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

Uso de Técnicas de Mineração de Dados para Identificar Estruturas Similares de Bancos de Dados

Débora Gomes dos Reis

Dissertação apresentada como requisito parcial para conclusão do
Mestrado Profissional em Computação Aplicada

Orientador
Prof. Dr. Marcelo Ladeira

Brasília
2018

Ficha catalográfica elaborada automaticamente,
com os dados fornecidos pelo(a) autor(a)

GD287u

Gomes dos Reis, Débora
Uso de Técnicas de Mineração de Dados para Identificar Estruturas Similares de Bancos de Dados / Débora Gomes dos Reis; orientador Marcelo Ladeira. -- Brasília, 2018.
90 p.

Dissertação (Mestrado - Mestrado Profissional em Computação Aplicada) -- Universidade de Brasília, 2018.

1. banco de dados. 2. similaridade de esquema. 3. metadados. 4. integração de dados. 5. deduplicação de esquema. I. Ladeira, Marcelo, orient. II. Título.

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

Uso de Técnicas de Mineração de Dados para Identificar Estruturas Similares de Bancos de Dados

Débora Gomes dos Reis

Dissertação apresentada como requisito parcial para conclusão do
Mestrado Profissional em Computação Aplicada

Prof. Dr. Marcelo Ladeira (Orientador)
CIC/UnB

Prof.a Dr.a Maristela Terto de Holanda
CIC/UnB

Dr. Marcio de Carvalho Victorino
FCI/UnB

Prof.a Dr.a Ana Carolina Brito de Almeida
IME/UERJ

Prof.a Dr.a Aletéia Patrícia Favacho de Araújo von Paumgartten
Coordenadora do Programa de Pós-graduação em Computação Aplicada

Brasília, 30 de julho de 2018

Dedicatória

Aos que buscam descobrir soluções mais inteligentes e eficientes com uso da tecnologia.

Agradecimentos

Em primeiro lugar, gostaria de agradecer à UnB por ter me dado a oportunidade de fazer parte do mestrado profissional do Programa de Pós-Graduação em Computação Aplicada (PPCA). Uma experiência que me trouxe grande aprendizado para toda a vida. Agradeço aos professores do PPCA, com quem aprendi bastante. Em especial, gostaria de agradecer ao Prof. Marcelo Ladeira que me orientou em meio a diversos obstáculos, sem ele não teria conseguido ter forças e o conhecimento necessário para continuar a pesquisa.

Também agradeço ao Prof. Rommel Carvalho pela orientação no início da pesquisa e ao Prof. Marcio Victorino pelas dicas no decorrer do trabalho. Não poderia deixar de agradecer também à Prof. Maristela e ao amigo Waldeyr por disponibilizarem acesso ao laboratório, onde pude realizar algumas execuções. Obrigada aos também alunos do PPCA que partilharam de conhecimentos e experiências.

Agradeço imensamente aos amigos do Ministério do Planejamento e do Ministério do Desenvolvimento Social - Cícero Padilha, Irna, Washington, Escobar, Caio Nakashima, Davi, Kovags, Gustavo Marques, Paulo Correia, Renata, Raul, Rozeane, Sandro, Thiago, Emerson, Matheus, Diogo, Bruno Costa, André Castro, Wesley Lyra, Cassiano Alves, toda a equipe da DELOG/SEGES - que me apoiaram em momentos diferentes durante o mestrado. Obrigada pelas dicas, pela paciência e pelo apoio durante o período do mestrado. Sei que muitos tiveram que cobrir as minhas ausências e agradeço por isso.

Por fim, e não menos importante, agradeço a compreensão dos meus familiares e amigos decorrente dos vários períodos de ausência. Em especial, agradeço ao apoio incondicional do meu querido e amado esposo Diego Henrique.

Obrigada à todos que fizeram parte desta conquista.

