UNIVERSIDADE FEDERAL DA FRONTEIRA SUL
CAMPUS CHAPECÓ
CURSO DE CIÊNCIA DA COMPUTAÇÃO

LEONARDO BELINSKI

AVALIAÇÃO DA BLOCAGEM DE GRANDES BASES DE DADOS NA PLATAFORMA DE TEMPO REAL APACHE STORM

CHAPECÓ
2018

LEONARDO BELINSKI

AVALIAÇÃO DA BLOCAGEM DE GRANDES BASES DE DADOS NA PLATAFORMA DE TEMPO REAL APACHE STORM

Trabalho de conclusão de curso de graduação apresentado como requisito para obtenção do grau de Bacharel em Ciência da Computação da Universidade Federal da Fronteira Sul.

Orientador: Prof. Dr. Guilherme Dal Bianco

CHAPECÓ

2018

Belinski, Leonardo

Avaliação da blocagem de grandes bases de dados na plataforma de tempo real Apache Storm / Leonardo Belinski. – 2018.
31 f.: il.; 30 cm.

Orientador: Guilherme Dal Bianco
Monografia (Graduação) - Universidade Federal da Fronteira Sul, Ciência da Computação, Curso de Ciência da Computação, SC, 2018.

1. Deduplicação de dados, Duplicação de registros, Plataforma de tempo real, Apache Storm. I. Dal Bianco, Guilherme. II. Título.

E-mail: leonardobelinski@yahoo.com.br

LEONARDO BELINSKI

AVALIAÇÃO DA BLOCAGEM DE GRANDES BASES DE DADOS NA PLATAFORMA DE TEMPO REAL APACHE STORM

Trabalho de conclusão de curso de graduação apresentado como requisito para obtenção do grau de Bacharel em Ciência da Computação da Universidade Federal da Fronteira Sul.

Orientador: Prof. Dr. Guilherme Dal Bianco

Este trabalho de conclusão de curso foi defendido e aprovado pela banca em: 07 / 12 / 2018

BANCA EXAMINADORA:

Dr. Guilherme Dal Bianco - UFFS

Me. Andressa Sebben - UFFS

Dr. Denio Duarte - UFFS

RESUMO

O processamento de dados em tempo real permite que sejam processadas grandes quantidades e volumes de dados de forma contínua. Neste grande volume de dados, podem existir dados que representem a mesma informação (duplicação de dados). Este problema é uma anomalia que desperdiça recursos de tempo de processamento, espaço em disco e de memória auxiliar. Para tratar essa anomalia, é utilizada a técnica de deduplicação de dados. Nesta técnica, são identificados e excluídos dados duplicados. Para ser aplicada em grandes bases de dados, durante o processo de identificação de duplicatas, é utilizada a técnica de agrupamento de dados (ou blocagem de dados). Nesta técnica, são agrupados os dados que possuem alguma similaridade conforme uma característica definida. Assim, ao agrupar os dados, no processo de identificação de duplicatas, serão apenas realizadas comparações entre os elementos que compõem cada bloco. Neste contexto, foi desenvolvida a ferramenta RIJOIN [7]. Esta ferramenta, desenvolvida na plataforma Apache Storm, combina a deduplicação de dados com o agrupamento de dados, índice invertido, filtros e o processamento de dados em tempo real. Assim, neste trabalho, serão propostas melhorias nesta ferramenta, sendo estas principalmente direcionadas ao processo de criação e uso do índice invertido, na etapa de criação dos blocos de registros. Através de testes, foi observado que as melhorias propostas neste trabalho alcançaram resultados promissores, dentre estes, a redução no número de reordenações de registros.

Palavras-chave: Deduplicação de dados, Duplicação de registros, Plataforma de tempo real, Apache Storm.

ABSTRACT

Real-time data processing allows large amounts and volumes of data to be processed continuously. In this large volume of data, there may be data representing the same information (data duplication). This problem is an anomaly that wastes resources on processing time, disk space and auxiliary memory.To treat this anomaly, the data deduplication technique is used. Duplicate data are identified and deleted in this technique. To be applied in large databases, during the process of identifying duplicates, the technique of data grouping (or blocking of data) is used. In this technique, the data that have some similarity are grouped according to a defined characteristic. Thus, when grouping the data, in the process of identifying duplicates, only comparisons will be made between the elements that make up each block. In this context, the RIJOIN tool [7] was developed. This tool, developed on the Apache Storm platform, combines data deduplication with data collation, inverted index, filters and real-time data processing. Thus, in this work, improvements will be proposed in this tool, these being mainly directed to the process of creation and use of inverted index, in the stage of creation of records blocks. Through the tests, it was observed that the improvements proposed in this work achieved promising results, among them, the reduction in the number of records reordering.

Keywords: Deduplication of data, Duplication of records, Real-time platform, Apache Storm.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

Nos últimos anos, todos os dias uma massiva quantidade de dados é produzida. Tal fato é possibilitado por diferentes fatores, entre os quais pode-se destacar a disponibilidade e expansão da internet e a popularização das redes sociais. Devido ao fato de uma grande parte destas informações produzidas estarem disponíveis, empresas e outros interessados começaram a utilizar estes dados para benefício próprio, sendo para fins comerciais ou de estudo. Para obter e processar este grande volume de dados, estes interessados passaram a desenvolver programas e ferramentas específicas para este fim.

O processamento de dados em tempo real consiste em processar fluxos de dados (também chamado de *stream*) de forma contínua, à medida em que os dados são gerados e/ou obtidos de alguma fonte. Esta abordagem, ao contrário da abordagem estática (*batch*), não necessita de uma base de dados inicial para ser utilizada. Uma desvantagem desta abordagem é que, por funcionar de forma contínua, não é possível mensurar a quantidade e o volume de dados que esta aplicação irá receber.

Com o decorrer do tempo e o acúmulo de informações em uma base de dados, podem surgir anomalias no conteúdo armazenado. Alguns exemplos de anomalias são: valores não preenchidos em registros, erros léxicos e registros duplicados [7]. Estes dados com anomalias, além de possibilitarem que falhas prejudiquem a obtenção de respostas do processamento destas bases, aumentam o custo operacional em tempo de execução e espaço de armazenamento no sistema. Como exemplo, em um cenário de uma base de dados com anomalias de registros duplicados, todas as vezes que a rotina de *backup* for executada, estes dados serão processados desnecessariamente. Assim, para remover tais problemas, devem ser utilizadas rotinas, técnicas e ferramentas para encontrar e remover dados redundantes.

Para tratar este problema de registros duplicados, é utilizada a técnica de deduplicação de dados [2]. Esta técnica consiste em analisar um bloco de dados, buscando identificar se entre estes dados existem registros que representam a mesma informação, ou seja, uma duplicata e, em seguida, caso seja constatado que existem duplicatas, eliminar as mesmas da base de dados, mantendo apenas um dos registros desta duplicata [3].

