PRISCILLA KELLY MACHADO VIEIRA AZEVÊDO

Um Processo Incremental e Orientado à Consulta para Resolução de Entidades em Sistemas de Integração de Dados

Tese de Doutorado

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
www.cin.ufpe.br/~posgraduacao

RECIFE
2017

PRISCILLA KELLY MACHADO VIEIRA AZEVÊDO

Um Processo Incremental e Orientado à Consulta para Resolução de Entidades em Sistemas de Integração de Dados

Tese apresentada ao Centro de Informática da Universidade Federal de Pernambuco como parte dos requisitos para obtenção do título de Doutora em Ciência da Computação.

Orientadora: Profª. Ana Carolina Salgado

Coorientadora: Profª. Bernadette Farias Lóscio

RECIFE
2017

Catalogação na fonte
Bibliotecária Elaine Cristina de Freitas CRB4-1790

A994p Azevêdo, Priscilla Kelly Machado Vieira
Um Processo Incremental e Orientado à Consulta para Resolução de Entidades em Sistemas de Integração de Dados/ Priscilla Kelly Machado Vieira Azevêdo. – 2017.
131f.: fig., tab.

Orientadora: Ana Carolina Brandão Salgado
Tese (Doutorado) – Universidade Federal de Pernambuco. CIn. Ciência da Computação, Recife, 2017.
Inclui referências.

1. Resolução de Entidades 2. Integração de Dados 3. Duplicação de dados. I. Salgado, Ana Carolina Brandão (Orientadora) II. Titulo.

005.74 CDD (22. ed.) UFPE-MEI 2017-279

Priscilla Kelly Machado Vieira

Um Processo Incremental e Orientado a Consulta para Resolução de Entidades em Sistemas de Integração de Dados

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Doutora em Ciência da Computação.

Aprovado em: 27/11/2017.

_____

Orientadora: Profa. Dra. Ana Carolina Brandão Salgado

BANCA EXAMINADORA

_____

Prof. Dr. Ricardo Bastos Cavalcante Prudêncio
Centro de Informática / UFPE

_____

Prof. Dr. Luciano de Andrade Barbosa
Centro de Informática / UFPE

_____

Prof. Dr. Altigran Soares da Silva
Departamento de Ciência da Computação / UFAM

_____

Prof. Dr. Carlos Eduardo Santos Pires
Departamento de Sistemas e Computação / UFCG

_____

Profa. Dra. Damires Yluska de Souza Fernandes
Instituto Federal da Paraíba / Campus João Pessoa

Dedico este trabalho:

Aos meus pais, ***Magna*** *e* ***Ricardo****, por todo amor, atenção e ensinamentos passados em minha vida.*

Ao meu esposo, **Wilker.** Sem seu amor incondicional, suporte e sacrifício, esta tese não seria possível.

Agradecimentos

A ajuda recebida foi imprescindível para a realização deste doutorado. Portanto, gostaria de externar sinceramente meus agradecimentos a todos que contribuíram para a conclusão desta etapa.

À Deus, por me dar forças para superar os obstáculos que surgem no caminho para se atingir objetivos.

Aos meus pais, José Ricardo e Magna Maria, pelo apoio e suporte em todos os momentos desta jornada.

Ao meu grande amigo e esposo, Wilker Victor, por todo incentivo, compreensão e paciência demonstrados ao longo deste longo período de dedicação aos estudos.

Aos colegas do laboratório no Cin, por todo companheirismo, apoio e dedicação, os quais foram fundamentais para o desenvolvimento do trabalho.

Às professoras Ana Carolina e Bernadette Farias, pelas discussões, comentários e orientação valiosa, além da dedicação e incentivo ao aperfeiçoamento profissional.

À Universidade Federal Rural de Pernambuco, Unidade Acadêmica de Garanhuns, por ter me concedido um afastamento de 1 (um) ano para que fosse possível a minha dedicação plena à esta pesquisa de doutorado.

A todos que, direta ou indiretamente, contribuíram para a concretização deste trabalho.

Resumo

A Resolução de Entidades (RE) é o problema de identificar grupos de tuplas (registros ou instâncias), em uma única ou múltiplas fontes de dados, que representam a mesma entidade do mundo real. Esta é uma etapa crucial do processo de integração de dados, que muitas vezes necessita integrar dados em tempo de consulta (*online*). Esta tarefa torna-se ainda mais onerosa quando são consideradas fontes dinâmicas e com grandes volumes de dados. Além disso, tais características, tornam o processo de RE mais desafiador, uma vez que a maioria das técnicas de RE (tradicional), processa todas as tuplas de uma única vez, ao invés de processar apenas as tuplas importantes para o usuário. Portanto, novas soluções são necessárias para contornar este problema. Neste trabalho é proposto um processo incremental e orientado à consulta para RE. O processo é considerado incremental porque a cada iteração um conjunto de novas tuplas é processado e adicionado às demais tuplas processadas previamente. O termo orientado à consulta é proveniente do fato do processo proposto ser aplicado apenas sobre resultados de consultas. As contribuições deste trabalho são: especificação, implementação e avaliação do processo proposto. O processo foi avaliado com diferentes algoritmos e sobre diferentes fontes de dados. Foram utilizadas medidas de qualidade e desempenho do processo. Observou-se que o processo proposto tem qualidade muito similar aos processos tradicionais de RE, contudo tem um desempenho melhor.

Palavras-chave: Resolução de Entidades. Integração de Dados. Dados Duplicados. Deduplicação

Abstract

The Entity Resolution (ER) is the problem of identifying groups of tuples (records or instances) from single or multiple data sources which represent the same real-world entities. ER is an essential step in data integration tasks, and it often demands to obtain results at query-time (online). Especially in settings containing dynamic data sources with large volumes of data, the ER process can be still more challenging. However, most traditional ER techniques process all tuples at once, instead of considering tuples based on a query. This lead to a need for solutions to get around this problem. This work proposes a query-driven incremental process for ER. In this case, incremental means that in each iteration phase, the currently processed tuples will increase the set of previous tuples. The term query-driven means that the process in each iteration considers only tuples regarding the query result. The contributions of this work are the specification, development, and evaluation of the proposed process. Regarding the evaluation, we have used it in existing algorithms on different data sources. We conclude that the use of previous results in ER tasks turns the process more efficient than comparing all pairs of tuples at query-time, without reducing the quality of results.

***Keywords**: Entity Resolution. Data Integration. Duplicate Data. Deduplication*

Lista de Ilustrações

Lista de Tabelas

Lista de Quadros

Lista de Algoritmos

Lista de Siglas

Sigla	Significado
RE	*Resolução de Entidades*
CG	*Cluster Global*
CL	*Cluster Local*
IC	*Índice de Clusters*
IS	*Índice de Similaridades*
QUIPER	*Query-time and Incremental Process for ER*

Sumário

1 INTRODUÇÃO

Um grande volume de dados é publicado e coletado diariamente por empresas e organizações governamentais no mundo todo. Estes dados são armazenados em múltiplas fontes e a necessidade de acesso e análise destes dados gera a demanda por estratégias de integração de dados [Lenzerini, 2011].

Integração de dados tem por objetivo combinar dados de diferentes fontes de dados, muitas vezes heterogêneas e autônomas, disponibilizando uma visão única para os usuários (i.e., pessoas ou sistemas), que necessitam de serviços de integração de dados. Neste processo, é possível que os mesmos dados estejam armazenados em fontes de dados diferentes, indicando dados duplicados. Por este motivo, no processo de integração de dados, a etapa de Resolução de Entidades (RE) é crucial [Elmagarmid, 2006; Dong, 2009; Christen, 2012].

A RE tem o objetivo de identificar se dois dados (tuplas, registros, objetos ou instâncias) representam a mesma entidade do mundo real. Uma entidade pode ser uma pessoa (por exemplo, cliente, paciente ou autor), um produto, um local, um negócio ou qualquer outro objeto que exista no mundo real. O conjunto de valores e atributos de uma entidade (por exemplo, paciente com nome Maria, Pedro ou João) denotamos por tupla. Neste trabalho, o processo de RE identifica tuplas que representam a mesma entidade do mundo real.

Por ser uma tarefa pesquisada em diferentes áreas do conhecimento, a RE tem diversas outras denominações: interligação entre registros (*record linkage*), referência entre objetos (*object reference*), interligação entre referência (*reference linkage*), deduplicação (*deduplication*), detecção de dados duplicados (*duplication data detection*).

Exemplos de tuplas duplicadas em múltiplas fontes podem ser: (i) Um paciente representado em fontes de dados de diferentes hospitais; (ii) Um mesmo produto representado em diferentes lojas; (iii) Um livro que foi representado de forma diferente em cada ocorrência. Estas ocorrências, se não forem detectadas e/ou resolvidas podem levar a graves erros em análises dos dados como, por exemplo, não recuperar todo o histórico médico de um paciente ou indicar um número errado de vendas de um produto.

Em muitos casos, empresas e organizações possuem fontes de dados dinâmicas (i.e., que são submetidas a operações de atualização, remoção e inserção de tuplas) e com um grande volume de dados. Estas características tornam o processo de RE mais desafiador, uma vez que a maioria das técnicas de RE processam todas as tuplas de uma única vez (processos tradicionais de RE) e, em um grande volume de dados, o processo de RE pode ser oneroso. Além disso, os processos tradicionais de RE são aplicados de forma *off-line*, considerando apenas fontes de dados estáticas (fonte de dados em um momento específico, sem considerar modificações) o que não é adequado para fontes de dados dinâmicas.

Para reduzir os custos do processo de RE e se adequar a cenários dinâmicos, a literatura apresenta soluções incrementais para RE [Gruenheid et al., 2014]. Neste caso, resultados de iterações prévias do processo de RE são reutilizados durante a comparação de novas tuplas. Além disso, para reduzir o volume de tuplas considerado em cada iteração do processo de RE, pode-se processar apenas um conjunto de tuplas selecionadas (por exemplo, resultados de consultas). Desta forma, o volume de tuplas avaliado no processo de RE é incrementado gradativamente, crescendo a cada novo resultado de consulta recebido [Bhattacharya et *al*., 2006; Bhattacharya et *al*., 2006; Altwaijry et *al*., 2014].

Por exemplo, considerando fontes de dados bibliográficas, quando o interesse é identificar todos os trabalhos da autora "Getoor", não é necessário aplicar a RE em todas as tuplas pertencentes às fontes bibliográficas disponíveis, e sim focar nas tuplas de "Getoor" que respondem a consulta.

Neste trabalho, é proposto um processo incremental e orientado à consulta para RE, denominado por QUIPER (*Query-time and Incremental Process for ER*). O QUIPER é orientado à consulta por ser aplicado sobre resultados de consultas executadas em múltiplas fontes de dados e incremental por armazenar resultados das suas etapas em execuções anteriores para serem reutilizados em iterações futuras do QUIPER, a fim de acelerar o processo de RE em tempo de consulta.

Alguns trabalhos da literatura investigam a RE para execução de consultas (recuperação de informações), recuperar todas as tuplas duplicadas que respondem a uma consulta específica [Ramadan et *al*., 2015], ou sobre resultados de consultas [Bhattacharya & Getoor, 2007; Su et *al*.,2010; Altwaijrt et *al*., 2014], que aplica a RE sobre os resultados

das consultas executadas em múltiplas fontes de dados. Não foram encontrados trabalhos que utilizam de forma incremental os resultados intermediários do processo de RE para serem reutilizados em futuros processos de RE em sistemas de integração de dados.

As principais contribuições deste trabalho são a especificação, implementação e avaliação do QUIPER. Os experimentos, realizados sobre duas fontes de dados reais e uma fonte de dados sintética, mostraram que o QUIPER é mais eficiente (tempo de processamento) que o processo tradicional de RE, e possui qualidade dos resultados próxima (precisão) aos observados no processo tradicional de RE.

1.1 Objetivos

Considerando a RE como parte integrante do processo de integração de dados, o QUIPER identifica tuplas duplicadas em resultados de consultas. Para minimizar o tempo de processamento, foi proposto o reúso de iterações de RE para identificar tuplas duplicadas em resultados de consultas futuras.

A fim de atingir o objetivo geral supracitado, os seguintes objetivos específicos foram definidos:

- Propor estratégias de indexação [Christen, 2012; Ramadan & Christen, 2014] viáveis para o QUIPER, que permitam reduzir o tempo de processamento do QUIPER em relação aos processos tradicionais de RE;

- Implementar e avaliar os índices propostos;

- Implementar algoritmos de clusterização [Christen, 2012] e comparar o tempo de processamento (eficiência) e a qualidade dos resultados obtidos em relação aos processos tradicionais de RE;

- Realizar experimentos com diferentes fontes de dados e com diferentes algoritmos de clusterização;

- Mostrar que o QUIPER aplica RE em resultados de consultas com qualidade semelhante aos obtidos por processos tradicionais de RE, mas com tempo de processamento menor;

- Mostrar que o QUIPER, após aplicar RE em um conjunto de resultados de consulta, que juntos representam todas as tuplas de uma fonte de dados, obtém qualidade semelhante ao processo tradicional de RE sobre todas as tuplas de uma fonte de dados.

1.2 Premissas e Hipóteses

Para a formalização e definição deste trabalho, considera-se um sistema de integração que adota uma abordagem virtual, i.e., as tuplas são recuperadas a partir das fontes de dados e integradas sob demanda.

Para este trabalho, foram assumidas as seguintes premissas:

Premissa 01: Todas as tuplas que respondem a uma determinada consulta foram recuperadas pelo sistema de integração.

Premissa 02: O mapeamento entre os esquemas das fontes de dados foi realizado previamente na etapa de correspondência entre esquemas do processo de integração de dados.

Premissa 03: Todos os grupos de tuplas (*cluster* de tuplas) são criados para um único conceito, por exemplo, agrupados por *Autor*, *Artigo* ou *Conferência*. Esta estratégia permite o maior reúso no processo de RE. Por exemplo, se uma consulta solicita informações sobre os conceitos *Autor* e *Artigo*, e para identificar um autor é necessário desambiguar os artigos, os dois conceitos são agrupados separadamente e o processo de RE combina os resultados. Em um outro momento, se a consulta requer apenas informações sobre o conceito *Autor*, os *clusters* de *Autor* identificados previamente podem ser reutilizados.

Premissa 04: Assumimos que as tarefas de pré-processamento dos dados foram realizadas previamente ao início da execução do QUIPER.

Premissa 05: As consultas realizadas em um sistema de integração são inter-relacionadas. Desta forma, é esperado que consultas consecutivas possuam interseções entre seus resultados.

Considerando o contexto no qual este trabalho está inserido, tal como suas premissas, a seguir são listadas as hipóteses desta pesquisa:

Hipótese 01: O processo de RE realiza comparação entre pares de tuplas. O uso do resultado de uma consulta reduz o número de tuplas a serem processadas, atenuando, portanto, o número de comparações a serem realizadas e, por conseguinte, o tempo de processamento.

Hipótese 02: A cada novo resultado de consulta a ser processado, o QUIPER pode recuperar resultados das etapas do processo de RE em iterações anteriores e/ou realizar a RE em tempo de consulta (incrementando ao resultado recuperado as tuplas não processadas em iterações anteriores). Esta opção pode ter uma precisão igual ou muito próxima aos observados nos processos tradicionais de RE, os quais processam todas as tuplas em tempo de consulta.

Hipótese 03: O uso de estratégias de indexação pode interferir positivamente no reúso dos resultados das etapas do processo de RE realizadas em iterações prévias, reduzindo as tuplas a serem reprocessadas a cada resultado de consulta e reduzindo o tempo de processamento.

Hipótese 04: O reúso das iterações prévias do QUIPER para a RE em novos resultados de consultas permite um menor tempo de processamento se comparado aos tempos de processamento do processo tradicional de RE.

Hipótese 05: A precisão dos resultados do QUIPER após o processamento de um conjunto de tuplas que juntas representam todas as tuplas de uma fonte de dados é igual ou muito próxima à precisão dos resultados do processo tradicional de RE sobre toda a fonte de dados em um único momento.

1.3 Estrutura do Trabalho

O conteúdo deste trabalho está distribuído em 6 (seis) capítulos. No Capítulo 1, Introdução, foi apresentada a motivação inicial para a proposta e elencados os principais objetivos gerais e específicos do trabalho.

O Capítulo 2 aborda a fundamentação teórica com os conceitos básicos das principais áreas temáticas que envolvem este trabalho: Resolução de Entidades como parte de um processo de integração de dados.

No Capítulo 3 são explanados alguns trabalhos que possuem interseção com a pesquisa apresentada neste trabalho.

No Capítulo 4 é detalhado o QUIPER, apresentando todas as etapas do processo proposto e discutidos os algoritmos implementados.

No Capítulo 5 são apresentados os experimentos que avaliam o QUIPER.

No Capítulo 6 são apresentadas as discussões e conclusões do trabalho.

Por fim, são apresentadas as Referências Bibliográficas utilizadas neste trabalho.

2 RESOLUÇÃO DE ENTIDADES

2.1 Introdução

Neste capítulo são apresentados os fundamentos teóricos necessários para a compreensão deste trabalho. Resolução de Entidade (RE) é uma etapa do processo de integração de dados, por este motivo o capítulo é iniciado com uma explanação do processo de integração de dados. Em seguida, são apresentados os conceitos básicos de RE, a classificação das estratégias para RE, assim como ferramentas e *frameworks* para RE.

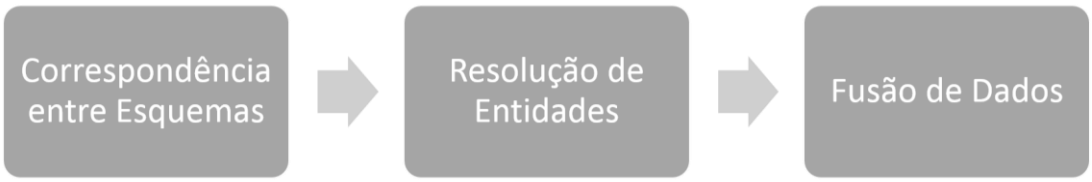
2.2 Integração de Dados

Integração de dados tem por objetivo combinar dados de diferentes fontes de dados, muitas vezes heterogêneas e autônomas, disponibilizando uma visão única e uniforme para os usuários [Lenzerini, 2011]. Esta visão unificada de múltiplas fontes de dados facilita, por exemplo, a cooperação entre agências governamentais, cada uma com as suas próprias fontes de dados, permitindo o compartilhamento de dados.

No processo de integração de dados, a uniformização do acesso a fontes de dados heterogêneas enfrenta uma gama de desafios [Dong, 2009; Dong & Srivastava, 2015]. Dentre eles destacam-se: (i) Variedade de esquemas: cada fonte de dados utiliza esquemas independentes uma das outras, com granularidades diferentes; (ii) Variedade de modelos: cada fonte de dados pode estar representada em modelos de dados distintos. Algumas vezes, a unificação de modelos leva à perda semântica dos dados; (iii) Heterogeneidade de instâncias: cada fonte de dados representa de formas distintas o mesmo objeto, instância ou tupla de uma entidade do mundo real e (iv) Redundância de Dados: dados redundantes em múltiplas fontes de dados, gerando problemas de veracidade e conflitos entres as informações provindas de diferentes fontes de dados.

Considerando tais desafios, o processo tradicional de integração de dados é dividido em 3 (três) etapas (Figura 1) [Dong & Naumann, 2009]: correspondência entre esquemas, Resolução de Entidades e fusão de dados.

Figura 1 - Etapas para integração de dados



Fonte: Adaptada de Dong & Naumann (2009)

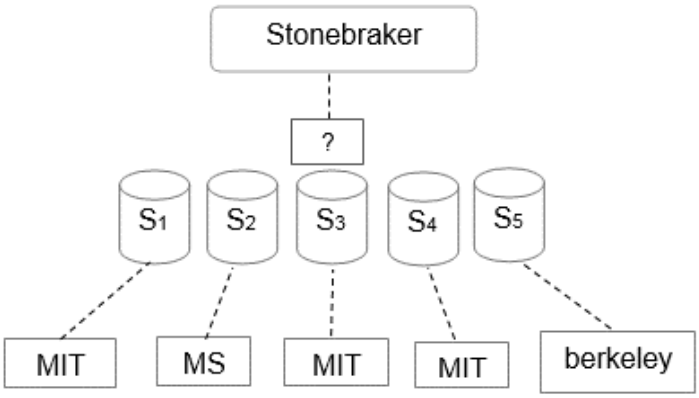
A etapa de correspondência entre esquemas (*schema matching*) [Bellahsene et *al*., 2011] é utilizada para contornar o problema de fontes com esquemas distintos. Considerando esquemas de fontes distintas, pode-se fazer a correspondência, por exemplo, entre a entidade (conceito) "*Autor*", em uma fonte de dados, com o conceito "*Escritor*", em outra fonte de dados.

A Resolução de Entidades detecta tuplas que representam a mesma entidade do mundo real [Halevy et *al*., 2006]. Considerando duas tuplas de entidades em fontes de dados distintas, pode-se, por exemplo, indicar que a autora "*Getoor*", em uma fonte e dados, é a mesma que a autora "*Getor*", em outra fonte de dados.

Adicionalmente, as fontes de dados a serem integradas podem ser conflitantes entre si. Tais conflitos podem ser provenientes de dados incompletos, errados ou desatualizados. Nestes casos, precisa-se saber quais fontes de dados possuem as tuplas com os valores corretos e como unir estes valores [Dong et *al*., 2014]. Este problema é definido como combinação de dados (*data merge*) ou fusão de dados (*data fusion*). Neste sentido, as tuplas que representam a mesma entidade do mundo real passam pelo processo de fusão e são unificadas em uma única representação.

Em [Dong & Srivastava, 2013] o problema de fusão de dados é exemplificado da seguinte forma: dadas 5 (cinco) fontes com dados (S_1, S_2, S_3, S_4 e S_5) (Figura 2) sobre filiação de pesquisadores, como identificar qual fonte tem a informação correta com relação ao pesquisador "Stonebraker"? Em [Dong & Naumann, 2009; Dong & Srivastava, 2013; Dong et *al*., 2014], são discutidas técnicas para solucionar este problema.

Figura 2 - Exemplo de fusão de dados



Fonte: Dong & Srivastava, 2013

A qualidade do resultado do processo de integração de dados depende da qualidade dos dados a serem integrados [Batini & Scannapieco, 2006; Dong et *al*., 2013; Rekatsinas et *al*., 2014]. Quando se considera um grande número de dados e fontes de dados, deve-se ter foco nos diferentes níveis de qualidade de cada fonte de dados a ser analisada. Dentre os problemas destacam-se dados: ultrapassados, incompletos, inconsistentes e duplicados [Batini & Scannapieco, 2006; Lee et *al*., 2009; Zaveri et *al*., 2012]. Estes problemas devem ser levados em consideração, uma vez que entradas de baixa qualidade geram saídas com baixa precisão [Christen, 2012]. Neste sentido, estratégias de Resolução de Entidades (RE) também podem ser utilizadas para mensurar a qualidade das fontes de dados. Na próxima seção são detalhados os conceitos de RE.

2.3 Conceitos de Resolução de Entidades

O problema de RE é abordado em diferentes processos de gerenciamentos de dados [Dorneles et *al*., 2010]: (i) Integração de dados; (ii) Mineração de dados, na etapa de limpeza de dados; (iii) Busca de dados similares (recuperação de informações) e (iv) Planos de consulta, indicando a sobreposição de fontes de dados. Neste trabalho, é dada ênfase em RE, como uma etapa do processo de integração de dados.

Os problemas de RE podem surgir em conjuntos de dados isolados, como arquivos e bases de dados, sendo ainda mais críticos quando múltiplas fontes de dados necessitam ser integradas [Adriaans, 1996]. Isto acontece em virtude das diversas fontes conterem tuplas redundantes com diferentes representações. De modo a possibilitar um acesso preciso e

consistente aos dados, é necessário consolidar as diferentes representações, detectar e eliminar possíveis duplicações.

A RE é pesquisada há pelo menos 60 anos. Na década de 50 já se tinha em média 100 trabalhos de pesquisa na área [Monge, 2012]. Os primeiros estudos foram realizados no contexto médico, para a identificação de um mesmo paciente em fontes de dados diferentes. Ao longo das décadas o volume de dados evoluiu, assim como novos modelos de dados surgiram, sendo necessário o aperfeiçoamento e a criação de novas técnicas.

Para a RE, é comum a utilização de técnicas sintáticas, ou seja, técnicas que analisam as tuplas com base em sua grafia [Oliveira, 2008]. São exemplos de técnicas sintáticas [Elmagarmid, 2006]: (i) Comparação numérica: determina se dois valores numéricos são iguais ou não; (ii) Distância de edição: considera o número de alterações que devem ser realizadas para transformar um dado em outro; (iii) Correspondência de nomes: identifica correspondências entre os dados devido ao uso de abreviações ou erros de digitação; e (iv) Fonética: compara os dados em função do som das suas pronúncias.

Em linhas gerais, o processo tradicional de RE, baseado no processamento *off-line* de tuplas de fontes de dados estáticas (fontes de dados em um instante de tempo específico), segue a Figura 3 [Christen, 2012]. Basicamente, todas as tuplas das fontes de dados *A* e *B* são processadas em um único momento. As etapas do processo tradicional de RE são: (i) Pré-processamento; (ii) Indexação; (iii) Comparação; (iv) Classificação; (v) Avaliação e (vi) Revisão. A seguir são apresentados os objetivos de cada etapa.

I. A etapa de pré-processamento (*data pre-processing*) é utilizada para uniformizar a representação dos dados, tal como, remover ou inserir caracteres e expandir abreviações [Herzog, 2007];

II. Na indexação (*indexing*) é realizado um filtro para os pares de tuplas que provavelmente são correspondentes. Estes são analisados na etapa de comparação entre pares de tuplas (*tuple pair comparison*). Uma das estratégias mais utilizadas na etapa de indexação é a blocagem (*blocking*) [Nin et *al*., 2007; Christen et al., 2009; Kenig & Gal, 2009; Ramadan et *al*., 2013; Ramadan & Christen, 2014];

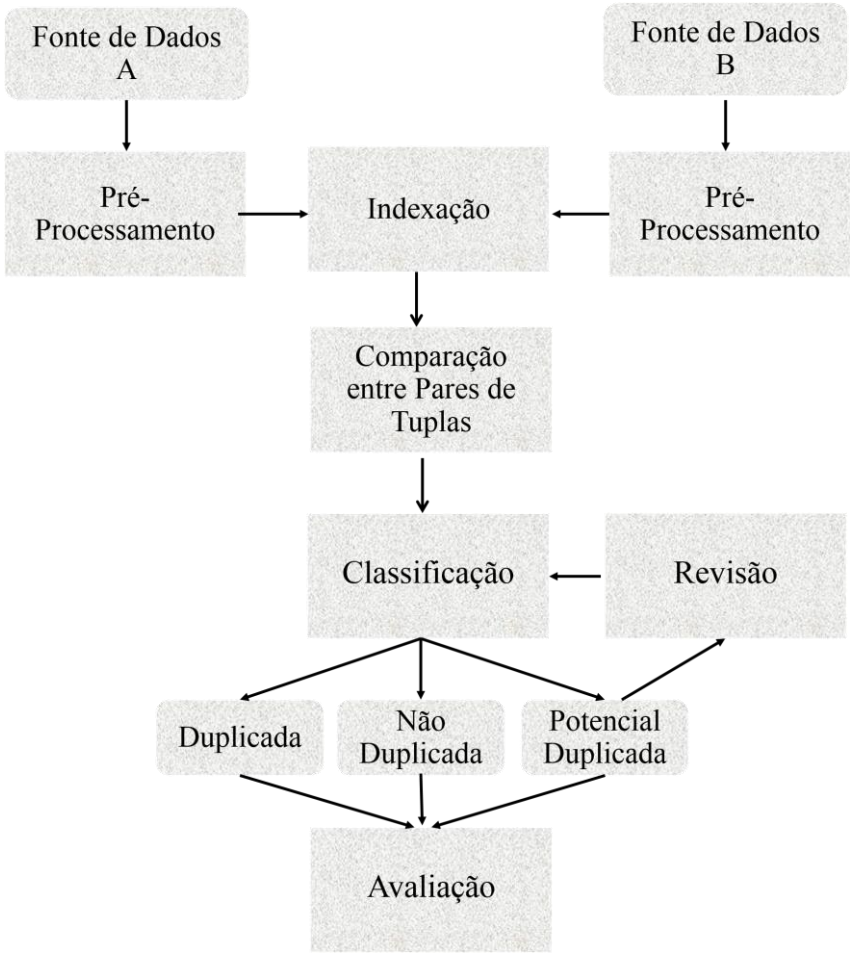
III. Na comparação entre pares de tuplas são utilizadas funções de similaridades [Elmagarmid et *al*., 2007; Herzog et *al*., 2007; Silva et *al*., 2007; Dorneles et *al*., 2010; Bharambe et *al*., 2012] que avaliam o quão duas tuplas são próximas, gerando uma medida de similaridade. Comumente, o grau de similaridade é normalizado entre 0 e 1, onde 0 representa a dissimilaridade e 1 representa similaridade total;

IV. Na etapa de classificação (*classification*), as tuplas são classificados em duplicadas, não duplicadas ou potencialmente duplicadas. Esta etapa pode ser realizada por meio de algoritmos de clusterização [Charikar et *al*., 2004; Young et *al*., 2010; Ackerman & Dasgupta, 2014; Gruenheid et *al*., 2014].

V. Na etapa de avaliação do processo (*evaluation*) vários critérios podem ser utilizados [Christen & Goiser, 2007], com foco em qualidade dos resultados (considerando a quantidade de acertos e a cobertura dos dados duplicados descobertos) ou em tempo de processamento (eficiência).

VI. Por fim, pode-se ter uma revisão supervisionada (*clerical review*) para tuplas indicadas como potenciais duplicadas (*potential matches*) na etapa de classificação.

Figura 3 – Processo tradicional de Resolução de Entidades entre duas fontes de dados



Fonte: Christen, 2012 (Modificada).

Em Dong & Srivastava (2015) o processo tradicional de RE é apresentado em apenas 3 (três) etapas: blocagem, comparação entre pares de tuplas e clusterização de tuplas com o objetivo de classificar quais pares representam a mesma entidade do mundo real. Neste

caso, se tem uma simplificação do processo apresentado por Christen (2012), destacando as principais etapas.

Para avaliar a qualidade dos resultados provindos do processo de RE, pode-se fazer uso de métricas, a exemplo de Precisão, Revocação e Medida-F. A Precisão mede a porcentagem de pares de tuplas que foram corretamente detectadas como duplicadas pelo processo de RE (Equação 1). A Revocação calcula a porcentagem de pares de tuplas duplicadas esperadas pelo usuário e que foram indicadas pelo processo de RE (Equação 2). A Medida-F calcula a proporção dos pares de tuplas detectados pelo processo de RE em relação a todos os pares de tuplas do sistema (Equação 3).

Nas Equações 1 e 2, *VerdadeirosPositivos* é o total de pares de tuplas detectados corretamente como pares de tuplas duplicados. *FalsosPositivos* é o total de pares de tuplas detectados erroneamente como pares de tuplas duplicadas. *FalsosNegativos* é o total de pares de tuplas detectados erroneamente como pares de tuplas não duplicadas.

$$Precisão = \frac{VerdadeirosPositivos}{VerdadeirosPositivos + FalsosPositivos} \qquad \text{(Equação 1)}$$

$$Revocação = \frac{VerdadeirosPositivos}{VerdadeirosPositivos + FalsosNegativos} \qquad \text{(Equação 2)}$$

$$Medida - F = \frac{2 * Precisão * Revocação}{Precisão + Revocação} \qquad \text{(Equação 3)}$$

É importante destacar que existem cenários onde o tempo gasto para o processamento é primordial, principalmente quando considerados ambientes com grandes volumes de dados. Esta restrição se torna um gargalo, quando utilizadas técnicas que comparam todos os pares de tuplas envolvidos [Kenig & Gal, 2009].

Em ambientes com grandes volumes de dados, cada vez mais necessita-se de processos de RE que considerem a característica dinâmica das fontes de dados e que atenuem o tempo de processamento. O processo tradicional de RE não é adequado para este tipo de cenário. Na literatura, se observa o surgimento de soluções incrementais, nas quais a cada iteração do processo de RE, novas tuplas das fontes de dados são processadas e integradas ao conjunto de tuplas processadas anteriormente.

Este novo conjunto de tuplas pode ser formado por: (i) Novas tuplas que não foram processadas anteriormente e que foram inseridas em fontes de dados que já foram submetidas ao processo de RE; (ii) Tuplas atualizadas em fontes de dados que já foram submetidas ao processo de RE e (iii) Tuplas pertencentes a fontes de dados que ainda não foram submetidas ao processo de RE. Em (iii), considera-se que o processo de RE é realizado em resultados de consultas e que as tuplas pertencentes aos resultados de consultas não foram consultadas previamente. O processo incremental permite a redução do volume de tuplas processado em cada iteração e o reúso dos resultados intermediários obtidos previamente pelo processo de RE reduz o custo de processamento do processo de RE como um todo.

2.3.1 Etapa de Indexação do Processo de Resolução de Entidades

Para a etapa de indexação do processo de RE a literatura apresenta uma série de técnicas, a exemplo da técnica de blocagem [Christen, 2012; Ramadan et *al*., 2015]. De modo geral, estas técnicas processam todas as tuplas de todas as fontes de dados a serem integradas e as inserem em um ou mais blocos. Cada bloco é criado de acordo com um critério ou as tuplas são ordenadas a fim de que tuplas mais semelhantes fiquem próximas.