Resumo

Com a crescente quantidade e tamanho das bases de dados, é cada vez mais difícil identificar as similares dentre grandes bases de dados armazenadas em diferentes Sistemas Gerenciadores de Banco de Dados (SGBDs). Por isso, é proposto o uso de técnicas de mineração de dados para identificar as estruturas similares de bancos de dados relacionais através da comparação de seus metadados. A quantidade de metadados é proporcional à quantidade de objetos de bancos de dados e as possibilidades de combinações para a comparação é quadrática em relação ao número de esquemas analisados. Em busca da técnica mais eficiente, foi proposta uma abordagem que calcula a similaridade de esquemas avaliando a distância de todos os esquemas para um esquema, considerado como origem. Obviamente, esquemas mais próximos são mais similares que esquemas mais distantes. Esta abordagem foi comparada com outras duas abordagens. A primeira compara todos os esquemas contra todos os outros esquemas, exceto a comparação inversa. A segunda abordagem compara os grupos de esquemas com tamanhos parecidos. Para validar as abordagens, é realizado um experimento com 354 esquemas reais de tamanhos pequeno a grande que variam de 2 a 20 mil objetos de banco de dados, advindos de 5 SGBDs de um Ministério do Brasil. Eles somam juntos mais de 26 mil tabelas e 238 mil colunas. Os metadados extraídos foram tratados e comparados em pares de esquemas. A similaridade das variáveis textuais é mensurada usando a distância cosseno e a das numéricas usando a distância euclidiana. A técnica de clusterização hierárquica é aplicada para facilitar a visualização dos esquemas mais similares. Como resultado, a abordagem proposta se mostrou a mais eficiente, identificou os esquemas com estruturas mais similares em menos de 2 minutos. Os metadados extraídos foram usados para criar o repositório de metadados da organização. Como isso, descobriu-se bancos de dados duplicados que, ao serem descontinuados, geraram economia de 10% no custeio e liberaram recursos de infraestrutura. Esta abordagem é flexível porque suporta uma variedade de SGBDs, de quantidade e de tamanhos de bases de dados.

Palavras-chave: banco de dados, similaridade de esquema, metadados, integração de dados, deduplicação de esquema

Abstract

With the expanding diversity of database technologies and database sizes, it is becoming increasingly hard to identify similar relational databases among many large databases stored in different Database Management Systems (DBMS). Therefore, we propose to use data mining techniques to automatically identify similar structures of relational databases by comparing their metadata. The amount of metadata is proportional to the size of the schema structure. The possibilities of combinations for comparison is quadratic in relation to the number of schemas analyzed. Looking for the most efficient technique, we propose to calculate the schema similarity evaluating a distance of all the schemas to just one schema, which is a start point. Obviously schemas with close distances are more similar than schemas with bigger distances. We compare this proposal against two other approaches. The first approach compares all schemas against all another schemas except for its inverse comparison. The second approach compares schemas in a group of schemas with similar sizes. To validate our proposal, an experiment is performed with 354 real schemas ranging in sizes from 2 to 20 thousand metadata, totaling together more than 26 thousand tables and 238 thousand columns. Those schemas came from 5 different DBMS. The metadata extracted is transformed and formatted for comparing pairs of a schema. The textual features are compared using Cosine Distance and numerical features are compared using Euclidean Distance. Then, the hierarchical cluster technique is used to facilitate the visualization of the schema that most closely resembled one another. Results showed that, our was the most efficient because it compared all schema and identified the most similar schema by its structure in less than 2 minutes. The extracted metadata was used to create the first version of the metadata repository and an initial version of a data catalog. Using this procedure, duplicated schemas were discovered and then discontinued, resulting in a cost savings of 10% of cost savings, while freeing up infrastructure resources. This solution is flexible, it supports a variety of number of schema, of schema sizes, and DBMS.

Keywords: database, schema matching, metadata, data integration, schema deduplication

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

BD Banco de Dados.

CRISP-DM *CRoss Industry Standard Process for Data Mining.*

CSV *Comma Separated Value.*

EGD Estratégia de Governança Digital.