Para que a deduplicação seja aplicada com melhor eficiência, principalmente em bases com grande volume de dados, é utilizada a técnica de agrupamento de dados (ou blocagem de dados). Esta técnica consiste em agrupar dados que contém alguma similaridade com base em

uma característica. A Figura 1.1 contém um exemplo de agrupamento de dados, sendo que os dados estão agrupados segundo o atributo montadora. Assim, ao separar os registros em blocos, a busca por duplicatas será reduzida somente a estes componentes, que naquele momento, formam o bloco. Outra vantagem do uso desta técnica é a redução do tempo necessário para a busca de duplicatas.

Figura 1.1: Exemplo de agrupamento de dados

Apesar da blocagem reduzir o número de elementos a serem comparados ao agrupar os dados, para comparar os registros de um bloco, é necessário que sejam criados pares de registros combinando todos os elementos do bloco. Assim, esta é uma tarefa que apresenta um crescimento de ordem quadrática em relação à quantidade de registros de cada bloco. Este aspecto mostra que esta técnica é limitada, pois para que seja executada em grandes bases de dados, demanda de recursos computacionais substanciais.

Neste contexto de agrupamento de dados e processamento em tempo real, foi desenvolvida a ferramenta RIJOIN [7], a qual apresenta um método para o agrupamento de dados, combinando o uso de índice invertido e filtros. O objetivo desta ferramenta é gerar pares candidatos a serem duplicatas de registros. Mais detalhes desta ferramenta serão apresentados no Capítulo 3.

Este trabalho propõe o desenvolvimento de melhorias na etapa de criação e uso do índice invertido e a implementação do filtro de sufixo da ferramenta PPJOIN+ [8] na ferramenta de processamento de fluxo de dados em tempo real RIJOIN [7].

1.1 Objetivos

1.1.1 Objetivo Geral

Desenvolvimento de melhorias no processo de agrupamento de dados da ferramenta de processamento de dados em tempo real RIJOIN [7].

1.1.2 Objetivos Específicos

- Desenvolver melhorias na etapa de criação e uso do índice invertido;
- Implementar o filtro de sufixo do método PPJOIN+ [8];
- Realizar testes (com o uso das bases de dados geradas) na ferramenta após a implementação das melhorias propostas;
- Realizar testes (com o uso das bases de dados geradas) na ferramenta RIJOIN [7];
- Comparar e avaliar os resultados destes testes.

1.2 Estrutura do Trabalho

Este trabalho é composto pela seguinte estrutura: no Capítulo 1 é apresentado o referencial teórico para este trabalho, no Capítulo 2 são apresentadas as ferramentas utilizadas para o desenvolvimento deste trabalho, no Capítulo 3 são apresentados os trabalhos relacionados, o Capítulo 4 contém a proposta desenvolvida para este trabalho, no Capítulo 5 são apresentados e discutidos os resultados dos testes da ferramenta proposta neste trabalho e no Capítulo 6 são apresentadas as considerações finais deste trabalho.

2 FERRAMENTAS UTILIZADAS

Neste capítulo é descrito o funcionamento básico das ferramentas Apache Storm e Redis.

2.1 Apache Storm

O Apache Storm é uma plataforma de desenvolvimento de aplicações para o processamento de *stream* (fluxo de dados) de forma contínua e em tempo real. As aplicações desenvolvidas nesta plataforma são capazes de processar grandes volumes de dados. Uma *stream* no Apache Storm é formada por uma sequência ilimitada de tuplas, sendo estas tuplas formadas por uma lista de campos do tipo chave-valor [4].

Algumas das características desta plataforma são: *open source*, distribuída, escalável e tolerante a falhas. Esta plataforma é desenvolvida pela Apache Software Foundation [4]. Um programa implementado no Storm é chamado de aplicação. Após estar em funcionamento, uma aplicação só irá parar de executar se for encerrada pelo usuário [4].

Para serem realizadas as tarefas, basicamente são necessários três componentes: Spout, Bolt e Topologia. Cada um destes componentes é apresentado a seguir com uma breve explicação de sua função em uma aplicação

O Spout é o responsável por receber os dados e repassar os mesmos para um ou mais bolts em forma de fluxos de dados. Estes dados podem ser obtidos de duas formas: através de base de dados estática (arquivos de dados ou banco de dados) ou de APIs conectadas a alguma fonte de fluxo de dados (por exemplo, redes sociais). Para a entrada de dados obtidos de base estática, o spout realiza a leitura de cada registro e repassa o conteúdo lido em forma de fluxo de dados, assim, simulando a geração e o recebimento de dados de forma dinâmica (recebimento dos dados em tempo real). Já para a entrada de dados através do uso de API, quando um fluxo de dados é enviado da API para o spout, este fluxo é enviado para um ou mais bolts para ser processado.

O Bolt é responsável por processar os dados recebidos através do fluxo de dados. Neste processamento, podem ser realizadas diferentes computações ou transformações nos dados, tais como: filtragens, contabilização de frequência de termos, agregação de dados, persistência de dados em disco, entre outros. Após o processamento, estes dados podem ser enviados como um fluxo de dados para um ou mais bolts ou persistidos em banco de dados.

A topologia é uma abstração da computação realizada por uma aplicação [5]. Esta com-

putação é um processamento sobre um conjunto de dados realizado por spouts e bolts. Tais formas de processamento foram mencionadas acima na apresentação de cada componente. Esta abstração é representada no formato de um grafo acíclico dirigido. Neste grafo, cada um dos nós representa um spout ou bolt e, as arestas, representam o encadeamento entres os componentes da aplicação e também o fluxo dos dados entres estes componentes [4]. A Figura 2.1 mostra um exemplo de uma topologia no Apache Storm, sendo esta composta por 2 spouts e 5 bolts.

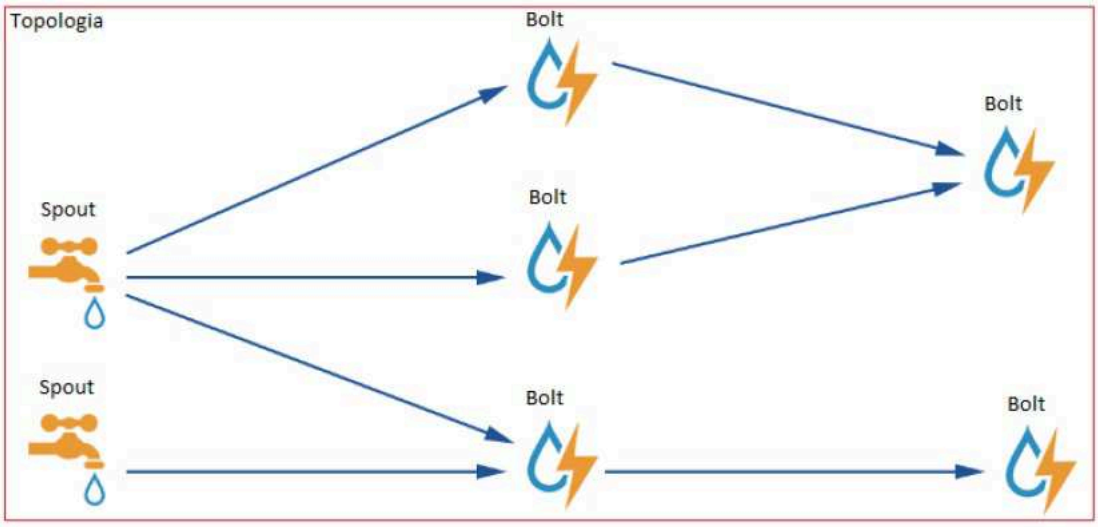

Figura 2.1: Componentes do Apache Storm [7]

2.2 Redis

A ferramenta Redis é um banco de dados NoSQL do tipo chave-valor, é open source, oferece alto desempenho pois é capaz de armazenar e manipular dados diretamente na memória, é escalável, consegue replicar dados de forma assíncrona e se necessário até persistir dados em disco. Contém diferentes tipos de estruturas de dados e também comandos para funções especificas que manipulam os dados [6].

