Comparação de Performance entre PostgreSQL e Cassandra NoSQL utilizando um Dataset de Vagas de Emprego do LinkedIn

Amanda de Medeiros Zepechouka[1], Juan C. F. R. R. de Morais[1], Leandro de Lima Minchuel[1], Leonardo J. Zanotti[1], Victoria K. Vieira[1], Wellington H. Kania[1]

[1]Setor de Educação Profissional e Tecnologia – Universidade Federal do Paraná (UFPR)
R. Dr. Alcides Vieira Arcoverde, 1225 - Jardim das Américas, Curitiba - PR, 81520-260

leonardozanotti@ufpr.br

juanfernandes@ufpr.br

victoriavieira@ufpr.br

Abstract. *This article presents a comparative performance study between PostgreSQL and Cassandra NoSQL using job vacancy data from LinkedIn. The technologies used, data modeling, the case study with research questions, the results obtained and performance evaluation are covered.*

Resumo. *Este artigo apresenta um estudo comparativo de performance entre PostgreSQL e Cassandra NoSQL utilizando dados de vagas de emprego do LinkedIn. São abordadas as tecnologias empregadas, a modelagem dos dados, o estudo de caso com perguntas de pesquisa, os resultados obtidos e a avaliação de desempenho.*

1. Introdução

Com o crescimento exponencial da quantidade de dados gerados e armazenados diariamente, torna-se essencial a utilização de tecnologias capazes de manipular grandes volumes de informação de maneira eficiente. O Big Data se refere a essa imensa quantidade de dados, estruturados ou não, que não podem ser tratados por métodos tradicionais devido ao seu volume, velocidade e variedade. Para lidar com esses desafios, surgiram diversas tecnologias de armazenamento e processamento de dados, entre as quais se destacam os bancos de dados relacionais, como o PostgreSQL[1], e os bancos de dados NoSQL, como o Cassandra[2].

O PostgreSQL é um sistema gerenciador de banco de dados objeto-relacional, conhecido por sua robustez, conformidade com padrões SQL e extensibilidade. É amplamente utilizado em aplicações onde a consistência dos dados e a integridade referencial são cruciais. Por outro lado, o Cassandra é um banco de dados NoSQL distribuído, desenvolvido para lidar com grandes volumes de dados distribuídos em muitos servidores, oferecendo alta disponibilidade e escalabilidade horizontal.

Neste estudo, comparamos o desempenho do PostgreSQL e do Cassandra utilizando um dataset de vagas de emprego do LinkedIn. Nosso objetivo é avaliar qual dessas tecnologias oferece melhor performance em diferentes tipos de consultas e operações, considerando aspectos como tempo de execução, simplicidade de modelagem e adequação ao tipo de dados e uso pretendido.

2. Tecnologia Big Data Utilizada

Nesta seção, descrevemos detalhadamente as tecnologias de Big Data empregadas no estudo: PostgreSQL e Cassandra. Discutimos suas características, modelagem de dados e exemplos reais de uso no mercado.

2.1. PostgreSQL

O PostgreSQL é um banco de dados objeto-relacional de código aberto, conhecido por sua conformidade com padrões SQL e capacidade de lidar com transações complexas. Ele oferece suporte a diversos tipos de dados e extensões, tornando-se uma escolha popular para uma variedade de aplicações.

2.1.1. Modelagem de Dados em PostgreSQL

A modelagem de dados em PostgreSQL segue um esquema relacional tradicional, onde os dados são armazenados em tabelas com colunas e linhas. Cada tabela representa uma entidade do modelo de dados, e as relações entre essas entidades são estabelecidas por meio de chaves estrangeiras.

2.1.2. Exemplo de Uso no Mercado

Um exemplo real de uso do PostgreSQL é a plataforma de comércio eletrônico **Shopify**. Shopify utiliza PostgreSQL para gerenciar dados transacionais, como informações de clientes, pedidos e inventários, garantindo consistência e integridade dos dados. A robustez e a conformidade com padrões SQL do PostgreSQL permitem que a plataforma lide com transações complexas e consultas SQL avançadas de forma eficiente.

