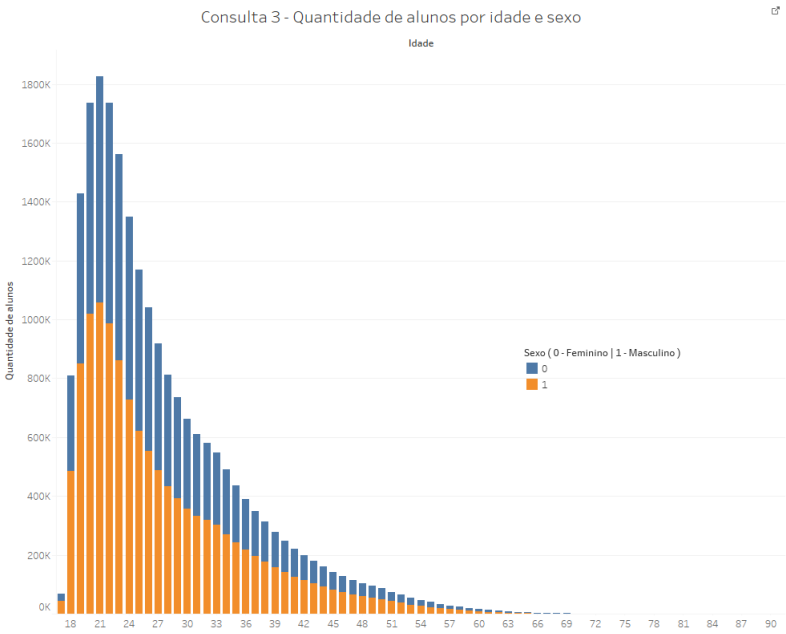

Figura 4.7: Gráfico da Consulta 3

Consulta 4 - Quantidade de alunos matriculados no ensino superior por região geográfica

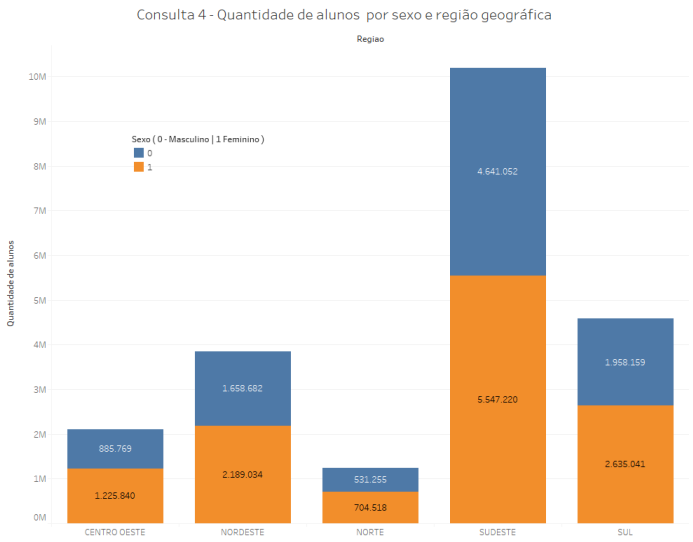

Figura 4.8: Gráfico da Consulta 4

Consulta 5 - Quantidade de alunos matriculados no ensino superior por sexo e Unidade da Federação

Figura 4.9: Gráfico da Consulta 5

Capítulo 5

Conclusões e Trabalhos futuros

Nesta monografia, houve um estudo de caso para analisar e comparar o desempenho da consulta de dados em um *Data Warehouse* NoSQL. Foi utilizado um modelo dimensional para um banco de dados relacional MySQL, e um modelo que foi proposto, seguindo as características do modelo dimensional adaptado para o banco de dados NoSQL Cassandra.