O critério usado para a criação de blocos é baseado no valor de um ou mais atributos das tuplas. Para cada bloco é gerada uma chave de blocagem (*Blocking Key*). Desta forma, tuplas com a mesma chave de blocagem são inseridas no mesmo bloco e são comparadas entre elas na etapa de comparação entre pares de tuplas do processo de RE.

Contudo, a maioria das técnicas disponíveis na literatura foram propostas para processos tradicionais de RE. Neste caso, todas as tuplas são processadas em um dado instante de tempo sem mudanças posteriores dos índices. Desta forma, as técnicas de indexação são denominadas de estáticas [Ramadan et *al*., 2015].

Para cenários incrementais de RE e em ambientes de RE em tempo de consulta, está emergindo na literatura o conceito de indexação dinâmica. Neste caso, os índices são atualizados e modificados a fim de refletir o estado atual das fontes de dados, com as novas iterações do processo de RE [Ramadan et *al*., 2015].

2.3.2 Etapa de Classificação do Processo de Resolução de Entidades

A etapa de classificação do processo de RE tem por objetivo gerar grupos de tuplas que representam a mesma entidade do mundo real. Uma das formas de se realizar esta classificação é por meio de algoritmos de clusterização. Neste caso, cada *cluster* corresponde a uma única entidade do mundo real (por exemplo, um lugar ou uma pessoa).

Clusterização é uma classificação não supervisionada de padrões em *clusters* (ou grupos), na qual cada *cluster* corresponde a um padrão detectado [Charikar et *al*., 2004; Young et *al*., 2010; Ackerman & Dasgupta, 2014; Gruenheid et *al*., 2014]. Existe uma vasta variedade de algoritmos de clusterização descritos na literatura [Jain et *al*., 1999; Berkhin, 2006]. A grande variedade é consequência da inexistência de um algoritmo universal que consiga trabalhar com todos os padrões de dados. A escolha do algoritmo para um determinado problema depende do tipo dos dados e do propósito da aplicação.

De forma geral, os algoritmos são divididos e classificados nos seguintes métodos de clusterização [Jain et *al*., 1999; Berkhin, 2006]:

- Métodos de Partição: Um valor *k* de *clusters* é fornecido pelo usuário e a base é particionada em *k* partes. O método divide o conjunto de dados em partes disjuntas, as partições iniciais são aleatórias e, posteriormente, a cada iteração objetos mais próximos são inseridos em um mesmo *cluster*.
- Métodos Hierárquicos: Envolve a construção de uma hierarquia do tipo árvore. Estes podem ser subdivididos em aglomerativos ou divisórios. Aglomerativos, inicia-se com cada instância como sendo um *cluster* e em seguida são realizadas junções entre *clusters*. Divisórios, inicia-se com um *cluster* único, com todas as instâncias, e em seguida são realizadas sucessivas divisões.
- Métodos Baseados em Densidade: Métodos especializados na descoberta de *clusters* de formatos arbitrários. Normalmente são regiões densas separadas por regiões de baixa densidade.

Cada uma das classificações dos métodos listados, pode ser refinada segundo as seguintes características:

- Incremental (*on-line*): Algoritmos incrementais são processados à medida que novas tuplas são recuperadas. A cada nova tupla uma escolha deve ser realizada (i) Criar um novo *cluster* ou (ii) Inserir em um *cluster* existente.
- Tradicional (em lote ou *off-line*): Algoritmos tradicionais são caracterizados por avaliar todos os dados em um único momento, detectando os padrões e classificando as tuplas por *clusters*.

Algoritmos incrementais de clusterização são definidos com o objetivo de manipular grandes volumes de dados, considerando limite de tempo para processamento, assim como espaço de memória. Clusterização incremental foi proposta em [Widyantoro et al., 2002; Huang et *al*., 2006; Young et *al*., 2010; Gruenheid et *al*., 2014; Whang & Garcia-Molina, 2014; Ackerman & Dasgupta, 2014] e o maior desafio é obter os resultados dos algoritmos de clusterização incremental os mais próximos possíveis dos resultados obtidos por algoritmos tradicionais de clusterização. Contudo, na literatura alguns trabalhos estão dando foco na minimização deste problema por meio de operações corretivas [Ackerman & Dasgupta, 2014; Gruenheid et *al*., 2014], as quais objetivam identificar dados que foram inseridos em *clusters* errados e os inserir em *clusters* mais adequados.

Considerando um processo de RE incremental, pode-se mapeá-lo para um problema de clusterização incremental, no qual, a cada novo conjunto de tuplas, a clusterização incremental determina qual o *cluster* mais adequado para cada uma das tuplas.

Para o escopo deste trabalho, é importante destacar 3 (três) algoritmos de clusterização existentes na literatura: *Single-Link*, *Average-Link* [Kogan et *al*., 2006] e *Hill-Climbing* [Guo et *al*., 2010]. Todos foram previamente utilizados em problemas de RE com grandes volumes de dados [Gruenheid et *al*., 2014; Bhattacharya & Getoor, 2006; Tan et *al*., 2006].

O algoritmo *Single-Link*, que é hierárquico e aglomerativo, combina (*merge*) os 2 (dois) *clusters* que possuem o par de tuplas com maior valor de similaridade e que pertencem a *clusters* distintos. Por outro lado, o algoritmo Average-Link, também hierárquico e aglomerativo, combina os 2 (dois) *clusters* com maior média de valores de similaridade entre suas tuplas. O algoritmo *Hill-Climbing*, maximiza uma função objetivo [Guo et *al*.,

2010] (por exemplo, coesão de *clusters*) em busca de valores maiores em cada iteração para decidir pela combinação ou não de 2 (dois) *clusters*.

2.3.3 Classificação de Processos de Resolução de Entidades

Na literatura existem diversos processos para Resolução de Entidades (RE). Considerando as classificações definidas em [Elmagarmid et *al*., 2007; Baumgartner et *al*., 2009; Bharambe et *al*., 2012] e os objetivos deste trabalho, foi realizada a classificação ilustrada na Figura 4.

Para a RE pode-se utilizar processos com abordagem: (i) Tradicional, que realiza a RE em todas as fontes de dados em um único momento, também chamada de RE com processamento em lote (*batch processing*); (ii) Orientada à consulta, objetiva realizar RE em tempo de consulta para uma dada consulta específica, sem fazer reúso de iterações anteriores [Kalpana et *al*., 2010] e (iii) Incremental, realiza RE em um subconjunto das tuplas das fontes de dados, incrementando o conjunto de tuplas processadas ao longo do tempo. Em (iii), um incremento pode ser: (a) Resultados de consultas, que são submetidos ao processo de RE incremental (i.e., o processo de RE orientado à consulta considerando incrementos); (b) Alterações nas fontes de dados previamente submetidas ao processo de RE; e (c) Ordenação das tuplas que serão processadas para atender a critérios de tempo de processamento, estas tuplas são processadas de forma incremental para que o processo de RE tenha um menor custo de processamento, também chamado de processo progressivo, *on-line* ou *pay-as-you-go*.

Figura 4 - Classificação de processos de Resolução de Entidades

Fonte: Próprio autor, 2017.

Os processos (ii) e (iii) são adaptações do processo tradicional (i) descrito na Figura 3. Neste trabalho é proposto um processo que combina as abordagens (ii) e (iii), tal que os incrementos considerados são resultados de consultas (a).

Em cada um dos tipos de processos de RE, podem ser utilizadas estratégias distintas para avaliar se duas tuplas representam a mesma entidade do mundo real. Estas estratégias foram divididas em (i) Estratégias supervisionadas e (ii) Estratégias não supervisionadas.

Em estratégias supervisionadas, faz-se uso de um conjunto rotulado (conjunto de treinamento) para inferir quais tuplas representam a mesma entidade do mundo real. Para este propósito, são utilizados algoritmos supervisionados que possam tirar proveito do conjunto de treinamento. Na literatura, tem-se trabalhos que objetivam minimizar a interferência humana, atividade manual de seleção, para a criação do conjunto de treinamento, assim como otimizar este processo [Bianco et *al*., 2015].

A desvantagem da estratégia supervisionada é a criação de conjuntos de treinamento. Esta atividade é complexa principalmente em ambientes com grandes volumes de dados, nos quais o conjunto de treinamento deve ser representativo e mínimo.

Em estratégias não supervisionadas, não existe um conjunto de treinamento para nortear o algoritmo. Neste sentido, os algoritmos buscam por padrões, a partir de uma característica de similaridade. Dentre os métodos não supervisionados, estão os algoritmos de clusterização [Jain et *al*., 1999; Fahad et *al*., 2014], utilizados neste trabalho como estratégia não supervisionada para grandes volumes de dados.

Considerando as estratégias dos algoritmos para RE (supervisionada e não supervisionada), existem as funções de similaridade utilizadas para definir o quão duas tuplas são similares, gerando uma medida de similaridade [Elmagarmid et *al*., 2007; Herzog et *al*., 2007; Silva et *al*., 2007; Dorneles et *al*., 2010; Bharambe et *al*., 2012]. Estas funções de similaridade podem considerar características sintáticas (que avaliam a escrita dos dados) e/ou semânticas (que avaliam o significado e os relacionamentos entre entidades).

2.4 Ferramentas e *Frameworks* para Resolução de Entidades

Para identificar lacunas em soluções existentes ou funcionalidades que pudessem ser reutilizadas para o cenário no qual esta pesquisa está inserida, foram avaliadas

ferramentas/*frameworks* que dispõem de estratégias de apoio a RE. O foco foi em soluções com implementações abertas para que fosse possível a análise das características dos algoritmos utilizados. As estratégias comerciais [Data Match, 2015; DTM Data Scrubber, 2015] foram excluídas por não serem amplamente difundidas e não possuírem os algoritmos disponíveis na literatura.

Inicialmente foram encontradas 17 (dezessete) soluções entre ferramentas e frameworks: WHIRL [Cohen, 1998], FLAMINGO [Jin et *al*., 2003], DuDe [Draisbach & Naumann, 2010], BN [Leitão et *al*., 2007], MOMA [Thor & Rahm, 2007], SERF [Benjelloun et *al*., 2009], Active Atlas [Tejada et *al*., 2002], MARLIN [Bilenko & Mooney, 2003], Multiple Classifier System [Zhao & Ram, 2005], Operator Trees [Chaudhuri et *al*., 2007], TAILOR [Elfeky et *al*., 2002], FEBRL [Christen, 2008; Christen, 2009], STEM [Köpcke & Rahm, 2008], DEDOOP [Kolb et *al*., 2012], A Generic Web-Based Entity Resolution Framework [Pereira et *al*., 2011], General Context-Aware Data Matching and Merging Framework [Žitnik et *al*., 2013] e D-Dupe [Bilgic et *al*., 2006; Kang et *al*., 2008]. Muitas destas soluções foram parcialmente avaliadas por Köpcke & Rahm (2009), Draisbach & Naumann (2010) e Elmagarmid (2007).

Das soluções analisadas, apenas 4 (quatro) possuem implementação disponível para avaliação: FEBRL, Dedoop, D-Dupe e DuDe. Algumas características das soluções escolhidas para análise foram sintetizadas por Köpcke & Rahm (2009), Draisbach, & Naumann (2010) e Elmagarmid (2007). Contudo, em contraste com estes trabalhos da literatura, esta seção não se concentra em algoritmos específicos e sim em avaliar características das soluções que possuem interseções com este trabalho. As características avaliadas estão listadas na Subseção 2.4.5.

2.4.1 FEBRL

FEBRL [Christen, 2008; Christen, 2009] é uma ferramenta desenvolvida desde 2002, na linguagem de programação Python [Python, 2016], para apoiar o processo de limpeza de dados e de detecção de dados duplicados. Foi desenvolvida para o domínio de saúde, mas pode ser utilizada e expandida para outros domínios.

Na ferramenta é possível realizar todas as atividades via interface gráfica: (i) selecionar as fontes; (ii) utilizar as estratégias de blocagem disponíveis, indicando inclusive, os atributos a serem utilizados; (iii) utilizar uma ou mais das 23 funções de comparação,

indicando por atributo qual função deve ser utilizada e o seu limiar; (iv) selecionar um algoritmo de classificação, indicando todos os parâmetros de entrada. É importante destacar que a ferramenta pode ser estendida e novos algoritmos podem ser inseridos e avaliados.

2.4.2 D-Dupe

D-Dupe [Bilgic et *al*., 2006; Kang et *al*., 2008] é uma ferramenta desenvolvida desde 2006, na linguagem de programação C#, a última versão disponível é de 2008, a versão Beta 2.0. Esta possui diversos algoritmos de comparação e classificação implementados para dados relacionais. A visualização dos resultados é feita na forma de grafos, representando uma rede de relacionamentos entre as instâncias. Desta forma, a RE é realizada de forma sintática (com funções de similaridade que avaliam a escrita dos dados) e semântica (avaliando os relacionamentos entre as tuplas).

O usuário é responsável por indicar manualmente os atributos que devem ser comparados e as funções que devem ser utilizadas. Adicionalmente, é possível realizar a fusão de tuplas duplicadas, indicar que duas tuplas não são duplicadas ou apenas indicar que duas tuplas representam a mesma tuplas do mundo real.

2.4.3 Dedoop

Dedoop (Deduplication with Hadoop) [Kolb et *al*., 2012] é um *framework* baseado em MapReduce para RE. Basicamente foi especificada uma infraestrutura, com acesso por meio de APIs, que possibilita o processamento paralelo de comparações entre pares de tuplas e do processo de RE como um todo.

Dedoop possui a implementação de diversos algoritmos de blocagem, comparação e classificação. Automaticamente o processo de RE é transformado em um *workflow* de MapReduce, realizando o balanceamento entre blocos, minimizando o tempo de processamento dos algoritmos.

Os arquivos de entrada e saída são CSV e, além disso, existe uma interface gráfica que permite a visualização dos resultados por diferentes perspectivas. A solução está sendo aperfeiçoada desde 2012, com código em Java.

2.4.4 DuDe

O *framework* DuDe (*Duplicate Detection*) [Draisbach & Naumann, 2010] é dividido em 7 (sete) componentes, implementados na linguagem de programação Java [Java, 2016], que podem ser estendidos e utilizados em outros processos de RE que os pesquisadores queiram avaliar.

DuDe permite RE em múltiplas fontes de dados, todos os dados são transformados e processados internamente no formato JSON [JSON, 2016]. Dentre os 7 (sete) componentes do DuDe tem-se como destaques: (i) Extração, responsável por extrair dados das fontes de dados. (ii) Pré-processamento, utilizado para extrair informações estatísticas dos dados de entrada, tais como quantidade de valores nulos ou distintos. (iii) Algoritmos, disponibiliza uma série de algoritmos de classificação, que indicam se duas tuplas são a mesma entidade do mundo real. IV) Comparação, disponibiliza um conjunto de funções de similaridade que podem ser utilizadas para comparar duas entidades. Adicionalmente, são disponibilizadas fontes de dados para testes, assim como os resultados esperados após o processo de RE ser aplicado.

2.4.5 Quadro Comparativo

Dentre as ferramentas e *frameworks* analisados, cada uma possui características específicas, mas, em geral, muitas destas são comuns e buscam atender aos requisitos de qualidade e tempo de processamento do processo de RE. Como resultado da análise realizada sobre as soluções descritas acima, o Quadro 1 foi gerado.

A principais características e diferenciais sumarizados no Quadro 1 são:

- Ano: Ano de início do desenvolvimento da solução (primeiros artigos publicados);
- Grupo de Pesquisa: Grupo ou universidade que desenvolve a solução;
- Seleção de Atributos: Estratégia utilizada para selecionar os atributos a serem comparados em funções de similaridades;
- Múltiplas fontes de dados: Se a solução suporta uma ou mais fontes de dados;
- Semântica: Se a solução avalia o significado das instâncias ou os relacionamentos entre instâncias;
- Algoritmos incrementais: Indica o uso de algoritmos incrementais para Resolução de Entidades;

- Permite ser estendido: Característica que avalia se a solução pode ser ampliada com outros algoritmos;
- Tipo de processamento: Avalia se a execução da solução é realizada de forma sequencial ou incremental;
- Modo de visualização de saída: Indica as estratégias utilizadas para avaliação dos resultados gerados.

Quadro 1 - Comparativo entre ferramentas de RE

Solução / Característica	FEBRL	D-Dupe	Dedoop	DuDe
Ano	2002	2006	2012	2010
Grupo de Pesquisa	Australian National University in Canberra e New South Wales Department of Health in Sydney	University of Maryland Institute for Advanced Computer Studies - UMIACS	University of Leipzig	Information Systems Group
Seleção de Atributos	Manual	Manual	Manual	Manual
Múltiplas Fontes de Dados	Sim (duas)	Não	Sim (duas)	Sim
Semântica	Não	Sim (Contexto de colaboração)	Não	Não
Algoritmos Incrementais	Não	Não	Não	Não
Permite ser Estendido	Sim	Sim	Sim	Sim
Tipo de Processamento	Sequencial	Sequencial	Paralelo	Sequencial
Modo Visualização Saída	Gráfica (histograma)	Gráfica (grafo)	Gráfica (diversos gráficos)	JSON, CSV, texto

Basicamente observa-se que as ferramentas e *frameworks* analisados não possuem suporte direto para um processo incremental de RE. Contudo, disponibilizam diversas funções de similaridade e algoritmos de indexação que podem ser reutilizados, tal como permitem que suas soluções sejam estendidas e que novos algoritmos sejam inseridos em suas plataformas.

2.5 Considerações

Com a necessidade de estratégias de RE em diferentes ambientes e para solucionar problemas distintos (integração de dados, mineração de dados, recuperação de informações, qualidade de fontes de dados, entre outros), houve uma intensificação nas pesquisas de processos incrementais de RE. Esta necessidade surgiu com o aumento na geração e publicação de dados e a necessidade cada vez mais eminente de estratégias mais eficientes, sem se distanciar da qualidade do resultados obtidos pelos processos tradicionais de RE. Considerando o problema de integração de grandes volumes de dados, este trabalho dá ênfase a RE como uma etapa da integração de dados. Desta forma, é proposto um processo de RE incremental e orientado à consulta que colabora para integrar fontes de dados.

Neste capítulo, foram apresentados os principais conceitos que embasam a proposta deste trabalho. Foram apresentados os conceitos básicos de integração de dados, seguindo pelas definições de RE e suas classificações. Por fim, foram explanadas as principais soluções, com implementação disponível, para RE em integração de dados.

3 ABORDAGENS PARA RESOLUÇÃO DE ENTIDADES

3.1 Introdução

No âmbito de Resolução de Entidades (RE) existe uma vasta quantidade de trabalhos que se diferenciam nas abordagens, estratégias, funções de similaridade e modelos de dados analisados. Neste capítulo são apresentados os trabalhos relacionados com o processo de RE desenvolvimento neste trabalho. Todos os trabalhos a serem apresentados, objetivam identificar se um par de objetos, instâncias, dados ou tuplas que representam a mesma entidade do mundo real.

Basicamente, os trabalhos relacionados foram divididos em 2 (duas) abordagens, seguindo a classificação definida na Seção 2.3.3: (i) Orientado à consulta, que realiza a RE em tempo de consulta e (ii) Incremental, que realiza a RE em partes dos dados (incrementos, porções), incrementando os resultados ao longo do tempo. Não foram encontrados trabalhos que unem o processo incremental com o orientado à consulta, ou seja, onde os incrementos considerados são resultados de consultas e processados em tempo de consulta com o objetivo de resolver problemas de integração de dados.

Por ser amplo o número de estratégias utilizadas, cada uma com características e objetivos específicos, ao término do capítulo há um quadro comparativo com o objetivo de sumarizar as principais características analisadas.

3.2 Abordagem Orientada à Consulta

Nesta seção são discutidos trabalhos com processos de RE orientados a consulta. É importante destacar que poucos trabalham dão ênfase ao processo de RE em tempo de consulta e que este problema possui características diferentes da Resolução de Entidades tradicional. Primeiro, tem-se a variável consulta a ser avaliada, que em algumas abordagens é avaliada para que planos de consultas sejam traçados a fim de minimizar as fontes de dados a serem acessadas e identificar quais pares de tuplas são suficientes para responderem a uma consulta específica. Segundo, as consultas possuem um curto tempo para serem respondidas e a Resolução de Entidades não deve interferir de forma a ampliar excessivamente o tempo de resposta. Desta forma são explanados os trabalhos de: (i)

Bhattacharya & Getoor (2007); (ii) Su et *al*. (2010); (iii) Salloum et *al*. (2013) e (iv) Altwaijry et *al*. (2013).

Bhattacharya & Getoor (2007)

Em Bhattacharya & Getoor (2007) é realizado o processo RE em tempo de execução da consulta. O trabalho possui como principal motivação o fato de que os usuários podem acessar o sistema frequentemente, mas não é necessário resolver toda a base de dados, assumindo que o usuário só necessite de parte dos dados. Desta forma, a principal ideia é identificar e processar apenas as tuplas que respondem a uma consulta específica. O trabalho está inserido na área de Resolução de Entidades para recuperação de informações.

Basicamente, a estratégia utilizada é uma adaptação do algoritmo definido em [Bhattacharya et *al*., 2006], que faz uso dos relacionamentos entre tuplas para inferir a duplicação. Em suma, são definidas duas etapas: (i) Expansão, expandir a consulta para extrair as tuplas relevantes. (ii) Resolução, aplicar a RE apenas nas tuplas extraídas. A extração é realizada de forma recursiva, analisando a dependência entre as tuplas e limitando o número de níveis a serem percorridos.

Na etapa (i) é utilizada uma estratégia de blocagem para identificar as potenciais tuplas duplicadas, diminuindo, portanto, o número de comparações a serem realizadas. No trabalho, é dado ênfase apenas a consultas onde são exigidos valores específicos para um atributo (por exemplo, consultas com um literal na cláusula *where*). O foco principal é identificar os falsos positivos, por meio dos coautores de trabalhos científicos. Desta forma, a estratégia é utilizada em redes científicas de relacionamentos.

A função de similaridade definida é uma ponderação da similaridade sintática entre duas tuplas e a similaridade de relacionamentos das tuplas comparadas com outras tuplas da fonte de dados. Apesar de considerarmos a avaliação dos relacionamentos como sendo uma análise semântica, Bhattacharya & Getoor (2007) não a definem assim.

Para validação da proposta, foram realizados experimentos em fontes de dados reais e sintéticas do domínio bibliográfico. As consultas foram realizadas em cada fonte de dados e o tempo de execução, profundidade da busca e precisão para responder uma consulta específica foram catalogados e comparados com outros algoritmos da literatura.

Dentre os trabalhos futuros destacados pelos autores tem-se o possível armazenamento de resultados intermediários de resoluções de entidades, sendo utilizados em futuras consultas.

Su et *al*. (2010)

O objetivo de Su et *al*. (2010) é comparar resultados de consultas realizadas em múltiplas fontes de dados web, detectando as tuplas que representam a mesma entidade do mundo real. A estratégia é não supervisionada para seleção de atributos a serem considerados na etapa de comparação entre tuplas (*Unsupervised Duplicate Detection*) e utiliza uma base de treinamento universal para auxiliar no algoritmo de RE. Os principais diferenciais destacados são: (i) detectar dados duplicados na web, considerando processamento de consultas *on-the-fly* e (ii) considerar a dissimilaridade das tuplas para apoiar o processo de RE.

Para a execução do processo proposto são assumidas as seguintes premissas: (i) todos os atributos das tuplas são do tipo *string*, trabalhando apenas com um tipo de dado; (ii) os resultados das consultas em cada fonte são mapeados para um esquema global; (iii) tradutores mapeiam as tuplas de HTML para o modelo relacional; (iv) as tuplas na mesma fonte de dados possuem o mesmo formato; (v) não há tuplas duplicadas em uma fonte de dados, apenas entre fontes de dados; (vi) atributos com valores nulos são preenchidos com a média do valor de similaridade entre os demais atributos da tupla com valor nulo e uma outra tupla que está sendo comparada.

Para integrar os resultados das consultas, é utilizado um esquema global, para o qual os resultados das consultas em cada fonte são mapeados. Estes resultados são armazenados em um banco de dados relacional, onde todo o processo é avaliado.

A estratégia proposta utiliza pesos diferentes para cada atributo, identificando quais atributos são mais relevantes para cada entidade. Estes pesos são ajustados durante o processo de aprendizagem do algoritmo proposto. A cada passo são avaliados os vetores que representam cada tupla (conjunto de atributos) e é ponderado o valor de similaridade, tal como o valor de dissimilaridade dos atributos de cada tupla. A este processo, é associado um conjunto de dados de treinamento, indicando vetores similares e não similares.

Os experimentos mostraram que utilizar um conjunto de treinamento, no qual a maioria das tuplas não são duplicadas, e aplicar o algoritmo não supervisionado de aprendizagem de máquina proposto (que detecta quais os pares são duplicados), tem resultado semelhante aos processos de RE com conjuntos de treinamento tradicionais, i.e., apenas com pares de tuplas corretamente detectadas como duplicadas.

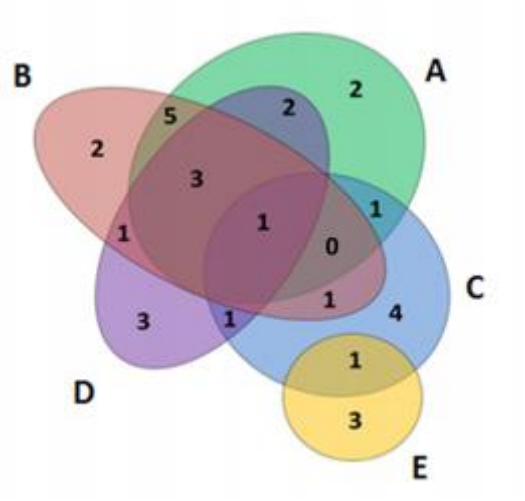
Salloum et *al*. (2013)

Em Salloum et *al*. (2013) é apresentado o OASIS (*Online query Answering System for overlappIng Sources*), responsável por ordenar fontes de dados de acordo com a sobreposição entre elas. Neste caso, é necessário aplicar um processo de RE para identificar quais fontes de dados possuem interseção entre suas tuplas. Um processo semelhante é realizado no trabalho de Saleem et *al*. (2013), diferenciando nos tipos de dados analisados e nas técnicas utilizadas. Basicamente, o processo de RE é aplicado na área de qualidade de fontes de dados para que no processo de processamento de uma consulta, o menor número de fontes de dados seja acessado, reduzindo, portanto, o tempo de processamento de uma consulta específica.

O OASIS possui a premissa de que a ordem em que as fontes de dados são acessadas interfere na cobertura dos resultados de uma consulta. Desta forma, pode-se acessar um número mínimo, em uma ordem ideal, de fontes de dados e se ter a mesma cobertura de resultados de uma consulta que acessa todas as fontes de dados. Neste sentido, na Figura 5 são representadas 5 (cinco) fontes de dados: A, B, C, D e E, com o número de retornos para uma determinada consulta. Se as fontes A e B forem consultadas, estas retornam 14 e 13 resultados, respectivamente, com 9 sobreposições. No entanto, se as fontes A e C fossem consultas, estas retornariam 21 resultados distintos, com apenas 3 resultados sobrepostos, obtendo, portanto, uma maior cobertura de resultados para uma consulta.

O OASIS possui 3 (três) componentes básicos: (i) Estimador de sobreposições de fontes de dados, considerando uma análise estatística de metadados. (ii) Ordenador de fontes de dados, ordena as fontes de dados de acordo com a quantidade de novas contribuições que cada uma pode oferecer às consultas. (iii) Enriquecedor estatístico, detecta valores estatísticos críticos e os enriquece para aperfeiçoar o processo de detecção de fontes de dados sobrepostas.

Figura 5 – Exemplificação sobreposição de fontes



Fonte: Salloum et al., 2013.

As principais variáveis avaliadas no algoritmo do OASIS são: (i) cobertura das fontes de dados e (ii) sobreposição entre as fontes de dados. Estas variáveis são atualizadas a cada nova consulta, e as estatísticas recentes são utilizadas para a próxima consulta. Para o cálculo da sobreposição são avaliadas as tuplas totalmente iguais sintaticamente, não considerando aproximação de termos.

O OASIS realiza ordenação de fontes de dados de forma estática ou dinâmica. Na ordenação estática, as estatísticas sobre as fontes de dados não são recalculadas, assim, algumas soluções podem ser incompletas e imprecisas. Por outro lado, a ordenação dinâmica avalia de forma incremental as estatísticas (por exemplo, de cobertura), acarretando em maiores precisões, porém com a desvantagem da adição de custos de processamento.

Na validação, é comprovado que a solução é escalável considerando um alto número de fontes de dados. No entanto, ressalta-se a importância do balanceamento entre precisão da estimativa da sobreposição de fontes de dados e o tempo de execução da solução. Como trabalho futuro, sugere-se combinar as técnicas utilizadas no OASIS com técnicas que avaliam outras características de qualidade das fontes de dados, tais como precisão e atualidade das tuplas.

Altwaijry et *al*. (2013)

O trabalho de Altwaijrt et *al*. (2013) propõe um processo de RE considerando consultas em SQL. Esta estratégia é relevante quando se deseja avaliar apenas uma porção das

tuplas disponíveis nas fontes de dados, de forma rápida, sem a necessidade de processar todas as tuplas.

A estratégia pode ser aplicada em uma ou múltiplas fontes de dados *on-line*, sendo complementar, ou substituindo, estratégias de blocagem. Adicionalmente, utiliza-se a semântica das consultas, avaliando funções de agregação e minimizando o número de tuplas a serem processadas, resolvendo, portanto, o menor número de tuplas possível, ou seja, apenas aquelas que influenciam na consulta.

O foco da pesquisa é em consultas onde a cláusula "*where"* possui apenas um predicado, podendo ser ampliado para múltiplos predicados conectados por meio de conectivos lógicos.

Basicamente, o algoritmo proposto, QDA (*Query Driven Approach*), é uma modificação da estratégia de fecho transitivo [Benjelloun et *al*., 2009], o qual escolhe iterativamente um par de tuplas para aplicar a RE, realizando o merge sobre os pares de tuplas que forem detectados como duplicados. O QDA utiliza-se do objetivo da consulta para minimizar o número de passos para solucioná-la, considerando, principalmente, consultas SQL com funções de agregação. O resultado retornado para a consulta é um subconjunto da solução ideal, o suficiente para responder corretamente a consulta realizada.

Diferentemente do fecho transitivo, o QDA usa sua própria estratégia de seleção de pares de tuplas para selecionar os pares de tuplas que serão submetidos ao processo de RE. O objetivo desta estratégia é minimizar o número de chamadas a serem resolvidas para responder a determinada consulta. Em segundo lugar, em vez de chamar a resolução do par de tuplas escolhido, o QDA primeiro tenta determinar rapidamente se pode evitar fazer esta chamada, verificando se o par escolhido não é duplicado, se considerado transitividade entre pares de tuplas.

As tuplas que são processadas para responder a consulta são organizadas em um grafo. O grafo tem seus vértices analisados até que a condição da função de agregação seja satisfeita. As avaliações são realizadas por meio de funções de similaridade sintáticas.

Experimentos concluíram que a solução tem um bom desempenho, se comparada com o fecho transitivo tradicional. Como trabalho futuro, Altwaijrt et *al*. (2013) indicam a expansão da estratégia para consultas conjuntivas. Neste trabalho não é avaliado o reúso

dos resultados para consultas posteriores, mas indica esta atividade como sendo um trabalho futuro.

3.3 Abordagem Incremental

Nesta seção são discutidos os trabalhos relacionados que utilizam a abordagem incremental para RE. Desta forma, são explanados os trabalhos de: (i) Benjelloun et *al*. (2009); (ii) Welch et *al*. (2012); (iii) Whang et *al*. (2013); (iv) Whang & Garcia-Molina (2014); (v) Gruenheid et *al*. (2014); (vi) Altowim et *al*. (2014); (vii) Ramadan et *al*. (2015) e (viii) Firmani et *al*. (2016).

Benjelloun et *al*. (2009)

Em Benjelloun et *al*. (2009) são apresentados processos genéricos de RE. Nestes processos as funções de similaridade não são estudadas a fundo, são definidas como sendo uma caixa preta, onde os processos propostos de RE devem minimizar o número de acessos às caixas pretas, potencialmente com alto custo de processamento.

No trabalho o processo de RE é identificado como tendo a etapa de correspondência entre tuplas e também o *merge* (fusão) entre tuplas. No *merge*, uma única tupla será a representante do conjunto de tuplas que representam a mesma entidade do mundo real.

Desta forma, foram propostas 4 (quatro) propriedades de correspondência e *merge* de tuplas que podem minimizar o tempo de processamento do processo de RE: (i) Propriedade associativa de tuplas duplicadas, se as detecções não são associativas, a ordem em que as tuplas são processadas pode afetar o resultado final. (ii) Propriedade reflexiva, uma tupla é duplicada em relação a ela mesma; (iii) Propriedade comutativa, se uma tupla t_1 é duplicada em relação a uma tupla t_2, a tupla t_2 também é duplicada em relação a tupla t_1; e (iv) Representatividade de *merge*, uma tupla t_3 obtida do merge entre as tuplas t_1 e t_2, qualquer nova tupla que seja duplicada em relação a t_1 ou t_2, também será duplicada em relação a t_3.