ETL *Extract, Transform, and Load.*

FK *Foreign Key.*

ICMLA *International Conference on Machine Learning and Applications.*

IP *Internet Protocol.*

PDA Plano de Dados Abertos.

SGBD Sistema Gerenciador de Banco de Dados.

SISP Sistema de Administração dos Recursos de Tecnologia da Informação.

SQL *Structured Query Language.*

SRL *Statistical Relational Learning.*

TI Tecnologia da Informação.

Capítulo 1

Introdução

Este capítulo apresenta o contexto da pesquisa, a definição do problema, a sua justificativa, o objetivo geral, os objetivos específicos, a contribuição e os resultados esperados.

1.1 Contexto da Pesquisa

Grandes organizações empresariais e organizações governamentais usualmente armazenam um grande número de grandes bases de dados e enfrentam várias situações que requerem a junção delas. Estas situações incluem quando as empresas são fundidas ou incorporadas por aquisição, quando departamentos separados da mesma instituição são combinados, quando duas ou mais instituições governamentais são unidas por reestruturação da estrutura governamental. Como exemplo, recentemente o Governo do Brasil fez uma reforma nos ministérios, alterando sua estrutura organizacional através da fusão de ministérios para diminuir a quantidade de ministérios [1]. Neste cenário, foi escolhido um caso de um Ministério que se originou da junção de ministérios antigos, sofreu fusão com outros ministérios, herdando centenas de bases de dados armazenadas em diferentes tecnologias que são responsáveis por manter um expressivo volume de dados de programas sociais, políticas públicas, dentre outros dados. Nestas situações, para economizar recursos é preciso identificar os esquemas similares, como forma de evitar a duplicidade e facilitar a integração de bases de dados.

Facilitar a identificação de bancos de dados similares em um tempo factível pode fazer a diferença para: revisar os bancos de dados, priorizar projetos de integração de dados, reduzir a duplicação de esquemas, suportar a decisão de arquivar ou integrar bases de dados como uma maneira de economizar recursos, melhorar a qualidade de dados para gerar análises de dados mais assertivas. Acima de tudo, a identificação de esquemas similares pode aprimorar o conhecimento dos dados que existe na organização e facilitar a análise de dados.

1.2 Definição do Problema

Existem vários motivos para uma grande quantidade bases de dados similares, que incluem o uso de diferentes modelos de armazenamento de dados, o massivo volume de dados mantido pela organização, a mudança da necessidade dos dados no decorrer do tempo e a distribuição dos dados em diferentes sistemas e tecnologias [2]. Um banco de dados relacional é organizado em esquema, tabelas e colunas, de forma que cada esquema possui tabelas, cada tabela armazena registros e possui uma ou mais colunas e as tabelas podem ter relacionamentos entre elas através destas colunas [3]. Cada coluna contém um valor de um tipo de dado definido. Os bancos de dados relacionais são armazenados e gerenciados por um Sistema Gerenciador de Banco de Dados (SGBD).

Dentro de um SGBD podem existir vários bancos e esquemas, cada um com centenas de tabelas e até milhares de colunas. Os esquemas podem ter estruturas similares, desde que não tenham o mesmo nome. Assim, dentro de um SGBD podem existir dois esquemas de dados com nomes diferentes, mas exatamente as mesmas tabelas, colunas e registros. O desafio consiste em encontrar quais são as estruturas comuns quando se tem muitos esquemas com nomes variados. Este problema se agrava com a existência de diferentes SGBDs e mais de um servidor de banco de dados, a qual potencializa a superposição de bases de dados. Assim, ao comparar bases de dados entre os diferentes SGBDs e servidores de banco de dados é possível encontrar bases de dados com nomes iguais, mas estrutura de dados diferentes, confundindo a identificação de banco de dados similares.