2.2. Cassandra NoSQL

O Cassandra é um banco de dados NoSQL distribuído e altamente escalável, projetado para armazenar e gerenciar grandes volumes de dados distribuídos por vários servidores. Ele oferece alta disponibilidade e particionamento automático, sendo ideal para aplicações que exigem escalabilidade horizontal e alta taxa de escrita.

2.2.1. Modelagem de Dados em Cassandra

A modelagem de dados em Cassandra é baseada em um esquema de tabela, similar ao modelo relacional, mas com algumas diferenças chave. As tabelas em Cassandra são otimizadas para consultas específicas e projetadas para evitar operações de junção complexas.

2.2.2. Exemplo de Uso no Mercado

Um exemplo real de uso do Cassandra é o **Netflix**. Netflix utiliza Cassandra para gerenciar grandes volumes de dados de visualização de usuários, distribuídos por várias regiões

geográficas. A alta disponibilidade e a capacidade de escalabilidade horizontal do Cassandra permitem que o Netflix lide com milhões de transações de leitura e escrita por segundo, garantindo uma experiência de usuário fluida e sem interrupções.

3. Estudo de Caso

Neste capítulo, descrevemos detalhadamente o dataset utilizado, apresentamos as perguntas de pesquisa específicas e discutimos os resultados esperados de toda a análise.

3.1. Dados Utilizados

O dataset[3] utilizado neste estudo é composto por um arquivo CSV contendo informações detalhadas sobre vagas de emprego. Abaixo estão os detalhes de cada arquivo e seus respectivos campos:

- **job_postings.csv**:
  - job_id: Identificador da vaga de emprego conforme definido pelo LinkedIn.
  - company_id: Identificador da empresa associada à vaga.
  - title: Título da vaga.
  - description: Descrição da vaga.
  - max_salary: Salário máximo.
  - med_salary: Salário mediano.
  - min_salary: Salário mínimo.
  - pay_period: Período de pagamento (Hora, Mês, Ano).
  - formatted_work_type: Tipo de trabalho (Tempo integral, Meio período, Contrato).
  - location: Localização da vaga.
  - applies: Número de aplicações submetidas.
  - original_listed_time: Tempo original em que a vaga foi listada.
  - remote_allowed: Indica se o trabalho remoto é permitido.
  - views: Número de visualizações da vaga.
  - job_posting_url: URL da postagem da vaga em uma plataforma.
  - application_url: URL onde as candidaturas podem ser submetidas.
  - application_type: Tipo de processo de candidatura (offsite, complexo/simples onsite).
  - expiry: Data ou tempo de expiração da listagem da vaga.
  - closed_time: Tempo para fechar a listagem da vaga.
  - formatted_experience_level: Nível de experiência exigido (início, associado, executivo, etc).
  - skills_desc: Descrição das habilidades exigidas para a vaga.
  - listed_time: Tempo em que a vaga foi listada.
  - posting_domain: Domínio do site com a aplicação.
  - sponsored: Indica se a listagem da vaga é patrocinada ou promovida.
  - work_type: Tipo de trabalho associado à vaga.
  - currency: Moeda em que o salário é fornecido.
  - compensation_type: Tipo de compensação para o trabalho.

3.2. Perguntas de Pesquisa

Para conduzir a análise comparativa entre PostgreSQL e Cassandra NoSQL, definimos várias perguntas de pesquisa específicas que cobrem uma variedade de operações e consultas comuns em bancos de dados. As perguntas são projetadas para explorar a eficiência de cada tecnologia em diferentes cenários. Abaixo estão as perguntas de pesquisa e os resultados esperados:

3.2.1. Consulta Envolvendo Máximo (Agrupamento)

- **Pergunta:** Qual é o salário máximo presente no banco de dados?
- **Resultado Esperado:** Espera-se que ambas as tecnologias retornem os mesmos valores para salário máximo mas com diferenças no tempo de execução devido à maneira como cada tecnologia lida com agregações e agrupamentos.