Considerando as 4 (quatro) propriedades propostas, foram desenvolvidos 3 (três) algoritmos: *G-Swoosh*, *R-Swoosh* e *F-Swoosh*. O algoritmo *G-Swoosh* é mais geral e indicado em casos em que as quatro propriedades não são seguras. O algoritmo *R-Swoosh* explora as quatro propriedades de correspondência e *merge*. O algoritmo *F-Swoosh* também aproveita as quatro propriedades, contudo evita refazer tarefas repetidas de

comparação de tuplas, podendo ser, portanto, significativamente mais eficiente do que o algoritmo *R-Swoosh*.

Foram realizados experimentos de desempenho dos algoritmos em fontes de dados reais. Os resultados mostraram que o algoritmo *G-Swoosh* só pode ser usado em fontes de dados pequenas, principalmente quando *merges* ocorrem com frequência, enquanto os algoritmos *R-Swoosh* e *F-Swoosh* podem lidar com fontes de dados maiores. No trabalho não é definido o que é considerado como fonte de dados pequena ou com grandes volumes de dados. Contudo, os experimentos apresentados mostram que quanto maior a fonte de dados pior o desempenho do algoritmo *G-Swoosh*.

O trabalho destaca o uso do algortimo *F-Swoosh* original em um cenário incremental. Neste caso, as tuplas processadas previamente não são processadas entre elas, sendo realizadas comparações apenas entre as novas tuplas e entre as novas tuplas e as tuplas processadas previamente. Contudo, não foram realizados experimentos em cenários incrementais, apenas a formalização da aplicação do algoritmo em um processo incremental de RE.

Welch et *al*. (2012)

Em Welch et *al.* (2012) é apresentado o sistema *FastPath* com processamento paralelo, que tem como foco melhorar a qualidade de fontes de dados, reduzindo as tuplas duplicadas e processando novas tuplas o mais rápido quanto for possível, para que os mapeamentos entre as tuplas fiquem sempre atualizados e as fontes de dados sejam sempre confiáveis segundo o critério de tuplas duplicadas.

Basicamente, o sistema particiona as tuplas em blocos e, posteriormente, compara todos os pares de tuplas dentro de um mesmo bloco. Todas as novas tuplas inseridas nas fontes de dados são inseridas em blocos e são comparadas com os grupos de tuplas pré-existentes.

Os experimentos foram realizados em um cenário de recuperação de informações e foram realizadas comparações entre o *FastPath* e um processo tradicional de RE. Os resultados mostraram que o *FastPath* alcança valores similares de precisão e revocação se comparado ao processo tradicional de RE, mas com tempos menores de processamento.

O trabalho não apresenta detalhes do funcionamento do sistema, tão pouco dos algoritmos utilizados para o processo incremental de RE. Não são apresentadas definições que subsidiam o processo implementado no *FastPath*. Contudo, os resultados apresentados dos experimentos em um grande volume de dados, proveniente de fontes de dados reais, dão indícios da importância do processo incremental de RE.

Whang et *al*. (2013)

Em Whang et *al*. (2013) é utilizada uma abordagem *pay-as-you-go* para RE, na qual o incremento são tuplas ordenadas com o objetivo de minimizar o tempo de execução do processo de RE. A abordagem apresentada é um refinamento da estratégia apresentada em Whang & Garcia-Molina (2010).

Nesta abordagem apresentada, o incremento foi definido como a obtenção de resultados parciais o mais rápido quanto for possível. É importante destacar que os resultados parciais podem não identificar todos os registros que correspondem a mesma entidade do mundo real, no entanto, as entidades mais similares são identificadas nos primeiros passos.

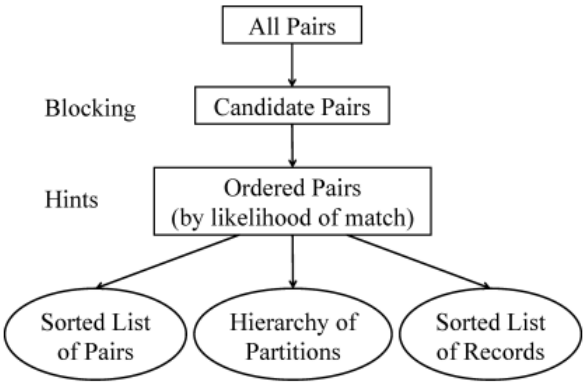
Para a implementação da abordagem *pay-as-you-go* são utilizadas dicas (pistas), coletadas em uma fase de análise prévia dos dados. O ideal é que considerando um mesmo intervalo restrito de tempo, a abordagem *pay-as-you-go* tenha resultados melhores que as abordagens tradicionais. No entanto, esta abordagem tem deficiências nos casos em que a fase de pré-análise tem um alto custo, ultrapassando, inclusive, o tempo total da execução do processo.

Em suma, as tuplas são agrupados em blocos, onde as comparações são realizadas entre as tuplas do mesmo bloco. Em sequência estes blocos são submetidos ao processo de ordenação (seguindo as dicas), para que o algoritmo de RE considere primeiro os registros mais próximos.

Considerando a Figura 6, 3 (três) tipos de dicas foram propostas: (i) Lista ordenada de pares de tuplas, (ii) Hierarquia de tuplas e (iii) Lista ordenada de tuplas. A estratégia de ordenar os pares de tuplas estima a distância entre pares de tuplas e os ordena em forma crescente. Na hierárquica, as tuplas são organizadas em *clusters* com menos detalhes de comparações no topo e a cada nível da hierarquia os *clusters* são refinados. Na lista

ordenada de tuplas (não são pares), as tuplas que aparecem no início da lista são mais prováveis de serem duplicadas que as do fim da lista.

Figura 6 – Framework *pay-as-you-go* para RE



Fonte: Whang et *al*., 2013.

Whang & Garcia-Molina (2014)

Em Whang & Garcia-Molina (2014) é apresentado uma extensão de Whang et *al*. (2013), com uma evolução das pistas e regras para detectar quais tuplas representam a mesma entidade do mundo real. Em Whang & Garcia-Molina (2014) o termo *pay-as-you-go* é substituído por incremental e a cada nova mudança nas fontes de dados, a Resolução de Entidades (RE) é aplicada e os mapeamentos entre tuplas são atualizados.

Em suma, o processo incremental de RE proposto consiste no processamento de um conjunto inicial de tuplas, gerando uma regra R_1 que define se duas tuplas representam a mesma entidade do mundo real. Ao longo do tempo, novas tuplas podem ser avaliadas e a regra R_1 é refinada, gerando uma nova regra R_2 e, assim, sucessivamente.

O principal foco do trabalho é identificar se a materialização de resultados intermediários do processo de RE pode melhorar o desempenho de processos futuros de RE. Experimentos em fontes de dados reais foram realizados a fim de identificar o impacto da evolução das regras e a eficiência de algoritmos de clusterização para RE.

Concluiu-se que evoluir as regras que identificam se duas tuplas representam a mesma entidade do mundo real possui um custo menor que reprocessar todas as tuplas das fontes de dados (RE tradicional). Adicionalmente, foi realizado um estudo sobre o espaço para armazenamento dos resultados parciais de RE e seu custo. Os experimentos mostraram que o custo é amortizado com a redução futura de evoluções das fontes de dados. Por fim,

foi apresentada a qualidade dos resultados alcançados com regras em um processo incremental de RE. Neste caso, detectou-se uma pequena perda na qualidade dos resultados, mas com ganho em desempenho

Gruenheid et *al*. (2014)

A solução proposta por Gruenheid et *al*. (2014) considerou desafios gerados pelas características de Big Data: (i) O grande volume de tuplas muitas vezes implica em um longo tempo de processamento da RE. (ii) A velocidade com a qual as tuplas são atualizadas, rapidamente torna os mapeamentos obsoletos. Estes problemas podem ser contornados por meio de soluções incrementais de forma eficiente e precisa.

É apresentado um *framework* que atualiza de forma incremental as evidências de *links* entre tuplas. A solução proposta baseia-se em técnicas de clusterização, associadas a estruturas de grafos.

Cada *cluster* corresponde a uma entidade do mundo real. Os incrementos são realizados sempre que uma tupla é inserida, removida ou atualizada. Adicionalmente, quando ocorre a alteração em um *cluster*, são avaliadas novas evidências, com o objetivo de corrigir erros anteriores na construção dos *clusters*. A solução proposta foi instanciada com 2 (dois) algoritmos de clusterização, ambos sem a necessidade de conhecer previamente o número de *clusters*: *Correlation* e *DB-index*. No trabalho foram propostos dois modelos de solução: Incremental e Gulosa.

No modelo incremental, os nodos a serem inseridos no grafo são propagados por meio dos vértices alcançáveis, que fazem ligações entre as tuplas indicando a similaridade. Esta estratégia produz solução ótima, mas em situações onde o grafo possui alta densidade, a complexidade da solução é exponencial.

A solução gulosa é uma variação do algoritmo incremental, contudo possui complexidade polinomial e menor índice de precisão. Neste cenário, apenas os *clusters* diretamente ligados ao incremento (novos nodos a serem inseridos) são avaliados e a cada iteração verifica-se a possibilidade de união entre dois *clusters*, divisão de um *cluster* e movimentação de um nodo de um *cluster* para outro. Estas operações têm como função objetivo intensificar o grau de coesão dentro de cada *cluster* e atenuar a correlação entre estes.

A estratégia foi validada para dados tabulares, no entanto, no trabalho é enfatizada a extensão da solução para qualquer tipo de dados que possam ser representados por um grafo, onde os vértices são as entidades e as arestas os valores de similaridade. Todas as funções de similaridade avaliadas consideram apenas aspectos sintáticos dos dados. Como trabalho futuro, identificou-se a avaliação do problema de RE em fontes web, assim como soluções incrementais para textos não estruturados.

Altowim et al. (2014)

No trabalho de Altowin et *al*. (2014) é proposto um processo de RE considerando dados relacionais. O principal foco é considerar a qualidade da solução e o tempo disponível, maximizando a qualidade sem ultrapassar a variável tempo estipulada. De forma progressiva, as tuplas mais importantes das fontes de dados são selecionadas e o plano de execução é traçado e executado. Neste contexto, tuplas mais relevantes são aquelas que proporcionam inferência de mais pares de tuplas duplicados a partir de regras de transitividade, reduzindo, portanto, o número de pares de tuplas a serem comparados.

Durante o plano de execução do processo são definidos quais blocos de tuplas devem ser processados e quais pares de tuplas são mais importantes, para que estes sejam solucionados primeiro. Para estas decisões são utilizadas probabilidades que estimam o impacto da inferência de afirmar ou negar que duas tuplas são duplicadas.

A solução é baseada em grafos, onde cada vértice é um par de tuplas e as arestas indicam dependências entre os vértices. Estas dependências indicam que se em um vértice o par é duplicado, no vértice dependente o par também o é. Estas dependências são originárias das dependências entre as relações do modelo relacional de dados.

Como função de similaridade é utilizado um conjunto pré-definido de funções que são utilizadas de acordo com o tipo de atributo a ser comparado. Possivelmente, mais de uma função pode ser utilizada por atributo. Para a realização do processo de agrupamento dos dados (*blocking*) são construídas chaves a partir de *substrings* de cada atributo.

A solução proposta é avaliada apenas em bancos de dados relacionais, obtendo planos de seleção de instâncias eficientes e que geram resultados de alta qualidade se comparado a outros trabalhos que seguem a mesma proposta.

Ramadan et *al*. (2015)

Para o processo de RE incremental onde o incremento considerado é um resultado de consulta, foi encontrado apenas o trabalho de Ramadan et *al*. (2015), como variação dos trabalhos de Ramadan et *al*. (2013), Ramadan & Christen (2014) e Ramadan & Christen (2015).

Ramadan et *al*. (2015) apresentam índices dinâmicos para Resolução de Entidades. O foco do trabalho é a recuperação de informações, com o objetivo de recuperar todas as tuplas que respondem a uma determinada consulta e identificar quais delas representam a mesma entidade do mundo real. Na área de recuperação de informações, as técnicas de indexação geralmente são usadas para extrair das fontes de dados um conjunto de tuplas candidatas que são semelhantes a uma tupla consultada e que devem ser comparadas em mais detalhes.

O trabalho destaca que a indexação tradicional é usada com sucesso em cenários que consideram apenas as fontes de dados em um dado instante de tempo, em processos de Resolução de Entidades em lote (tradicionais). Estes índices não são adequados para Resolução de Entidades em fontes de dados dinâmicas. Com esta motivação, o trabalho adapta o método tradicional de indexação de vizinhança ordenada [Christen, 2012] e propõe um método de indexação dinâmica baseado em árvores adequado para Resolução de Entidades em tempo de consulta.

Para reduzir o tempo de processamento da Resolução de Entidade em tempo de consulta, é proposto um pré-cálculo dos valores de similaridade dos atributos, representados como nós vizinhos em uma árvore. Foram realizados experimentos em fontes de dados reais e sintéticas e realizadas comparações entre o índice proposto e índices tradicionais definidos na literatura em um cenário de recuperação de informações. De forma geral, o índice proposto mostrou-se escalável, com rápida inserção de valores, atenuando o tempo de resposta de uma consulta específica.

Firmani et *al*. (2016)

Firmani et *al*. (2016) apresenta um processo *online* para RE. Neste caso, a motivação é que o usuário final pode querer tomar decisões durante o processo de RE, ao invés de aguardar a conclusão de todas as etapas da RE. O processo *online* de RE tem os mesmos princípios do processo *pay-as-you-go* ou incremental para RE.

No processo *online* apresentado, um *oracle* é utilizado para responder perguntas booleanas que determinam se duas tuplas representam a mesma entidade do mundo real. O foco da estratégia é reduzir o número de perguntas submetidas ao *oracle*, de acordo com a ordem em que pares de tuplas são avaliados. Presume-se que o menor número de acessos ao *oracle* implica em um melhor desempenho da abordagem. Para reduzir o número de acessos ao *oracle*, é utilizada a função de transitividade entre pares de tuplas.

Para o processo, a entrada considerada é um grafo, onde os vértices são as tuplas e as arestas os valores de similaridade entre um par de tuplas. Para realizar a RE *online* é proposto um algoritmo guloso, baseado na ordenação das arestas e dos vértices proposta por Wang et *al*. (2013) e Vesdapunt et *al*. (2014).

Foram realizados experimentos em fontes de dados reais e sintéticas, com versões do algoritmo proposto sequencial e paralelo. Os experimentos mostraram que o algoritmo proposto tem resultados com cobertura melhor que os resultados apresentados em estratégias tradicionais, mesmo em cenários com muitas tuplas duplicadas. Como trabalho futuro é indicado uma melhoria do algoritmo em casos que o *oracle* não possui a resposta correta, minimizando os erros do processo de RE.

3.4 Quadro Comparativo

De forma comparativa, foram avaliadas algumas características de cada um dos trabalhos apresentados (Quadro 2 e Quadro 3). As células com valor "(-)" indica que a informação sobre a característica não foi encontrada no trabalho A última linha do quadro apresenta as características do QUIPER, proposto neste trabalho:

- Área: Em qual área da computação o trabalho deu ênfase: integração de dados (ID) ou recuperação da informação (RI).

- Redução de escopo: Indica qual abordagem é utilizada para reduzir o número de dados a serem processados, objetivando a redução do número de comparações entre entidades. Esta característica pode ser definida como: (i) Bloco, utiliza estratégia de criação de blocos; (ii) Consulta, utiliza consulta para selecionar apenas as entidades que são importantes para responder à consulta; (iii) Consulta + Bloco, utiliza uma combinação das duas características anteriores.

- Abordagem: Sintetiza o tipo de abordagem utilizada no processo de RE. Esta foi diferenciada em: orientada à consulta ou incremental.

- Algoritmo: Indica a estratégia utilizada no algoritmo de RE. Esta pode ser: supervisionada (S) ou não supervisionada (NS), como descrito no Capítulo 2, na Seção 2.2.3.

- Similaridade: Identifica se a função de similaridade utilizada para definir se duas tuplas representam a mesma entidade do mundo real, faz análise sintática, semântica ou híbrida.

- Seleção de Atributos: Indica se a estratégia utiliza uma solução manual ou automática.

- Modelo de Dados: Identifica os modelos de dados aos quais a abordagem foi aplicada.

- Processamento: Indica se o algoritmo foi implementado de forma sequencial ou paralela.

Quadro 2 - Comparativo dos trabalhos relacionados

Critério / Trabalho	Área	Redução de escopo	Abordagem	Algoritmo	Similaridade	Seleção de Atributos	Modelo de Dados	Processamento
Bhattacharya & Getoor (2007)	RI	Consulta + Blocagem	Orientada à Consulta	NS	Híbrida	Manual	Relacional	Sequencial
Su et *al*. (2010)	ID	Consulta + Blocagem	Orientada à Consulta	NS	Sintática	Automática	Relacional	Sequencial
Salloum et *al*. (2013)	RI	Consulta + características incrementais das fontes	Orientada à Consulta	NS	Manual	Manual	(-)	Sequencial
Altwaijry et *al.* (2013)	ID	Consulta + Blocagem	Orientada à Consulta	NS	Sintática	Manual	Relacional	Sequencial
Benjelloun et *al*. (2009)	ID	Blocagem	Incremental (alteração nas fontes de dados) e Tradicional	NS	Sintática	Manual	(-)	Sequencial
Welch et *al*. (2012)	RI	Blocagem	Incremental (alteração nas fontes de dados)	NS	Sintática	Manual	(-)	Paralelo

Quadro 3 - Continuação do comparativo dos trabalhos relacionados

Critério / Trabalho	Área	Redução de escopo	Abordagem	Algoritmo	Similaridade	Seleção de Atributos	Modelo de Dados	Processamento
Whang et *al.* (2013)	ID	Blocagem	Incremental (ordenação das tuplas)	NS	Sintática	Manual	(-)	Sequencial
Whang & Garcia-Molina et *al.* (2014)	ID	Blocagem	Incremental (alteração nas fontes de dados)	NS	Sintática	Manual	Relacional	Sequencial
Gruenheid et *al.* (2014)	ID	Blocagem	Incremental (alteração nas fontes de dados)	NS	Sintática	Manual	Relacional	Sequencial
Altowim et *al.* (2014)	ID	Blocagem	Incremental (ordenação das tuplas)	S	Sintática	Todos os atributos	Relacional	Sequencial
Ramadan et *al.* (2015)	RI	Blocagem + Consulta	Incremental (resultados de consultas)	NS	Sintática	Manual	Relacional	Sequencial
Firmani et *al.* (2016)	ID	Blocagem	Incremental (ordenação das tuplas)	NS	Sintática	Manual	(-)	Sequencial e Paralelo
QUIPER	ID	Blocagem + Consulta	Incremental (resultados de consultas)	NS	Sintática	Manual	Relacional	Sequencial

A partir das informações sintetizadas nos Quadros 2 e 3, observa-se que a grande maioria dos trabalhos relacionados avaliam o processo de RE sobre dados relacionais e com funções de similaridade sintáticas. Com relação a abordagem de RE aplicada, observa-se que a grande maioria se concentra na RE orientada à consulta ou incremental, sem considerar uma combinação das duas abordagens.

O trabalho de Ramadan et *al*. (2015) é o único que apresenta uma solução para RE incremental e orientada à consulta. Contudo, tem o foco em recuperação de informação, onde a RE é aplicada com o objetivo de recuperar todas as tuplas que respondem a uma consulta específica. Ademais, o foco principal da pesquisa é apresentar índices dinâmicos para o processo de RE em recuperação de informação. Neste caso, não é apresentado, nem avaliado, o processo de RE como um todo.

É importante destacar lacunas em trabalhos que investigam a seleção automática de atributos para as funções de similaridade da etapa de comparação entre pares de tuplas, tal como funções de similaridade que avaliam a semântica dos dados. Por fim, observou-se que trabalhos recentes de RE em grandes volumes de dados estão investigando o processamento paralelo das etapas do processo de RE a fim minimizar o tempo de execução do processo de RE como um todo.

3.5 Considerações

Este capítulo apresentou os principais trabalhos que possuem interseção com esta pesquisa. Os trabalhos relacionados foram divididos de acordo com a abordagem utilizada para o processo de RE: orientada à consulta ou incremental com suas variações.

Também foi realizada uma análise comparativa dos trabalhos relacionados, realizando uma síntese e destacando as diferenças com relação a este trabalho de doutorado. Nesta classificação, não foram encontrados na literatura trabalhos na área de Integração de Dados que aplicam o processo de RE incremental e em tempo de consulta.

4 RESOLUÇÃO DE ENTIDADES INCREMENTAL E ORIENTADA À CONSULTA

4.1 Introdução

Este trabalho propõe um processo incremental e orientado à consulta para Resolução de Entidades, denominado por QUIPER (*Query-time and Incremental Process for Entity Resolution*). Neste sentido, objetiva-se reduzir o número de comparações realizadas no processo de Resolução de Entidades (RE), processando apenas as tuplas que compõem os resultados de uma consulta, ao invés de considerar todo o conjunto de tuplas disponível. Com o objetivo de minimizar o tempo de processamento em tempo de consulta, é proposta a utilização de resultados anteriores do processo a fim de auxiliar o processo de RE em resultados de novas consultas.

O conteúdo do capítulo é estruturado como segue: Seção 4.2 apresenta um exemplo motivacional; Seção 4.3 define conceitos e notações necessárias para o entendimento do processo proposto; Seção 4.4 apresenta o QUIPER; Seções 4.5 e 4.6 apresentam os algoritmos do QUIPER para as etapas nas quais este trabalho tem maiores contribuições; Seção 4.7 apresenta um exemplo completo para o entendimento do QUIPER; Seção 4.8 discorre as considerações finais do capítulo.

4.2 Exemplo Motivacional

Para facilitar o entendimento do processo QUIPER, considere um conjunto de fontes de dados bibliográficos, as quais devem ser integradas a fim de obter uma visão uniforme referente a artigos, conferências e autores. Assuma um sistema de integração que adota uma abordagem virtual, i.e., as tuplas são recuperadas das fontes de dados e integradas sob demanda.

Durante o processo de integração de dados, um conjunto de consultas acessa as mesmas fontes de dados e a RE incremental não necessita resolver todas as tuplas disponíveis nas fontes de dados. É necessário aplicar a RE apenas nas tuplas que respondem a consulta e que não foram processadas previamente.

Suponha uma consulta q_1 que é submetida ao sistema de integração: "*Recupere todos os autores que publicaram no ano de 2012"*. Com a finalidade de recuperar as tuplas, o sistema de integração processa q_1 sobre as múltiplas fontes de dados disponíveis. O resultado de q_1 é um conjunto de tuplas que são submetidas como entrada ao processo QUIPER. O processo de RE objetiva identificar autores de diferentes fontes de dados que representam a mesma entidade do mundo real. Por exemplo, o autor '*Tony Presley*' da fonte de dados '*1*' pode representar a mesma entidade que o autor '*Tonia Presley*' da fonte de dados '*3*'.

Este exemplo motivacional será detalhado ao longo deste capítulo à medida que detalhes do processo QUIPER forem sendo apresentados.

4.3 Definições Gerais e Notações

Como mencionado anteriormente, o processo de RE é essencialmente um problema de gerar grupos de tuplas que representam a mesma entidade do mundo real, podendo ser mapeado para um problema de clusterização, no qual cada *cluster* contém um conjunto de tuplas que representam a mesma entidade do mundo real. É importante notar que quando resultados de consultas (conjuntos de tuplas) são considerados em um sistema de integração de dados, cada tupla pode ser proveniente de uma fonte de dados diferente. Considerando as características de um sistema de integração de dados, assume-se que:

- O mapeamento entre esquemas foi realizado na etapa de *Correspondência entre Esquemas* do processo de integração de dados;

- Todos os *clusters* são criados considerando apenas um conceito, i.e., a clusterização é feita por *Autor*, *Artigo* ou *Conferência*, por exemplo. Isto permite o reúso de *clusters* previamente criados. Por exemplo, se uma consulta requer informações sobre os conceitos *Autor* e *Artigo*, e para identificar um autor é necessário desambiguar artigos, os dois conceitos são clusterizados separadamente e o processo de RE combina os dois resultados. Em um segundo momento, se uma consulta requer informações apenas do conceito *Autor*, os *clusters* previamente criados para *Autor* podem ser reusados.

Para facilitar a leitura deste trabalho, segue uma sumarização das notações utilizadas:

- S: Um conjunto de fontes de dados;
- s_k: Uma fonte de dados;

- $s_k.Id$: O identificador da fonte de dados;
- *L:* Um conjunto de conceitos do mundo real, por exemplo *Autor, Pessoa, Universidade,* entre outros;
- l_z: Um conceito do mundo real;
- A_z: Um conjunto de atributos;
- $A_z.re$: Um conjunto de atributos relevantes;
- a_x: Um atributo;
- *Q:* Um conjunto de consultas;
- q_d: Uma consulta;
- *T:* Um conjunto de tuplas;
- t_k: Uma tupla;
- $t_k.Id$: O identificador da tupla;
- *BlockingKey*: Uma chave de blocagem;
- *ChaveDeAcesso*: Uma chave de acesso ao índice;
- *ClusterId*: O identificador do *cluster*.

Para as definições a seguir, considere $S = \{s_1, s_2, ..., s_m\}$, um conjunto de fontes de dados, $Q = \{q_1, q_2, ..., q_n\}$, um conjunto de consultas executadas sobre *S*. O resultado de q_i é um conjunto de tuplas $T = \{t_1, t_2, ..., t_p\}$, no qual cada tupla está relacionada com um conceito l_z que representa um conjunto de objetos do mundo real (*Autor*, *Artigo* e *Conferência*, são exemplos de conceitos). Cada fonte de dados é descrita por um conjunto de conceitos *L*.

Cada conceito $l_z \in L$ é descrito por um conjunto de atributos, $A = \{Id, a_2, a_3, ..., a_s\}$. Dentre os atributos de l_z, existe um atributo *Id*, cujos valores identificam unicamente cada instância de l_z. Neste trabalho, considerou-se que cada consulta de *Q* é reformulada e seleciona o atributo *Id* e os atributos relevantes de um conceito. Os atributos relevantes de um conceito são definidos como segue.

Definição 1 (Conjunto de Atributos Relevantes de um Conceito): Dentre os atributos associados a um conceito l_z, $A_z = \{a_1, a_2, ..., a_s\}$, existe um subconjunto de atributos relevantes, denotado por $A_z.re$, onde $A_z.re \subseteq A_z$ e A_z.re é formada por atributos que possibilitam a distinção/semelhança entre duas instâncias (tuplas). Por exemplo, o conceito *Autor* possui o seguinte conjunto de atributos: *A = {Nome, Filiação, Endereço}* e para identificar se dois autores representam a mesma entidade do mundo real, têm-se

que o conjunto de atributo relevantes é: *A.re = {Nome, Filiação}*. Este conjunto de atributos relevantes pode ser definido por um especialista de domínio ou por meio de uso de técnicas automáticas apresentadas na literatura [Su et *al*., 2010; Canalle et *al*, 2015; Canalle et *al*., 2017].

Desta forma, suponha que uma consulta q_d solicita a *Filiação* do conceito *Autor*. A consulta reformulada de q_d consistirá em solicitar, de forma transparente ao usuário, os atributos *Nome* e *Filiação*, dado que os atributos *Nome* e *Filiação* são atributos relevantes, apesar de não estarem especificados em q_d.

Após a execução de q_d, o processo QUIPER, considera como entrada o resultado de q_d. O resultado de uma consulta é definido como segue:

Definição 2 (Resultado de uma Consulta q_d): O resultado de uma consulta é definido como um conjunto de tuplas *T*. Uma tupla é definida como segue.

Definição 3 (Tupla): Uma tupla t_k pertencente a *T* é definida como um par, $t_k = (s_k.Id, AT)$, no qual $s_k.Id$ é o identificador da fonte de dados à qual t_k pertence e *AT* denota uma lista de pares $[(t_{k.Id}, v_1)(a_2, v_2), (a_3, v_3), \ldots, (a_q, v_q)]$, na qual a_x denota um atributo pertencente ao conceito ao qual t_k está relacionada e v_x é seu valor. A tupla deve conter todos os atributos relevantes do conceito (*A.re*), assim como o atributo que a identifica unicamente. Neste caso, a tupla t_k tem um par $(t_k.Id, v_y)$, no qual $t_k.Id$ é o atributo identificador e v_y é seu valor.

Considerando o exemplo motivacional (Seção 4.2), a consulta q_1 foi executada sobre o conceito *Autor*. A Figura 7 apresenta as tuplas pertencentes ao resultado da consulta q_1. Cada linha da tabela é uma tupla com identificador da fonte ($s_k.Id$), identificador da tupla ($t_k.Id$), *Nome* e *Filiação* de *Autor*.

Para cada conjunto de tuplas *T* que corresponde ao resultado de uma consulta, blocos de tuplas são criados. Blocos são definidos como segue.

Figura 7 - Lista de tuplas do resultado de q_1

Autor			
s_k.Id	t_k.Id	Nome	Filiação
1	110	Tony Presley	University of Manchester
2	113	Toni	Stanford University
3	120	Tonia Presley	University of Manchester
4	115	Tony	Stanford University
5	134	Tonia	Stanford University
4	245	Alicea	Stanford University
2	849	Alice	Stanford University

Fonte: Próprio autor, 2017.

Definição 4 (Bloco de Tuplas): Um bloco é definido como um par, Bloco = (*BlockingKey*, *R*), no qual *BlockingKey* é a chave de blocagem gerada para o bloco por meio de uma função de blocagem e *R* denota uma lista de tuplas pertencentes a um resultado de q_i [t_1, $t_{2...,}$ t_n] que possuem a mesma *BlockingKey*.

A Figura 8 apresenta 2 (dois) blocos de tuplas do resultado de q_1. Para cada tupla do resultado da consulta, uma *BlockingKey* é gerada. Tuplas com a mesma *BlockingKey* são inseridas no mesmo bloco. As *BlockingKey* geradas foram '*tn*' e '*als*'. Estes valores foram gerados por meio da função de blocagem Double-Metaphone [Christen, 2012] sobre o atributo *Nome*.

Figura 8 – Blocos de tuplas pertencentes a q_1

	Autor				
	BlockingKey	s_k.Id	t_k.Id	Nome	Filiação
Blocos	tn	1	110	Tony Presley	University of Manchester
	tn	2	113	Toni	Stanford University
	tn	3	120	Tonia Presley	University of Manchester
	tn	4	115	Tony	Stanford University
	tn	5	134	Tonia	Stanford University
	als	4	245	Alicea	Stanford University
	als	2	849	Alice	Stanford University

Fonte: Próprio autor, 2017.

A abordagem proposta faz uso de dois tipos de *clusters*: os globais e os locais, os quais serão descritos a seguir.

Um *Cluster* Global armazena um conjunto de identificadores de tuplas que correspondem à mesma entidade do mundo real. A estratégia é utilizar estes *Clusters* Globais para armazenar informações (histórico) sobre tuplas que foram consideradas duplicadas por processos de RE aplicados previamente. Estes resultados prévios são utilizados para reduzir o tempo de processamento de comparação entre tuplas em novas tarefas de RE.

Definição 5 (*Cluster* Global): Um *Cluster* Global (CG) é definido por um par, CG = (*ClusterId, P*), no qual *ClusterId* é o identificador do *Cluster* Global e *P* denota uma lista de pares [($s_{1...m}$.*Id*, $t_{1...n}$.*Id*)], no qual s_k.*Id* denota o identificador da fonte à qual a tupla com identificador t_k.*Id* pertence.

Cada bloco de tuplas (Definição 4) tem uma ou mais tuplas. Estas tuplas pertencem a um *Cluster* Local. Um *Cluster* Local armazena uma lista de tuplas que correspondem à mesma entidade do mundo real. Os *Clusters* Locais são criados e, posteriormente, inicializados a partir das informações dos *Clusters* Globais. Diferentemente dos *Clusters* Globais, os *Clusters* Locais são criados para um resultado de consulta específico e não são preservados, i.e., ao término do processo de RE eles são descartados. Um *Cluster* Local é definido como segue.

Definição 6 (*Cluster* Local): Um *Cluster* Local (CL) é definido por um par, CL = (*ClusterId*, T_C), no qual *ClusterId* é o identificador do *Cluster* Local e $T_C = [t_1, t_2, \ldots, t_n]$ denota uma lista de tuplas que pertencem ao CL.

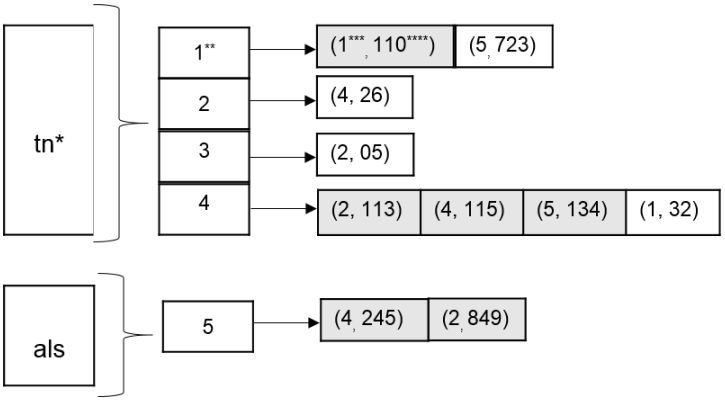
Um Índice de *Clusters* (IC) indexa uma lista de *Clusters* Globais e é definido como segue.