Para a área de Tecnologia da Informação (TI), a comparação manual de bases de dados é tão trabalhosa e demorada que muitas vezes nem chega a ser executada. Muitas corporações preferem manter duas estruturas similares funcionando, mesmo que isto signifique mais custos de manutenção e problema na qualidade de dados. Esta situação piora em casos com grande quantidade de bases de dados que possuem tamanho expressivo. Manualmente escanear centenas de bancos de dados até encontrar os similares é uma atividade propensa a erros e pode demorar meses até ser finalizada. Uma busca interna recente num dos ministérios do Brasil indicou mais de 300 bases de dados relacionais ativas em ambiente de produção. Este Ministério já possui mais de 70 *Terabytes* de dados divididos entre cerca de 15 *Terabytes* de dados transacionais e cerca de 55 *Terabytes* de dados históricos. Não conseguir identificar os esquemas similares gera uma série de problemas, como por exemplo:

1. diferentes versões do mesmo dado em cada coordenação do órgão, causando diferentes níveis de qualidade de dados;

2. equipes espalhadas usando versões diferentes da mesma base;

3. complexidade para integrar dados e suportar relatórios gerenciais envolvendo diferentes áreas do órgão;

4. dificuldade em mapear os dados existentes, uma etapa do processo de abertura de dados abertos exigida pela política estratégica de TI do Governo Federal;

5. dificuldade em conferir previamente se já existe uma estrutura semelhante de banco de dados antes da criação de uma nova base de dados;

6. alto custo com a infraestrutura que sustenta os bancos de dados devido à repetição de bases de dados em diversos SGBDs;

7. alto custo de recursos humanos para serviços de profissionais especializados em diferentes SGBDs para realizar a administração dos bancos de dados, os serviços de monitoramento e de *backup*. Uma contabilidade interna do Ministério [1] estima o custo dos serviços de manutenção de bancos de dados relacionais em R$828.784,16 por ano.

1.3 Justificativa

A semelhança entre dois bancos de dados é definida como uma comparação entre os seus pares de metadados de objetos de banco de dados. O número de possíveis combinações de estrutura de bancos de dados é quadrático com o número de objetos de banco de dados analisados, exigindo muito processamento. Esta complexidade traz a necessidade em focar em técnicas mais eficientes para encontrar estruturas similares de bancos de dados relacionais.

O Ministério precisa saber quais são as bases similares para decidir se estas serão arquivadas ou integradas, como forma de economizar recursos, melhorar a qualidade de dados e gerar análises de dados mais assertivas. Acima de tudo, o Ministério necessita mapear suas bases de dados para melhorar o gerenciamento dos seus dados. Com isso, contribuir para *Mapear os Dados sob a responsabilidade do ministério* que consta no Plano de Dados Abertos (PDA) do Ministério de vigência de 2017 a 2019 [4]. Secundariamente, facilitar o alcance dos objetivos estratégicos de *Fomentar a disponibilização e o uso de dados abertos*, *Melhorar a governança e a gestão por meio do uso da tecnologia* e de *Compartilhar e integrar dados, processos, sistemas, serviços e infraestrutura* constantes na Estratégia de Governança Digital (EGD) de vigência de 2016 a 2019 do Governo Federal[5]. Dessa forma, a escolha deste objeto de estudo vem da necessidade da instituição em

[1] A estimativa de custo foi realizada em janeiro de 2017 pela equipe de TI do ministério.

organizar as bases de dados relacionais, diminuir estes problemas que existem hoje, além de facilitar o alcance dos seus objetivos institucionais.

Devido à expressiva quantidade de registros dos esquemas analisados, que vão de 290 milhões a 12 bilhões de registros, foram utilizados apenas os metadados dos objetos de banco de dados. Assim, o conjunto de dados a ser abordado é composto pelos metadados dos objetos de banco de dados dos bancos de dados relacionais gerenciados pela área de TI apenas do ambiente de produção, aqueles que representam os bancos de dados de sistemas estruturantes do ministério. Dessa forma, não serão considerados neste momento as bases de dados que não estão sob a gerência da TI, as bases de dados históricas, *data warehouse*, bases de dados de ambiente de desenvolvimento e de homologação.