O objetivo principal deste trabalho foi a comparação de desempenho de consulta destes dados entre o banco de dados MySQL e o banco de dados Cassandra. Para isso foi feita um processo de ETL dos dados baixados do INEP, para as ambas as modelagens, o que demonstra que essa é uma etapa possível para uma abordagem de um DW com banco de dados Cassandra. O banco Cassandra não possui um sistema de consulta de dados compatível com o padrão SQL, uma imposição para a consulta de dados *ad-hoc*. Para contornar essa adversidade foi utilizado o motor de consulta de dados Presto, visto que uma das desvantagens do uso do Cassandra, é que seu modelo de dados tem que ser projetado de acordo com a consulta, e para projetar o banco é necessário saber exatamente como os dados serão utilizados. O uso deste motor de consulta, permite que se faça qualquer tipo de consulta no banco Cassandra usando a linguagem SQL. Esse motor de consulta ainda permite que se trabalhe com o paralelismo dos seus componentes chamados de Presto *workers*, que dividem a consulta de acordo com o número de *workers* ativos, o que é mais uma vantagem. Foi construido o ambiente e consultado os dados, e o banco Cassandra se mostrou menos eficaz que o banco MySQL na configuração do *cluster* 1, que tem apenas 1 nó Cassandra, sendo 8,9% menos eficaz que o MySQL. A partir de dois nós no *cluster*, houve uma melhora de desempenho em relação ao MySQL, ficando 11,2% e 14,3% mais eficaz, sendo esses resultados do *cluster* 1 e do *cluster* 2, respectivamente. Nota-se também um aumento de 20,1% do *cluster* 1 para o *cluster* 2. Essa melhora no desempenho tem vários fatores de origem, como o balanceamento de carga dos dados no *cluster*, a harmonia entre o número de Presto *workers* trabalhando em paralelo e o número de nós Cassandra, além das características da sua *keyspace*. O trabalho apresenta

que os resultados seguem as características de um banco Cassandra, que é um banco de baixo custo e que provê um desempenho escalável que pode atender vários tipos de demandas, de acordo com a necessidade do projeto. Os painéis de análise foram criado com sucesso em uma ferramento OLAP Tableau, mostrando assim que o *Datawarehouse* com uma modelagem NoSQL proposta neste trabalho atende as demandas de um projeto de *Business Intelligence* completo.

É possível fazer um futuro trabalho comparando o desempenho entre outros bancos NoSQL com o Cassandra. Também é possível fazer um trabalho de mineração desses dados ou de cruzamento com outros dados públicos, para outros tipos de relatórios e análises.

Referências

[1] Wakulicz, Gilmar J.: *Sistemas de Informacoes Gerenciais.* e-Tec BRASIL, primeira edição edição, 2016, ISBN 978-85-9450-002-1. x, 9, 10

[2] Laudon. Kenneth, Laudon, Jane: *Sistemas de informações gerenciais.* Pearson Education, Inc., São Paulo, sétima edição, 2007, ISBN 978-85-7605-089-6. x, 10, 11, 12, 13

[3] Kimball, Ralph e Margy Ross: *The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling.* Wiley Publishing, 3rd edição, 2013, ISBN 1118530802, 9781118530801. x, 13, 14, 16, 17, 18, 19, 39

[4] Silberschatz, Abraham, Henry Korth e S. Sudarshan: *Database Systems Concepts.* McGraw-Hill, Inc., New York, NY, USA, 6ª edição, 2010, ISBN 0072958863, 9780072958867. x, 23, 25

[5] Sadalage, Pramod J. e Martin Fowler: *NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence.* Addison-Wesley Professional, 1st edição, 2012, ISBN 0321826620, 9780321826626. x, 31

[6] Rao, Srinivasa: *Cassandra: Practical Data Warehousing NoSQL Data Architecture and Modelling - Volume 1.* CreateSpace Independent Publishing Platform, USA, 1st edição, 2015, ISBN 152292941X, 9781522929413. x, 30, 31, 32, 33

[7] Victorino. Marcio, Magalhães. Halley, Cardoso. Luan: *Análise de dados abertos sobre o ensino superior brasileiro.* 1, 2016. x, 37, 38, 43

[8] *Presto*, 2017. `https://prestodb.io/`, acesso em 05/07/2017. x, 41, 44, 45

[9] INEP: *Sobre o inep*, 2017. `http://portal.inep.gov.br/web/guest/sobre-o-inep`, acesso em 11/04/2017. 1, 3, 35, 36

[10] *Amazon dynamodb*, 2017. `https://aws.amazon.com/pt/dynamodb/`, acesso em 09/08/2017. 1