Em um banco de dados de chave-valor, as informações são armazenadas no formato de conjunto de pares formados por uma chave e um valor. Este valor, ao ser gravado no banco de dados, é associado a esta chave que passa a ser o identificador único e exclusivo para este valor. Na Figura 2.2 é apresentado um exemplo de uma associação de chave-valor.

O Redis permite que através de chaves sejam mapeados diferentes tipos de estruturas de dados, entre eles: strings, listas, conjuntos, hash, conjuntos classificados. Existem também comandos para funções específicas que manipulam os dados, tais como: interseção de conjuntos,

Chave	Valor
Nome	João
Sobrenome	Da silva

Figura 2.2: Exemplo de chave-valor.

adicionar elemento em uma lista, recuperar o membro melhor ranqueado em uma lista. Estes comandos facilitam e até mesmo otimizam a transformação dos dados.

Devido a sua versatilidade, pode ser usado em diferentes casos de uso, tais como: banco de dados, cache de memória, gerenciamento de sessões, processamento de *streamings*, classificações em tempo real, entres outros. Devido às suas vantagens, este banco de dados é utilizado na ferramenta RIJOIN [7], auxiliando no manuseio dos dados e, devido a estas vantagens também será utilizado na ferramenta proposta neste trabalho, a RIJOIN+.

3 TRABALHOS RELACIONADOS

3.1 PPJOIN

Neste trabalho é proposto um método para filtrar os pares de registros candidatos a serem duplicatas. Para ser realizada a filtragem dos registros, são propostos três filtros que são combinados ao uso de agrupamento de registros por característica. Estes filtros são: filtro de posição, filtro de prefixo e filtro de tamanho [8]. Este método foi proposto para o processamento de dados no modelo estático (*batch*).

Inicialmente, são lidos todos os registros da base de dados e, durante este processo, é realizada a contagem da frequência de cada termo nos registros. Ao finalizar a contagem, os termos são ordenados de forma crescente, conforme a sua frequência e é gerado um número sequencial (id) para cada termo. Este número sequencial inicia em 1, sendo que este será atribuído ao termo com a menor frequência. Esta sequência termina em *n*, sendo *n* o número total de termos. Em seguida, os termos dos registros são substituídos pelo número sequencial atribuído a eles.

Após o processo de substituição ser concluído, os termos de cada registro são ordenados de forma crescente, utilizando como critério a frequência do termo. Este processo de ordenação é necessário para que sejam facilmente identificados os termos menos frequentes. Em seguida é criado o índice invertido. Para cada termo do registro é criada uma entrada no índice utilizando como chave o número sequencial do termo, sendo este procedimento realizado em todos os registros. Em cada entrada do índice é criado um bloco e neste bloco é inserido o número do registro que contém aquele termo.

Ao concluir a criação do índice invertido, a partir dos índices serão gerados os pares de registros. Ou seja, são realizadas as combinações (dois a dois) entre todos os registros deste bloco, tendo como resultado os pares de registros. Nesta combinação, serão gerados aproximadamente x^2 pares de registros, sendo *x* o número de elementos deste bloco.

Após serem gerados todos os pares é iniciada a etapa de filtragem nos pares de registros. Nesta etapa são aplicados três filtros sobre os registros.

O primeiro filtro aplicado nos registros é o filtro de tamanho. Neste filtro é avaliado, com base em um limiar definido, se a diferença de tamanho entre os registros respeita este limiar. Caso respeite, o par de registros é aceito e enviado para o próximo filtro.

A Figura 3.1 apresenta um exemplo do filtro de tamanho. Sendo o primeiro registro composto por 8 termos e o segundo registro composto por 5 termos, a diferença de tamanho entre os registros é de 3 termos. Para que estes registros sejam aceitos por este filtro, é necessário que esta diferença não seja maior que o valor do limiar.

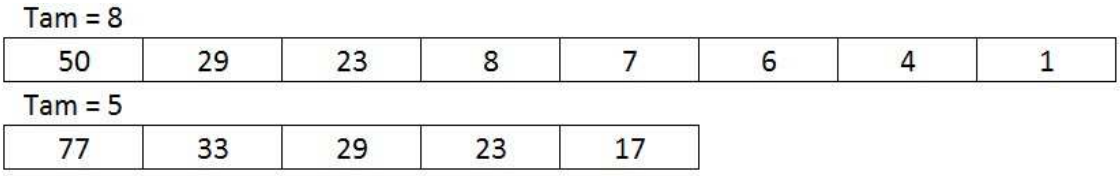

Figura 3.1: Exemplo do funcionamento do filtro de tamanho [7]

O segundo filtro aplicado nos registros é o filtro de prefixo. Com o uso de um limiar, é calculado o tamanho do prefixo para cada registro. Em seguida, é verificado se entre estes prefixos existe pelo menos um termo em comum. Se esta condição é válida, este par de registros é aceito e enviado para o próximo filtro.

A Figura 3.2 apresenta um exemplo do uso do filtro de prefixo. Após ser calculado o tamanho do prefixo, que neste caso será o mesmo para ambos os registros, é verificada a quantidade de termos em comum nestes registros. Neste exemplo, a regra de ao menos existir um termo em comum é atendida. Logo, este par de registros é aceito pelo filtro.

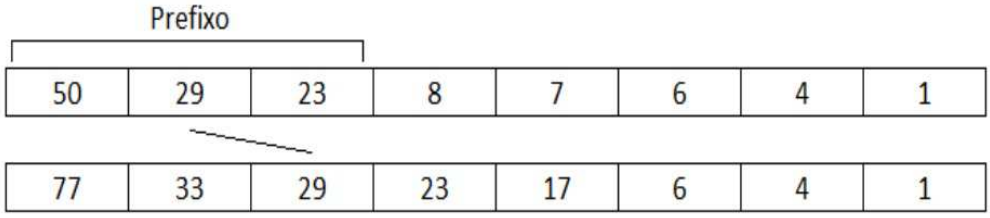

Figura 3.2: Exemplo do uso do filtro de prefixo [7]

O terceiro filtro aplicado nos registros é o filtro de posição. Neste filtro, é avaliado se a distância do termo em comum entre os registros é inferior ao valor encontrado na multiplicação entre o tamanho dos registros e um limiar definido. Esta distância é a diferença da posição do termo em comum nos registros. Se esta distância respeitar o limite calculado, o par de registros será aceito. Utilizando como base os registros da Figura 3.2, a distância entre o termo em comum dos registros é 1. Se o valor encontrado entre a multiplicação do tamanho dos registros e o limiar for 1, este par de registros será aceito.