Definição 7 (Índice de *Clusters*): Um Índice de *Clusters* (IC) é definido por uma lista de pares, $IC = [(ChaveDeAcesso_1, CGList_1), (ChaveDeAcesso_2, CGList_2), \ldots, (ChaveDeAcesso_n, CGList_n)]$, na qual $ChaveDeAcesso_k$ é a chave de acesso ao índice e $CGList_k$ denota uma lista de *Clusters* Globais $[CG_1, CG_2, \ldots, CG_n]$. O valor da chave de acesso é o mesmo valor da chave de blocagem (*BlockingKey*) do bloco de tuplas apresentado na Definição 4.

A Figura 9 apresenta Índices de *Clusters* para o exemplo motivacional. As chaves de acesso (*ChaveDeAcesso*) são '*tn*' e '*als*'. Os identificadores dos *Clusters* Globais

(*ClusterId*) são ‘*1*’, ‘*2*’, ‘*3*’, ‘*4*’ e ‘*5*’. No par (*1, 110*), ‘*1*’ é o identificador da fonte e ‘*110*’ o identificador da tupla. Na Figura 9 existem alguns pares destacados (por exemplo, (*1, 110*)). Estes serão utilizados ao longo da explicação do processo QUIPER.

Figura 9 – Índices de *Clusters* Globais



* tn: A chave de acesso (*ChaveDeAcesso)*
** 1: O identificador do *Cluster* Global *(ClusterId)*
*** 1: O identificador da fonte ($s_k.Id$)
**** 110: O identificador da tupla ($t_k.Id$)

Fonte: Próprio autor, 2017.

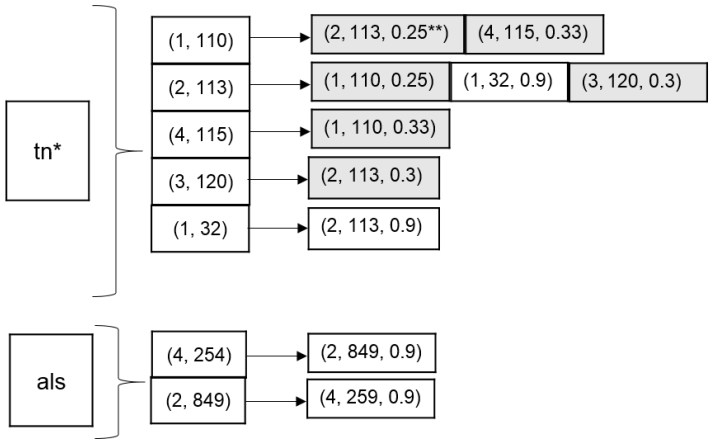
Para identificar se uma tupla é duplicada com relação a um conjunto de tuplas em um resultado de consulta (se pertencem ao mesmo *Cluster* Local), é necessário comparar esta tupla com todas as demais tuplas do mesmo bloco de tuplas (i.e., com a mesma *BlockingKey*). O Índice de Similaridades (IS) indexa valores de similaridade entre pares de tuplas e é definido como segue.

Definição 8 (Índice de Similaridades): O Índice de Similaridades (IS) é definido por uma lista de pares, $IS = [(ChaveDeAcesso_1, Lis_1), (ChaveDeAcesso_2, Lis_2), \ldots, (ChaveDeAcesso_n, Lis_n)]$, na qual $ChaveDeAcesso_i$ é a chave de acesso ao índice e Lis_i denota uma lista de pares $Lis_i = [(s_{1\ldots m}.Id, t_{1\ldots n}.Id)]$, nos quais $s_k.Id$ é o identificador da fonte e $t_k.Id$ o identificador da tupla. Cada par $(s_k.Id, t_k.Id)$ está relacionado a uma lista de triplas $Sim_k = [(s_{1\ldots x}.Id, t_{1\ldots y}.Id, simValue_{1\ldots z})]$, nas quais $simValue_l$ é o valor de similaridade entre a tupla $t_k.Id$ da fonte de dados $s_k.Id$ e a tupla $t_1.Id$ da fonte de dados $s_1.Id$. O valor da chave de acesso ($ChaveDeAcesso_i$) é o mesmo valor da chave de blocagem (*BlockingKey*) do bloco de tuplas (Definição 4).

Figura 10 apresenta Índices de Similaridades para o exemplo motivacional. As chaves de acesso (*ChaveDeAcesso*) são ‘*tn*’ e ‘*als*’. Os valores indexados são valores de

similaridade entre pares de tuplas. Por exemplo, o valor de similaridade entre a tupla '*110*'da fonte de dados '*1*' e a tupla '*113*' da fonte de dados '*2*' é '*0.25*'. Na Figura 10 existe algumas triplas destacadas (por exemplo, (2, *113*, *0.25*)). Estas serão utilizadas ao longo da explicação do processo QUIPER.

Figura 10 - Índices de Similaridades



* tn: A chave de acesso (*ChaveDeAcesso*)
** 0.25: O valor de similaridade entre a tupla '*110*' da fonte de dados '*1*' e a tupla '*113*' da fonte de dados '*2*'

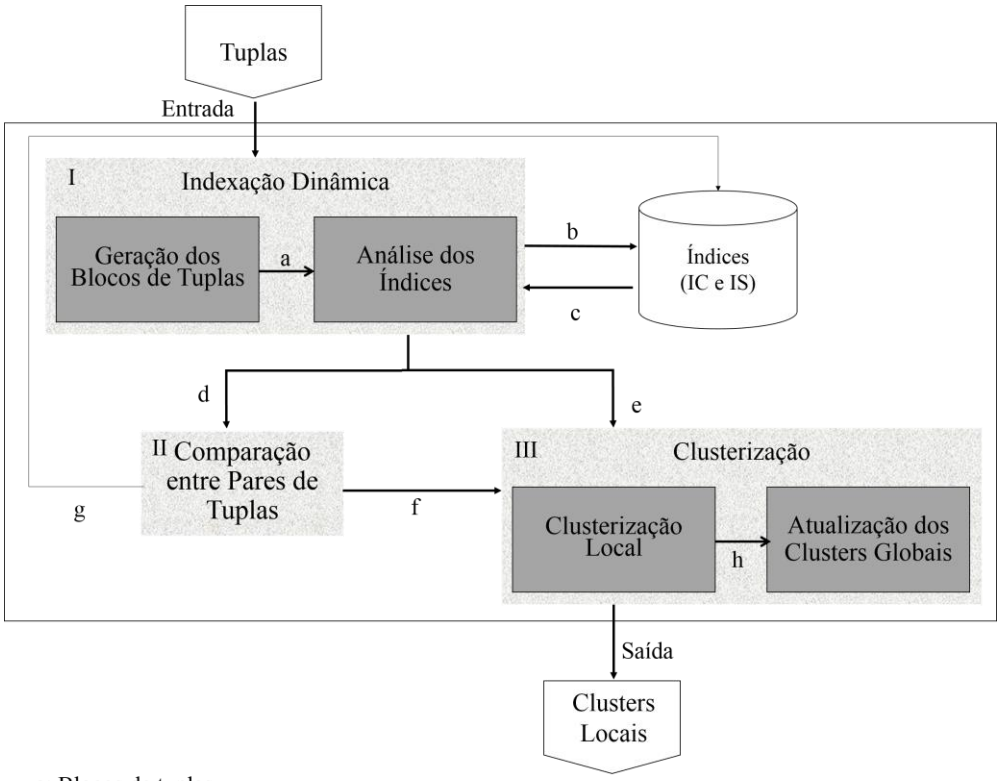
Fonte: Próprio autor, 2017.

4.4 Processo QUIPER

Considerando um conjunto de tuplas *T*, provenientes do resultado de uma consulta, o processo de Resolução de Entidades (RE) consiste, basicamente, em um problema de clusterização, onde cada *cluster* contém, de forma simplificada, as tuplas t_k que representam uma mesma entidade do mundo real.

O processo QUIPER é apresentado na Figura 11. Em estratégias onde as tuplas a serem processadas são selecionadas via consulta (por um usuário ou sistema), não se tem a garantia de que os atributos especificados na consulta são suficientes para distinguir duas tuplas. Nestes casos, os trabalhos disponíveis na literatura realizam expansões na consulta, a fim de incluir atributos que podem não ter sido especificados na consulta, mas que auxiliam na decisão de definir se duas tuplas representam a mesma entidade do mundo real [Bhattacharya & Getoor, 2007; Ioannou et *al*., 2010; Su et *al*., 2010; Altwaijry et *al*., 2013; Altwaijry et *al*., 2014; Canalle et *al*, 2015]. Desta forma, a entrada do processo QUIPER é um conjunto de tuplas segundo as Definições 2 e 3.

Figura 11 - QUIPER



a: Blocos de tuplas
b: Blocos de tuplas a serem pesquisados nos índices
c: Clusters Globais e valores de similaridade
d: Tuplas não comparadas em iterações anteriores
e: Tuplas comparadas em iterações anteriores
f: Novos valores de similaridade
g: Novos valores de similaridade
h: *Clusters* Locais

Fonte: Próprio autor, 2017.

Como a seleção de atributos relevantes é uma tarefa que não é o foco deste trabalho, foram utilizadas estratégias já existentes na literatura para selecionar os atributos mais relevantes a serem utilizados [Su et *al*., 2010; Canalle et *al*, 2015]. Além disso, considera-se que esta seleção foi um pré-processamento e que o conjunto de tuplas *T* pertencente a cada resultado de consulta processado possui os atributos relevantes.

Em linhas gerais, o processo proposto contempla as seguintes etapas: (I) *Indexação Dinâmica*, (II) *Comparação entre Pares de Tuplas* e (III) *Clusterização*.

I. A etapa de *Indexação Dinâmica* objetiva reduzir o tempo de execução do processo de RE em tempo de consulta. Esta etapa é dividida em duas tarefas: (a) *Geração dos Blocos de Tuplas* e (b) *Análise dos Índices*.

a. A tarefa de *Geração dos Blocos de Tuplas*, funciona como um filtro que é aplicado sobre cada conjunto de tuplas *T* que corresponde ao resultado de uma consulta q_i. Para cada tupla t_k pertencente a *T* é gerada uma chave de blocagem (*BlockingKey*). Esta chave é utilizada como chave de acesso aos índices propostos neste trabalho para recuperar informações pré-calculadas (calculadas para resultados de consultas processadas anteriormente). Esta chave de blocagem é criada baseada em um ou múltiplos valores de atributos. A Figura 8 apresenta o resultado desta tarefa considerando o resultado de q_1 do exemplo motivacional (Figura 7). Nesta tarefa, os pares de tuplas que provavelmente representam a mesma entidade do mundo real são inseridos no mesmo bloco de tuplas.

b. A tarefa de *Análise dos Índices* acessa índices para recuperar resultados de etapas do processo QUIPER, executadas sobre resultados de consultas recebidos anteriormente, a fim de reutilizar em execuções futuras. Para este propósito, para cada bloco de tuplas, dois índices são acessados: Índice de *Clusters* (IC) e Índice de Similaridades (IS). Ao término desta tarefa, tuplas de um mesmo bloco de tuplas que não foram comparadas em iterações anteriores, são identificadas e enviadas como entrada para a etapa de *Comparação entre Pares de Tuplas* (Etapa II). Adicionalmente, informações sobre *Clusters* Globais (indexados no IC), com tuplas que foram comparadas em execuções anteriores do QUIPER, são enviadas como entrada para a etapa de *Clusterização* (Etapa III).

II. Durante a etapa de *Comparação entre Pares de Tuplas*, os valores de similaridade entre pares de tuplas que não foram comparadas em iterações anteriores do processo QUIPER, e que pertencem ao mesmo bloco, são calculados e seus valores correspondentes enviados para a etapa de *Clusterização* (Etapa III). Posteriormente, estes novos valores são utilizados para a atualização do IS.

III. A etapa de *Clusterização* identifica se um par de tuplas representa a mesma entidade do mundo real ou não. Esta etapa tem duas tarefas: (a) *Clusterização Local* e (b) *Atualização de Clusters Globais*.

a. A tarefa de *Clusterização Local* cria e inicializa os *Clusters* Locais (CL). Nesta tarefa, tuplas são inseridas em CL existentes ou novos CL são criados. Tuplas duplicadas são inseridas em um mesmo CL.

b. A tarefa de *Atualização dos Cluster Globais* mapeia os CL para os *Clusters* Globais (CG), i.e., os CG são atualizados de acordo com os CL gerados durante a tarefa de *Clusterização Local*.

A saída do processo QUIPER consiste em uma lista de CL. Diferentemente dos CG, os CL são criados para um conjunto de tuplas específico e não são preservados, i.e., ao término do processo de RE eles são descartados.

O Algoritmo 1 apresenta detalhes do processo QUIPER (Figura 11). O algoritmo é executado sobre um conjunto de tuplas (*T*) que corresponde ao resultado de uma consulta q_i. Para cada tupla pertencente a *T*, o algoritmo gera uma chave de blocagem (*BlockingKey*) por meio do procedimento *GeraçãoChaveBlocagem* e insere as tuplas com mesma chave de blocagem no mesmo bloco de tuplas. Esta chave de blocagem é gerada por meio de uma função da literatura [Christen, 2012] (linha 14). Depois, o Índice de *Clusters* (IC) é acessado, por meio da chave de acesso que é similar a chave de blocagem gerada. Nesta etapa, os *Clusters* Locais (CL) são inicializados a partir dos *Clusters* Globais (CG) (linha 15). Por fim, os CL são inseridos em uma lista de CL (*ListaCL*). As tuplas de um mesmo CL pertencem a um mesmo bloco de tuplas. Desta forma, os CL são agrupados pela chave de blocagem de suas tuplas (Detalhes na Seção 4.5.1).

Em seguida, para cada par de tuplas que pertencem ao mesmo bloco de tuplas, o Índice de Similaridades (IS) é acessado (Detalhes na Seção 4.5.2). Neste caso, o valor de similaridade é recuperado (a partir do IS) ou calculado por meio de uma função de similaridade (*FunçãoSimilaridade*). Por fim, todos os valores de similaridade são inseridos em uma lista (*ListaValorSimilaridade*) (linha 16).

Para cada bloco de tuplas, uma matriz de valores de similaridade entre CL é criada. Estes valores de similaridade são calculados de acordo com o algoritmo de clusterização escolhido (*AlgoritmoClusterização*). Esta matriz é quadrada e sua dimensão é igual ao número de CL (linha 20). Os valores de similaridade inseridos na matriz são calculados por meio do procedimento *ClusterizaçãoLocal* (linha 21). A *ClusterizaçãoLocal* identifica tuplas que representam a mesma entidade do mundo real e as inserem no mesmo CL (Detalhes na Seção 4.6.1). Por fim, o procedimento *AtualizaçãoClustersGlobais* (linha 22) atualiza os CG (Detalhes na Seção 4.6.2).

Algoritmo 1 – Detalhamento do processo QUIPER

```
1.  QUIPER (T, FunçãoBlocagem, FunçãoSimilaridade, AlgoritmoClusterização,
    Limiar)
2.  Entrada: T: Conjunto de tuplas que pertencem a um resultado de consulta
3.               FunçãoBlocagem: Função que calcula a chave de blocagem
4.               FunçãoSimilaridade: Função que calcula o valor de similaridade entre um
                 par de tuplas
5.               AlgoritmoClusterização: Algoritmo de clusterização local que calcula o
                 valor de similaridade entre um par de  Clusters Locais (por exemplo, o
                 algoritmo Single-Link ou Average-Link)
6.               Limiar: Limiar para o algoritmo de clusterização
7.  Saída: ListaCL: Lista de Clusters Locais agrupados por blocos
8.  Início
9.  Var: ListaBlocoTuplas: Lista de tuplas que pertencem a T agrupadas por bloco
10.      ListaValorSimilaridade: Lista de valores de similaridade entre pares de tuplas
         agrupadas por blocos
11.      NúmeroBlocos: Número de blocos de tuplas gerados
12.      NúmeroCL: Número de Clusters Locais por bloco de tuplas
13.      MatrizSim: Matriz de valores de similaridade entre Clusters Locais

14. ListaBlocoTuplas ← GeraçãoChaveBlocagem (T, FunçãoBlocagem)/*gera uma
    chave de blocagem para cada tupla e retorna as tuplas agrupadas por  bloco*/
15. ListaCL ←  AcessaIC (ListaBlocoTuplas)
16. ListaValorSimilaridade ← AcessaIS (ListaCL, FunçãoSimilaridade)/*acessa o IS e
    os valores de similaridade não encontrados são calculados por meio da
    FunçãoSimilaridade*/
17. NúmeroBlocos ←  tamanho (ListaCL) /*retorna o número de blocos gerados*/
18. Para i←1 até NúmeroBlocos Faça /*Faz as tarefas de Clusterização Local e
    Atualização dos Clusters Globais por bloco de tuplas*/
19. NúmeroCL ← tamanho(ListaCL[i]) /* retorna o número de CL de um bloco*/
20. initializaMatrizQuadrada (MatrizSim, NúmeroCL) /*inicializa a matriz quadrada
    com dimensão igual ao número de CL*/
21. ListaCL←ClusterizaçãoLocal (MatrizSim,  ListaValorSimilaridade[i], ListaCL [i],
    AlgoritmoClusterização, Limiar) /*identifica tuplas que representam a mesma
    entidade do mundo real e as insere no mesmo CL */
22. AtualizaçãoClustersGlobais (ListaCL [i]) /* faz o merge e atualiza os Clusters
    Globais */
23. FimPara
24. Retorna ListaCL
25. Fim
```

4.5 Indexação Dinâmica

Esta seção detalha a etapa de *Indexação Dinâmica* do processo QUIPER. Para subsidiar o processo QUIPER é necessário que a etapa de indexação seja dinâmica, permitindo que

um subconjunto de tuplas seja indexado e *clusterizado* a cada vez (a cada novo resultado de consulta), e não todo o conjunto de tuplas em um único momento. Considerando que a RE é realizada em tempo de consulta, é necessário que o tempo de processamento seja viável (menor que o tempo de processamento da RE tradicional sem reúso). Desta forma, uma solução para garantir esta viabilidade consiste no reaproveitamento de resultados de processamentos realizados previamente, incluindo valores de similaridade e mapeamentos entre pares de tuplas duplicadas detectados em etapas prévias de *clusterização*. Além disso, estes valores devem ser indexados para que o acesso seja facilitado. Sendo assim, este trabalho propõe dois índices dinâmicos: Índice de *Clusters* (IC) e Índice de Similaridades (IS).

4.5.1 Índice de Clusters

Durante a tarefa de *Análise dos Índices* na etapa de *Indexação Dinâmica*, uma busca é realizada no Índice de *Clusters* (IC) a fim de recuperar os identificadores das tuplas processadas em iterações anteriores do QUIPER. O IC indexa uma lista de *Clusters* Globais que contém identificadores de tuplas e identificadores de fontes de dados às quais estas tuplas pertencem (Definidos na Seção 4.3). O Algoritmo 2 apresenta o acesso do QUIPER ao IC.

Para cada tupla t_k pertencente a uma lista de blocos de tuplas (*ListaBlocoTuplas*), sua chave de blocagem é recuperada (*BlockingKey*) (linha 10). Se existir uma entrada correspondente no IC para a chave de acesso (similar a chave de blocagem), para o identificador da fonte (s_k.*Id*) ao qual t_k pertence e para o identificador da tupla (t_k.*Id*), isto é, a tupla t_k foi comparada em iterações anteriores do QUIPER e está indexada no IC, então o identificador do *Cluster* Global (*ClusterId*) correspondente é recuperado (linhas 12-13). As tuplas que pertencem ao mesmo *Cluster* Global são inseridas no *Cluster* Local correspondente com o mesmo *ClusterId*. Nos casos em que a tupla não foi comparada em iterações anteriores, um novo *Cluster* Local com um novo *ClusterId* (*NovoClusterLocalId*) é criado (linha 16). Os *Clusters* Locais criados e inicializados são inseridos em uma lista (*ListaCL*) (linhas 14 e 17) que é a saída do Algoritmo 2.

A Figura 12 mostra o resultado do algoritmo *AcessaIC* sobre os blocos pertencentes a q_1 do exemplo motivacional, apresentado na Figura 8. Assuma que os *Clusters* Globais estão no IC apresentado na Figura 9 e que os *Clusters* Locais foram inicializados de acordo

com estes *Clusters* Globais. Na Figura 9 estão destacadas as tuplas que tiveram seus *Clusters* Globais recuperados. Na Figura 12 a coluna *ClustersId* apresenta o identificador dos *Clusters* Locais (CL) de cada tupla. Desta forma, a tupla '*110*' da fonte de dados '*1*' foi inserida no CL '*1*', as tuplas '*113*', '*115*' e '*134*' das fontes de dados '*2*', '*4*' e '*5*', respectivamente, foram inseridas no CL '*4*', as tuplas '*245*' e '*849*' das fontes de dados '*4*' e '*2*', respectivamente, foram inseridas no CL '*5*'. A tupla '*120*' da fonte de dados '*3*' não foi indexada no IC, por este motivo, um CL com um novo *ClusterId* foi criado e o identificador atribuído foi '*6*'.

Algoritmo 2 – Detalhamento do acesso ao IC

1. ***AcessaIC*** *(ListaBlocoTuplas)*
2. ***Entrada:*** *ListaBlocoTuplas: Lista de tuplas que pertencem a T agrupadas por bloco*
3. ***Saída:*** *ListaCL: Lista de Clusters Locais agrupados por blocos*
4. ***Início***
5. ***Var*** *NúmeroBlocos: Número de blocos*
6. *BlockingKey: A chave de blocagem do bloco*
7. *ClusterId: Identificador do Cluster Global*

8. *NúmeroBlocos ← tamanho (ListaBlocoTuplas)*
9. ***Para*** *i←1* ***até*** *NúmeroBlocos* ***Faça***
10. *BlockingKey ← getBlockingKey(ListaBlocoTuplas[i])*
11. *Para cada $t_k \in$ ListaBlocoTuplas[i] Faça /* insere cada tupla em um Cluster Local da ListaCL */*
12. *ClusterId ← getClusterGlobalId(t_k, BlockingKey) /*recupera o identificador do Cluster Global da tupla t_k no CI */*
13. ***Se****(ClusterId !=0)* ***Então***
14. *insereNoClusterLocal(CLLista, t_k, ClusterId, BlockingKey) /*insere a tupla t_k no Cluster Local com identificador ClusterId*/*
15. ***Senão***
16. *ClusterId ← novoClusterLocalId()*
17. *insereNoClusterLocal(ListaCL, t_k, ClusterId, BlockingKey)*
18. ***FimPara***
19. ***FimPara***
20. ***Retorna*** *ListaCL*
21. ***Fim***

Figura 12 – Inicialização dos *Clusters* Locais agrupados por chave de blocagem (*BlockingKey)*

Autor					
BlockingKey	**s_k.Id**	**t_k.Id**	**Nome**	**Filiação**	***ClusterId***
tn	1	110	Tony Presley	University of Manchester	1
tn	2	113	Toni	Stanford University	4
tn	3	120	Tonia Presley	University of Manchester	6
tn	4	115	Tony	Stanford University	4
tn	5	134	Tonia	Stanford University	4
als	4	245	Alicea	Stanford University	5
als	2	849	Alice	Stanford University	5

Fonte: Próprio autor, 2017.

Para a definição desta estratégia de indexação foram analisadas técnicas de indexação dinâmicas encontradas na literatura. Dentre elas, destacam-se as estratégias utilizadas em cenários de Recuperação da Informação, onde atributos/valores são indexados [Ramadan & Christen, 2013]. Contudo, em cenários de integração de dados com grandes volumes de dados, e que se pode fazer uso de mais de um atributo para comparação entre pares de tuplas, a materialização e indexação de um grande número de valores de atributos distintos de cada tupla pode tornar-se oneroso, se considerado o tamanho do espaço de memória a ser utilizado. Desta forma, foi considerada a indexação dos próprios *clusters* e seus acessos no processo de RE.

4.5.2 Índice de Similaridades

Por se tratar de um processo de RE, para identificar se duas tuplas representam a mesma entidade do mundo real, i.e., pertencem ao mesmo *cluster*, é necessário realizar uma comparação entre as duas tuplas. Para este propósito, funções de similaridade são utilizadas.

O Índice de Similaridades (IS) indexa dinamicamente valores de similaridade entre pares de tuplas. A cada novo resultado de consulta recebido, valores de similaridades podem ser recuperados do IS ou calculados e inseridos no IS. Desta forma, objetiva-se reduzir o custo para calcular valores de similaridade em tempo de consulta, podendo reduzir significativamente o tempo necessário para o processo de RE como um todo. O IS é definido conforme a Definição 8 da Seção 4.3.

O Algoritmo 3 detalha o acesso ao IS. Neste algoritmo foi considerada a comparação entre pares de tuplas de *Clusters* Locais (CL) distintos. Contudo, caso necessário, é possível adaptá-lo para considerar valores de similaridade entre pares de tuplas de um mesmo CL. Esta necessidade depende do algoritmo de clusterização a ser utilizado na etapa de *Clusterização* do QUIPER.

Para cada bloco de tuplas (linhas 14-15) (tuplas com mesma chave de blocagem agrupadas por *Clusters* Locais), o número de CL é calculado (linha 17). As tuplas de cada par de CL pertencentes ao mesmo bloco (*BlockingKey*) são inseridas em uma lista (*ListaCLTuplasMerge*) (linhas 21-22). Para cada par de tuplas destes CL, o IS é acessado (linha 28) para verificar se o valor de similaridade entre o par de tuplas foi previamente calculado. Se existir uma entrada correspondente no IS, i.e., o valor de similaridade entre as tuplas *TuplaCL_1* e *TuplaCL_2* está indexado no IS, então o valor de similaridade é recuperado e inserido na lista de valores de similaridades (*ListaValorSimilaridade*) (linhas 29 e 30). Se o valor de similaridade entre o par de tuplas não foi calculado em iterações anteriores do QUIPER, então os valores dos atributos das tuplas são comparados com o objetivo de obter o valor de similaridade entre o par de tuplas (linhas 32-33).

Para simplificar, inserimos o procedimento *ComparaçãoentreParesdeTuplas* no algoritmo *AcessaIS*. Quando um novo valor de similaridade é calculado, o IS é atualizado (linha 34). A saída do Algoritmo 3 é uma lista dos valores de similaridade entre pares de tuplas (*ListaValorSimilaridade*).

A Figura 13 mostra o resultado do algoritmo *AcessaIS* sobre o exemplo motivacional das tuplas pertencentes ao bloco '*tn*' (Figura 8). Assuma que o Índice de Similaridades é o da Figura 10. Na Figura 10 e na Figura 13 foram destacados os valores de similaridades recuperados do IS. Por exemplo, o valor de similaridade entre a tupla *'110'* da fonte de dados *'1' e a tupla* '*113*' da fonte de dados '*2*' é '*0.25*'. Os valores de similaridade não encontrados no IS (por exemplo, o valor de similaridade entre a tupla '*110*' da fonte de dados '*1*' e a tupla '*120*' da fonte dados '*3*'), foram calculados por meio da função de similaridade de *Levenshtein* [Christen, 2012] sobre o atributo '*Nome'* da entidade '*Autor'*.

Algoritmo 3 - Detalhamento do acesso ao IS

```
1.  AcessaIS (ListaCL, FunçãoSimilaridade)
2.  Entrada: ListaCL: Lista de Clusters Locais agrupados por blocos
3.               FunçãoSimilaridade: Calcula o valor de similaridade entre tuplas
4.  Saída: ListaValorSimilaridade: Lista de valores de similaridade tuplas
5.  Início
6.  Var NúmeroBlocos: Número de blocos
7.        BlockingKey: A chave de blocagem do bloco de tuplas
8.        NúmeroCL: Número de Clusters Locais em um bloco
9.        ListaCLBloco: Lista de Clusters Locais com tuplas de um mesmo bloco de
          tuplas
10.     ListaCLTuplasMerge: Lista de tuplas do merge de um par de CL
11.     NúmeroListaCLTuplasMerge: Número de tuplas do merge de dois CL
12.    TuplaCL_1, TuplaCL_2: Tuplas de um CL específico
13.    ValorSim: Valor de similaridade entre um par de tuplas

14. NúmeroBlocos ← tamanho (ListaCL)
15.Para m←1 até NúmeroBlocos Faça
16.BlockingKey ← getBlockingKey(ListaCL[m])
17.NúmeroCL ← tamanho(ListaCL[m]) /* retorna o número de CL em um bloco */
18.insereClusterBloco(ListaCLBloco, ListaCL[m])
19.Para i ← 1 até NúmeroCL Faça /* recupera tuplas de um par de CL*/
20.Para j ← i+1 até NúmeroCL Faça /*insere as tuplas de um CL na lista*/
21.insereTuplasCL(ListaCLTuplasMerge, ListaCLBloco [i])
22.insereTuplasCL(ListaCLTuplasMerge,ListaCLBloco[j])
23.NúmeroListaCLTuplasMerge ← tamanho (ListaCLTuplasMerge)
24.Para k← 1 até NúmeroListaCLTuplasMerge Faça /*acessa tuplas de CL*/
25.TuplaCL_1 ← ListaCLTuplasMerge[k] /* acessa uma tupla da lista*/
26.Para l ← k+1 até NúmeroListaCLTuplasMerge Faça
27.TuplaCL_2← ListaCLTuplasMerge [l]
28.ValorSim←getValorSim(TuplaCL_1,TuplaCL_2,BlockingKey)
29.Se ValorSim!=null Então /* valor de similaridade existe no IS*/
30. insereValorSim(ListaValorSimilaridade,TuplaCL_1,TuplaCL_2,ValorSim,
        BlockingKey)
31.Senão
32.ValorSim←ComparaçãoEntreParesDeTuplas(TuplaCL_1,TuplaCL_2,
     FunçãoSimilaridade)
33.insereValorSim(ListaValorSimilaridade,TuplaCL_1,TuplaCL_2,ValorSim,
        BlockingKey)
34.insereValorSimIS (TuplaCL_1, TuplaCL_2, ValorSim, BlockingKey)
35.FimPara
36.FimPara
37.FimPara
38.FimPara
39.FimPara
40.Retorna ListaValorSimilaridade
41.FimInício
```

Figura 13 – Saída do algoritmo *AcessaIS com a etapa de Comparação entre Pares de Tuplas*

(1, 110), (2, 113), 0.25*
(1, 110), (3, 120), 0.84
(1, 110), (4, 115), 0.33
(1, 110), (5, 134), 0.24
(2, 113), (3, 120), 0.3
(2, 113), (4, 115), 0.75
(2, 113), (5, 134), 0.8
(3, 120), (4, 115), 0.3
(3, 120), (5, 134), 0.38
(4, 115), (5, 134), 0.75

* Valor de similaridade entre a tupla *'110'* da fonte de dados *'1'* e a tupla *'113'* da fonte de dados '*2'*

Fonte: Próprio autor, 2017.

Neste exemplo foram calculados ou recuperados todos os valores de similaridade entre pares de tuplas. Contudo, a depender do algoritmo de clusterização a ser utilizado na tarefa de *Clusterização* Local, não é necessário calcular os valores de similaridade entre pares de tuplas que já estão em um mesmo *Cluster* Local (por exemplo, o valor de similaridade entre a tupla '*113*' da fonte de dados '*2*' e a tupla '*115*' da fonte de dados '*4*' (Figura 12)). Neste caso, no Algoritmo 3, deve ser inserida uma condição para evitar que valores de similaridade sejam calculado ou recuperados de forma desnecessária.

4.6 Clusterização

Para completar o processo QUIPER, foi definida a etapa de *Clusterização*. Esta é subdividida nas tarefas de *Clusterização Local* e *Atualização dos Clusters Globais* detalhadas a seguir.

4.6.1 Clusterização Local

Após as etapas de *Indexação Dinâmica* e *Comparação entre Pares de Tuplas*, a etapa de *Clusterização* recebe como entrada (Figura 11): (i) Tuplas comparadas em iterações anteriores (passo *f*) com a identificação dos *Clusters* Locais e (ii) Novos Valores de Similaridade (passo *e*).

O principal objetivo da tarefa de *Clusterização Local*, da etapa de *Clusterização* é identificar as tuplas de um resultado de consulta que representam a mesma entidade do mundo real. Para reduzir os custos desta etapa, os *Clusters* Locais são inicializados considerando resultados de iterações passadas do QUIPER (*Indexação Dinâmica*).

O Algoritmo 4 detalha a tarefa de *Clusterização Local*. Os valores da matriz de similaridade entre pares de *Clusters* Locais (*MatrizSim*) são calculados por meio de *AlgoritmoClusterização*. Estes valores de similaridade são calculados com base no valor de similaridade entre os pares de tuplas pertencentes a cada *Cluster* Local.

O algoritmo inicializa *MatrizSim*. Esta matriz é simétrica, i.e., o valor de similaridade entre um par de *Clusters* Locais (c_1, c_2) é o mesmo que (c_2, c_1). Neste caso, são calculados apenas os valores dos elementos acima da diagonal principal (linhas 16-25). Para não realizar comparações desnecessárias, por convenção, os valores de similaridade da diagonal principal foram assumidos como sendo '*0*' (linhas 18-19). Os demais valores de *MatrizSim* são inseridos pelo procedimento *insereValorSimilaridadeCL* (linha 23).