3.2.2. Leitura simples por chave primária

- **Pergunta:** Qual a vaga com job_id igual a ”2147609816”?
- **Resultado Esperado:** Ambas as consultas devem retornar a vaga respectiva ao job_id. No entanto, espera-se que Cassandra seja mais rápido devido à sua arquitetura de busca otimizada.

3.2.3. Atualização de dados por chave primária

- **Pergunta:** Atualize o título da vaga com job_id igual a ”2974397965”.
- **Resultado Esperado:** Ambas as tecnologias devem ser capazes de atualizar os dados corretamente, mas espera-se que PostgreSQL seja mais eficiente em operações de atualização devido ao seu suporte a transações.

3.2.4. Filtragem por índice secundário

- **Pergunta:** Selecione todas as vagas que são de tempo integral.
- **Resultado Esperado:** Por se basear em um índice secundário, o PostgreSQL deve levar vantagem sobre o Cassandra, que é mais robusto para consultas baseadas na chave primária.

3.3. Comparação Entre NoSQL e Relacional

Para cada uma das perguntas de pesquisa acima, serão comparados os seguintes aspectos entre PostgreSQL e Cassandra:

- **Velocidade:** Tempo de execução das consultas e operações.
- **Simplicidade:** Facilidade de escrita e compreensão das consultas.
- **Modelagem Adequada:** Adequação da modelagem de dados para cada tecnologia.

4. Resultados e Avaliações

Este capítulo descreve a implementação do código utilizado para comparar o desempenho entre PostgreSQL e Cassandra utilizando o conjunto de dados de vagas de emprego do LinkedIn. O foco principal está na inserção dos dados, criação das tabelas e execução das consultas para avaliar a eficiência de ambos os sistemas de gerenciamento de banco de dados.

4.1. Configuração do Ambiente

Para a realização dos experimentos, utilizamos a seguinte configuração de ambiente:

- **PostgreSQL**: Versão 16.3, rodando localmente.
- **Cassandra**: Versão 3.4.6, rodando em um container Docker
- **Dataset**: Utilizamos um arquivo CSV com 500MB (cerca de 124 mil linhas) de dados de vagas de emprego do LinkedIn.

4.2. Configuração das Máquinas

Para testes, utilizou-se de um ambiente Linux Mint 21.1, rodando em um Intel I5-1155G7 com 8GB de memória ram.

4.3. Modelagem dos Dados

Para configurar uma análise justa entre ambos os bancos de dados, a modelagem dos dados foi mantida igual ao do arquivo original, conforme mostrado na sessão 3.1. Entretanto, é conhecido que o Cassandra levaria vantagem ao utilizar de uma modelagem de dados específica para o seu ambiente.

4.4. Resultados das Consultas

Conforme mostrado na figura seguinte, os resultados seguem o esperado, com o Cassandra ultrapassando o PostgreSQL em consultas por chave primária quando o volume de dados aumenta consideravelmente.