3.2 PPJOIN+

Este método é um aprimoramento do método PPJOIN em [8]. Neste método é proposto um filtro chamado de filtro de sufixo, sendo este filtro complementar aos filtros de prefixo e o de posição. É considerado sufixo do registro todos os termos que não foram classificados como prefixo, como pode ser observado na Figura 3.3. O objetivo deste filtro é verificar o quão similares estes registros podem ser.

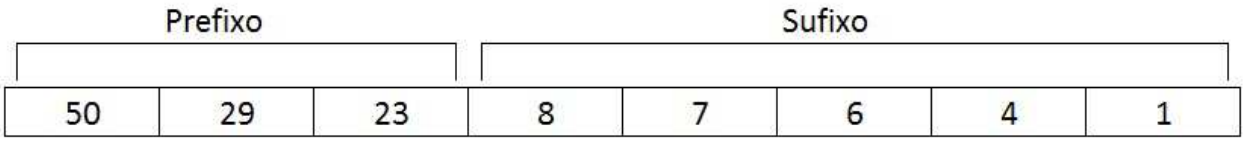

Figura 3.3: Exemplo de uso do filtro de sufixo

Para validar os registros, é calculado um limite (*threshold*) mínimo de similaridade entre os registros, com o auxílio da restrição de similaridade de Jaccard [8]. Para verificar os termos do sufixo dos registros, é utilizada a estratégia de busca do tipo divisão e conquista (busca binária). Para equilibrar o tempo de execução desta busca, um limite máximo de recursões é determinado heuristicamente [8].

3.3 RIJOIN

No trabalho [7], é apresentada uma ferramenta desenvolvida na plataforma de tempo real Apache Storm. Esta ferramenta, no contexto da deduplicação de dados, tem como objetivo o agrupamento de dados e com o auxílio de filtros, a detecção de pares de registros candidatos a serem duplicatas. Basicamente esta ferramenta é baseada na proposta do PPJOIN [8], mas aplicada no contexto de processamento de dados em tempo real.

A topologia desta ferramenta, apresentada na Figura 3.4, é composta por um spout e quatro bolts. Logo abaixo serão apresentados tais componentes e a sua função nesta ferramenta.

O objetivo de cada componente nesta ferramenta é:

- Spout: ler o arquivo de dados e enviar os registros para o primeiro bolt;
- Bolt 1: gerar os blocos de registros;
- Bolt 2: receber um bloco de registros e o termo chave deste bloco, manter estes registros ordenados e gerar os pares de registros para serem filtrados nos bolts seguintes;

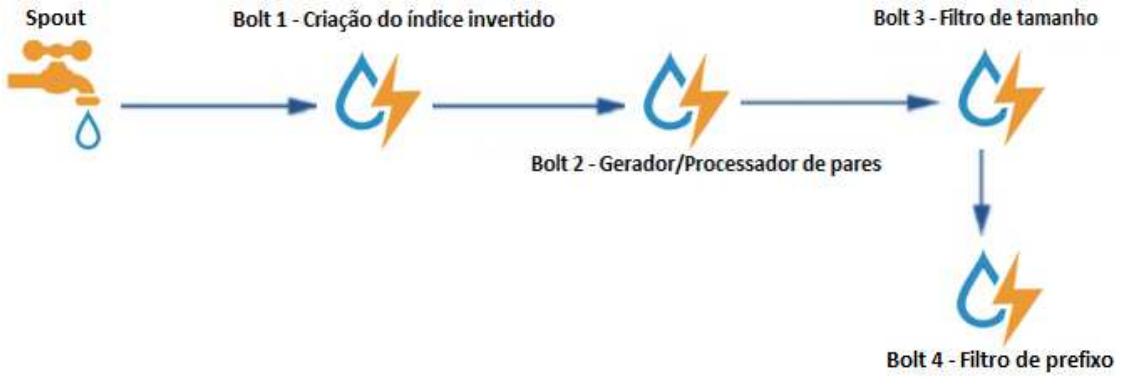

Figura 3.4: Topologia do RIJOIN [7].

- Bolt 3: analisar a diferença de tamanho entre os registros pertencentes ao par;
- Bolt 4: analisar o prefixo dos registros pertencentes ao par e identificar se os mesmos são semelhantes.

Primeiramente, ao ler cada registro, o spout envia o mesmo para o componente seguinte. Esta ação simula o efeito de que a aplicação está recebendo os dados de forma *online*.

Ao receber o registro, o primeiro bolt armazena o mesmo em uma tabela de registros. Em seguida, para cada termo deste registro, é contabilizada a sua frequência global e é criada uma chave (caso não exista esta chave) na tabela de índice invertido, utilizando o próprio termo como chave. Após este processo, o código deste registro que está sendo processado é adicionado no bloco referente a este termo.

Após este processamento, são enviados para o próximo bolt todos os blocos do índice invertido que possuem como chave algum termo deste registro.

Com o auxílio da Figura 3.5, será exemplificado o processo de recebimento de um registro, criação do índice invertido e a contagem de frequência dos termos deste registro. Ao receber o registro, apresentado no topo da figura, para cada termo do registro é contabilizada a atualizada a sua frequência. A tabela no lado esquerdo da figura mostra como estavam as frequências antes das mesmas serem contabilizadas e, no lado direito da figura, o resultado após a atualização das frequências dos termos.

Após terminar esse processo, para cada termo deste registro, é adicionado o número do registro no seu respectivo bloco do índice invertido. Na Figura 3.6, tomando como base o registro presente na Figura 3.5, o id “1138” do registro foi adicionado nos blocos referentes aos termos presente neste registro, sendo estes os termos: “20”, “Felipe”, “Martins” e “9999-9999”.

No segundo bolt, para cada termo do registro mais recente do bloco (último registro que

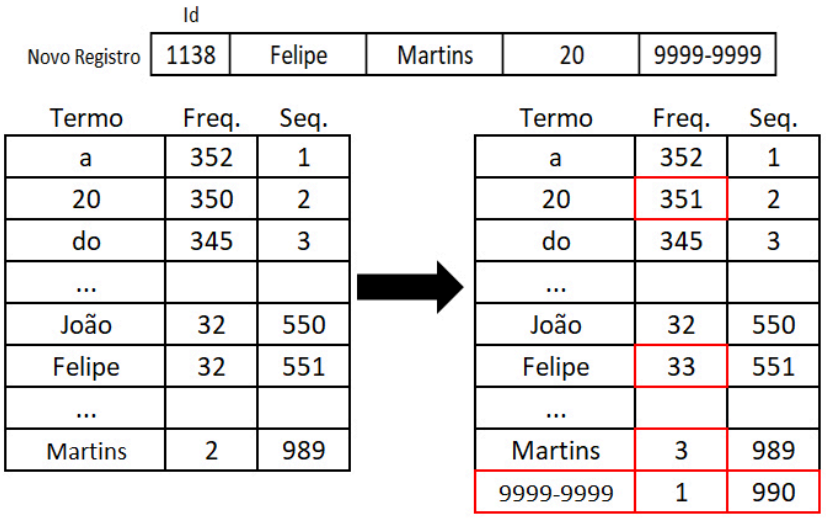

Figura 3.5: Exemplo da atualização da frequência dos termos de um novo registro [7].