Depois do preenchimento de *MatrizSim*, o procedimento *mergeCL* identifica qual par de *Clusters* Locais representa a mesma entidade do mundo real. Se *AlgoritmoClusterização* encontrar um par de *Clusters* Locais com valor de similaridade maior que o *Limiar* (linha 28), *MatrizSim* é atualizada com o valor de similaridade entre o novo *Cluster* Local (oriundo do *merge* de um par de *Clusters* Locais) e os demais *Clusters* Locais (linha 32). O algoritmo finaliza se nenhum par de *Clusters* Locais tiver o valor de similaridade maior que o limiar (linhas 33-34) ou se todas as tuplas de um mesmo bloco já pertencerem ao mesmo *Cluster* Local (linha 27). Em ambos os casos, o procedimento *mergeCL* retorna *null* (linha 29). Ao término do algoritmo, a lista de *Clusters* Locais atualizada é retornada (linha 36).

No Algoritmo 4, *AlgoritmoClusterização* corresponde a um algoritmo existente da literatura. Para avaliar a tarefa de *Clusterização Local*, este trabalho fez uso de 3 (três) diferentes algoritmos: *Single-Link*, *Average-Link* [Kogan et al., 2006] e *Hill-Climbing* [Guo et al., 2010]. Estes algoritmos foram escolhidos por já terem sido utilizados anteriormente em processos de RE e serem avaliados como bons algoritmos em cenários com grandes volumes de dados [Tan et al., 2006; Bhattacharya & Getoor, 2006; Gruenheid et al., 2014].

Algoritmo 4 - Detalhamento da etapa de *Clusterização Local*

```
1.  ClusterizaçãoLocal (MatrizSim, ListaValorSimilaridade, ListaCL,
    AlgoritmoClusterização, Limiar)
2.  Entrada: MatrizSim: Matriz de valores de similaridade entre Clusters Locais
3.               ListaValorSimilaridade: Lista de valores de similaridade entre tuplas
4.               ListaCL: Lista de Clusters Locais agrupados por bloco
5.               AlgoritmoClusterização: Algoritmo de clusterização local que calcula o
                 valor de similaridade entre um par de Clusters Locais
6.               Limiar: Limiar para o algoritmo de clusterização
7.  Saída: ListaCL: Lista atualizada de Clusters Locais agrupados por blocos
8.  Início
9.  Var Linha: Número de linhas de MatrizSim
10.      Coluna: Número de colunas de MatrizSim
11.      Controle: Indica se todas as tuplas de um mesmo bloco estão em um único CL
12.      ListaCLResultado: Lista de Clusters Locais que devem ser submetidos ao merge
13.      CL_1, CL_2: CL para serem submetidos ao cálculo do valor de similaridade
14.      ClusterLocal_1, ClusterLocal_2: CL a serem submetidos ao merge

15. Linha, Coluna← tamanho (ListaCL) /*retorna o tamanho da lista*/
16. Para i ← 1 até linha Faça /*insere o valor de similaridade entre um par de CL em
    MatrizSim */
17. Para j ← i até Coluna Faça
18. Se i==j Então
19. MatrizSim [i][j] ← 0
20. Senão
21. CL_1← ListaCL [i] /*retorna o Cluster Local da posição "i"da lista */
22. CL_2 ← ListaCL [j]
23. MatrizSim [i][j] ← insereValorSimilaridadeCL (CL_1, CL_2,
    ListaValorSimilaridade, AlgoritmoClusterização) /* o valor de similaridade é
    calculado por meio do AlgoritmoClusterização utilizando ListaValorSimilaridade*/
24. FimPara
25. FimPara
26. Controle ← verdadeiro
27. Enquanto(Controle==verdadeiro)
28. CLResultadoLista← mergeCL (MatrizSim, Limiar) /* seleciona o par de Clusters
    Locais a serem submetidos ao merge */
29. Se CLResultadoLista != null Então
30. ClusterLocal_1 ← CLResultadoLista [1] /*retorna um CL da lista */
31. ClusterLocal_2 ← CLResultLista [2]
32. AtualizaMatrizSim (MatrizSim, ListaCL, ClusterLocal_1, ClusterLocal_2) /* merge
    o par de Clusters Locais e atualiza o valor de similaridade entre o novo Cluster
    Local e os demais */
33. Senão
34. Controle← falso
35. FimEnquanto
36. Retorna ListaCL
37. Fim
```

A estratégia adotada para o algoritmo *Single-Link*, por exemplo, combina o par de *Clusters* Locais no qual um de seus pares de tuplas tem maior valor de similaridade e este é maior que um limiar. Por outro lado, para o algoritmo *Average-Link*, dois *Clusters* Locais são submetidos ao merge se a média dos valores de similaridade entre seus pares de tuplas for a maior de todas e maior que um limiar. O algoritmo *Hill-Climbing* maximiza uma função objetivo [Guo et *al*., 2010] (por exemplo, coesão entre *Clusters* Locais) em busca sempre de aumentar seu valor e aplica o merge entre dois *Clusters* Locais sempre que o valor da função objetivo é aumentado.

Desta forma, assuma que os algoritmos *Single-Link* e *Average-Link* foram utilizados para calcular o valor de similaridade entre os pares de *Clusters* Locais do resultado de q_1 do exemplo motivacional (Figura 12). Considere que os valores de similaridade entre os pares de tuplas foram apresentados na Figura 13.

Figura 14 - (a) Matriz inicial para o QUIPER; (b) Matriz inicial para o processo tradicional de RE

Clusters Locais	(1, {(1, 110)})	(4, {(2,113), (4,115), (5, 134)})	(6, {(3, 120)})
(1, {(1, 110)})	0	0.33/0.27	0.84/0.84
(4, {(2,113), (4,115), (5, 134)})	-	0	0.38/0.33
(6, {(3, 120)})	-	-	0

(a)

Clusters Locais	(7, {(1, 110)})	(8, {(2, 113)})	(9, {(3, 120)})	(10, {(4, 115)})	(11, {(5, 134)})
(7, {(1, 110)})	0	0.25	0.84	0.33	0.24
(8, {(2, 113)})	-	0	0.3	0.75	0.8
(9, {(3, 120)})	-	-	0	0.3	0.38
(10, {(4, 115)})	-	-	-	0	0.75
(11, {(5, 134)})	-	-	-	-	0

(b)

Fonte: Próprio autor, 2017.

A Figura 14(a) apresenta os valores de similaridade entre pares de *Clusters* Locais do bloco '*tn*' utilizando o QUIPER e a Figura 14(b) utilizando um processo tradicional de RE. Na Figura 14(a) os valores de similaridade apresentados do lado esquerdo são provenientes do uso do algoritmos *Single-Link* e os do lado direito provenientes do algoritmo *Average-Link* (por exemplo, 0.33/0.27). Todas as tuplas pertencentes ao mesmo *Cluster* Local representam o mesmo '*Autor*' (por exemplo, *(4, {(2,113), (4,115), (5,*

134)}), o *Cluster* Local '*4*' tem as tuplas '*113*', '*115*' e '*134*' das fontes de dados '*2*', '*4*' e '*5*', respectivamente).

Na Figura 14(a), o valor de similaridade entre o cluster '*1*' e o *cluster* '*4*' utilizando o algoritmo *Single-Link*, é o maior valor de similaridade entre os pares de tuplas (*0.25*, *0.33* e *0.24*) (Figura 13). Para o algoritmo *Average-Link*, utiliza-se a média dos valores de similaridade entre os pares de tuplas (*0.25*, *0.33* e *0.24*) que neste caso é *0.27*. Observe que alguns valores de similaridade entre pares de tuplas apresentados na Figura 13 foram recuperados do Índice de Similaridades e outros foram calculados em tempo de consulta (Seção 4.5.2).

Assuma que o limiar utilizado para os algoritmos de clusterização foi 0.6. Desta forma, os *Clusters* Locais '*1*' e '*6*' seriam os eleitos a serem submetidos ao merge. Desta forma, a tupla '*120*' da fonte de dados '*3*' seria inserida no *Cluster* Local '*1*' ou a tupla '*110*' da fonte de dados '*1*' seria inserida no *Cluster* Local '*6*'. Neste exemplo, arbitrariamente convencionamos que a tupla '*120*' da fonte de dados '*3*' seria inserida no *Cluster* Local '*1*'.

A Figura 14(b) apresenta a matriz com os resultados provenientes do processo tradicional de RE. Observe que neste caso, cada *Cluster* Local é unitário (i.e., tem apenas uma tupla) e todos os valores de similaridade entre pares de tuplas devem ser calculados (Figura 13). Como apresentado na Figura 14(a), o QUIPER reduz as dimensões da matriz e, consequentemente, o número de comparações entre pares de tuplas.

4.6.2 Atualização dos Clusters Globais

Na tarefa de *Atualização dos Clusters Globais*, os *Clusters* Globais são atualizados para representar os *Clusters* Locais identificados na execução do QUIPER durante o processamento de um resultado de consulta específico. Intuitivamente, quando tuplas são inseridas em um mesmo *Cluster* Local, é necessário incluí-las no *Cluster* Global correspondente (*Cluster* Global que representa a mesma entidade do mundo real). Um par de *Clusters* Globais deve ser combinado (submetido a uma operação de *merge*) se pelo menos um de seus pares de tuplas for inserido em um mesmo *Cluster* Local.

O Algoritmo 5 apresenta os detalhes da tarefa de *Atualização dos Clusters Globais*. A entrada do algoritmo é a lista de *Clusters* Locais com tuplas de um mesmo bloco de tuplas (Definição 4). O identificador do *Cluster* Local (*LocalClusterId*) (linhas 12-13) e sua lista

de tuplas são recuperados a partir de *ListaCL*. Cada tupla de um *Cluster* Local (CL) é buscada no Índice de *Clusters* (IC) que indexa os *Clusters* Globais (linhas 15-22). Se a tupla de um CL não pertence a nenhum *Cluster* Global, ela é inserida em um *Cluster* Global (existente ou novo) com o identificador idêntico ao do CL no qual ela foi inserida (linha 18). Se a tupla de um CL já pertence a um *Cluster* Global, o identificador do *Cluster* Global é atualizado com o valor do identificador do *Cluster* Local (*LocalClusterId*) (linha 21). Neste caso, todas as tuplas que pertencem ao *Cluster* Global com o identificador anterior, são movidas para o *Cluster* Global com identificador *LocalClusterId*.

Algoritmo 5 - Detalhamento da etapa de *Atualização dos Clusters Globais*

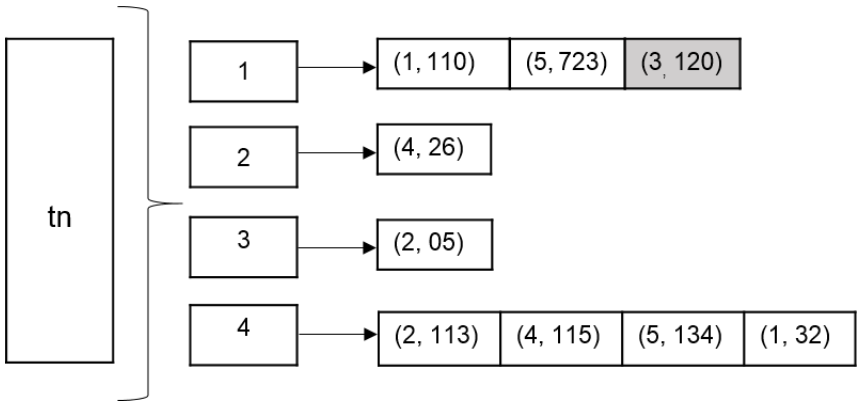
```
AtualizaçãoClusterGlobal (ListaCL)
2.  Entrada: ListaCL: Lista de Clusters Locais agrupados por bloco
3.  Início
4.  Var NúmeroCL: Número de Cluster Locais
5.       ListaTuplasCL: Lista de tuplas de um Cluster Local
6.       NúmeroTuplasCL: Número de tuplas de um Cluster Local
7.      LocalClusterId: Identificador do Cluster Local
8.     GlobalClusterId: Identificador do Cluster Global
9.     ClusterGlobal: Um Cluster Global

10. NúmeroCL ← tamanho (ListaCL) /*retorna o tamamho da lista */
11. Para i ← 1 até  NúmeroCL Faça /*recupera as tuplas de um Cluster Local */
12. LocalClusterId ← getLocalClusterId (ListaCL[i])
13. insereTuplasLocais (ListaTuplasCL, ListaCL[i]) /* insere as tuplas de um Cluster
    Local na lista */
14. NúmeroTuplasCL ← tamanho (ListaTuplasCL)
15. Para j ← i até NúmeroTuplasCL Faça /* aplica o merge nos Clusters Globais que
      possuem tuplas em um mesmo Cluster Local */
16.  GlobalClusterId ← getGlobalClusterId(ListaTuplasCL[j].Id) /*recupera o
       ClusterId de uma tupla no Índice de Clusters */
17. Se GlobalClusterId == null Então /* se a tupla não está indexada */
18.  insereTuplaCG (LocalClusterId, ListaTuplasCL[j].Id ) /*insere a tupla no Cluster
    Global com identificador igual ao do Cluster Local*/
19. Senão Se GlobalClusterId != LocalClusterId Então
20. ClusterGlobal ← getCG (GlobalClusterId) /*recupera o Cluster Global com
     ClusterId igual a GlobalClusterId*/
21.  setClusterGlobalId (LocalClusterId) /* altera o ClusterId do Cluster Global para
      o valor de LocalClusterId*/
22. FimPara
23. FimPara
24. Fim
```

A Figura 15 apresenta os *Clusters* Globais com chave de acesso '*tn*' no Índice de *Clusters* (IC) atualizado. Este foi apresentado anteriormente (antes da tarefa *Atualização dos*

Clusters Globais ser aplicada) na Figura 9. A configuração do IC apresentada na Figura 15 reflete o *merge* entre o *Cluster* Local '*1*' e '*6*' apresentado na Seção 4.6.1. Neste caso o algoritmo *AtualizaçãoClusterGlobal* insere a tupla '*120*' da fonte de dados '*3*' no *Cluster* Global '*1*', sem realizar comparações entre a tupla '*120*' e as demais tuplas do *Cluster* Global '*1*' (Por exemplo, não realiza comparações entre a tupla '*120*' da fonte de dados '*3*' e a tupla '*723*' da fonte de dados '*5*'). Na Figura 15, foi destacada a nova tupla inserida no *Cluster* Global com *ClusterId* igual a '*1*'.

Figura 15 – Índice de *Clusters* atualizado

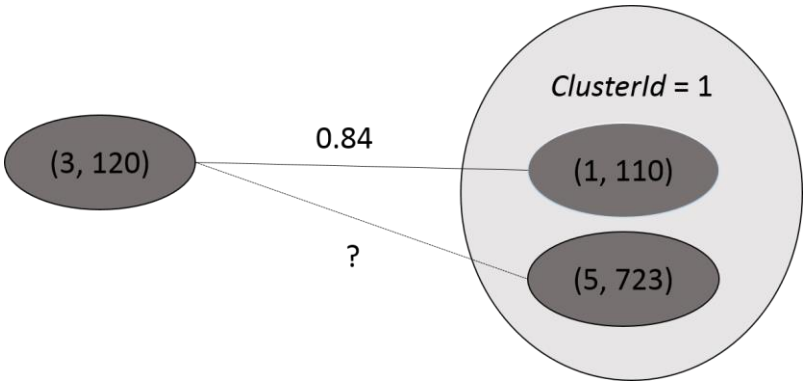


Fonte: Próprio autor, 2017.

Esta inserção de tuplas em um *Cluster* Global sem a necessidade de se comparar as novas tuplas com todas as demais tuplas pré-existentes no *Cluster* Global é baseada na propriedade de transitividade entre pares de tuplas duplicadas definida na literatura [Benjelloun et *al.*, 2009]. Desta forma, mesmo sem saber o valor de similaridade entre a tupla '*120*' da fonte de dados '*3*' e a tupla '*723*' da fonte de dados '*5*', se é de conhecimento que as tuplas '*120*' da fonte de dados '*3*' e '*723*' da fonte de dados '*5*' são duplicadas em relação a tupla '*110*' da fonte de dados '*1*', então, pode-se presumir que as tuplas '*120*' e '*723*' são duplicadas entre si. A Figura 16 ilustra esta transitividade entre pares de tuplas.

Os experimentos desenvolvidos durante a execução deste trabalho, mostraram que o uso da propriedade de transitividade permitiu que o processo QUIPER alcançasse resultados de Medida-F semelhantes aos dos processos tradicionais de RE.

Figura 16 - Transitividade entre pares de tuplas



Fonte: Próprio autor, 2017.

4.7 Exemplo para as Etapas do QUIPER

Nesta seção será apresentado um exemplo com uma sequência de 2 (dois) resultados de consultas (q_1 e q_2) sendo processados de acordo com o QUIPER. O resultado de q_1 representará um primeiro conjunto de tuplas sendo processado no QUIPER (sem ter nenhum valor a ser recuperado dos índices). O resultado de q_2 representará um segundo conjunto de tuplas, processado de acordo com as etapas do QUIPER, o qual parte de suas tuplas possui informações a serem recuperadas dos índices.

Este exemplo dará uma visão geral do processo QUIPER, facilitando o entendimento de como ocorre o reúso de iterações passadas para detectar pares de tuplas duplicadas em novos resultados de consultas. Para este fim, considere o mesmo cenário de fontes de dados bibliográficas do exemplo motivacional (Seção 4.2) e a seguinte parametrização assumida:

- As consultas q_1 e q_2 foram realizadas sobre o conceito *Autor*;
- O filtro para dividir as tuplas que provavelmente representam a mesma entidade do mundo real foi a blocagem. A chave de blocagem utilizada no filtro foi gerada a partir da função *Double-Metaphone* [Christen, 2012] sobre o atributo *Nome*;
- A chave de acesso aos índices também foi gerada pela função *Double-Metaphone* sobre o atributo *Nome*;
- A função de similaridade utilizada para comparar pares de tuplas foi *Levenshtein* [Christen, 2012] sobre os atributos *Nome* e *Filiação*;
- O algoritmo de clusterização para definir o *merge* entre um par de *Clusters* Locais foi o *Single-Link* com limiar de 0.5.

Suponha a consulta q_1: "*Recupere todos os autores que publicaram entre o ano de 2012 e 2013"*. A Figura 17 apresenta as tuplas pertencentes ao resultado da consulta q_1 (entrada do QUIPER). Na etapa de *Indexação Dinâmica*, na tarefa de *Geração dos Blocos de Tuplas* são geradas chaves de blocagem (*BlockingKey*) para cada tupla do resultado de q_1 (Figura 18). Tuplas com a mesma *BlockingKey* são inseridas no mesmo bloco. As *BlockingKey* geradas foram '*tn*' e '*als*'.

Figura 17 - Resultado de q_1

Autor			
s_k.Id	t_k.Id	Nome	Filiação
1	110	Tony Presley	University of Manchester
5	723	Tony Presle	University of Manchester
2	113	Toni	Stanford University
4	115	Tony	Stanford University
5	134	Tonia	Stanford University
4	245	Alicea	Stanford University
2	849	Alice	Stanford University

Fonte: Próprio autor, 2017.

Figura 18 – Blocos de tuplas do resultado de q_1

	Autor			
BlockingKey	s_k.Id	t_k.Id	Nome	Filiação
tn	1	110	Tony Presley	University of Manchester
tn	5	723	Tony Presle	University of Manchester
tn	2	113	Toni	Stanford University
tn	4	115	Tony	Stanford University
tn	5	134	Tonia	Stanford University
als	4	245	Alicea	Stanford University
als	2	849	Alice	Stanford University

Fonte: Próprio autor, 2017.

Em seguida, a tarefa de *Análise dos Índices* acessa o Índice de *Clusters* (IC) e o Índice de Similaridade (IS). Neste caso, como o resultado de q_1 é o primeiro conjunto de tuplas processados segundo o QUIPER, os índices estão vazios. Desta forma, todos os valores de similaridade, entre todos os pares de tuplas de um mesmo bloco de tuplas, devem ser calculados.

Na etapa de *Comparação entre Pares de Tuplas* os valores de similaridade entre os pares de tuplas que pertencem ao mesmo bloco são calculados e, posteriormente, armazenados no IS. A Figura 19 apresenta em destaque na cor cinza os valores de similaridade entre as tuplas do bloco de tuplas com chave '*tn*' e em branco os valores de similaridade entre as tuplas do bloco de tuplas com chave '*als*'.

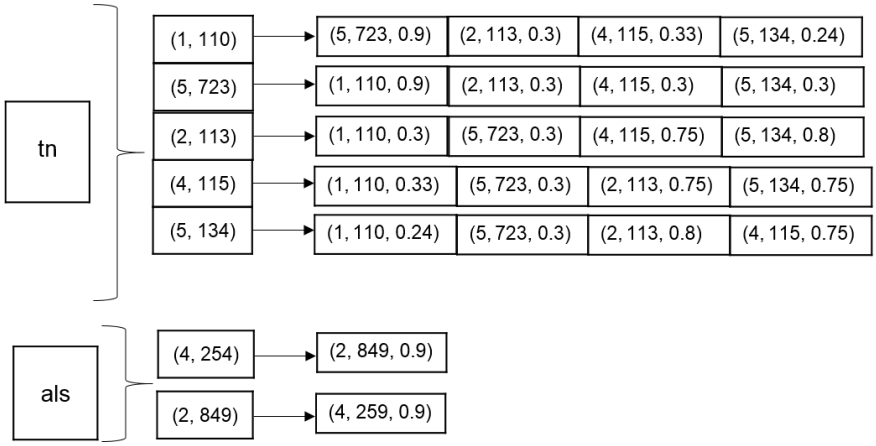
Figura 19 – Saída da etapa *Comparação entre Pares de Tuplas* para o resultado de q_1

(1, 110), (5, 723), 0.9
(1, 110), (2, 113), 0.3
(1, 110), (4, 115), 0.33
(1, 110), (5, 134), 0.24
(5, 723), (2, 113), 0.3
(5, 723), (4, 115), 0.3
(5, 723), (5, 134), 0.3
(2, 113), (4, 115), 0.75
(2, 113), (5, 134), 0.8
(4, 115), (5, 134), 0.75
(4, 245), (2, 849), 0.9

Fonte: Próprio autor, 2017.

A Figura 20 apresenta o IS com os novos valores de similaridade armazenados. Neste caso, tem-se uma lista de adjacência com os valores de similaridade entre pares de tuplas e as chaves de acesso ao índice são ''*tn*' e '*als*' (iguais as chaves de blocagem geradas para o bloco de tuplas).

Figura 20 – IS atualizado com a etapa de *Comparação entre Pares de Tuplas* para o resultado de q_1



Fonte: Próprio autor, 2017.

Na etapa de *Clusterização*, a tarefa de *Clusterização Local* inicia sem que nenhum *Cluster* Local tenha sido inicializado na etapa de *Indexação Dinâmica* (IC vazio). Desta forma, cada tupla é inserida em um novo *Cluster* Local. Estes são submetidos ao algoritmo de clusterização. A Figura 21 apresenta os *Clusters* Locais inicializados com uma única tupla.

Figura 21 – Inicialização dos *Clusters* Locais para o resultado de q_1

Autor

BlockingKey	s_k.Id	t_k.Id	Nome	Filiação	ClusterId
tn	1	110	Tony Presley	University of Manchester	1
tn	5	723	Tony Presle	University of Manchester	2
tn	2	113	Toni	Stanford University	3
tn	4	115	Tony	Stanford University	4
tn	5	134	Tonia	Stanford University	5
als	4	245	Alicea	Stanford University	6
als	2	849	Alice	Stanford University	7

Fonte: Próprio autor, 2017.

Após a inicialização dos *Clusters* Locais, o algoritmo de clusterização é aplicado considerando os valores de similaridade entre os pares de tuplas de um mesmo bloco de tuplas (Figuras 19 e 20). A Figura 22 apresenta o resultado do algoritmo *Single-Link* sobre os *Clusters* Locais apresentados na Figura 21. Neste caso, por exemplo, o algoritmo indicou que as tuplas '*110*' e '*723*'das fontes de dados '*1*' e '*5*', respectivamente, representam a mesma entidade do mundo real e devem ser inseridas o mesmo *Cluster* Local (i.e., *Cluster* Local com identificador '*1*').

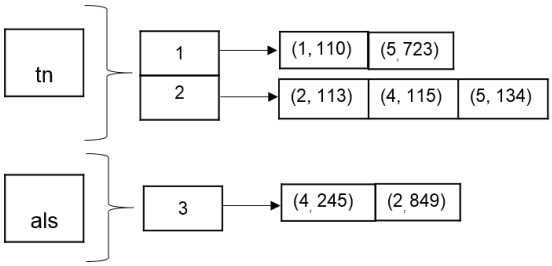
Por fim, a tarefa de *Atualização dos Clusters Globais* é aplicada. Neste caso, ainda não existem *Clusters* Globais e o IC está vazio. Portanto, os *Clusters* Locais são apenas copiados para o IC. A Figura 23 apresenta o IC após a execução da tarefa de atualização. As chaves de acesso ao IC são '*tn*' e '*als*', os identificadores dos *Clusters* Globais foram os mesmos dos *Clusters* Locais: '*1*', '*2*' e '*3*'.

Figura 22 – *Clusters* Locais gerados para o resultado de q_1

Autor					
BlockingKey	**s_k.Id**	**t_k.Id**	**Nome**	**Filiação**	**ClusterId**
tn	1	110	Tony Presley	University of Manchester	1
tn	5	723	Tony Presle	University of Manchester	1
tn	2	113	Toni	Stanford University	2
tn	4	115	Tony	Stanford University	2
tn	5	134	Tonia	Stanford University	2
als	4	245	Alicea	Stanford University	3
als	2	849	Alice	Stanford University	3

Fonte: Próprio autor, 2017.

Figura 23 – IC atualizado após a tarefa de *Atualização dos Clusters Globais* para o resultado de q_1



Fonte: Próprio autor, 2017.

Agora suponha a consulta q_2 (sequencial a consulta q_1): "*Recupere todos os autores que publicaram entre os anos de 2011 e 2012*". A Figura 24 apresenta as tuplas pertencentes ao resultado da consulta q_2. A etapa de *Indexação Dinâmica*, na tarefa de *Geração dos Blocos de Tuplas*, gerou as mesmas chaves de blocagem (Figura 25) apresentadas no resultado de q_1.

Figura 24 - Resultado de q_2

Autor			
s_k.Id	t_k.Id	Nome	Filiação
1	110	Tony Presley	University of Manchester
2	113	Toni	Stanford University
3	120	Tonia Presley	University of Manchester
4	115	Tony	Stanford University
5	134	Tonia	Stanford University
4	245	Alicea	Stanford University
2	849	Alice	Stanford University

Fonte: Próprio autor, 2017.

Em seguida, a tarefa de *Análise dos Índices* acessa o Índice de *Clusters* (IC) (Figura 23) e o Índice de Similaridade (IS) (Figura 20). Neste caso, algumas tuplas não precisarão ser comparadas entre si e alguns valores de similaridade serão recuperados do IS. A Figura 26 apresenta os *Clusters* Locais inicializados a partir dos *Clusters* Globais recuperados do IC (Figura 23). Observe que a tupla '*110*' da fonte de dados '*1*' pertence ao *Cluster* Global '*1*' (Figura 23) e foi inserida no *Cluster* Local '*1*' (Figura 26). A tupla '*120*' da fonte de dados '*3*' não está em nenhum *Cluster* Global e foi inserida em um novo *Cluster* Local '*4*' (Em destaque na Figura 26).

Figura 25 - Blocos de tuplas do resultado de q_2

Autor				
BlockingKey	**s_k.Id**	**t_k.Id**	**Nome**	**Filiação**
tn	1	110	Tony Presley	University of Manchester
tn	2	113	Toni	Stanford University
tn	3	120	Tonia Presley	University of Manchester
tn	4	115	Tony	Stanford University
tn	5	134	Tonia	Stanford University
als	4	245	Alicea	Stanford University
als	2	849	Alice	Stanford University

Fonte: Próprio autor, 2017.

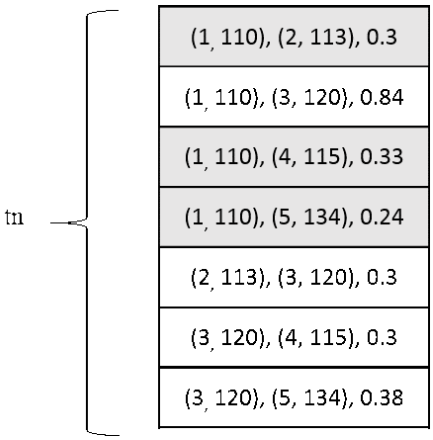
Após a inicialização dos *Clusters* Locais, os valores de similaridade, entre pares de tuplas com mesma chave de blocagem, são recuperados do IS (*acessaIS*) ou calculados em tempo de consulta (etapa de *Comparação entre Pares de Tuplas*). A Figura 27 apresenta em destaque cinza os valores de similaridade recuperados do IS (Figura 20) e em branco os valores de similaridade calculados em tempo de consulta.

Figura 26 - Inicialização dos *Clusters* Locais para o resultado de q_2

BlockingKey	**s_k.Id**	**t_k.Id**	**Name**	**Affiliation**	***ClusterId***
tn	1	110	Tony Presley	University of Manchester	1
tn	2	113	Toni	Stanford University	2
tn	3	120	Tonia Presley	University of Manchester	4
tn	4	115	Tony	Stanford University	2
tn	5	134	Tonia	Stanford University	2
als	4	245	Alicea	Stanford University	3
als	2	849	Alice	Stanford University	3

Fonte: Próprio autor, 2017.

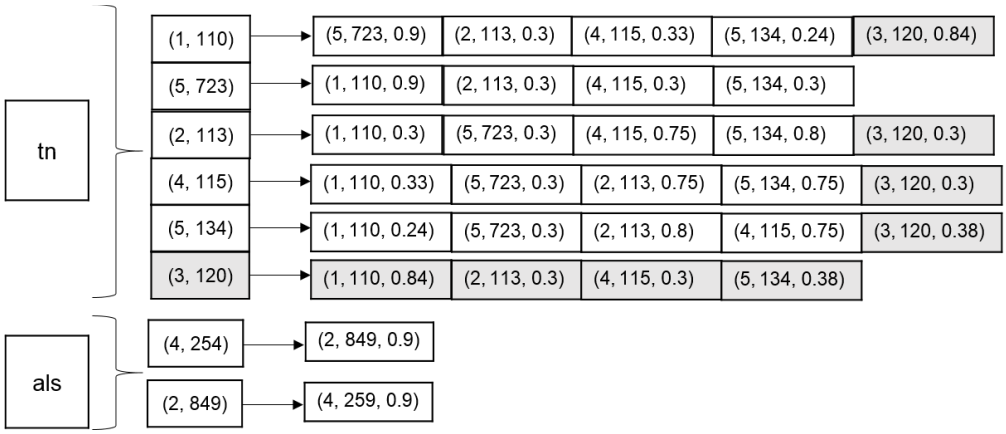
Figura 27 - Saída da etapa *Comparação entre Pares de Tuplas* para o resultado de q_2



Fonte: Próprio autor, 2017.

Observe que alguns valores de similaridade não precisaram ser calculados ou recuperados (por exemplo, o valor de similaridade entre a tupla '*113*' da fonte de dados '*2*' e a tupla '*120*' da fonte de dados '*3*'). Nestes casos, as tuplas já estão no mesmo *Cluster* Local (Figura 26) e, de acordo com o algoritmo de clusterização *Single-Link*, não serão mais comparadas. Adicionalmente, foram indicados os valores de similaridade referentes às tuplas com chave de blocagem '*tn*', pois nenhuma nova comparação para o bloco de tuplas com chave de blocagem '*als*' foi necessária.

Figura 28 - IS atualizado com a etapa de *Comparação entre Pares de Tuplas* para o resultado de q_2



Fonte: Próprio autor, 2017.

Com a etapa de *Comparação entre Pares de Tuplas*, novos valores de similaridade são inseridos no IS. A Figura 28 apresenta o IS atualizado após o processamento do resultado

de q_2. Observe a configuração do IS antes do processamento do resultado de q_2 na Figura 20. Na Figura 28 foram destacados os novos valores de similaridade inseridos.

Na etapa de *Clusterização*, a tarefa de *Clusterização Local* aplica o algoritmo de clusterização considerando os valores de similaridade entre os pares de tuplas de um mesmo bloco de tuplas (Figuras 27 e 28) e os *Clusters* Locais inicializados (Figura 26). A Figura 29 apresenta o resultado do algoritmo *Single-Link*. Neste caso, o algoritmo indicou que as tuplas '*120*' e '*110*'das fontes de dados '*1*' e '*3*', respectivamente, representam a mesma entidade do mundo real e devem ser inseridas no mesmo *Cluster* Local (i.e., *Cluster* Local com identificador '*1*'). Na Figura 29 foi destacada a tupla que mudou de *Cluster* Local.

Figura 29 - *Clusters* Locais gerados para o resultado de q_2

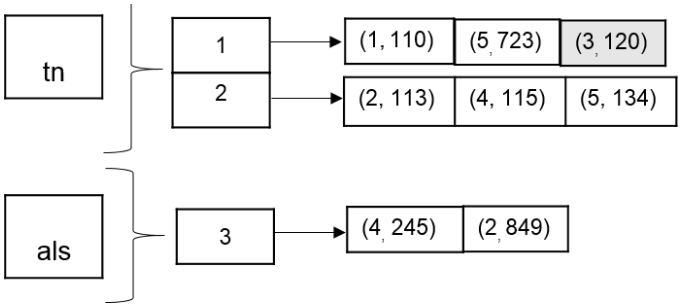
Author					
BlockingKey	**s_k.Id**	**t_k.Id**	**Name**	**Affiliation**	***ClusterId***
tn	1	110	Tony Presley	University of Manchester	1
tn	2	113	Toni	Stanford University	2
tn	3	120	Tonia Presley	University of Manchester	1
tn	4	115	Tony	Stanford University	2
tn	5	134	Tonia	Stanford University	2
als	4	245	Alicea	Stanford University	3
als	2	849	Alice	Stanford University	3

Fonte: Próprio autor, 2017.