1.4 Objetivo Geral

O objetivo geral deste trabalho é identificar as estruturas similares de bancos de dados relacionais a partir dos metadados dos seus objetos de banco de dados e usar os metadados extraídos para criar o repositório de metadados com atualização periódica.

1.5 Objetivos Específicos

Para atingir o objetivo geral, foram definidos os seguintes objetivos específicos:

- Construir o repositório de metadados capaz de reunir os metadados dos objetos de banco de dados, pesquisar os esquemas de dados relacionais existentes. Criar também procedimento de manutenção e atualização periódica do repositório de metadados;
- Aplicar técnicas de mineração de dados nos metadados extraídos dos objetos de banco de dados para identificar estruturas similares de dados relacionais, reduzindo a sobrecarga de analisar manualmente todos os conjuntos de dados;
- Utilizar técnicas de visualização de dados para mostrar os esquemas com maior similaridade de metadados de objetos de bancos de dados.

1.6 Contribuição

Este trabalho traz contribuições científicas, tecnológicas e de inovações. Como contribuição científica, apresenta um estudo de um caso real de criação de um banco de metadados de objetos de bancos de dados e propõe o cálculo de distância em relação a uma origem como ferramenta para auxiliar a identificação de esquemas similares de bancos de dados.

Essa abordagem permite lidar com o problema de cálculo de similaridade (em geral com complexidade quadrática) com uma abordagem de complexidade linear. Como contribuição tecnológica, propõe o primeiro projeto que faz uso de técnicas de *Data Mining* na área de TI do Ministério. Como contribuição de inovação, semiautomatiza o processo manual de identificação de esquemas repetidos em bases de dados do Ministério. Este estudo também pode favorecer outras áreas do Ministério que ainda realizam o gerenciamento local de seus bancos de dados, pode beneficiar outros ministérios e também empresas privadas que armazenam grande quantidade de esquemas de bancos de dados.

1.7 Resultados Esperados

A partir deste trabalho, espera-se como resultados:

1. disponibilizar a primeira versão do repositório de metadados de bancos de dados relacionais com atualização periódica. Um repositório capaz de mostrar dados sumarizados e detalhados, realizar busca de metadados, realizar filtros na busca de metadados técnicos;
2. contribuir para o conhecimento dos dados através da apresentação da análise exploratória e das descobertas sobre os metadados de objetos de bancos de dados mantidos pelo ministério;
3. sugerir uma norma para evitar a criação de bancos de dados relacionais duplicados ao exigir a busca prévia no repositório de metadados de objetos de bancos de dados por bancos relacionais similares;
4. possibilitar a revisão das bases de dados relacionais a partir da identificação das bases similares. Com estas informações será possível priorizar projetos de integração de bases de dados e de diminuição da duplicação de bases de dados;
5. facilitar o gerenciamento das bases de dados relacionais a partir do repositório de metadados de objetos de bancos de dados; e
6. contribuir para alcançar o objetivo estratégico de *Mapear os dados sob a responsabilidade do ministério* conforme consta no PDA.

Este documento foi dividido em 7 capítulos e o restante deste trabalho está organizado conforme é descrito a seguir. O Capítulo 2 apresenta a fundamentação teórica. O Capítulo 3 descreve o estado da arte e os trabalhos correlatos a esta pesquisa. O Capítulo 4 apresenta a metodologia e detalha a solução proposta. No Capítulo 5 são descritos os experimentos realizados e no Capítulo 6 são apresentados os resultados obtidos com

a pesquisa. Por fim, o Capítulo 7 apresenta as conclusões e dá o direcionamento para os trabalhos futuros. O Apêndice A contém um exemplo do *Structured Query Language* (SQL) para extração dos metadados de objetos de banco de dados para cada SGBD. No Apêndice B é possível visualizar por completo o dendrograma resultante do experimento realizado e o Apêndice C contém a identificação do artigo publicado em 2017.