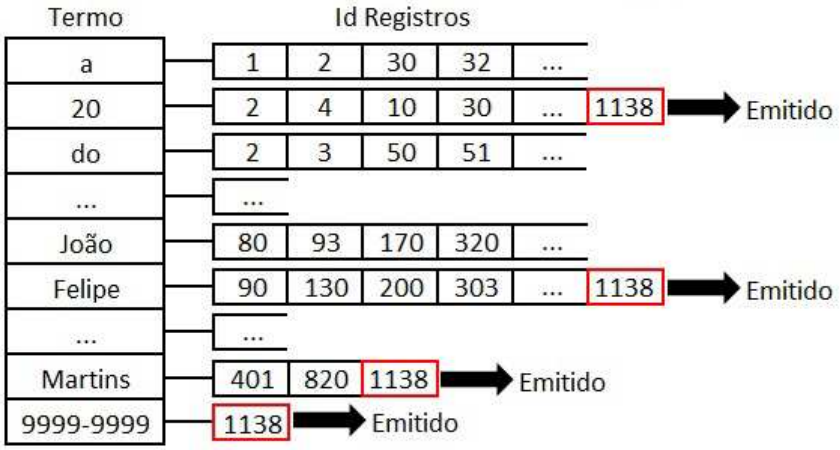

Figura 3.6: Exemplo de inserção nos blocos do índice invertido [7].

foi adicionado a este bloco) é realizada a busca da sua posição na lista ordenada. Em seguida, são ordenados os termos do registro de forma não crescente, utilizando como critério a frequência global de cada termo. Após esta ordenação, é atualizado este registro na tabela de registros. Esta atualização facilita e agiliza as próximas ordenações, pois o registro estará preordenado na próxima vez em que for processado por este bolt. Assim, se este bloco é formado por mais de um registro, este processo será realizado para cada registro que compõem o bloco que foi enviado para este bolt.

Durante o processo de ordenação dos demais registros deste bloco, são formados os pares de registros. Cada par é formado pelo registro recentemente adicionado ao bloco e um dos demais registros deste bloco. Assim, no final do processo, cada um dos demais registros contido neste bloco formará um par com o registro recentemente adicionado. Após formar cada par, este bolt envia o mesmo e o termo chave do bloco para o próximo bolt.

Na Figura 3.7 é exemplificado o processamento que ocorre no segundo bolt. Ao ser inserido o id "1138" no bloco de registros formado pela chave "Martins", os termos do registro representado por esse id são ordenados conforme a sua frequência global. Em seguida, são gerados os pares de registros, sendo estes o par [1138,401] e o par [1138,820].

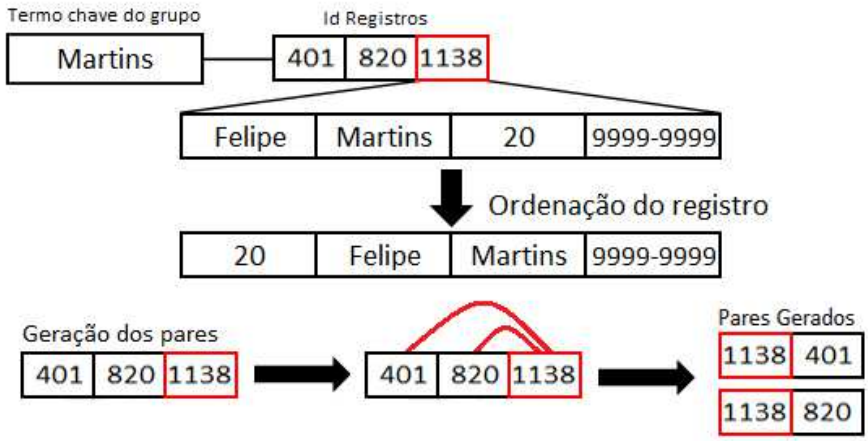

Figura 3.7: Exemplo da ordenação dos termos do registro e a criação dos pares de registros [7]

Ao chegar ao terceiro bolt, é iniciada a etapa de filtragem. Ao receber um par de registros, utilizando como critério o tamanho do registro, é verificado qual dos registros é o maior e qual é o menor entre os mesmos. Para este par ser aceito, o tamanho do menor registro deve ser no mínimo igual ao valor obtido da multiplicação entre o tamanho do maior registro e um limiar definido na aplicação. Se este critério for atendido, este par é enviado para o próximo bolt.

Após ser aceito e enviado, este par de registros será novamente filtrado no quarto bolt. Com o auxílio de um limiar, é calculado o tamanho do prefixo de cada registro do par recebido. Em seguida, é verificada a quantidade de termos em comum entre os prefixos. Se esta quantidade for maior do que o limiar, este par é aceito pela ferramenta como par candidato a ser duplicata de registro.

Esta ferramenta apresenta uma limitação, pois, é necessário que a cada novo registro sejam atualizadas as frequência globais dos termos. Assim, para manter os registros atualizados, é necessário que sejam reordenados. Assim, o processo de criação dos pares de registros é o mais custoso, no quesito de tempo de processamento e memória auxiliar, entre as tarefas realizadas por esta ferramenta.

Para solucionar este problema, neste trabalho é proposta uma modificação neste processo de criação dos pares de registros com o objetivo de reduzir o número de ordenações

realizadas nos blocos de registros durante o processo de ordenação dos termos e criação dos pares de registros.

4 PROPOSTA - FERRAMENTA *RIJOIN+*

Neste capítulo serão descritos os detalhes sobre as melhorias propostas e implementadas na ferramenta RIJOIN [7], chamada de RIJOIN+. Para este trabalho, foram realizadas modificações na ferramenta base deste trabalho, sendo realizadas na etapa da criação e uso do índice invertido e a implementação do filtro de sufixo, tendo como o base o método proposto em PPJOIN+ [8].

4.1 Limiares

Para o funcionamento desta ferramenta, assim como em [7], são utilizados limiares nos filtros para auxiliar na verificação dos pares de registros, sendo este limiar global G, comum a todos os bolts (variando entre 0.1 a 0.9). Durante a descrição do funcionamento de cada bolt, será descrito o papel deste limiar.