Por fim, a tarefa de *Atualização dos Clusters Globais* é aplicada. Neste caso, apenas a tupla '*120*' da fonte de dados '*3*' necessitará ser inserida em um *Cluster* Global, pois as demais tuplas dos *Clusters* Locais já pertenciam a um *Cluster* Global correspondente e não foram movidas para nenhum outro *Cluster* Local. Portanto, a tupla movida foi inserida no mesmo *Cluster* Global da tupla '*110*' da fonte de dados '*1*' (*Cluster* Global '*1*') (Figura 23).

A Figura 30 apresenta o IC atualizado (em destaque a nova tupla inserida). Observe que a nova tupla foi inserida no *Cluster* Global sem ser comparada com a tupla '*723*' da fonte de dados '*5*', pois esta não se encontrava no resultado da consulta q_2. Contudo, anteriormente (no resultado de q_1), a tupla '*723*' foi comparada com a tupla '*110*', que é duplicada em relação a nova tupla '*120*'. Se, em um resultado de uma suposta consulta q_3, os pares de tuplas '*120*' e '*723*' surgissem, estas não seriam mais comparadas. Seriam inseridas no mesmo *Cluster* Local.

Figura 30 - IC atualizado após a tarefa de *Atualização dos Clusters Globais* para o resultado de q_2



Fonte: Próprio autor, 2017.

4.8 Considerações

Este capítulo apresentou a especificação da principal contribuição deste trabalho: o processo incremental e orientado à consulta para Resolução de Entidades, denominado por QUIPER (*Query-time and Incremental Process for Entity Resolution*). Inicialmente, foi apresentada uma visão geral do processo, assim como as definições gerais e as notações necessárias para o entendimento do processo. Em seguida, foi detalha a etapa de *Indexação Dinâmica*, apresentando os dois índices propostos neste trabalho: (i) Índice de *Clusters* e (ii) Índice de Similaridades.

Por fim, foi detalhada a etapa de Clusterização, destacando as tarefas de *Clusterização Local* e *Atualização dos Clusters Globais*. Para o entendimento geral do processo QUIPER, um exemplo foi desenvolvido à medida que os conceitos e detalhes das etapas do QUIPER eram apresentados. Por fim, foi apresentado um exemplo completo de uma sequência de resultados de consultas sendo processadas pelo QUIPER. O próximo capítulo apresenta os experimentos realizados para avaliação do QUIPER.

5 EXPERIMENTOS

5.1 Introdução

Este capítulo apresenta a implementação realizada para a execução dos experimentos que avaliam o processo QUIPER. Primeiro serão apresentados os artefatos utilizados para a implementação, em seguida serão apresentados os experimentos e as discussões sobre os resultados alcançados.

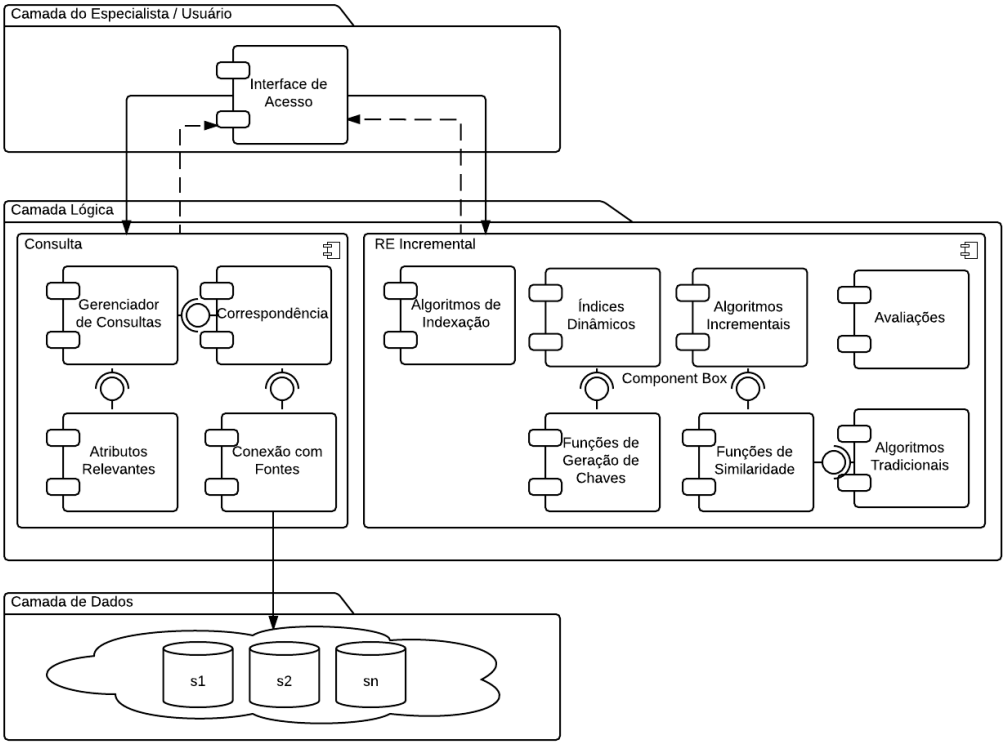
O conteúdo do capítulo é estruturado como segue: Seção 5.2 apresenta o protótipo desenvolvido e a infraestrutura utilizada; Seção 5.3 apresenta as configurações dos experimentos, especificando as fontes de dados e as métricas utilizadas; A Seção 5.4 apresenta metodologia dos experimentos com foco em eficiência e qualidade dos resultados do QUIPER. Por fim, a Seção 5.5 apresenta as discussões finais do capítulo.

5.2 Implementação

A arquitetura do protótipo desenvolvido é apresentada na Figura 31. O protótipo foi desenvolvido na linguagem de programação Java [JAVA, 2016]. Como destaque, foram incorporados ao protótipo alguns algoritmos de indexação, implementados no *framework* Dude [Draisbach & Naumann, 2010] e funções de similaridade em bibliotecas consolidadas, como a *SecondString* [Cohen et *al*., 2003]. O protótipo foi dividido em três camadas básicas: (i) Especialista, (ii) Lógica e (iii) Dados. As setas com linha contínua indicam solicitações e as setas com linha tracejada indicam o retorno das solicitações.

Na camada do especialista, o componente *Interface de Acesso* permite que uma consulta seja realizada pelo especialista. A camada Lógica do protótipo é subdividida em dois conjuntos de componentes: (i) *Consulta* e (ii) *RE Incremental*. Na camada de dados estão as fontes de dados a serem consultadas e submetidas ao processo de Resolução de Entidades.

Figura 31 - Arquitetura do protótipo QUIPER



Fonte: Próprio autor, 2017.

No conjunto de componentes *Consulta* tem-se: (i) O componente *Gerenciador de Consulta* executa a consulta em múltiplas fontes de dados. (ii) O componente *Correspondência* realiza a correspondência entre os esquemas de múltiplas fontes de dados, neste protótipo as correspondências são realizadas de forma manual. (iii) O componente *Atributos Relevantes* implementa a estratégia proposta por [Canalle, 2016; Canalle et *al.*, 2017] e permite selecionar os atributos relevantes de uma entidade. Por fim, o componente *Conexão com Fontes*, na versão atual do protótipo, permite fontes de dados em diferentes SGBDRs e fontes de dados em arquivos CSV.

No conjunto de componentes *RE Incremental* tem-se: (i) O componente *Algoritmos de Indexação* que disponibiliza diferentes algoritmos de indexação estática de dados (blocagem padrão, vizinhança ordenada, ...). (ii) O componente *Índices Dinâmicos* implementa os dois índices propostos neste trabalho (IC e IS). Este componente utiliza o componente *Funções de Geração de Chaves* para geração das chaves de acesso. (iii) O componente *Algoritmos Incrementais* implementa os algoritmos de clusterização dentro da perspectiva incremental do QUIPER (*Single-Link*, *Average-Link* e *Hill-Climbing*) e

para este propósito faz uso do componente *Funções de Similaridade*. (iv) O componente *Algoritmos Tradicionais* implementa os algoritmos de clusterização em uma perspectiva do processo tradicional de RE (lote). Isto permite a comparação dos resultados do processo QUIPER com o processo tradicional de RE. Por fim, o componente *Avaliações* implementa métricas que podem ser utilizadas para avaliação da qualidade dos resultados, permitindo que arquivos com o espelho das respostas esperadas (*gold standard*) sejam utilizados para avaliar os resultados alcançados. Relatórios de qualidade e eficiência do processo são gerados. Os componentes *Algoritmos Incrementais*, *Índices Dinâmicos* e *Algoritmos Tradicionais* tiveram implementação própria para validação deste trabalho, seguindo os algoritmos apresentados no Capítulo 4.

5.3 Definição dos Experimentos

Os experimentos foram realizados com duas fontes de dados reais e uma fonte de dados sintética. Para cada fonte de dados, um arquivo de *gold standard* apresenta os pares de tuplas duplicadas. Estes pares de tuplas duplicadas são a combinação de pares de tuplas que representam a mesma entidade do mundo real, sem a inclusão de pares repetidos. Por exemplo, o conjunto de tuplas $T= \{t_1, t_2, t_3, t_4\}$, tem 4 (quatro) tuplas, e o conjunto de pares de tuplas duplicadas pode ser $\{(t_1, t_2), (t_1, t_3), (t_1, t_4), (t_2, t_3), (t_3, t_4)\}$, o qual tem 5 (cinco) pares de tuplas duplicadas.

A primeira fonte de dados**,** CD[1], possui informações sobre artistas e os seus CDs, tem 9763 tuplas randomicamente extraídas do freeDB[2], com aproximadamente 300 pares de tuplas duplicadas. A fonte de dados CD possui os seguintes atributos {*Nome Artista*, *Título do Álbum*, *Categoria*, *Gênero*, *Ano*, *Faixa 1*, *Faixa 2*, ...*Faixa 14*}.

A segunda fonte de dados real, Cora[3], amplamente utilizada em trabalhos de RE, tem 1916 tuplas de publicações acadêmicas, com aproximadamente 66 mil pares de tuplas duplicadas. A fonte de dados Cora possui os seguintes atributos {*Autor*, *Título da Publicação*, *Nome Revista*, *Volume*, *Ano*, *Evento*, *País*, *Instituição*}.

[1] http://hpi.de/naumann/projects/repeatability/datasets/cd-datasets.html

[2] http://www.freedb.org/

[3] http://www.cs.umass.edu/~mccallum/code-data.html

A fonte de dados sintética, foi gerada com o gerador de dados Febrl[4]. Foram geradas 130 mil tuplas, com informações pessoais, com aproximadamente 40 mil pares de tuplas duplicadas. Na parametrização do gerador foi definido que para cada tupla duplicada existiram outras 10 tuplas duplicadas em relação a ela e que esta duplicação seria proveniente de 2 modificações em atributos distintos da fonte de dados. A fonte de dados Febrl possui os seguintes atributos {*Sexo, Idade, Data de Nascimento, Vocativo, Nome, Sobrenome, Rua, Bairro, Cep, Cidade, Estado, Telefone, Chave de Blocagem*}.

Estas três fontes de dados foram escolhidas por: (i) Serem amplamente utilizadas em experimentos de RE. (ii) Possuírem o espelho das respostas esperadas (*gold standard*), necessário para avaliação da qualidade do processo. (ii) Possuírem dados de domínios distintos, com volume de tuplas e total de pares de tuplas duplicadas variados.

Para os experimentos não se tinha um sistema real de integração de dados em funcionamento. Desta forma, foram avaliados *log*s de consultas disponíveis para experimentos[5]. Contudo, observou-se que apesar das consultas serem inter-relacionadas, o que facilitaria o reúso de tuplas previamente comparadas, não se conseguiu mapear as consultas disponíveis nos *logs* para as fontes de dados utilizadas para validação deste trabalho. Uma outra opção para se obter consultas seria a geração manual de consultas considerando as fontes de dados utilizadas nos experimentos. Contudo esta decisão poderia enviesar as análises a serem realizadas, sendo um risco para validação do QUIPER. Por esse motivo, para simular uma sequência de resultados de consultas sendo recebidos no processo QUIPER, amostras aleatórias de tuplas das fontes de dados foram extraídas. Os tamanhos destas amostras variaram de acordo com o propósito de cada experimento e estão explicitados nas próximas seções.

Com relação aos resultados de consultas processados, os experimentos também deviam verificar o impacto da ordem que os resultados de consultas eram processados. No canário no qual este trabalho está inserido, o desempenho do QUIPER tem uma dependência direta que os resultados de consulta possuam intersecção e a metodologia do experimento deveria garantir um cenário mais próximo do real possível. Desta forma, cada

[4] http://sourceforge.net/projects/febrl/.

[5] http://jeffhuang.com/search_query_logs.html

experimento foi repetido inúmeras vezes com o objetivo de calcular o caso médio (cenários bons e ruins para o QUIPER). Com relação a qualidade dos resultados obtidos, por exemplo, a Medida-F foi calculada para cada resultado de consulta e para cada fonte de dados após ser submetida a uma sequência de processos de RE incremental e RE tradicional. Para cada um dos casos, o experimento foi repetido 100 (cem) vezes e os resultados apresentados foram a média alcançada.

Para a criação das chaves de acesso aos índices dinâmicos (Índice de *Clusters* e Índice de Similaridades) foi aplicada a função *Double-Metaphone* [Christen, 2012] sobre o atributo: (i) "*Título do Álbum*" da fonte de dados CD, (ii) "*Título da Publicação*" da fonte de dados Cora. Para a fonte de dados Febrl, foi utilizado o atributo *"BlockingKey"* (chave de blocagem) gerado automaticamente.

As tuplas das fontes de dados CD e Cora foram agrupadas como possíveis duplicadas por meio do algoritmo de blocagem padrão [Christen, 2012], utilizando a mesma função de criação das chaves de acesso aos índices, *Double-Metaphone*. Para a escolha dos atributos para aplicação da função de blocagem, foram realizados experimentos para determinar a melhor combinação de atributos. Esta seleção de atributos foi comparada com a seleção identificada pelos experimentos de Papadakis et *al*. (2016) para as mesmas fontes de dados com o objetivo de avaliar se o mesmo grupo de atributos para as mesmas fontes de dados foi escolhido. Para a fonte de dados Febrl, todas as tuplas com o mesmo valor do atributo "*Chave de Blocagem"* (gerado automaticamente) foram inseridas no mesmo bloco.

É importante destacar que a mesma função de blocagem utilizada para o QUIPER foi utilizada no processo tradicional de RE. Isto permite uma comparação justa entre as duas abordagens, dado que diferentes funções de blocagem podem gerar diferentes blocos o que causa impacto direto no desempenho do processo de RE como um todo. Por ter sido utilizada a mesma função de blocagem para as duas abordagens, este trabalho não deu foco à avaliação da qualidade dos blocos gerados, dado que independente da qualidade dos blocos, as duas abordagens foram aplicadas sobre os mesmos blocos.

Os algoritmos de clusterização escolhidos foram: *Single-Link, Average-Link* [Kogan et al., 2006] e *Hill-Climbing* [Guo et al., 2010]. Estes algoritmos foram escolhidos por já terem sido utilizados anteriormente em processos de RE incrementais e serem avaliados

como bons algoritmos em cenários com grandes volumes de dados. É importante destacar que a escolha do algoritmo de clusterização tem uma dependência direta com os dados analisados e que estes algoritmos já foram anteriormente avaliados sobre as mesmas fontes de dados utilizadas para validação do QUIPER [Tan et al., 2006; Bhattacharya & Getoor, 2006; Gruenheid et al., 2014].

As funções de similaridade e os limiares foram definidos de acordo com a execução dos algoritmos de clusterização (*Single-Link*, *Average Link* e *Hill-Climbing*) no protótipo desenvolvido para a validação deste trabalho (Seção 5.2). Como ponto de partida, foi considerado um estudo realizado por Draisbach & Naumann (2013), que objetiva identificar os melhores valores de limiar. A partir deste estudo e considerando os algoritmos de clusterização em um processo tradicional de RE, foram realizados experimentos com variações nas funções de similaridade e nos valores de limiar. O limiar foi incrementado em 0.1 de 0.5 até 1.0 com o objetivo de detectar o valor de limiar que alcançasse o maior valor da métrica Medida-F [Christen, 2012].

Em todas as fontes de dados a função de similaridade aplicada foi *Levenshtein* [Christen, 2012]. A Tabela 1 apresenta os atributos e os valores de limiar definidos para os algoritmos *Single-Link*, *Average-Link* e *Hill-Climbing*. Quando mais de um atributo foi selecionado, o valor de similaridade considerado foi a média simples dos valores de similaridade de cada atributo separadamente. Os valores "-" apresentados para os algoritmos *Single-Link* e *Average-Link* em algumas células da Tabela 1, representam que o limiar não foi o melhor para a combinação de atributos apresentados. Para o algoritmo *Hill-Climbing* os valores "-" representam que o limiar não foi avaliado para a fonte de dados em questão ou o limiar não foi o melhor para a combinação de atributos apresentados.

Os experimentos foram executados em uma máquina Dell Inspiron, com um processador Intel Core i5 (2.2 GHz), 4GB de memória RAM e 1 TB de disco rígido. O sistema operacional utilizado foi o Windows 8.1.

Tabela 1 – Valores de limiar

Fonte de Dados	Atributos	*Single-Link*	*Average-Link*	*Hill-Climbing*
CD	*"Título do Álbum"*, *"Nome Artista"* e *"Faixa 1"*	0.9	0.6	-
CD	*"Título do Álbum"*, *"Nome Artista"*	-	-	0.9
Cora	*"Autor"* e *"Título da Publicação"*	0.9	0.5	-
Febrl	*"Nome"* e *"Sobrenome"*	0.9	0.9	-

Para medir a qualidade dos resultados obtidos no processo de RE, foram utilizadas as métricas de Precisão, Revocação e Medida-F, comumente utilizadas para avaliar processos de RE. Os principais objetivos dos experimentos são três. Primeiro, mostrar que o uso dos índices propostos é viável para um processo de RE incremental e orientado à consulta. Segundo, mostrar que o reúso de resultados de iteração passadas do processo de RE, é mais eficiente do que refazer a RE a cada novo resultado de consulta a ser processado. Terceiro, algoritmos de clusterização em um processo de RE incremental podem ter uma qualidade similar aos algoritmos de clusterização em um processo tradicional de RE (com clusterização em lote), com uma eficiência maior.

É importante destacar que as comparações realizadas nos experimentos foram sempre entre algoritmos implementados durante o desenvolvimento desta pesquisa, tanto para avaliação do QUIPER quanto do processo tradicional de RE. Isto ocorreu devido à dificuldade de acesso aos algoritmos apresentados nos trabalhos relacionados para que fosse possível realizar uma comparação entre o QUIPER e os processos apresentados nos trabalhos relacionados. Muitos dos trabalhos relacionados apresentam uma visão geral do algoritmo, mas não disponibilizam uma implementação para que os experimentos sejam replicados ou outros experimentos sejam realizados.

5.4 Metodologia dos Experimentos

Esta seção apresenta a metodologia usada para a realização dos experimentos, detalhando e discutindo os resultados obtidos.

Os experimentos realizados são apresentados nas 7 (sete) subseções seguintes. O Quadro 4 sumariza os principais objetivos de cada experimento:

Quadro 4 – Lista de experimentos

Subseção do Experimento	Objetivo
5.4.1	Define um limiar para o índice de similaridade
5.4.2	Verifica o tempo de processamento à medida que mais resultados de consultas são processados
5.4.3	Verifica tempo de processamento à medida que mais valores são recuperados dos índices propostos
5.4.4	Mede tempo de acesso aos índices propostos
5.4.5	Verifica a qualidade dos resultados alcançados e os tempos de processamento medidos
5.4.6	Amplia o experimento da Seção 5.4.5 e verifica a qualidade dos resultados alcançados e os tempos de processamento medidos em cenários com grande volume de resultados de consultas
5.4.7	Verifica a qualidade dos resultados considerando toda a fonte de dados e não apenas a qualidade para um dado resultado de consulta

5.4.1 Define um Limiar para o Índice de Similaridade

Em cenários com grandes volumes de dados, o Índice de Similaridades (IS) pode ter um grande número de valores de similaridade indexados e isto pode causar um problema de desempenho. Para avaliar este problema, foram criados conjuntos de amostras de tuplas da fonte de dados CD, nos quais cada amostra simula um resultado de consulta passado como entrada para o QUIPER. Inicialmente, considerou-se que o IS estava vazio e que 100% das tuplas eram consideradas como novas (i.e., não haviam sido processadas em

iterações anteriores). Posteriormente, considerou-se que 10% das tuplas haviam sido processadas previamente e 90% eram consideradas tuplas novas e assim sucessivamente até atingir o valor que 80% das tuplas do resultado da consulta já haviam sido processadas previamente.

Adicionalmente, para cada configuração de porcentagem, foram considerados 2 (dois) cenários: i) IS ingênuo: todos os valores de similaridade entre pares de tuplas comparadas durante o processo de RE são indexados. ii) IS modificado: apenas os valores de similaridade acima de um limiar são indexados. No cenário ii o limiar utilizado foi 0.7. Cada cenário foi executado com os algoritmos *Single-Link* e *Hill-Climbing*.

A Tabela 2 apresenta os tempos de execução dos 2 (dois) cenários para cada um dos algoritmos de clusterização considerados. Pode-se observar que quanto mais valores de similaridade são encontrados no IS, menor é o tempo de processamento do QUIPER. A partir dos dados da Tabela 2, não se encontrou uma influência direta e significativa entre o aumento do tamanho dos índices e o aumento no tempo de processamento do QUIPER.

Em algumas configurações, o IS Modificado tornou o QUIPER ligeiramente mais eficiente, principalmente nos cenários em que poucos valores de similaridade estavam indexados no IS. Nestes casos, acreditamos que o custo de acessar um índice ligeiramente maior, sem recuperar um grande volume de informações (comparado ao total de valores de similaridade a serem calculados para um determinado resultado de consulta), teve um custo mais elevado que recuperar algumas informações em um índice menor (IS Modificado).

Neste experimento foi observado que o mais importante para o desempenho do QUIPER é que o Índice de Similaridades tenha muitos valores de similaridade que diminua o volume de comparações entre pares de tuplas em tempo de consulta. Encontrar estes valores de similaridade é mais importante que o tamanho do índice propriamente dito. Em cenários, independente do tamanho do IS, em que os valores de similaridade necessários para um determinado resultado de consulta não são encontrados no IS, o QUIPER não consegue diminuir a número de comparações entre pares de tuplas, não reduzindo, portanto, o tempo de processamento.

Portanto, o IS modificado pode ter um pequeno ganho médio de tempo sobre o IS ingênuo, se valores de similaridade desnecessários não estejam indexados, apresentando

um meio termo entre encontrar alguns valores de similaridade no IS em relação a encontrar todos os valores de similaridade em um IS de grande tamanho.

Tabela 2 – Tempos de execução do QUIPER em milissegundos

Porcentagem de tuplas no IS	***Single-Link* IS Ingênuo**	***Single-Link* IS Modificado**	***Hill-Climbing* IS Ingênuo**	***Hill-Climbing* IS Modificado**
0%	3688	3594	2669	2640
10%	3771.6	3043.6	2011	1390
20%	3584.4	2900.2	2159	1312
30%	3456.2	2859.2	1883	1221
40%	3255.8	2937.4	1875	1187
50%	3072	2659.4	1531	994
60%	2621.8	2353	1584	949
70%	1925	2093.8	1360	1093
80%	906	937	922	875

Para medir a qualidade dos resultados do QUIPER, foi calculada a média da Medida-F de um conjunto de execuções do experimento apresentado. Para a Medida-F, foram considerados os cenários com IS modificado. Por fim, foram comparados os valores da Medida-F do QUIPER com os valores da Medida-F do processo tradicional de RE (sem o uso dos índices – IS e IC).

O valor da Medida-F do QUIPER foi próximo ao valor apresentado pelo processo tradicional de RE. Para o algoritmo *Hill-Climbing*, a maior diferença entre as Medidas-F foi 0.105 e a menor diferença foi 0.016. Em todos os cenários onde se observou diferenças entre os valores de Medida-F, o QUIPER obteve o menor valor. Para o algoritmo *Single-Link*, não foram observadas diferenças relevantes nos valores de Medida-F.

Neste experimento, observou-se baixos valores de Medida-F com o algoritmo *Hill-Climbing* (em média 0.5) comparados aos demais algoritmos avaliados nesta pesquisa. É possível que este baixo valor de Medida-F seja consequência da versão do algoritmo implementada neste trabalho, a qual considerou apenas a função de coesão entre *clusters*. Neste caso, quando o algoritmo encontra um *cluster* com coesão máxima de suas tuplas, ele não permite o *merge* com outros *clusters* e o algoritmo finaliza a busca por tuplas duplicadas. Para minimizar este problema, é possível que uma implementação do algoritmo *Hill-Climbing* com outras funções objetivo (por exemplo, separação entre *clusters*) tenha melhores resultados.

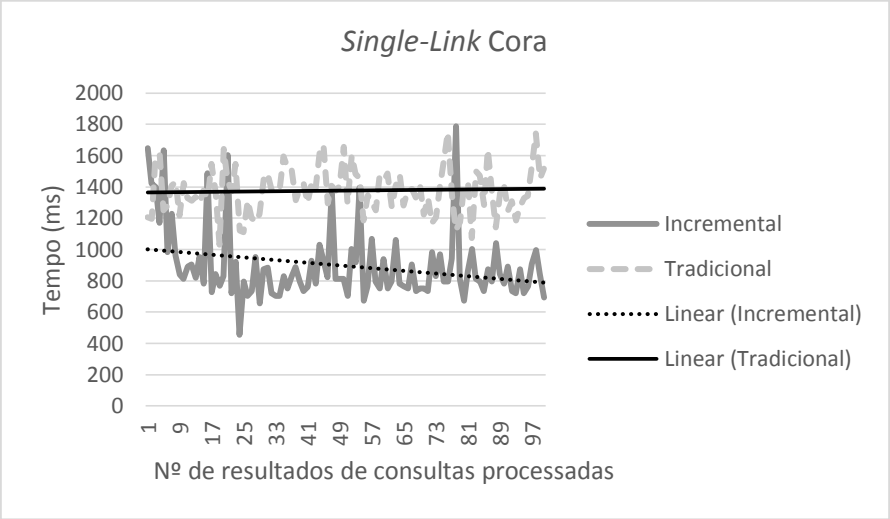
5.4.2 Avaliação do Tempo de Execução com o Aumento dos Resultados Aleatórios de Consultas

Para simular situações adversas de resultados de consultas sendo processados, foram gerados resultados de consultas aleatórios de cada uma das fontes de dados. É importante destacar que os resultados de consultas aleatórios permitem que, a cada execução do experimento, a configuração dos índices seja distinta e reflita amostras de dados com características distintas. Por exemplo, amostras de tuplas onde se tenha: (i) Muitas tuplas indexadas no mesmo *cluster* do IC, o que minimiza o número de comparações a serem realizadas (maior ganho da abordagem proposta), (ii) Poucas tuplas indexadas no mesmo *cluster* do IC, o que não minimiza significativamente o número de comparações a serem realizadas (menor ganho da abordagem proposta), (iii) Muitas tuplas selecionadas para o grupo de tuplas previamente comparadas foram comparadas entre si (alocadas em um mesmo bloco) e, desta forma, muita informação é recuperada do IS e (iv) Muitas tuplas selecionadas para o grupo de tuplas previamente comparadas não foram comparadas entre si (alocadas em blocos distintos) e, desta forma, pouca informação é recuperada do IS.

O objetivo deste experimento foi mostrar que à medida que novos resultados de consultas são processados, o processo tem uma tendência decrescente de tempo de processamento, dado que os índices acumulam informações e, consequentemente, mais informações vão sendo recuperadas dos índices. Para este fim, os parâmetros foram modificados considerando as seguintes variáveis: (i) Tamanho dos resultados de consultas: resultados com um volume considerado pequeno (até 10% das tuplas da fonte), médio (até 20% das tuplas da fonte) e grande (até 30% das tuplas da fonte) e (ii) Número de consultas processadas: um número pequeno (até 100 consultas), médio (até 300 consultas) e grande (até 500 consultas).

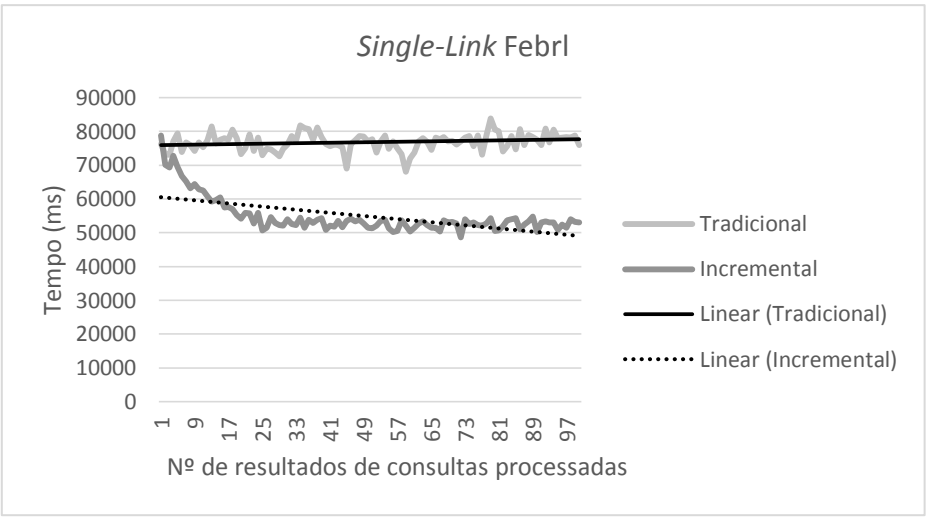
As Figuras 32 e 33 apresentam os gráficos resultantes de uma das configurações dos experimentos, com o algoritmo *Single-Link* executado sobre as fontes de dados Cora e Febrl, respectivamente. Para o cenário com o QUIPER e com o processo tradicional de RE, foi traçada uma reta para destacar a tendência dos valores quando o número de tuplas processadas aumenta (tendência linear). Esse tipo de linha de tendência cria uma linha reta de melhor ajuste para conjuntos de dados, mostrando que o tempo de processamento está constante, aumentando ou diminuindo.

Figura 32 - Tempos de processamento do algoritmo *Single-Link* com resultados de consultas aleatórias na fonte de dados Cora



Fonte: Próprio autor, 2017.

Figura 33 - Tempos de processamento do algoritmo *Single-Link* com resultados de consultas aleatórias na fonte de dados Febrl



Fonte: Próprio autor, 2017.

O número de resultados de consultas processadas até que uma tendência linear decrescente fosse mais perceptível variou de acordo com as características de cada fonte de dados. A fonte de dados CD, por ter poucas tuplas duplicadas, não mostrou um expressivo comportamento decrescente do tempo de processamento, mesmo em situações extremas, onde 500 resultados de consultas, de tamanho grande, foram processados. Isto demonstra que a abordagem proposta neste trabalho é mais eficiente quando se tem resultados de consultas com interseções e que possuam tuplas duplicadas.

Nos experimentos realizados sobre a fonte de dados Cora (Figura 32), que tem como característica um alto grau de duplicidade das tuplas, pode-se perceber, em um cenário com poucos resultados de consultas (100) de tamanho pequeno, uma tendência decrescente no tempo de processamento do QUIPER e uma tendência constante para o processo tradicional de RE.

Nos experimentos realizados sobre a fonte de dados Febrl (Figura 33), que tem por característica um grau médio de duplicidade das tuplas, pode-se perceber, em um cenário onde poucos resultados de consultas (100) e tamanho médio, uma tendência decrescente linear no tempo de processamento do QUIPER e uma tendência constante para o processo tradicional de RE.