Capítulo 2

Fundamentos

Este capítulo apresenta os conceitos básicos sobre metadados, comparação de esquemas, mineração de dados e o modelo de referência *CRoss Industry Standard Process for Data Mining (CRISP-DM).*

2.1 Metadados

Os metadados representam qualquer informação necessária na tecnologia da informação para analisar, desenvolver, implementar, usar, gerenciar, buscar e entender os dados [6]. Os metadados são obtidos através da extração das informações sobre os dados, mas não os valores dos dados em si. Outra forma de obter metadados é através da leitura do repositório de metadados, se existir. Vaduva and Vetterli [6] explicam que o repositório de metadados é o mais importante passo para o gerenciamento integrado de metadados. Eles explicam ainda que um repositório de metadados gerencia todos os metadados num repositório central. Quando existe um vasto número de bases de dados, é ideal que se tenha um repositório central de metadados para as bases de dados relacionais.

De acordo com Mosley et al. [7], os metadados estabelecem o contexto dos dados. Eles explicam que através dos metadados é possível identificar e reduzir os processos, os dados redundantes, o retrabalho e o uso de dados desatualizados ou incorretos. Ainda segundo Mosley et al. [7], os metadados são classificados em 4 tipos principais, que são:

1. Metadados de negócio: contém nomes de negócio e definições conceituais da própria área que representam a perspectiva do negócio, como suas regras de negócio, por exemplo.

2. Metadados técnicos: são dados providos por técnicos e contém informações sobre seus sistemas e bancos de dados, contém, dentre outras informações, os detalhes do desenho físico dos bancos de dados, que incluem, por exemplo, os tipos dos dados,

o tamanho, o domínio, a origem, o uso de cada coluna e a estrutura de chaves e índices de cada tabela.

3. Metadados de processos: contém dados que definem e descrevem as características dos processos, programas, papéis, ferramentas e incluem, por exemplo, as ordem das atividades e os responsáveis.

4. Metadados da administração de dados: contém dados do monitoramento dos dados e inclui, por exemplo, informações sobre os donos dos dados, sobre a política de compartilhamento dos dados, usuários dos dados e sobre os responsável pela regulamentação da governança dos dados.

Nesta pesquisa, foi utilizado especificamente o metadado do tipo técnico porque este estudo tem o objetivo de comparar a estrutura dos esquemas de bancos de dados relacionais, comparação que utilizará os metadados dos objetos de bancos de dados compostos por detalhes físicos dos bancos de dados. A forma de comparação é detalhada na seção a seguir.

2.2 Comparação de Esquemas

A similaridade entre duas bases de dados é definida como a comparação entre seus pares de metadados técnicos originados dos objetos de banco de dados. O número de possíveis combinações é quadrática com o número de objetos de banco de dados analisados. O que traz a necessidade de usar técnicas mais eficientes para encontrar estruturas similares de bancos de dados relacionais.

De acordo com Christen [2], as razões para a quantidade e variedade de dados similares do nível mais granular como um registro até uma base de dados inteira inclui o uso de diferentes modelos de armazenamento de dados, o massivo volume de dados mantidos pelas organizações, mudando a necessidade dos dados com o tempo e a distribuição dos dados armazenados em diferentes sistemas. Ele explica que dados similares podem ser replicados e redundantes em múltiplas localidades e diferentes formatos. Para ele, a detecção de esquemas similares identifica tabelas e estruturas contendo dados que correspondem ao mesmo tipo de informação [2].