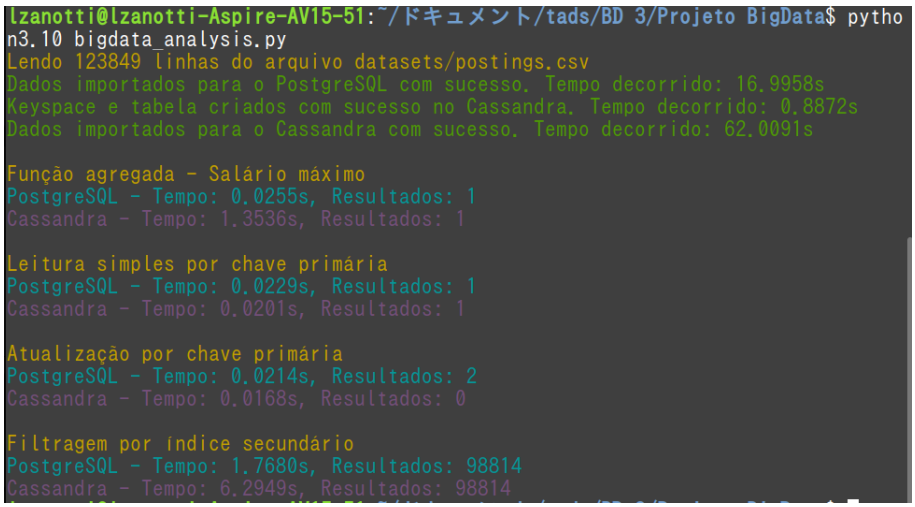

Figure 1. Resultado da análise

5. Conclusão

O Cassandra não apresentou bons resultados para as consultas de função agregada e filtragem por índice secundário, ficando bem atrás do concorrente, entretanto, para as consultas de leitura simples e atualização, o Cassandra superou o PostgreSQL quando o volume de dados tornou-se grandioso.

Entretando, além da velocidade do PostgreSQL ter sido superior, esse SGBD também demonstrou uma alta simplicidade de escrita e a modelagem de dados foi a mantida inalterada do arquivo. Para o Cassandra, todavia, uma modelagem simples, como a proposta neste trabalho, não encaixou muito bem com a sua metologia, que requer uma modelagem específica e otimizada.

Desse modo, o ambas as tecnologias tiveram seus pontos fracos e fortes e performaram conforme o previsto, de modo que, para um estudo futuro, propõe-se o uso de um dataset ainda maior, com alguns milhões de linhas, visando aumentar a discrepância entre as tecnologias.

6. Referências

References

[1] PostgreSQL. Disponível em: `http://postgresql.org`. Acesso em: 17 jul. 2024.

[2] Apache Cassandra. Disponível em: `http://cassandra.apache.org`. Acesso em: 17 jul. 2024.

[3] Kaggle. LinkedIn Job Postings Dataset. Disponível em: `https://www.kaggle.com/datasets/arshkon/linkedin-job-postings`. Acesso em: 17 jul. 2024.

7. Anexos

Repositório com código fonte do projeto: `https://github.com/LeonardoZanotti/BigData-Project`

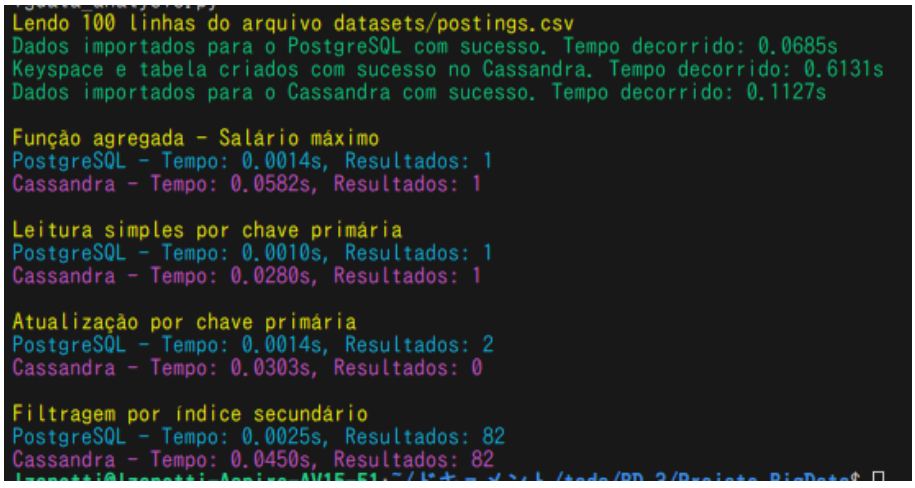

Figure 2. Resultado da análise (100 linhas)