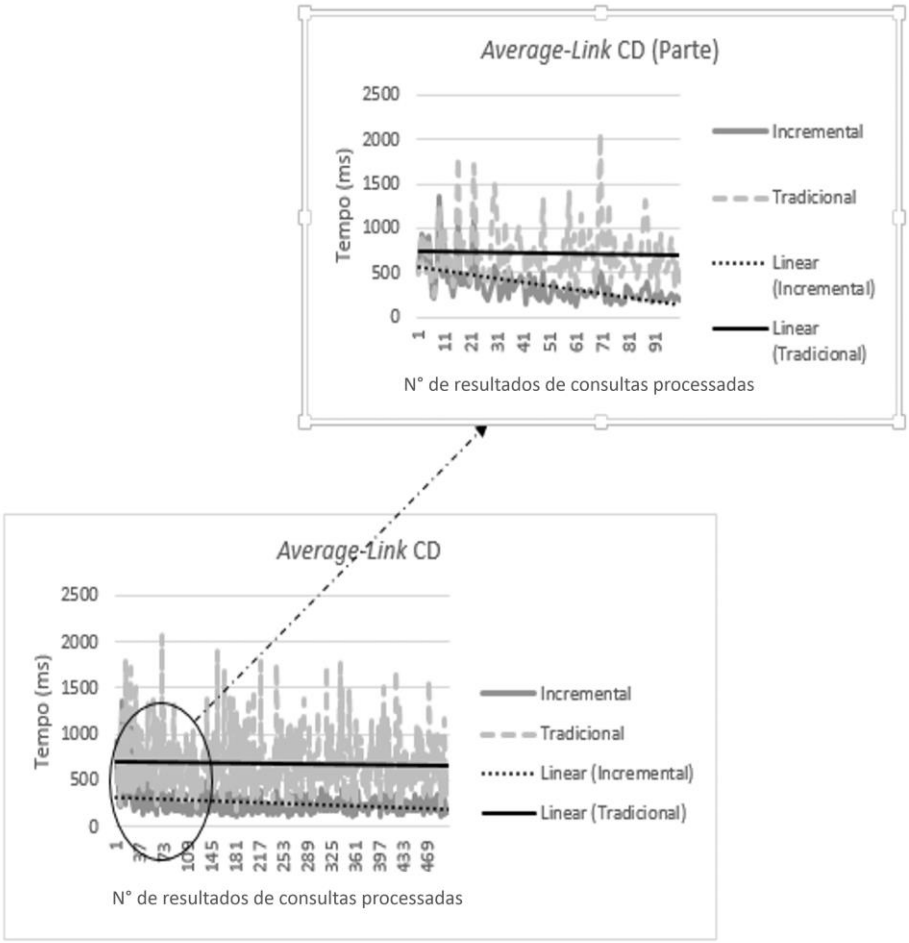
As Figuras 34-36 apresentam os gráficos resultantes, de uma das configurações dos experimentos, com o algoritmo *Average-Link* executado sobre as fontes de dados CD, Cora e Febrl, respectivamente.

O experimento realizado sobre a fonte de dados CD (Figura 34), por ter poucas tuplas duplicadas, não mostrou um expressivo comportamento decrescente do tempo de processamento, mesmo em situações extremas, onde 500 resultados de consultas, de tamanho grande, foram processados. Contudo, destacamos um pequeno ganho em alguns pontos onde sequências de resultados de consultas possuíam interseções. Desta forma, observou-se que a abordagem proposta neste trabalho é mais eficiente em cenários onde resultados de consultas tenham intersecções e que possuam tuplas duplicadas.

Nos experimentos realizados sobre a fonte de dados Cora (Figura 35), pode-se perceber, em um cenário com poucos resultados de consultas (100) e tamanho pequeno, uma tendência decrescente no tempo de processamento do QUIPER e uma tendência constante para o processo tradicional de RE. Comportamento semelhante ao algoritmo *Single-Link* apresentado anteriormente.

Nos experimentos realizados sobre a fonte de dados Febrl (Figura 36), que tem por característica um grau médio de duplicidade das tuplas, pode-se perceber, em um cenário com poucos resultados de consultas (100) e tamanho médio, uma tendência decrescente no tempo de processamento do QUIPER e uma tendência constante para o processo tradicional de RE. Comportamento semelhante ao algoritmo *Single-Link* apresentado anteriormente.

Figura 34 - Tempos de processamento do algoritmo *Average-Link* com resultados de consultas aleatórias na fonte de dados CD

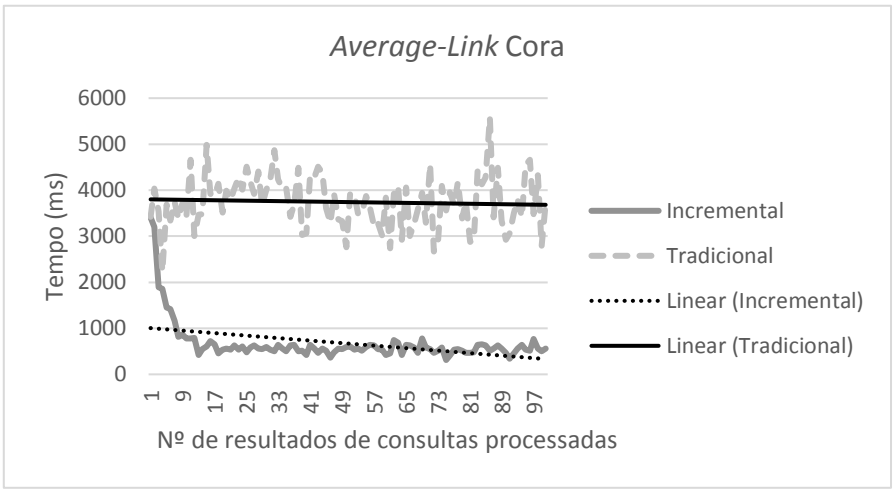


Fonte: Próprio autor, 2017.

A partir destes experimentos, foi observado que o principal fator que contribui para redução do tempo do QUIPER sobre o processo tradicional de RE é o Índice de *Clusters* (IC). Este índice permite que as tuplas clusterizadas em iterações anteriores e inseridas em um mesmo *Cluster* Global não necessitem mais ser comparadas, reduzindo, portanto, o tempo de processamento à medida que mais tuplas são inseridas em um mesmo *cluster*.

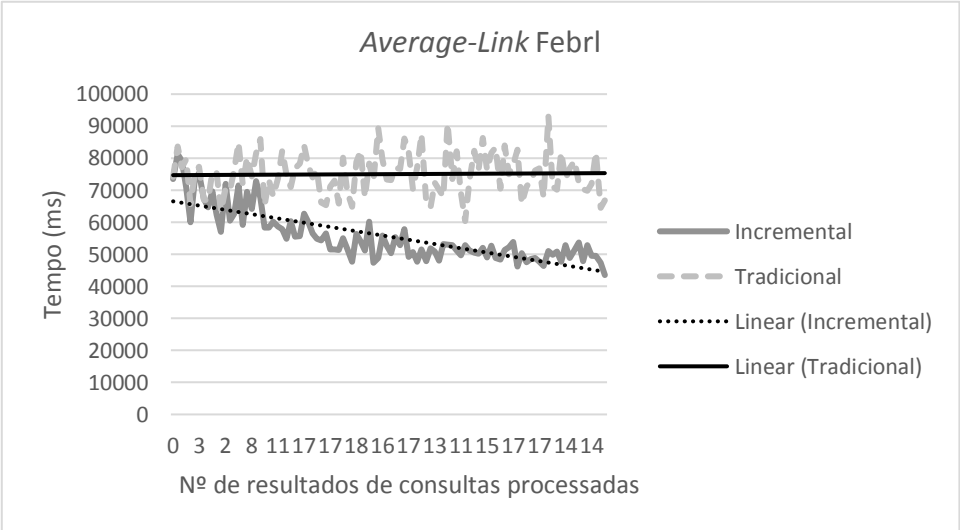
Esta característica foi observada em um *log* que indicou, em cada resultado de consulta, quantos pares de tuplas estavam previamente em *Clusters* Globais e haviam sido recuperados. Nos resultados de consultas nos quais mais tuplas estavam em *clusters* globais, observou-se experimentos com maior ganho de tempo de processamento, nestes casos, um número maior de pares de tuplas eram recuperados do IC.

Figura 35 - Tempos de processamento do algoritmo *Average-Link* com resultados de consultas aleatórias na fonte de dados Cora



Fonte: Próprio autor, 2017.

Figura 36 - Tempos de processamento do algoritmo *Average-Link* com resultados de consultas aleatórias na fonte de dados Febrl



Fonte: Próprio autor, 2017.

5.4.3 Tempo do Processo de RE com Garantia de Tuplas Duplicadas

A partir dos experimentos anteriores, foi observado que o volume de tuplas indexadas como duplicadas tem relação com a eficiência do processo proposto. O objetivo deste experimento foi estimar qual o ganho (eficiência) no processamento do resultado de uma

consulta, considerando que parte (uma porcentagem) das informações necessárias para o processo de RE estejam indexadas nos índices dinâmicos. Para este fim, o experimento da Seção 5.4.1 foi ampliado.

Os parâmetros foram variados e, em um primeiro momento, considerou-se que 90% das tuplas não haviam sido comparadas em iterações anteriores do QUIPER e 10% sim e, destes 10%, cada tupla tinha pelo menos uma tupla duplicada. Posteriormente, considerou-se que 80% das tuplas não haviam sido comparadas em iterações anteriores e 20% sim, e assim sucessivamente até considerar que 0% das tuplas não haviam sido comparadas em iterações anteriores e 100% sim.

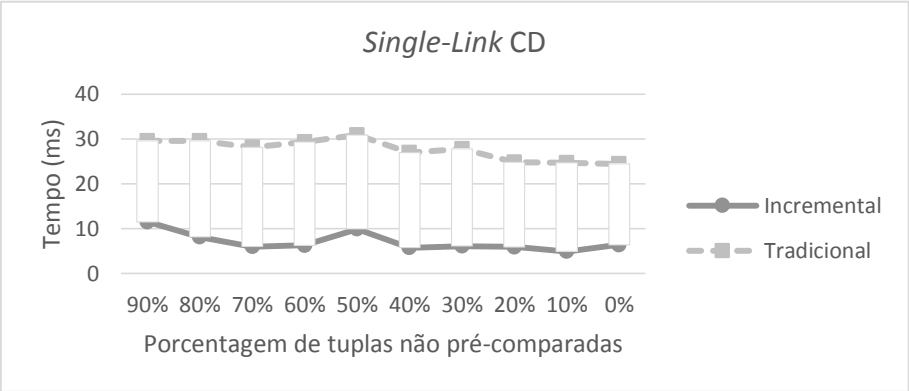
É importante destacar que tanto as tuplas selecionadas para estarem no grupo das tuplas comparadas em iterações anteriores, quanto as tuplas no grupo das tuplas não previamente comparadas (tuplas novas), foram selecionadas randomicamente, a fim de evitar enviesamento dos experimentos. Desta forma, foi possível avaliar configurações distintas dos índices e dos resultados de consultas a serem processados.

Considerando as distintas características de cada um dos resultados de consultas gerados, e objetivando mensurar o caso médio, cada caso foi repetido 100 vezes e o tempo de execução foi calculado como sendo a média dos tempos em todas as execuções. Para cada amostra, foi medido o tempo de execução do processo tradicional de RE e do QUIPER. A seguir, são apresentados os resultados obtidos usando o algoritmo *Single-Link*.

Algoritmo *Single-Link*

As Figuras 37-39 apresentam os gráficos resultantes dos experimentos com o algoritmo *Single-Link* executado sobre as fontes de dados CD, Cora e Febrl, respectivamente. Em todos os casos o QUIPER teve desempenho melhor que o processo tradicional de RE.

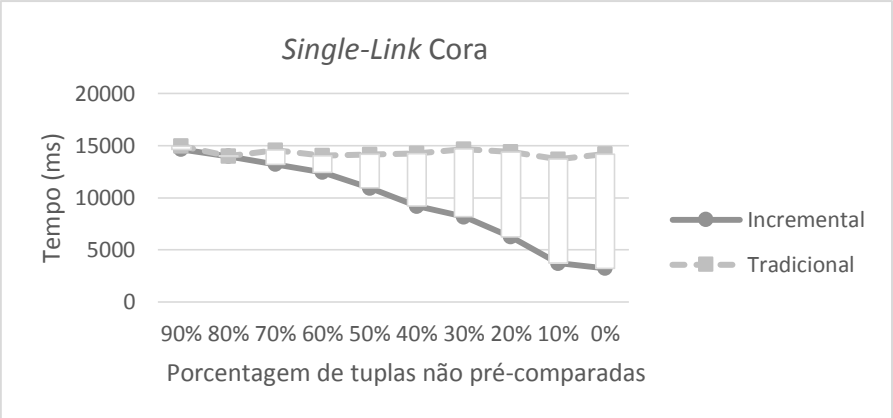
Figura 37 - Tempos de execução do algoritmo *Single-Link* para a base de dados CD



Fonte: Próprio autor, 2017.

Na fonte de dados CD (Figura 37) o menor ganho do QUIPER sobre o processo tradicional de RE foi de aproximadamente 60%, enquanto que o maior ganho foi de aproximadamente 80%. Esta fonte de dados é caracterizada por ter poucas tuplas duplicadas, por este motivo o tempo de processamento do algoritmo incremental não foi significativamente decrescente à medida que o volume de tuplas previamente comparadas aumentou. Contudo, o ganho é significativo em relação ao algoritmo Single-Link no processo tradicional de RE.

Figura 38 - Tempos de execução do algoritmo *Single-Link* para a base de dados Cora

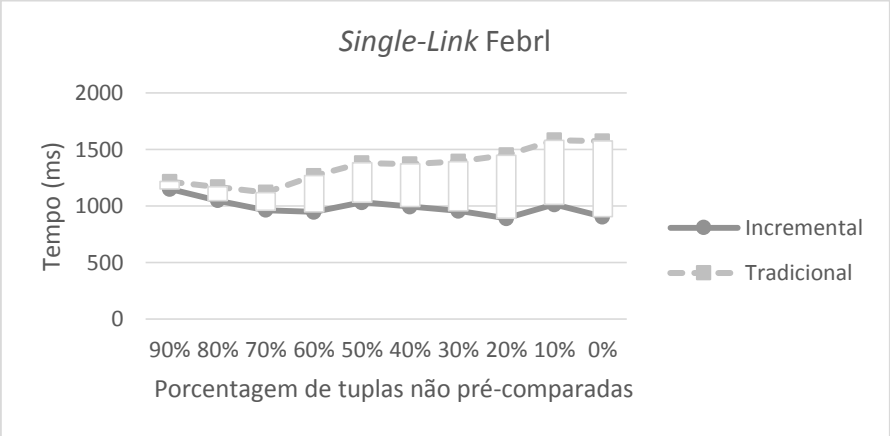


Fonte: Próprio autor, 2017.

Na fonte de dados Cora (Figura 38) o menor ganho do QUIPER foi de aproximadamente 2%, enquanto que o maior ganho foi de aproximadamente 78%. Esta fonte de dados é caracterizada por ter muitas tuplas duplicadas, por este motivo o tempo de processamento do algoritmo incremental foi significativamente decrescente à medida que o volume de

tuplas previamente comparadas aumentou. Adicionalmente, o ganho é significativo em relação ao processo tradicional de RE.

Figura 39 - Tempos de execução do algoritmo *Single-Link* para a base de dados Febrl



Fonte: Próprio autor, 2017.

Na fonte de dados Febrl (Figura 39) o menor ganho do QUIPER foi de aproximadamente 5%, enquanto que o maior ganho foi de aproximadamente 43%. Esta fonte de dados é caracterizada por ter muitas tuplas, a maior fonte de dados avaliada, e o fato de terem sido extraídos resultados aleatórios de consultas pode ter ocasionado a geração de resultados de consultas sem (ou com poucas) tuplas duplicadas. Esta fonte de dados foi avaliada como tendo um número médio de tuplas. Por este motivo, o tempo de processamento do algoritmo incremental teve tendência decrescente à medida que o volume de tuplas previamente comparadas aumentou.

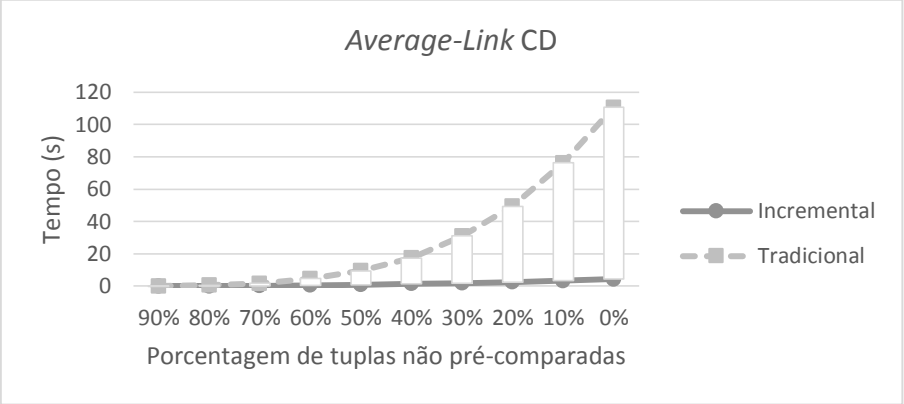
A seguir, são apresentados os resultados obtidos usando o algoritmo *Average-Link* no QUIPER, comparado ao *Average-Link* no processo tradicional de RE.

Algoritmo *Average-Link*

Os experimentos realizados com o *Single-Link* foram replicados e realizados com os algoritmo *Average-Link*. Naturalmente, o algoritmo *Average-Link* é mais custoso que o algoritmo *Single-Link*. Por este motivo, os tempos alcançados nesta segunda bateria de experimentos foram maiores que os medidos na primeira bateria de experimentos. Adicionalmente, à medida que menos comparações são necessárias, as diferenças se tornam mais evidentes entre o QUIPER e o processo tradicional de RE.

As Figuras 40-42 apresentam os gráficos resultantes dos experimentos com o algoritmo *Average-Link* executado sobre as fontes de dados CD, Cora e Febrl, respectivamente. Em todos os casos o QUIPER teve desempenho melhor que o processo tradicional de RE.

Figura 40 - Tempos de execução do algoritmo *Average-Link* para a base de dados CD



Fonte: Próprio autor, 2017.

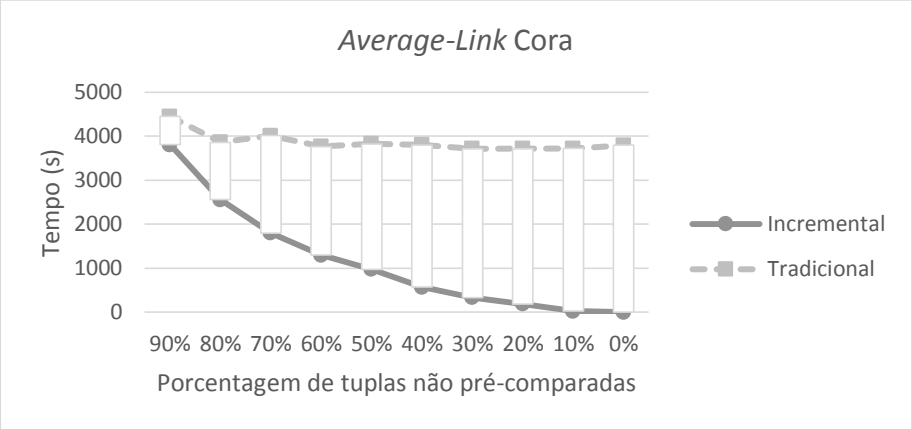
Na fonte de dados CD (Figura 40) o menor ganho do QUIPER foi de aproximadamente 56%, enquanto que o maior ganho foi de aproximadamente 96%. Por esta fonte de dados ter poucas tuplas duplicadas, ao serem fixados resultados de consultas com um número de tuplas duplicadas crescente a cada caso considerado, observou-se que o processo tradicional de RE necessitou de um tempo maior à medida que o número de tuplas duplicadas aumentou. Esta característica da fonte de dados justifica o comportamento crescente do processo tradicional. O QUIPER não se comportou da mesma forma, pois observou-se que a maior parte das tuplas duplicadas dos resultados de consultas considerados estavam indexados, o que reduziu drasticamente o tempo de processamento.

Desta forma, considerando um conjunto fixo de tuplas duplicadas e indexadas, observou-se que o QUIPER tem um ganho significativo à medida que o volume de tuplas duplicadas são indexadas.

Na fonte de dados Cora (Figura 41) o menor ganho do QUIPER foi de aproximadamente 14%, enquanto que o maior ganho foi de aproximadamente 99%. A fonte de dados Cora é considerada como tendo muitas tuplas duplicadas. Esta característica implica em tempo médio de processamento do processo tradicional de RE constante (considerando resultados de consultas de mesmo tamanho e, em média, com mesmo número de tuplas duplicadas).

O experimento mostrou que, similar ao que ocorreu com a fone de dados CD, à medida que mais informações são recuperadas dos índices, o ganho de eficiência do QUIPER sobre o processo tradicional de RE é significativo, tal como o tempo de processamento é decrescente.

Figura 41 - Tempos de execução do algoritmo *Average-Link* para a base de dados Cora



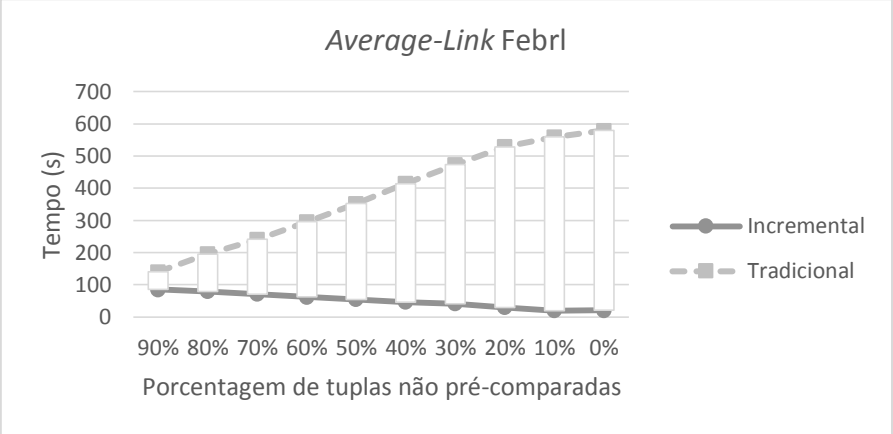
Fonte: Próprio autor, 2017.

Na fonte de dados Febrl (Figura 42) o menor ganho do QUIPER foi de aproximadamente 39%, enquanto que o maior ganho foi de aproximadamente 96%. A extração aleatória de resultados de consultas em uma fonte de dados como a Febrl, que tem muitas tuplas não duplicadas, implica em uma maior probabilidade de serem extraídos resultados de consultas sem tuplas duplicadas. Desta forma, observou-se que os resultados de consultas extraídos tinham como tuplas duplicadas basicamente a porcentagem fixada. Por este motivo, o tempo de processamento do processo tradicional de RE teve tendência crescente à medida que o volume de tuplas previamente comparadas aumentou.

Com estes experimentos com tuplas duplicadas fixas observou-se que quanto maior o número de tuplas duplicadas nos resultados de consultas e quanto maior o número de tuplas duplicadas indexadas, melhor o desempenho do QUIPER. Em todos os experimentos conclui-se que consultas com pelo menos 10% de tuplas indexadas são beneficiadas com o reúso de informações dos índices.

Adicionalmente, observou-se que o algoritmo *Single-Link* é menos sensível a grandes volumes de dados duplicados e tem uma convergência de *clusters* mais eficiente.

Figura 42 - Tempos de execução do algoritmo *Average-Link* para a base de dados Febrl



Fonte: Próprio autor, 2017.

5.4.4 Tempo de Acesso aos Índices

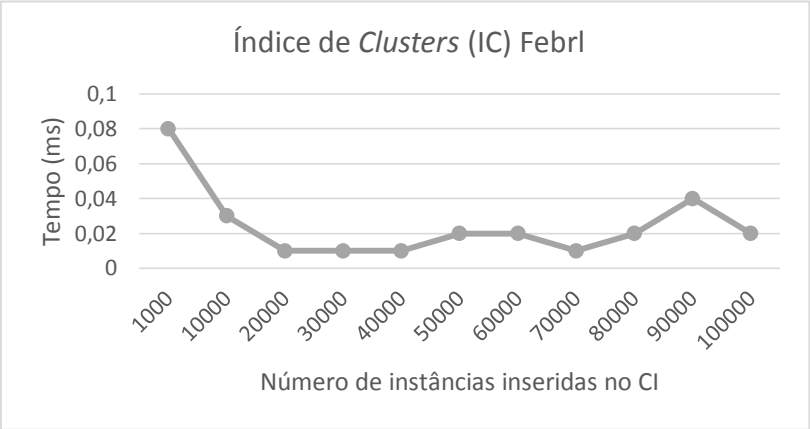
Para avaliar o impacto do acesso aos índices propostos no QUIPER, foram modelados dois experimentos, um para cada índice. O principal objetivo foi verificar a escalabilidade dos índices, tal como o tempo de acesso à medida que os índices crescem. Neste sentido, calculou-se o tempo médio de acesso considerando a busca de um valor aleatório. Neste caso, a busca nos índices foi realizada em momentos que os índices tinham diferentes tamanhos e o valor buscado poderia estar em qualquer posição ou ser inexistente.

Para o Índice de *Clusters* (IC) coletou-se resultados aleatórios de consultas da fonte de dados Febrl. Estes resultados de consultas foram processados e seus *clusters* indexados no IC. Iniciou-se com resultados de consultas com mil tuplas, posteriormente 10 mil tuplas, e assim sucessivamente até alcançar o valor de 100 mil tuplas. Para cada resultado de consulta gerado e processado, foram selecionadas aleatoriamente tuplas para serem buscadas no IC. A Figura 43 apresenta a média de tempos de acesso em cada caso. Cada caso foi repetido por 500 vezes, com o objetivo de se aproximar de buscas em um sistema real.

É importante observar que a variação do tempo de acesso ao IC em relação ao volume de dados indexados é pequena. Contudo, no primeiro caso (mil tuplas) obteve-se um tempo destoante em relação aos demais casos. Isto ocorreu porque as tuplas aleatórias escolhidas para serem buscadas no IC quase sempre não foram encontradas, dado que a fonte Febrl possui 130 mil tuplas e apenas 0.7% estavam indexadas. Este cenário pode ser equiparado

a um cenário real, com o sistema de integração sendo inicializado, onde ainda não se tem muitas informações sobre os dados.

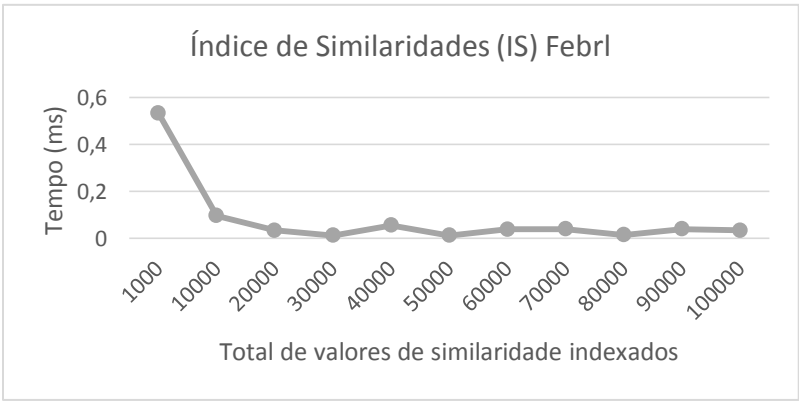
Figura 43 - Tempos de acesso ao IC



Fonte: Próprio autor, 2017.

Para o Índice de Similaridades (IS) coletou-se resultados aleatórios de consultas também da fonte de dados Febrl. Estes resultados de consultas foram processados e os valores de similaridade entre pares de tuplas foram indexados. Iniciou-se com mil valores de similaridade, e foram sendo incrementados na mesma proporção do experimento com o IC. Para cada resultado de consultas gerado e processado, foram selecionadas aleatoriamente pares de tuplas da fonte de dados Febrl para terem seus valores de similaridade buscados no IS. A Figura 44 apresenta a média de tempos de acesso medidos em cada caso. Cada caso foi repetido por 500 vezes.

Figura 44 - Tempos de Acesso ao IS



Fonte: Próprio autor, 2017.

É importante observar que neste experimento, assim como no experimento com o IC, a variação do tempo de acesso ao IS em relação ao volume de dados indexados é pequena.

No cenário em que o IS tem poucos valores de similaridade indexados, obteve-se o mesmo comportamento do IC, um tempo destoante em relação aos demais casos. Neste caso, os valores de similaridade entre tuplas a serem buscados no IS quase sempre não foram encontradas, dado que, nesta configuração, a fonte Febrl possuía aproximadamente apenas 0.3% das tuplas com algum valor de similaridade indexado.

5.4.5 Relação entre Qualidade dos Resultados e Tempo de Processamento

Para medir a qualidade dos resultados obtidos pelo QUIPER foram realizados experimentos em termos de Medida-F e tempo de execução. Foi calculada a média de tempo de execução e a média dos valores de Medida-F dos resultados alcançados em todos os cenários apresentados na Seção 5.4.2. Estes cenários simulam situações adversas de resultados de consultas sendo processados por meio do QUIPER e do processo tradicional de RE sobre cada resultado de consulta.

As Tabelas 3, 4 e 5 apresentam a média dos tempos de execução, juntamente com a média dos valores de Precisão, Revocação e Medida-F coletados para uma sequência de resultados de consultas extraídos das fontes de dados CD, Cora e Febrl, respectivamente.

Em termos de qualidade dos resultados do QUIPER, para as três fontes de dados, observou-se uma pequena perda de precisão. Acreditamos que esta queda da precisão é resultado do número de falsos-positivos gerados por influência errada da transitividade entre pares de *clusters* na tarefa de *Atualização dos Clusters Globais* (Seção 4.6.2). Contudo, obteve-se valores maiores de Revocação e Medida-F com relação ao processo tradicional de RE. Este ganho se dá porque nos processos tradicionais de RE são avaliadas apenas as tuplas em um único resultado de consulta isolado, ao passo que o QUIPER considera resoluções de entidades realizadas em iterações prévias, mantendo o histórico de resultados de consultas processados.

Para a fonte de dados CD, o QUIPER com *Single-Link* apresentou uma perda de eficiência em relação ao processo tradicional de aproximadamente 13%. Esta perda se deu pelo fato de poucas tuplas duplicadas estarem indexadas e serem acessadas nos índices propostos. Para o *Average-Link* observou-se um ganho médio em relação ao processo tradicional de 78%. Para este ganho, observou-se que apesar da fonte de dados CD conter poucas tuplas duplicadas, as informações coletadas dos índices impactam mais bruscamente no

algoritmo *Average-Link*, que tem desempenho mais lento que o *Single-Link*, tornando os ganhos mais evidentes.

Tabela 3 – Comparação da qualidade dos resultados e o tempo de execução dos processos de RE para a fonte de dados CD

Algoritmo / Métrica	*Single-Link* Tradicional	*Single-Link* QUIPER	*Average-Link* Tradicional	*Average-Link* QUIPER
Tempo (ms)	312.74	352.96	1604.52	341.12
Precisão	0.977	0.977	0.994	0.991
Revocação	0,545	0,548	0.903	0.914
Medida - F	0.700	0.702	0.947	0.951

Para a fonte de dados Cora, em termos de tempo de processamento, observou-se um ganho do algoritmo *Single-Link* no QUIPER em relação ao processo tradicional de RE de aproximadamente 29%. Para o algoritmo *Average-Link* o QUIPER apresentou um ganho médio em relação ao processo tradicional de RE de 71%. Para estes ganhos, observou-se que a característica da base de dados Cora, de conter muitas tuplas duplicadas, influencia para que muitas informações sejam coletadas dos índices.

Tabela 4 - Comparação da qualidade dos resultados e o tempo de execução dos processos de RE para a fonte de dados Cora

Algoritmo / Métrica	*Single-Link* Tradicional	*Single-Link* QUIPER	*Average-Link* Tradicional	*Average-Link* QUIPER
Tempo (ms)	869.76	613.31	1818.18	520.05
Precisão	0.851	0.852	0.821	0.820
Revocação	0.728	0.736	0.868	0.869
Medida - F	0.785	0.790	0.844	0.844

Para a fonte de dados Febrl, em termos de tempo de processamento, o algoritmo *Single-Link* no QUIPER apresentou um ganho de aproximadamente 19% em relação ao processo tradicional de RE. O algoritmo *Average-Link* teve um ganho médio no QUIPER em relação ao processo tradicional de RE de 22%. Para estes ganhos, observou-se que a característica da base de dados Febrl, de conter muitas tuplas e uma quantidade média de

tuplas duplicadas, influenciou para que uma quantidade média de informações sejam coletadas dos índices.

Tabela 5 - Comparação da qualidade dos resultados e o tempo de execução dos processos de RE para a fonte de dados Febrl

Algoritmo / Métrica	*Single-Link* Tradicional	*Single-Link* QUIPER	*Average-Link* Tradicional	*Average-Link* QUIPER
Tempo (ms)	34407.83	27828.05	32032.62	24877.52
Precisão	0.876	0.875	0,892	0,892
Revocação	0.701	0.702	0.663	0.663
Medida - F	0.779	0.779	0.761	0.761

Na fonte de dados Febrl, o ganho do algoritmo *Average-Link* no QUIPER foi menos evidente que nos experimentos apresentados anteriormente (Tabelas 3 e 4). Isto pode ter ocorrido pelo falo de terem sido considerados resultados de consultas com volume maior que os anteriores e com características mais adversas, dado que a fonte de dados Febrl é a maior das três fontes de dados analisadas.

Nestes experimentos se observou que o tempo de processamento da RE sobre cada resultado de consulta pode ser alto, caso se tenha uma restrição de tempo para se responder à uma consulta específica. Contudo, é importante destacar que com o QUIPER o tempo de resposta a uma consulta é reduzido se comparado ao processo tradicional de RE e que quanto mais informações são recuperadas de iterações prévias do QUIPER, melhor é seu desempenho.