Rahm and Bernstein [8] classificaram as abordagens de comparação de esquemas primeiro pela realização de comparações individuais e depois pela comparação de uma combinação múltipla de critérios de comparações. Segundo eles, para classificação individual os critérios podem ser baseado em instância ou em esquema, a nível de elemento ou de estrutura, que analisam critérios linguísticos ou baseado em limitações. Uma ilustração desta abordagem pode ser visualizada na Figura 2.1, adaptada de Rahm and Bernstein

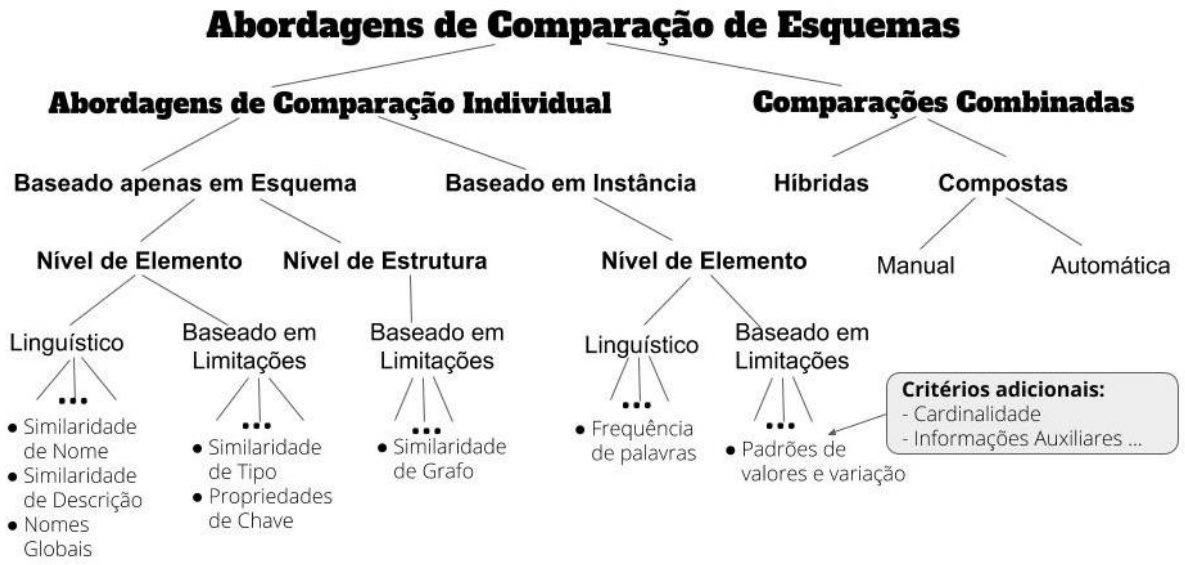


Figura 2.1: Abordagens de Comparação de Esquemas.

[8]. Desta abordagem, a comparação dos esquemas utilizará a abordagem *Baseada apenas em Esquema*, pois representa as informações existentes nos metadados dos objetos de banco de dados. Não foi considerado o comparativo baseado em instância por se tratar da comparação do dado em si, o que tornaria o volume de dados analisado muito maior, necessitando de mais recursos e inviabilizando a realização deste projeto. Também foi utilizada a comparação combinada composta do tipo automática. Dessa forma, foi descartada e não utilizada a comparação manual.

2.3 Mineração de Dados

De acordo com Tan et al. [9], a mineração de dados ou *Data Mining* é o processo de descobrir novos padrões nos dados de forma automática ou semiautomática, que possuam significados úteis. Existem diversas tarefas para a mineração de dados, a classificação, a visualização e a clusterização são exemplos destas tarefas. De acordo com Friedman et al. [10], a classificação é uma tarefa que requer a construção de um classificador, que é uma função que atribui um rótulo de classe às instâncias descritas por um conjunto de atributos. A clusterização consiste na análise das descrições dos conjuntos de dados com o objetivo de descobrir relações para indicar quais dados são similares entre si, e com isso, oferecer um modelo de agrupamento de dados de forma que a similaridade entre os dados de um grupo seja máxima e que a similaridade entre os diferentes grupos seja mínima [11]. Este estudo utiliza e aplica as tarefas de clusterização e de visualização de dados.