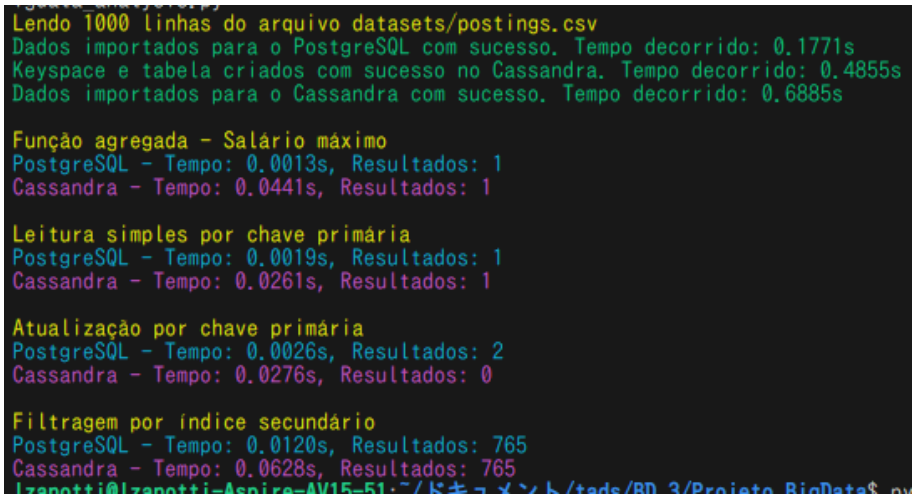

Figure 3. Resultado da análise (1000 linhas)

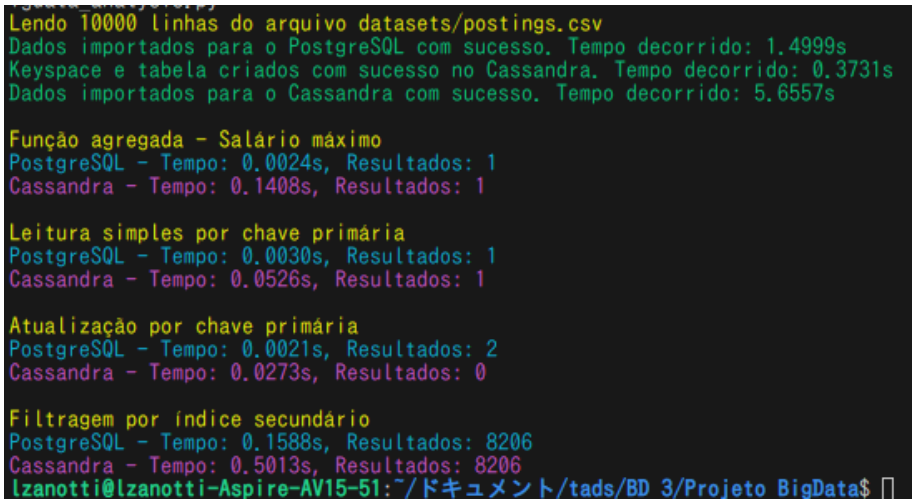

Figure 4. Resultado da análise (10000 linhas)

Figure 5. Resultado da análise (100000 linhas)