4.2 Ferramenta RIJOIN+

Para a modificação na etapa de criação e uso do índice invertido, foram realizadas alterações no primeiro e no segundo bolt desta ferramenta. Neste novo método proposto, a frequência global do termo chave do bloco será contabilizada no segundo bolt, diferentemente do processo que ocorre na ferramenta RIJOIN em [7], na qual este processo ocorre no primeiro bolt. Esta alteração permite que um controle sobre a ordenação do registro possa ser realizado. Para tal controle, antes de incrementar a frequência do termo, é realizada uma busca na tabela de frequência global, tendo como objetivo obter a posição atual do termo na tabela. Após este processo, é incrementada a frequência global da chave e, novamente buscada a posição do termo na tabela. Assim, caso a posição do termo for diferente da posição inicial, o registro será ordenado, caso contrário, os registros destes blocos não serão ordenados.

A segunda modificação proposta neste trabalho é a implementação do filtro de sufixo, tendo como base o apresentado no método PPJOIN+ [8]. Para esta modificação, foi incluído a topologia da ferramenta RIJOIN [7] um bolt, sendo este após o bolt de filtro de prefixo.

Assim, a ferramenta proposta neste trabalho será composta por 1 spout e 5 bolts. A topologia do Storm para este trabalho é apresentada na Figura 4.1.

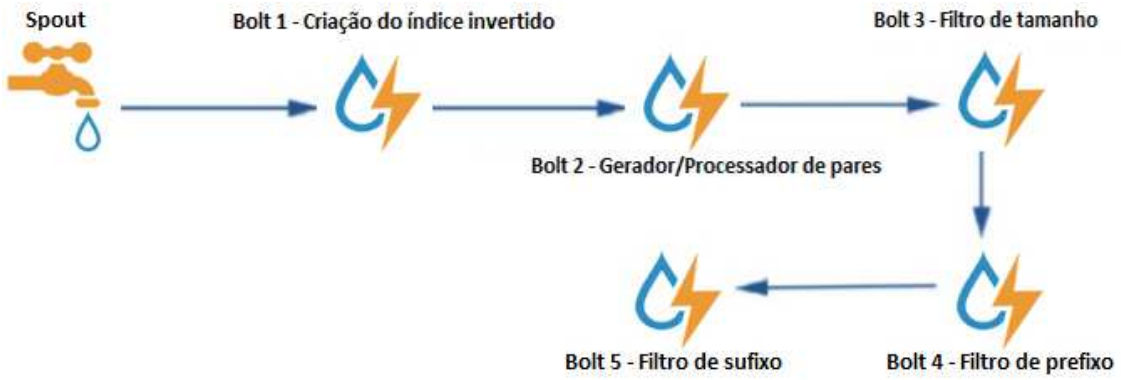

Figura 4.1: Topologia do RIJOIN+.

O objetivo de cada componente nesta ferramenta é:

- Spout: ler o arquivo de dados e enviar os registros para o componente seguinte da topologia;
- Bolt 1: gerar os blocos de registros;
- Bolt 2: receber um bloco de registros e o termo chave deste bloco, contabilizar a frequência global dos termos, manter estes registros ordenados e gerar os pares de registros para serem filtrados nos bolts seguintes;
- Bolt 3: verificar a diferença de tamanho entre os registros do par;
- Bolt 4: verificar o prefixo dos registros do par recebido e analisar se os mesmos são semelhantes;
- Bolt 5: verificar o sufixo dos registros do par recebido e identificar se os mesmos são semelhantes.

Primeiramente, ao ler um registro, o spout envia o mesmo para o componente seguinte da topologia, o primeiro bolt. Esta ação simula o efeito de que a aplicação está recebendo os dados de forma *online*.

Ao receber o registro, o primeiro bolt armazena o mesmo em uma tabela de registros e, em seguida, para cada termo deste registro é criada uma chave (caso não exista esta chave) na tabela de índice invertido, utilizando o termo como chave. Após este processo, o código do registro é adicionado no bloco referente a este termo.

Após este processamento, são enviados para o próximo bolt os blocos do índice invertido que foram atualizados pelos termos deste registro.

No segundo bolt, ao receber o bloco de registros, é realizada a verificação para que seja avaliado se será realizada a ordenação dos registros deste bloco. Se não forem efetuadas as ordenações, apenas serão formados os pares e estes enviados para o próximo bolt. Caso sejam realizadas as ordenações, este processo é iniciado pelo último registro que foi adicionado a este bloco. Nesta ordenação, os termos de cada registro são ordenados de forma não crescente, utilizando como critério a frequência global de cada termo. Após esta ordenação, é atualizado este registro na tabela de registros. Esta atualização facilita e agiliza as próximas ordenações, pois o registro estará preordenado na próxima vez em que for processado por este bolt. Assim, se este bloco é formado por mais de um registro, este processo será realizado para cada registro que compõem o bloco que foi enviado para este bolt.

Durante este processo de ordenação, cada registro pertencente a este bloco, após ser ordenado, formará um par com o último registro que foi inserido neste bloco e será enviado para o bolt seguinte. Assim, para o terceiro bolt será enviado um par de registros e o termo chave do bloco a qual pertencem.

Ao chegar ao terceiro bolt, serão filtrados os registros utilizando como critério o tamanho dos mesmos. Nesta validação, primeiramente é identificado qual dos registros é o maior entres os mesmos. Para este par ser aceito, o tamanho do menor registro deve ser no mínimo igual ao valor obtido da multiplicação entre o tamanho do maior registro e um limiar global G definido na aplicação. Se este critério for atendido, este par é enviado para o próximo bolt. Para calcular o tamanho do prefixo é utilizada a Equação 4.1, sendo *n* o tamanho do maior registro e *G* o limiar global.

$$T = (n * (1 - G)) \tag{4.1}$$

No quarto bolt, este par de registros será novamente filtrado pelo prefixo do registro. Com o auxílio de um limiar, é calculado o tamanho do prefixo de cada registro do par recebido. O cálculo do tamanho prefixo é definido na Equação 4.2, sendo *n* o tamanho do registro e *G* o limiar global.

$$Prefixo = \lceil n * \frac{G}{2} \rceil \tag{4.2}$$

Após calcular o tamanho do prefixo, é calculada a quantidade mínima de termos *L* em comum entre os registros. Primeiramente, é verificado qual dos registros possui o maior tamanho. Em seguida, é multiplicado o tamanho do maior registro pelo limiar global G. Para ser

aceito, este par de registro devem conter no mínimo *L* elementos em comum, caso contrário o par de registros será rejeitado. Se aceito, este par de registros e o tamanho de seus respectivos prefixos são enviados para o próximo bolt.

No último bolt, o filtro do sufixo, ao receber o par de registros e o tamanho do prefixo de cada elemento do par, é calculado o tamanho do sufixo de cada registro. Para obter o tamanho do sufixo, é subtraído do tamanho total do registro o tamanho do seu prefixo.