[11] Hewitt, Eben: *Cassandra: The Definitive Guide.* O'Reilly Media, Inc., 1st edição, 2010, ISBN 1449390412, 9781449390419. 1, 31, 32

[12] Setzer, V.W.: *Os Meios Eletrônicos e a Educação: Uma Visão alternativa.*, volume 10. São Paulo: Editora Escrituras, Coleção Ensaios Transversais, 2001. 5, 6

[13] Machado, Felipe Neri Rodrigues: *Projeto de Banco de Dados*. Editora Érica Ltda., São Paulo, 2004. 6

[14] *The open definition*, 2017. http://opendefinition.org/, acesso em 07/06/2017. 6

[15] Bauer, Florian. Kaltenböck, Martin: *Linked Open Data : The Essentials*. edition mono/monochrom, Vienna, Austria, Reading, Mass., 1994, ISBN 0201529831 9780201529838. 6, 8

[16] Bernadette, Hyland.: *Best pratices for publishing linked data*, 2014. https://www.w3.org/TR/ld-bp/, acesso em 05/07/2017. 7

[17] Data, Open Government: *The memorandum on transparency and open government*, 2017. http://www.whitehouse.gov/the_press_office/TransparencyandOpenGovernment, acesso em 27/04/2017. 8

[18] abertos, Portal brasileiro de dados: *Portal brasileiro de dados abertos*, 2017. http://dados.gov.br/paginas/dados-abertos, acesso em 20/04/2017. 8

[19] Open Government Data, The Annotated 8 Principles of: *The annotated 8 principles of open government data*, 2017. https://opengovdata.org, acesso em 26/04/2017. 8

[20] O'Brien, James A.: *Sistemas de informação e as decisões gerenciais na era da Internet*. Editora Saraiva, segunda edição edição, 2004, ISBN 8502044079, 9788502044074. 9, 10, 11, 12, 13

[21] Inmon, W.H. e A.M.N. Guz: *Como construir o data warehouse*. Campus, 1997, ISBN 9788535201413. 12, 14, 15, 16, 17

[22] GORDON, Steven R.; GORDON, Judith R.: *Sistemas de informações: uma abordagem gerencial*. LTC, Rio de Janeiro, 3. ed. edição, 2011. 12

[23] Sen, Arun e Varghese S. Jacob: *Industrial-strength data warehousing*. Commun. ACM, 41(9):28–31, setembro 1998, ISSN 0001-0782. http://doi.acm.org/10.1145/285070.285076. 13, 14, 16

[24] Date, C.J.: *Introdução a Sistemas de BANCOS DE DADOS*, volume 1993. EDITORA AFILIADA, São Paulo, 1986. 15, 21, 23

[25] Elmasri, R. e S.B. Navathe: *Sistemas de banco de dados*, volume 4 edição. PEARSON BRASIL, 2011. 21, 24, 25

[26] Codd, E. F.: *The Relational Model for Database Management: Version 2*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1990, ISBN 0-201-14192-2. 22, 25

[27] Codd, E. F.: *A relational model of data for large shared data banks*. Commun. ACM, 13(6):377–387, junho 1970, ISSN 0001-0782. http://doi.acm.org/10.1145/362384.362685. 22, 24

[28] Oracle: *Oracle Database Concepts*. Addison-Wesley Longman Publishing Co., Inc., 2015. 22

[29] *Why nosql*. White paper, International Business Machines Corp., IBM, March 2015. 26, 27, 28, 30

[30] Mohamed. Chajii, Velasco. Julio Cesar: *NoSQL Database : Raven DB*. edition mono/monochrom, Vienna, Austria, 1st edição, 2014. 27

[31] Tiwari, Shashank: *Professional NoSQL*. John Wiley Sons, Inc., 1st edição, 2011, ISBN 978-0-470-94224-6. 28, 30

[32] Fowler, Adam: *NoSQL For Dummies*. For Dummies, 1st edição, 2015, ISBN 1118905741, 9781118905746. 30

[33] Foundation, Apache Software: *Apache cassandra*, 2017. `http://cassandra.apache.org/`, acesso em 05/07/2017. 30, 41