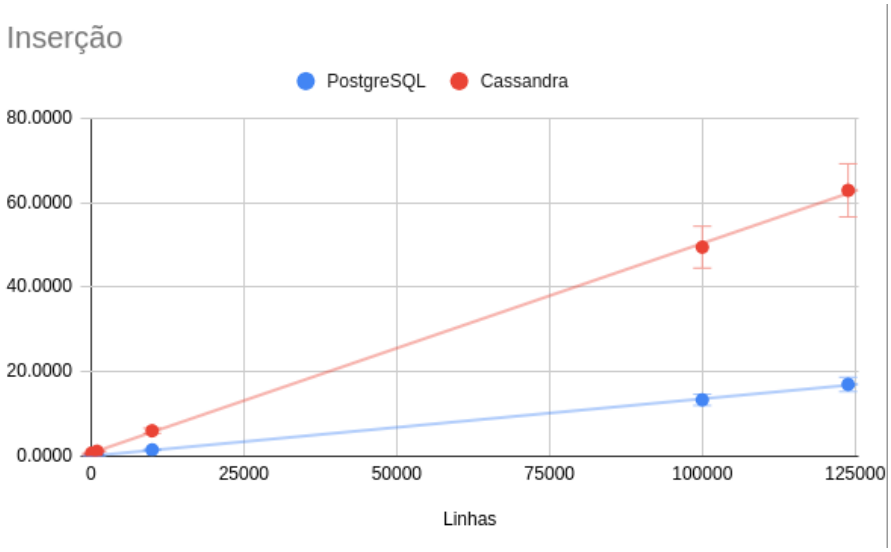

Figure 6. Regressão linear (inserção)

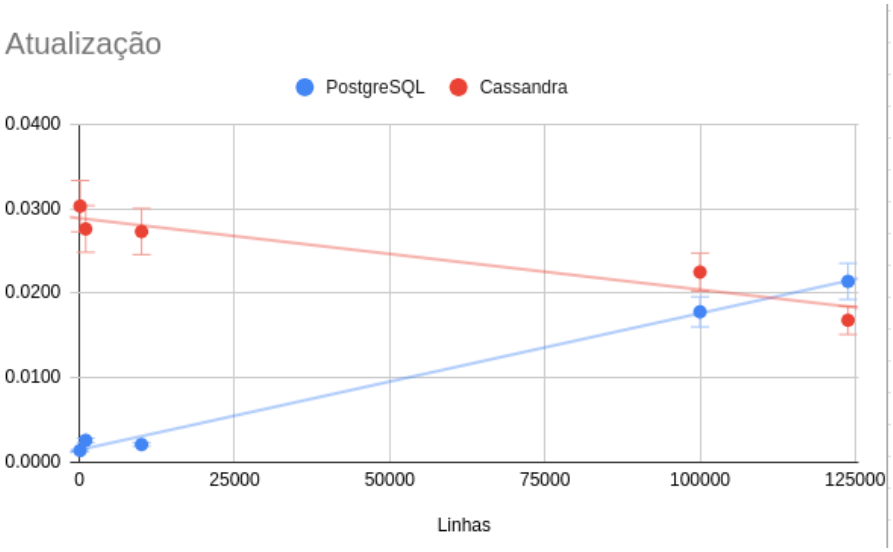

Figure 7. Regressão linear (Atualização)

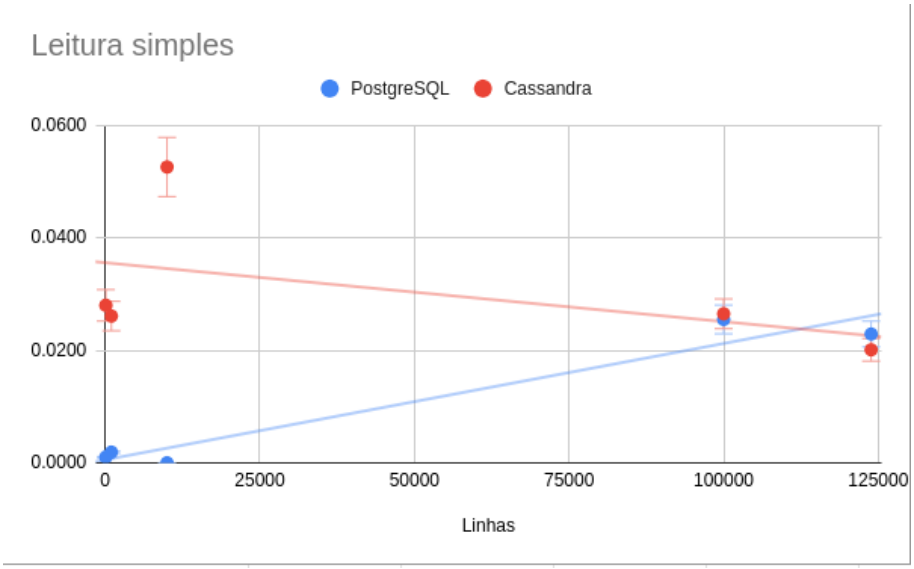

Figure 8. Regressão linear (Leitura simples)