Em seguida é estipulado o valor mínimo de semelhança entre os registros, chamado de *overlap*. Este *overlap* α é usado como referência para decidir se o par de registros será aceito como candidato a duplicata de registro. Para estipular este valor é utilizada a Equação 4.2, sendo *n* tamanho do primeiro registro e *G* o limiar global.

$$Overlap = \lfloor n * \frac{G}{2} \rfloor \tag{4.3}$$

Para a próxima etapa da validação deste filtro, será realizada uma busca no sufixo do segundo registro tendo como objetivo encontrar um termo pivô *A* contido no sufixo do primeiro registro. Para esta busca é utilizado o método de busca binária. Após calculado o tamanho do sufixo de cada registro, para o primeiro registro, é calculada a posição do termo central do sufixo. Em seguida, será realizada a busca no sufixo do segundo registro tendo como objetivo encontrar o termo *A* no sufixo do segundo registro, e caso encontre, retornar a posição deste termo dentro do segundo registro. Caso não encontre este termo, esta busca será realizada até atingir o nível de recursão *z* definido na ferramenta.

Em seguida, após ser realizada a busca, são verificados os prefixos deste par de registros e contabilizados os termos em comum. Após esta verificação, é calculado o *overlap* de semelhança entres os registros. Para o cálculo deste *overlap*, é utilizada a Equação 4.4, apresentada em [8], sendo I a quantidade de termos em comum entre prefixos dos registros, P a posição do termo pivô do sufixo do primeiro registro no segundo registro, T o tamanho do prefixo do segundo registro, U o tamanho do primeiro registro e V a posição do termo pivô no primeiro registro.

$$OverlapReg = I + (P - T) + 1 + (U - V) \tag{4.4}$$

Após calcular o *overlap* do registro, é verificado se este é no mínimo igual ou superior ao *overlap* α. Caso esta validação seja verdadeira, este par de registros é aceito como candidatos a duplicata de registro.

5 EXPERIMENTOS

Neste capítulo são descritos os experimentos realizados com o método proposto neste trabalho.

5.1 Base de Dados

Para serem realizados os testes, foram criadas bases de dados genéricas X e Y utilizando a ferramenta Febrl [1]. Estas bases de dados foram criadas seguindo o padrão dos atributos e padrão de duplicatas utilizadas no trabalho [7]. A quantidade de registros para a base X será de 1.000.000 (um milhão) de registros e a base Y terá 3.000.000 (três milhões) de registros. Cada uma destas bases, assim como no trabalho [7], terá 10% de seus registros sendo duplicata de algum outro registro. Assim, a base X terá 100.000 (cem mil) duplicatas e a base Y terá 300.000 (trezentas mil) duplicatas.

5.2 Métricas

As métricas utilizadas para avaliar o desempenho do método RIJOIN+ foram a precisão e a revocação.

A precisão representa a porcentagem de elementos cujo valor é relevante a classificação e que foram corretamente classificados. Para o cálculo da precisão é utilizada a Equação 5.1, sendo *TP* o número de pares aceitos corretamente e *FP* o número de pares que não são duplicatas e que foram aceitos pela ferramenta.

$$Prec = \frac{TP}{TP + FP} \tag{5.1}$$

A revocação representa a porcentagem de elementos relevantes que foram corretamente classificados como verdadeiros. Para realizar o cálculo da revocação é utilizada a Equação 5.2, sendo *TP* o número de pares aceitos corretamente e *FN* o número de pares de registros que não foram aceitos pela ferramenta.

$$Rev = \frac{TP}{TP + FN} \tag{5.2}$$

5.3 Configurações

Para realização dos experimentos propostos neste trabalho, foram utilizados o Apache Storm em sua versão 0.9.6 e o banco de dados Redis na versão 3.2.2. O computador utilizado para realizar os experimentos possui o processador Intel Core I7-3770 3.4 Ghz, 8 GB de memória RAM e sistema operacional Ubuntu 16.04 LTS 64 bits.

5.4 Eficácia

Nas Figuras 5.1 e 5.2 são demonstrados os resultados da execução da ferramenta *RI-JOIN+*, sendo a Figura 5.1 e 5.2 referentes as bases de dados base *X* e *Y*, respectivamente. Nestes gráficos, o eixo *x* representa o limiar global, o qual varia do 0.1 ao valor 0.9, e o eixo *y* é o valor percentual obtido. Para avaliar os resultados destes testes foram utilizadas as métricas de precisão e revocação descritas na Seção 5.2.

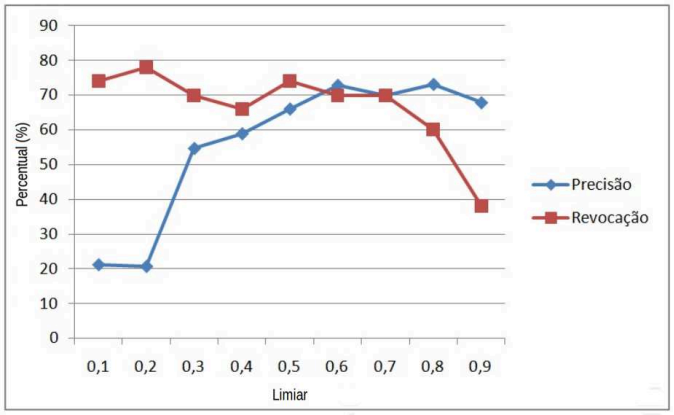

Figura 5.1: Gráfico do resultado da precisão e revocação da base *X*.

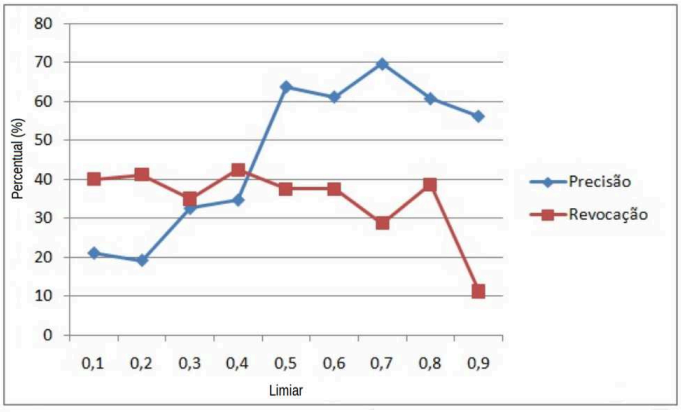

Figura 5.2: Gráfico do resultado da precisão e revocação da base *Y*.

Em todos os testes realizados o valor da precisão, a partir do limiar 0.3 se mostrou promissor, sendo este um limiar baixo, no qual registros com pouca similaridade são facilmente aceitos pela ferramenta como supostos pares de duplicatas.

É possível também observar que os valores da precisão se comportaram de forma parecida a partir do limiar 0.5. A revocação apresentou bons resultados, principalmente nos testes realizados na base *X*.

Em comparação aos resultados obtidos no trabalho [7], na métrica de precisão, o maior percentual entres os dois testes realizados pela ferramenta RIJOIN+ foi de um pouco mais de 70%. Já para a ferramenta RIJOIN [7], foi de 86.56%. Em relação à métrica de revocação, a ferramenta RIJOIN [7] se mostrou mais consistente nos testes, sendo que manteve em todos os testes, do limiar 0.1 ao limiar 0.5, quase 100% de revocação. Na ferramenta proposta neste trabalho, a melhor marca alcançada foi de aproximadamente 80%.