Para minimizar ainda mais o tempo de processamento em tempo de consulta é possível avaliar se uma combinação de algoritmos de clusterização pode tornar o QUIPER mais rápido e minimizar as perdas de precisão. Por exemplo, utilizar um algoritmo mais custoso (e com maiores valores de precisão) para os primeiros resultados de consultas e, posteriormente, quando o Índice de *Clusters* tiver mais estável (com um grande volume de tuplas indexadas), aplicar algoritmos menos custosos;

5.4.6 Comportamento com um conjunto Grande de Resultados de Consultas Processado

Neste experimento, foi avaliado o comportamento do QUIPER em cenários em que o volume de tuplas nos resultados de consultas cresce. Para este propósito, foram gerados resultados aleatórios de consultas da fonte de dados Febrl. Inicialmente 500 resultados de consultas e posteriormente os resultados foram incrementados até 5000 resultados de consultas acumuladas. A Tabela 6 apresenta a média dos tempos de execução e dos valores de Medida-F para o QUIPER e para o processo tradicional de RE.

Tabela 6 – Comportamento do QUIPER sobre a fonte de dados Febrl

Algoritmo / Métrica	*Single-Link* Tradicional	*Single-Link* QUIPER	*Average-Link* Tradicional	*Average-Link* QUIPER
Tempo (ms)	156.1817	132.9143	166.0859	131.4505
Medida - F	0.839827	0.82356	0.843535	0.808827

Os algoritmos *Single-Link* e *Average-Link*, foram 15% e 21%, respectivamente, mais rápidos no QUIPER do que no processo tradicional de RE. Neste caso, observa-se que mesmo em cenários em que os índices armazenam um grande volume de valores, o QUIPER alcança bons índices de desempenho.

Com relação à qualidade dos resultados, a média dos valores de Medida-F, utilizando o algoritmo *Single-Link* e *Average-Link* no QUIPER foi um pouco menor (aproximadamente 1% e 4% respectivamente) com relação ao processo tradicional de RE. Esta pequena perda, dependendo dos objetivos do sistema de integração, pode ser compensada pelos ganhos em tempo de execução.

5.4.7 Qualidade dos Resultados em Toda a Fonte de Dados

Os experimentos anteriores avaliaram qualidade dos resultados e tempo de processamento para cada resultado de consulta processado e comparou o QUIPER com o processo tradicional de RE. Este experimento compara a qualidade do resultado do processo tradicional de RE sobre todas as tuplas de uma fonte de dados em relação à qualidade do QUIPER aplicado sobre um conjunto de resultados de consultas que juntas representam

todas as tuplas de uma fonte de dados. Neste caso, após QUIPER ser aplicados sobre um conjunto de resultados de consulta, avalia-se se a porcentagem de pares de tuplas duplicadas encontrados, é a mesma dos pares de tuplas duplicadas encontrados pelo processo tradicional de RE sobre toda a fonte de dados.

Para este experimento, foram gerados resultados de consultas da fonte de dados Cora. Considerou-se cenários com 200 resultados de consultas ou até que todas as tuplas da fonte de dados tivessem sido acessadas. Por fim, avaliou-se os *Clusters* Globais gerados (acessando o Índice de *Clusters* -IC) e calculou-se o valor da Medida-F. Foram comparadas as médias dos valores de Medida-F do QUIPER (100 execuções) e a Medida-F encontrada no processo tradicional de RE.

O QUIPER com o algoritmo *Single-Link* obteve média de Medida-F menor que o processo tradicional de RE (aproximadamente 1%). Para o *Average-Link*, o QUIPER obteve um valor de Medida-F aproximadamente 3.5% menor que o valor obtido pelo processo tradicional de RE.

5.4.8 Síntese dos Experimentos

Os experimentos realizados tiveram por atividade fim a validação ou refutação das hipóteses definidas na Seção 1.2. O Quadro 5 sumariza as principais características de cada experimento.

- Experimento: Subseção na qual o experimento é descrito;
- Objetivo: Principal objetivo do experimento;
- Métrica: Métricas utilizadas para avaliar os resultados obtidos no experimento;
- Hipótese: Quais as hipóteses que o experimento objetiva avaliar;
- Resultado alcançado: Indica se as hipóteses foram validadas ou refutadas.

Quadro 5 – Síntese dos experimentos

Subseção do Experimento	Objetivo	Métrica	Hipótese	Resultado Alcançado
5.4.1	Definição de um limiar para o índice de similaridade	Tempo de processamento e Medida-F	Hipóteses 01, 03, 02	Hipóteses validadas
5.4.2	Verifica o tempo de processamento à medida que mais resultados de consultas são processados	Tempo de processamento	Hipóteses 01, 03, 04	Hipóteses validadas
5.4.3	Verifica tempo de processamento à medida que mais valores são recuperados dos índices propostos	Tempo de processamento	Hipóteses 01, 04	Hipóteses validadas
5.4.4	Mede tempo de acesso aos índices propostos	Tempo de processamento	Hipótese 03	Hipótese validada
5.4.5	Verifica a qualidade dos resultados alcançados e os tempos de processamento medidos	Tempo de processamento e Medida-F	Hipótese 02	Hipóteses validadas
5.4.6	Amplia o experimento da Seção 5.4.5 e verifica a qualidade dos resultados alcançados e os tempos de processamento medidos em cenários com grande volume de resultados de consultas	Tempo de processamento e Medida-F	Hipótese 02	Hipótese validada
5.4.7	Verifica a qualidade dos resultados considerando toda a fonte de dados e não apenas a qualidade para um dado resultado de consulta	Medida-F	Hipótese 05	Hipótese validada

5.5 Considerações

O desenvolvimento do protótipo para execução e avaliação do QUIPER foi primordial para a implantação dos experimentos. É importante destacar que o protótipo desenvolvido, tal como os algoritmos de clusterização, podem ser utilizados para um processo de RE incremental, onde um incremento é qualquer amostra de tuplas, não dependendo, portanto, que uma consulta seja executada.

Com os resultados obtidos nos experimentos, foi possível confirmar o benefício de se realizar Resolução de Entidades incremental por consulta, minimizando o tempo de processamento e mantendo a qualidade dos resultados obtidos. Por outro lado, foi possível observar os cenários em que a abordagem proposta não possui um bom desempenho, podendo ser possível a identificação das características necessárias para que a abordagem proposta seja mais eficiente.

Observou-se que, para que o QUIPER tenha bom desempenho, os resultados das consultas devem possuir interseções e tuplas duplicadas. Os índices propostos trazem benefícios quando pelo menos 10% das tuplas duplicadas de um resultado de consulta estejam indexadas no Índice de *Clusters* (IC). Neste caso, as informações recuperadas dos índices apoiam o QUIPER e reduzem o tempo de processamento.

6 CONCLUSÕES

Neste trabalho foi proposto um processo Incremental e orientado à Consulta para RE (QUIPER). Foram projetados e executados experimentos que avaliam a qualidade dos resultados obtidos pelo QUIPER e os compara com o processo tradicional de RE, assim como tempo de processamento (eficiência dos processos). Os experimentos foram realizados em fontes de dados reais e sintéticas.

Um protótipo que implementa o QUIPER foi desenvolvido e 3 (três) algoritmos de clusterização (*Single-Link*, *Average-Link* e *Hill-Climbing*) foram implementados com o objetivo de detectar quais pares de tuplas representam a mesma entidade do mundo real. Estes algoritmos foram avaliados dentro do QUIPER e do processo tradicional de RE.

Para a indexação, foi proposta uma indexação dinâmica com dois índices: Índice de *Clusters* (IC) e Índice de Similaridades (IS). Estes índices foram propostos de forma a se adequarem às necessidades do QUIPER. Os experimentos mostraram que o IC e o IS são viáveis para o funcionamento do QUIPER e subsidiam a Resolução de Entidades incremental.

Basicamente, o principal objetivo do QUIPER é ter uma qualidade de resultados próxima ao processo tradicional de RE, mas com um tempo menor de processamento. Os experimentos mostraram que o reúso de iterações prévias do processo de RE o torna significativamente mais rápido que executar a RE em tempo de consulta. Esta constatação dá subsídios para que o processo incremental seja aplicado em ambientes com grandes volumes de dados dinâmicos.

O capítulo é organizado como segue: Seção 6.1 apresenta as contribuições desta pesquisa; Seção 6.2 indica alguns direcionamentos para trabalhos futuros. Seção 6.3 discute os desafios identificados na literatura para o processo de Resolução de Entidades e pesquisas em aberto;

6.1 Contribuições

As contribuições deste trabalho foram práticas e teóricas. Estas são listadas como segue:

- Foi proposto, implementado e avaliado o QUIPER com diferentes algoritmos de clusterização e fontes de dados;
- Foi proposto o uso de índices dinâmicos no cenário de Resolução de Entidades incremental;
- Foi mostrado, em fontes de dados reais e sintéticas, que os índices propostos são adequados para o QUIPER;
- Os experimentos mostraram que o valor de Medida-F do QUIPER sobre cada resultado de consulta é próximo do valor de Medida-F do processo tradicional de Resolução de Entidades. Contudo, o tempo de processamento do QUIPER é substancialmente menor que o do processo tradicional de RE.
- Os experimentos realizados com toda a fonte de dados submetida ao processo tradicional de Resolução de Entidades e a mesma fonte de dados particionada em resultados de consulta e submetida ao QUIPER, apresentou valores de Medida-F próximos.
- O QUIPER foi publicado no *Alberto Mendelzon International Workshop on Foundations of Data Management* em 2016 [Vieira et *al*. 2016].
- Os experimentos iniciais dos índices propostos foram publicados na *International Conference on Enterprise Information Systems* em 2017 [Vieira et *al*. 2017].

6.2 Trabalhos Futuros

Toda pesquisa tem limitações e merecem ampliações e/ou melhorias. Assim, algumas limitações foram detectadas e/ou problemas que podem ser contornados, apontando direcionamentos futuros para o processo de Resolução de Entidades orientada à consulta. Estes são descritos a seguir:

- Avaliar outros algoritmos de clusterização e identificar quais categorias de algoritmos são mais adequadas para a Resolução de Entidade incremental;
- Avaliar se uma combinação de algoritmos de clusterização pode tornar o QUIPER mais rápido e minimizar as perdas de precisão. Por exemplo, utilizar um algoritmo mais custoso (e com maiores valores de precisão) para os primeiros resultados de consultas e, posteriormente, quando o Índice de *Clusters* tiver mais estável (com um grande volume de tuplas indexadas), aplicar algoritmos menos custosos;

- Identificar estratégias de atualização dos *Clusters* Globais com novas funções de união ou separação de *clusters* com o objetivo de aumentar os valores de Medida-F alcançados. Isto provocaria alterações nas tarefas de *Clusterização Local* e *Atualização dos Clusters Globais* da etapa de *Clusterização*;
- Avaliar estratégias de indexação que reflitam diretamente as alterações realizadas nas tuplas das fontes de dados. Por exemplo, se uma tupla tem um ou mais valores de atributos alterados, esta tupla deve ser considerada como nova para o Índice *Clusters* e o Índice de Similaridades;
- Realizar operações de manutenção dos índices e avaliar o impacto destas operações no desempenho do QUIPER. Por exemplo, informações que são pouco acessadas ou que não serão utilizadas pelos algoritmos de clusterização podem ser excluídas. Ademais, é possível que seja necessário reestruturar os índices por completo à medida que um grande volume de resultados de consultas é processado pelo QUIPER;
- Paralelizar o QUIPER, principalmente nas etapas de *Indexação Dinâmica* e *Clusterização*. Apesar do QUIPER ter sido definido como naturalmente paralelizável (os blocos de tuplas podem ser processados em paralelo, os índices podem ser acessados de forma paralela, tal como os algoritmos de clusterização podem ser executados de forma paralela), não foram realizados experimentos para identificar o que de fato é independente e pode ser executado de forma paralela. Contudo, se espera um ganho de desempenho razoável seguindo a tendência da literatura de processamento paralelo de grandes volumes de dados;
- Realizar experimentos com resultados reais de consultas em um sistema de integração de dados, possibilitando uma avaliação real do quão resultados de consultas inter-relacionadas podem ter interseções e fazer uso das vantagens do QUIPER.

6.3 Pesquisas em Aberto na Área de Resolução de Entidades

Considerando o processo de Resolução de Entidades (RE), nesta seção são explanados alguns desafios discutidos na literatura, assim como problemas em aberto para futuras pesquisas.

Uma das etapas da RE é a avaliação dos resultados gerados. Neste contexto, detectou-se a necessidade de um *framework* unificado com o objetivo de realizar comparações entre diversas categorias de algoritmos, assim como a possibilidade de realização de testes

considerando modelos distintos de dados (XML, Relacional, RDF, Geográficos, entre outros) [Christen, 2012]. Idealmente, seria interessante a possibilidade de inserção de novos algoritmos e funções de similaridade.

Em consonância com o *framework*, no processo de realização de testes, é importante que as fontes de dados sejam as mais fidedignas possíveis, refletindo problemas de fontes reais. Atualmente, utilizam-se geradores de dados, mas estes ainda não são ideais, principalmente quando se deseja avaliar algoritmos de alta precisão, de domínio geral ou escaláveis [Christen, 2012].

Adicionalmente, é importante destacar a necessidade de se avaliar mais profundamente o impacto da semântica no processo de comparação de dados [Christen, 2012]. As técnicas existentes já são robustas o suficiente? O uso da semântica é indispensável? Qual o ganho efetivo ao realizar o estudo semântico dos dados?

Considerando a qualidade dos dados, quais cenários os problemas de qualidade mais influenciam nos resultados do processo RE? Quais critérios de qualidade são mais importantes nesta análise? A qualidade é um fator determinante para a RE? [Christen, 2012].

Em [Getoor & Machanavajjhala, 2012] é destacada a possibilidade de se realizar RE, considerando dados de tipos distintos, interligando textos, vídeos e imagens referentes a uma mesma pessoa, por exemplo.

Um dos grandes gargalos para a RE é a etapa de comparação das tuplas. Neste sentido, quando considerados principalmente ambientes com grandes volumes de, precisa-se de algoritmos escaláveis, *online* e de alta qualidade. Atualmente, alguns esforços estão sendo concentrados em estratégias incrementais de RE [Gruenheid et *al*. 2014] (incluindo este trabalho) e no uso de probabilidade [Salloum et *al*., 2013] para inferir que duas tuplas representam a mesma entidade do mundo real.

Apesar de Resolução de Entidades ser um problema antigo e pesquisado há pelo menos 60 anos, ainda existem problemas em aberto que merecem ser vistos com mais atenção pela comunidade científica. É um problema que ocorre em diferentes áreas da computação e cada vez mais com o avanço na geração e manipulação de grandes volumes de dados, novos gargalos na Resolução de entidades emergem.

REFERÊNCIAS

Ackerman, M.; Dasgupta, S. *Incremental Clustering*: *The Case for Extra Clusters*. *Advances in Neural Information Processing Systems* (NIPS). Pp 307-315. *Montreal, Quebec*. 2014.

Altowim, Y., Kalashnikov, D. V., Mehrotra, S. *Progressive Approach to Relational Entity Resolution*. VLDB. V 7, issue 11, pp 999-1010. *Hangshou, China*. 2014.

Altwaijry, H., Kalashnikov, D. D., Mehrotra, S. *Query-Driven Approach to Entity Resolution*. VLDB. V 6, issue 14, pp 1846-1857. *Trento, Italy*. 2013.

Batini, C., Scannapieco, M. *Data Quality: Concepts, Methodologies and Techniques*. *Data-Centric Systems and Applications*. Springer. 2006.

Baumgartner, N. et *al*. "*Same, Same but Different" A Survey on Duplicate Detection Methods for Awareness*. *On The Move to Meaningful Internet Systems: OTM*. V 5871, pp 1050-1068. *Vilamoura*, Portugal. 2009.

Bhattacharya, I.; Getoor, L. *Entity Resolution in Graphs*. *Mining Graph Data*. *John Wiley & Sons, Inc*. 2006.

Bellahsene, Z., Bonifati, A., Rahm, E.: *Schema Matching and Mapping*. *Data-Centric Systems and Applications*. Springer. 2011.

Benjelloun, O. et *al*. *Swoosh: A Generic Approach to Entity Resolution*. VLDB *Journal*. V 18, issue 1, pp 255-276. 2009.

Berkhin, P. *A Survey of Clustering Data Mining Techniques*. *Grouping Multidimensional Data: Recent Advances in Clustering*. Pp 25 – 71. Springer Berlin Heidelberg. 2006.

Bilenko, M., Mooney, J. M. *Adaptive Duplicate Detection Using Learnable String Similarity Measures*. *9thACM SIGKDD international conference on knowledge discovery and data*. Pp 38-48. *New* York, *USA*. 2003.

Bilgic, M.; Licamele, L.; Getoor, L; Shneiderman, B. *D-Dupe: An Interactive Tool for Entity Resolution in Social Networks*. *IEEE Symposium on Visual Analytics Science and Technology*. B*altimore, USA*. 2006.

Bhattacharya, I., Licamele, L.; Getoor, L. *Query-Time Entity Resolution*. *ACM International Conference on Knowledge Discovery and Data Mining* (SIGKDD). *Philadelphia, USA*. 2006.

Bhattacharya, I., Getoor, L. *Query-Time Entity Resolution*. *Journal of Artificial Intelligence Reserche*. 2007.

Bianco, G. D. et *al*. *A Practical and Effective Sampling Selection Strategy for Large Scale Deduplication*. *IEEE Transactions on Knowledge and Data Engineering*. V 27, issue 9, pp 2305 – 2319. 2015.

Bilenko, M.; Mooney, R.J. *Adaptive Duplicate Detection Using Learnable String Similarity Measures*. *ACM International Conference on Knowledge Discovery and Data Mining* (SIGKDD). Pp 39-48. *Washington, USA*. 2003.

Bharambe, D.; Jain, S.; Jain, A. *A Survey: Detection of Duplicate Record*. *International Journal of Emerging Technology and Advanced Engineering*. 2012.

Canalle, G. K.; Lóscio, B. F.; Salgado, A. C. Uma Estratégia para Seleção de Atributos Relevantes no Processo de Resolução de Entidades. Simpósio Brasileiro de Banco de Dados. Rio de Janeiro, Brasil. 2015.

Canalle, G. K. Uma Estratégia Para Seleção De Atributos Relevantes No Processo De Resolução De Entidades. Dissertação de Mestrado. Universidade Federal de Pernambuco. Pernambuco, Brasil. 2016.

Canalle, G. K.; Lóscio, B. F.; Salgado, A. C. *A Strategy for Selecting Relevant Attributes for Entity Resolution*. *International Conference on Enterprise Information Systems* (ICEIS). V 1, pp 80-88. *Porto, Portugal*. 2017

CD. Disponível em: <http://hpi.de/naumann/projects/data-quality-and-cleansing/dude-duplicate-detection.html#c114715>. Acessado em 23/02/2017.

Charikar, M. et *al*. *Incremental Clustering and Dynamic Information Retrieval*. *Society for Industrial and Applied Mathematics*. V33, issue 6, pp 1417 – 1440. 2004.

Chaudhuri, S. et *al*.*Example-driven Design of Efficient Record Matching Queries*. *33rd International Conference on Very Large Data Bases* (VLDB). Pp 327-338. Vienna, Austria. 2007.

Christen, P., Goiser, K. *Quality And Complexity Measures for Data Linkage and Deduplication*. *Data Mining, Studies in Computational Intelligence*. V 43, pp. 127–151. Springer. 2007.

Christen, P. *Febrl – An Open Source Data Cleaning, Deduplication and Record Linkage System with a Graphical User Interface*. *International Conference on Knowledge Discovery and Data Mining* (KDD). Pp 1065-1068. *Las Vegas, USA*. 2008.

Christen, P.; Gayler, R.; Hawking, D. *Similarity-Aware Indexing for Real-Time Entity Resolution*. *18th ACM conference on Information and knowledge management* (CIKM). Pp 1565-1568. *Hong Kong, China*. 2009.

Christen, P. *A Survey of Indexing Techniques for Scalable Record Linkage and Deduplication*. *IEEE Transactions on Knowledge and Data Engineering*. V 9, issue 9, pp 1537-1555. 2011.

Christen, P. *Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection*. Springer. 2012.

Cora. Disponível em: http://www.cs.utexas.edu/users/ml/riddle/data.html. Acessado em 21/08/2017.

Cohen, W. W. *Integration of Heterogeneous Databases without Common Domains Using Queries Based on Textual Similarity*. *ACM Special Interest Group on Management of Data* (SIGMOD). Pp 201-212. *Seattle*, *USA*. 1998.

Cohen,W.; Ravikumar, P.; Fienberg, S. *A Comparison of String Metrics for Matching Names and Records*. *KDD Workshop on Data Cleaning and Object Consolidation*. V 3, pp 73–78, 2003.

Data Match. Disponível em: <http://www.dataladder.com/index.html>. Acessado em: 21/08/2017.

Dong, X.; Halevy, A.; Madhavan, J., D. *Reference Reconciliation in Complex Information Spaces*. *ACM SIGMOD International Conference on Management of Data*. Pp 85-96. *Baltimore*, *Maryland*. 2005.

Dong, X.; Naumann, F. *Data Fusion – Resolving Conflicts for Integration*. VLDB. V 2, issue 2, pp 1654-1655. 2009.

Dong, X.; Saha, B.; Srivastava, D. *Less is More: Selecting Sources Wisely for Integration*. VLDB. V 6, issue 2, pp 37-48. 2012.

Dong, X.; Srivastava, D. *Big Data Integration*. VLDB. V 6, issue 11, pp 1188-1189. 2013.

Dong, X.; Srivastava, D. *Compact Explanation of Data Fusion Decisions*. *22nd International Conference on World Wide Web* (WWW). Pp 379-390. *Rio de Janeiro*, *Brazil*. 2013.

Dong, X. De et *al*. *From Data Fusion to Knowledge Fusion*. VLDB. V 7, issue 10, pp 881-892. 2014.

Dong, X. L.; Srivastava, D. *Big Data Integration*. Morgan & Claypool. 2015.

Dorneles, C. F.; Gonçalves, R.; Mello, R. S. *Approximate Data Instance Matching: A Survey*. *Knowledge and Information Systems*. V 27, issue 1, pp 1-21. 2010.

Draisbach, U.; Naumann, F. *Dude: The Duplicate Detection Toolkit*. *VLDB: Workshop on Quality*. Singapura, 2010.

Draisbach, U.; Naumann, F. *On Choosing Thresholds for Duplicate Detection*. *18th International Conference on Information Quality*. *Arkansas, USA*. 2013.

DTM Data Scrubber. Disponível em: <http://www.sqledit.com/scr/index.html>. Acessado em: 21/08/2017.

Elfeky, M. G. et *al*. *TAILOR: A Record Linkage Tool Box*. *International Conference on Data Engineering* (ICDE). Pp 17-28. *San* Jose, USA. 2002.

Elmagarmid, A. K.; Ipeirotis, P. G.; Verykios, V. S. *Duplicate Record Detection: A Survey*. *IEEE Transactions on Knowledge and Data Engineering*. V 19, issue 1, pp 1-16. 2007.

Fahad, A. et *al*. *A Survey of Clustering Algorithms for Big Data: Taxonomy and Empirical Analysis*. *IEEE Transactions on Emerging Topics in Computing*. 2014.

Firmani, D.; Saha, B.; Srivastava, D. *Online Entity Resolution Using an Oracle*. *Journal VLDB*. 2016.

FreeDB. Disponível em: http://www.freedb.org/en/download__server_software.4.html. Acessado em: 21/08/2017.

Getoor, L.; Machanavajjhala, A. *Entity Resolution: Tutorial*. VLDB. *Istambul*. 2012.

Gruenheid, A.; Dong, X. L.; Srivastava, D. *Incremental Record Linkage*. VLDB. V 4, issue 9, pp 697-708. 2014.

Guo, S.; Dong, X.; Srivastava, D.; Zajac, R. *Record linkage with Uniqueness Constraints and Erroneous Values*. PVLDB. V 3, issue 1-2, pp 417-428. 2010.

Halevy, A., Rajaraman, A., Ordille, J. *Data Integration: The Teenage Years*. VLDB. Pp 9-16. *Seoul, Korea*. 2006.

Herzog, T., Scheuren, F., Winkler, W. *Data Quality and Record Linkage Techniques*. Springer. 2007.

Huang, J.; Ertekin, S.; Giles, S. *Efficient Name Disambiguation for Large-Scale Databases*. *10th European Conference on Principle and Practice of Knowledge Discovery in Databases* (PKDD). Pp 536-544. *Berlin, Germany*. 2006.

Ioannou, E. De el *al*. *Efficient Semantic-Aware Detection of Near Duplicate Resources*. *The Semantic Web: Research and Applications*. V 6089, pp 136-150. 2010.

Jain, A. K.; Murty, M. N; Flynn, P. J. *Data Clustering: A Review*. *ACM Computing Surveys* (CSUR). V 31, issue 3, pp 264-323. 1999.

Java. Disponível em: < https://www.java.com/pt_BR/>. Acessado em 02/08/2017.

Jin, L.; Li, C.; Mehrotra, S. *Efficient Record Linkage in Large Data Sets*. *8th International Conference on Database Systems for Advanced Applications*. *Washington, USA*. 2003.

JSON. Disponível em: <http://www.json.org/json-pt.html>. Acessado em 02/08/2017.

Kang, H. *Interactive Entity Resolution in Relational Data: A Visual Analytic Tool and Its Evaluation*. *IEEE Transactions on Visualization and Computer Graphics*. V 14, issue 5, pp 999-1014. 2008.

Kalpana, G.; Kumar, R. P.; Ravi, T. *Classifier Based Duplicate Record Elimination for Query Results from Web Databases*. *Trendz in Information Sciences & Computing*. Pp 50-53. *Chennai, India*. 2010.

Kenig, B.; Gal, A. *Efficient Entity Resolution with MFIBlocks*. VLDB. *Lyon, France*. 2009.

Kogan, J.; Nicholas, C.; Teboulle, M. *Grouping Multidimensional Data: Recent Advances in Clustering*. Springer. 2006.

Kolb, L.; Thor, A.; Rahm, E. *Dedoop: Efficient Deduplication with Hadoop*. VLDB. V5, issue 12, pp 1878-1881. 2012.

Köpcke, H.; Rahm, E. *Training Selection for Tuning Entity Matching*. *6th International Workshop on Quality in Databases and Management of Uncertain Data*. Pp 3-12. *Enschede, Netherlands*. 2008.

Köpcke, H.; Rahm, E. *Frameworks for Entity Matching: A Comparison*. *Data & Knowledge Engineering*. V 69, issue 2, Pp 197-210. 2009.

Lee, Y. De et *al*. *Journey to Data Quality*. The MIT Press. 2009.

Leitão, L.; Calado, P.; Weis, M. *Structure-based Inference of XML Similarity for Fuzzy Duplicate Detection*. *16th ACM Conference on Information and Knowledge Management* (CIKM). Pp 293-302. *Lisbon, Portugal*. 2007.

Lenzerini, M. *Ontology-Based Data Management*. *International Conference on Information and Knowledge Management* (CIKM). Pp 5-6. 2011.

Lenzerini, M. *Data Integration: A Theoretical Perspective*. *21th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database systems*. USA. 2002.

Lóscio, B. F.; Oliveira, M. I. S.; Salgado, A. C. *Towards Goal-oriented Data Integration of WoT Data Sources*. *1st French-Brazilian Spring School on Big Data and Smart Cities*. France. 2015.

Madhavan et al. *Web-scale Data Integration: You can only afford to Pay as You Go*. *Conference on Innovative Data System Research* (CIDR). 2007.

Monge, A. E. *Matching Algorithms within a Duplicate Detection System*. *IEEE Data Eng. Bull*. 2000.

Nin, J. et *al*. *On The Use of Semantic Blocking Techniques for Data Cleansing and Integration*. *11th International Database Engineering and Applications Symposium* (IDEAS). *Alberta, Canada*. 2007.

Oliveira, J. P. Detecção e Correção de Problemas de Qualidade dos Dados: Modelo, Sintaxe e Semântica. Tese De Doutoramento em Informática. Universidade do Minho. 2008.

Papadakis, G. et *al*. *Comparative Analysis of Approximate Blocking Techniques for Entity Resolution*. VLDB. V 9, issue 9, pp 684-695. 2016.

Pereira, D. A. et *al*. *A Generic Web-based Entity Resolution Framework*. *Journal of the American Society for Information Science and Technology*. V 62, issue 5, pp 919-932. 2011.

Python. Disponível em: < http://wiki.python.org.br/>. Acessado em 02/08/2017.

Ramadan, B. et *al*. *Dynamic Similarity-Aware Inverted Indexing for Real-Time Entity Resolution*. *Trends and Applications in Knowledge Discovery and Data Mining*. Pp 47-58. 2013.

Ramadan, B.; Christen, P. *Forest-Based Dynamic Sorted Neighborhood Indexing for Real-Time Entity Resolution*. *23rd ACM International Conference on Conference on Information and Knowledge Management* (CIKM). Pp 1787-1790. *Shanghai, China*. 2014.

Ramadan, B.; Christen, P. *Unsupervised Blocking Key Selection for Real-time Entity Resolution*. Advances in Knowledge Discovery and Data Mining, LNCS.CIKM'14. 2015.

Ramadan, B. et al. Dynamic Sorted Neighborhood Indexing for Real-Time Entity Resolution. *Journal of Data and Information Quality* (JDIQ) - *Challenge Papers and Regular Papers*. V 6, issue 4. 2015.

Rekatsinas, T.; Dong, X. L.; Srivastava, D. *Characterizing and Selecting Fresh Data Sources*. *ACM SIGMOD International Conference on Management of Data*. Pp 919-930. *Utah, USA*. 2014.

Saleem, M. De et *al*. *DAW: Duplicate-aware Federated Query Processing over the Web of Data*. *The Semantic Web* (ISWC). V 8218, pp 574-590. 2013.

Salloum, M. De et *al*. *Online Ordering of Overlapping Data Sources*. VLDB. *Hangzhou, China*. 2013.

Silva, R. Da. et *al*. *Measuring Quality of Similarity Functions in Approximate Data Matching*. *Journal of Informetrics*. 2007.

Su, W., Wang, J., Lochovsky, F, H. *Record Matching over Query Results from Multiple Web Databases*. *IEEE Transactions on Knowledge and Data Engineering*. V 22, issue 4, pp 578 - 589. 2010.

Tan, P.; Steinbach, M.; Kumar, V. *Introduction to Data Mining*. First Edition. Addison Wesley. 2006.

Tejada, S.; Knoblock, C.A.; Minton, S. *Learning Domain-independent String Transformation Weights for High Accuracy Object Identification*. *ACM International Conference on Knowledge Discovery and Data Mining* (SIGKDD). Pp 350-259. *Edmonton, Canada*. 2002.

Thor, A.; Rahm, E. *MOMA – A Mapping-based Object Matching System*. *Biennial Conference on Innovative Data Systems Research* (CIDR). Pp 247-258. *California, USA*. 2007.

Vesdapunt, N.; Bellare, K.; Dalvi, N. *Crowdsourcing Algorithms for Entity Resolution*. VLDB. V 7, issue 12, pp 1071–1082. 2014.

Vieira, P. K. M.; Salgado, A. C.; Lóscio, B. F. *A Query-driven and Incremental Process for Entity Resolution*. *Alberto Mendelzon International Workshop on Foundations of Data Management*. *Panama City, Panama*. 2016.

Vieira, P. K. M.; Salgado, A. C.; Lóscio, B. F. *Dynamic Indexing for Incremental Entity Resolution in Data Integration Systems*. *International Conference on Enterprise Information Systems* (ICEIS). Porto, Portugal. 2017.

Wang, J.; Li, G.; Kraska, T.; Franklin, M.J.; Feng, J. *Leveraging Transitive Relations for Crowdsourced Joins*. *ACM SIGMOD International Conference on Management of Data*. Pp 229–240. *New York, USA*. 2013.

Weis, M. et *al*. *Industry-scale Duplicate Detection*. VLDB. V 1, issue 2, pp 1253–1264. 2008.

Welch, M.; Sane, A.; Drome, A. *Fast and Accurate Incremental Entity Resolution Relative to an Entity Knowledge Base*. *21st ACM international conference on Information and knowledge management* (CIKM). Pp 2667-2670. *Hawaii, USA*. 2012.

Whang, S.; Garcia-Molina, H. *Entity Resolution with Evolving Rules*. VLDB. V 3, issue 1-2, pp 1326–1337. 2010.

Whang, S. E.; Marmaros, D.; Garcia-Molina, H. *Pay-As-You-Go Entity Resolution*. *IEEE Transactions on Knowledge and Data Engineering*. V 25, issue 5, pp 1111-1124. 2013.

Whang, S. E.; Garcia-Molina, H. *Incremental Entity Resolution on Rules and Data*. VLDB. V 23, issue 1, pp 77-102. 2014.

Widyantoro, D.H.; Ioerger, T. R.; Yen, J. *An Incremental Approach to Building a Cluster Hierarchy*. *IEEE International Conference on Data Mining* (ICDM). *Maebashi City*, *Japan*. 2002.

Young, S. et *al*. *A Fast and Stable Incremental Clustering Algorithm*. *Information Technology: New Generations* (ITNG). *Nevada*, *USA*. 2010.

Zaveri, A. De et *al*. *Quality Assessment Methodologies for Linked Open Data - A Systematic Literature Review and Conceptual Framework*. *Semantic Web Journal*. 2012.

Zhao, H.; Ramo, S. *Entity Identification for Heterogeneous Database Integration – A Multiple Classifier System Approach and Empirical Evaluation*. *Information Systems*. V 20, issue 2, pp 119-132. 2005.

Žitnik, S. et *al*. *General Context-Aware Data Matching and Merging Framework*. *Journal Informatica*. V 24, issue 1, pp 119-152. 2013.