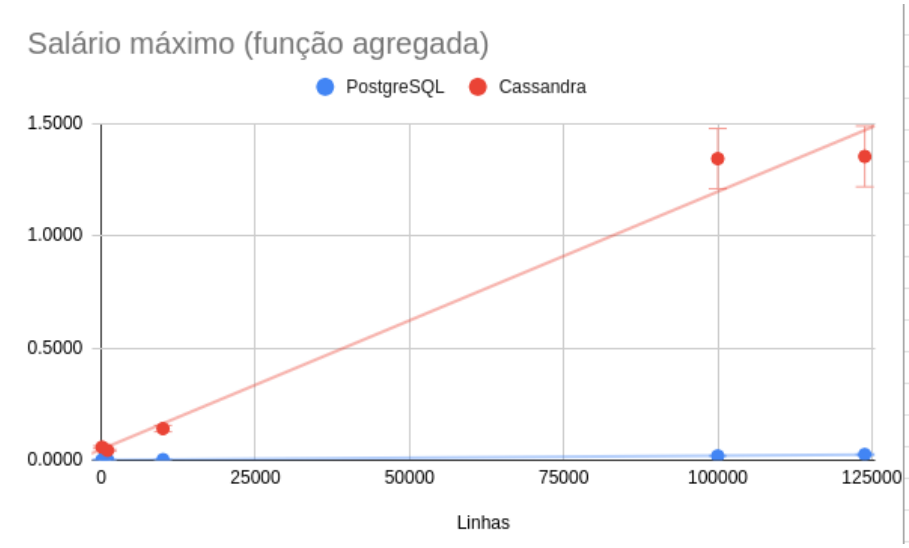

Figure 9. Regressão linear (Salário máximo - Agregação)

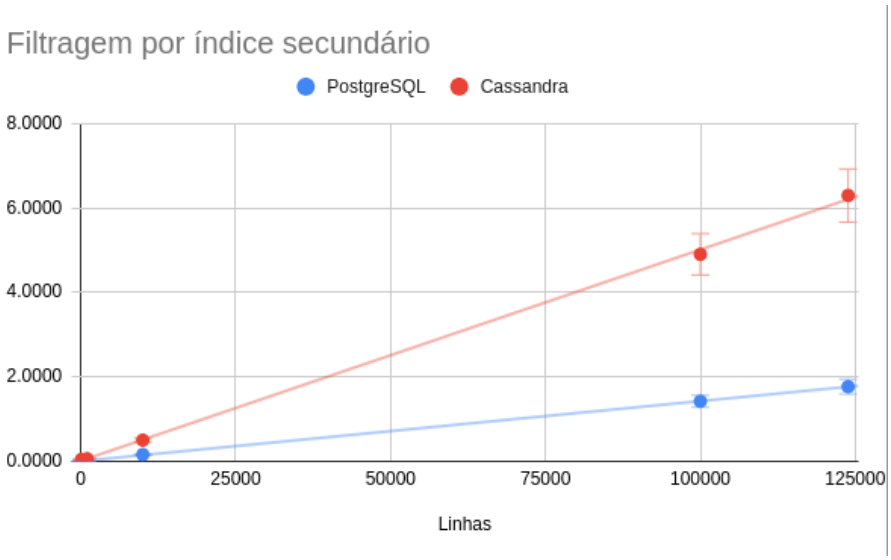

Figure 10. Regressão linear (Filtragem por índice secundário)