Para a base de dados *X*, o valor da revocação e da precisão se mantiveram praticamente acima dos 50% em todos os limiares.

Na base de dados *Y*, do limiar 0.1 ao limiar 0.6 a revocação ficou entre 30% e 50%

Nos experimentos, foi constatado que ao adicionar o processo de verificação da lista com as frequências dos termos, que tem o objetivo de avaliar se os registros serão ordenados ou não, fez com que houvesse uma redução de 16% no número de ordenações realizadas.

5.5 Eficiência

Nesta seção serão demonstrados os resultados do método em relação ao tempo de execução para as bases de teste e comparar estes aos resultados dos testes do trabalho [7].

Bases de Dados	Quantidade de registros	Tempo Médio de Execução dos Testes
X	1100	24.47s
Y	5500	134.7s
Z	11000	284.1s

Figura 5.3: Tabela de tempo de execução dos testes com a ferramenta RIJOIN+.

Para comparar a eficiência entre a ferramenta RIJOIN+ e a ferramenta RIJOIN [7], foram criadas bases de dados utilizando a ferramenta Febrl [1]. Tais bases de dados possuem a mesma quantidade de registros que as bases utilizadas para os testes na ferramenta RIJOIN [7]. Tais bases, X, Y e Z, são constituídas de 1100 registros, 5500 registros e 11000 registros,

respectivamente.

Na figura são apresentados os tempos obtidos nos testes realizados na ferramenta RIJOIN+ e os tempos dos testes da ferramenta RIJOIN, apresentados no trabalho [7].

É possível observar que em todos testes, o tempo de execução da ferramenta RIJOIN+ foi superior ao tempo obtido com a ferramenta RIJOIN [7]. Apenas no primeiro teste, realizado com a base *X*, o tempo obtido foi próximo ao tempo do teste para a mesma base realizado para a ferramenta RIJOIN [7].

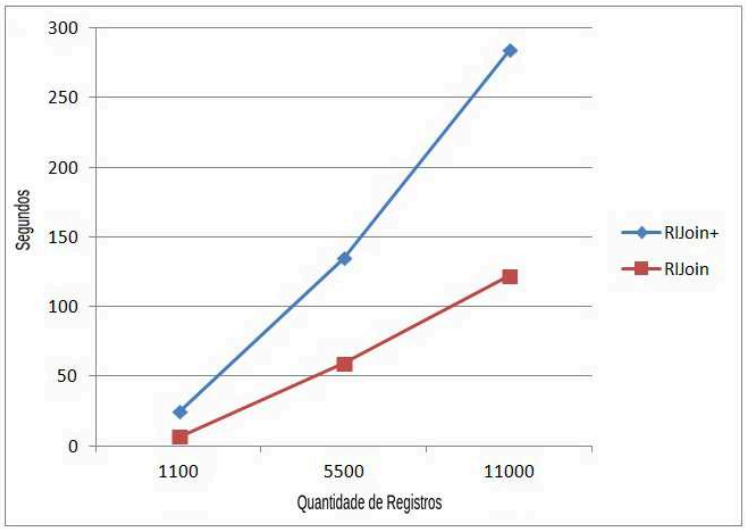

Figura 5.4: Tabela de tempo de execução dos testes com a ferramenta RIJOIN+ e RIJOIN [7].

Esta diferença de tempo pode ser atribuída ao filtro de sufixo, pois para realizar a tarefa de validação dos pares de registros, através da verificação do *overlap* do registro, são realizadas buscas de forma sequencial entre os termos dos prefixos de ambos os registros. Realizando no pior caso a comparação entre todos os termos dos registros, aumentando consideravelmente o tempo de processamento deste bolt e afetando o tempo total do processamento da base de dados pela ferramenta RIJOIN+.

6 CONSIDERAÇÕES FINAIS

Neste trabalho são propostas melhorias na ferramenta de blocagem de dados RIJOIN [7]. Esta nova ferramenta, chamada de RIJOIN+ é implementada na plataforma de processamento em tempo real Apache Storm e em seu funcionamento utiliza o banco de dados Redis para operar as informações. Nesta ferramenta foi adicionado o filtro de sufixo e realizadas alterações no processo de criação dos pares de registros.

A ferramenta RIJOIN+ apresentou promissores resultados para a métrica de precisão. Para a métrica de revocação, apresentou resultados consideráveis. As alterações propostas para o controle de reordenação dos registros, com o objetivo de reduzir a quantidade de reordenações, também apresentaram resultados promissores.

Como trabalho futuros e possíveis melhorias a serem desenvolvidas na ferramenta RIJOIN+:

- *Reduzir as ordenações de registros*: para cada bloco emitido para o segundo bolt, todos os registros pertencentes a este bloco são verificados e ordenados, o que aumenta o custo da perfomance deste componente e de toda a ferramenta. O controle desenvolvido neste método trouxe melhoras no quesito performance, mas refletiu no resultado. Para tal tarefa, será necessário desenvolver um controle para evitar que verificações e re-ordenações desnecessárias sejam realizadas;

- *Melhorar a busca no filtro do prefixo*: no processo de verificação da quantidade de termos em comum entres os prefixos, no filtro de prefixo, é realizada busca termo a termo, o que acaba aumentando o tempo de processamento deste bolt. Uma solução para este problema seria aplicar a busca utilizada para o sufixo também ao prefixo do registro;

- *Melhorar a busca no filtro do sufixo*: no filtro do sufixo, a verificação da quantidade de termos em comum entres os prefixos é realizada através de uma busca termo a termo, o que acaba aumentando o tempo de processamento deste bolt. Uma solução para este problema seria aplicar a busca utilizada para o sufixo também ao prefixo do registro;

- *Aplicação da ferramenta*: desenvolver um deduplicador de dados em tempo real no *Apache Storm* e utilizar o método *RIJOIN+*, proposto neste trabalho, no processo da blocagem dos dados deste deduplicador.

REFERÊNCIAS

[1] P. Christen. Febrl: a freely available record linkage system with a graphical user interface. *In Proceddings of the second Australasian workshop on Health data and Knowledge Management - Volume 80, pages 17-25. Australian Computer Society, Inc.*, 2008.

[2] M. Dutch. Understanding data deduplicatios ratios, Junho 2008.

[3] I. P. Fellegi and A. B. Sunter. A theory for record linkage. *Journal of the American Statistical Association*, Vol. 64, 1969. p. 1183–1210.

[4] A. S. Foundation. storm.apache.org, 2018.

[5] A. Jain and N. Anand. *Learning Storm*. Packt Publishing Ltd., Agosto 2014.

[6] Redis. redis.io, 2018.

[7] M. H. Ulrich. Rijoin - Uma ferramenta para o agrupamento de dados utilizando processamento online. 2017.

[8] C. Xiao, W. Wang, X. Lin, J. X. Yu, and G. Wang. Efficient similarity joins for near duplicate detection. *ACM Trans. Datab. Syst.V, N, Article A*, 2011.