Para lidar com o relacionamento entre os dados, Getoor and Taskar [12] explicam que existe uma subárea de mineração de dados chamada de mineração de dados relacionais

que busca padrões em banco de dados relacional onde os dados residem em várias tabelas. Porém, de acordo com Nickel et al. [13], os relacionamentos são uma informação valiosa somente quando se procura a previsão de relacionamentos faltantes, a previsão de propriedades de nós e tarefas como análise de redes sociais. Para Khosravi and Bina [14], o relacionamento com a Aprendizagem Relacional Estatística ou *Statistical Relational Learning (SRL)* geralmente não é escalável e muito ineficiente para grandes conjuntos de dados, o que leva a dificuldades de escalabilidade e eficiência na aprendizagem estrutural. Por essa razão, optou-se por não aplicar a mineração de dados relacionais ou SRL. Em vez disso, escolheu-se realizar uma abordagem proposicional relacionada à comparação de esquemas baseada na agregação de metadados de vários bancos de dados relacionais, aceitando perder a informação do relacionamento entre as tabelas.

Os algoritmos que executam as tarefas de mineração de dados podem ser executados de forma paralela ou sequencial. Para Tsai et al. [15], o método paralelo é um conceito baseado na divisão de um grande problema em pequenos e cada um deles é executado individualmente por um único processador, de forma que estes processos são realizados simultaneamente de forma paralela. Zeng et al. [16] explica que os algoritmos de mineração de dados paralelos são comumente usados quando os dados são espalhados em muitos locais distribuídos fisicamente ou quando a computação é paralelizada em muitos nós de computadores ou núcleos de processador.

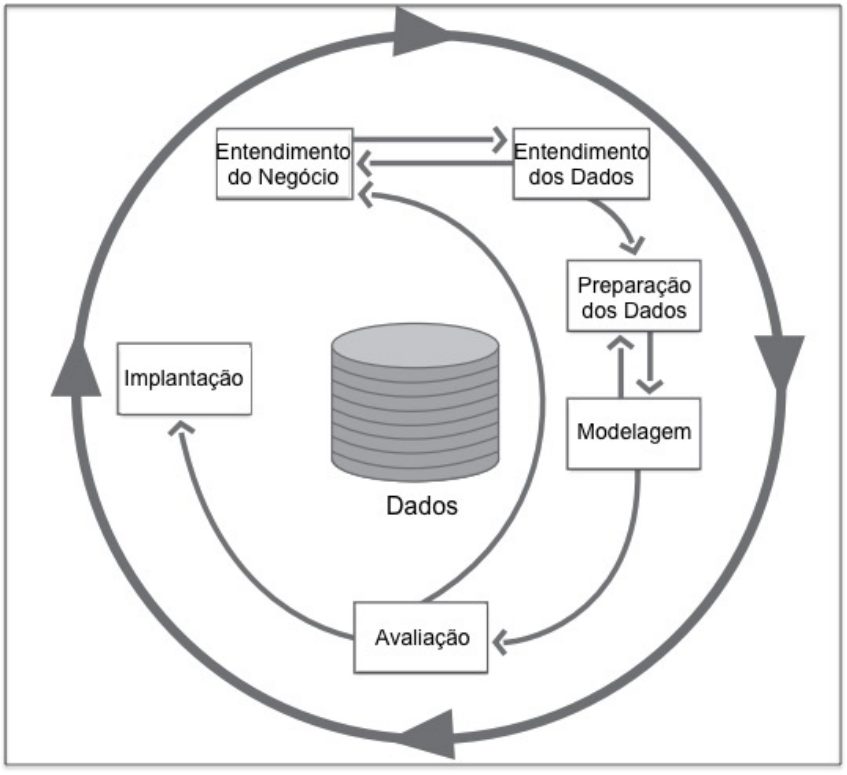


Figura 2.2: Fases do CRISP-DM.

2.4 CRISP-DM

O *CRoss Industry Standard Process for Data Mining* (CRISP-DM) é descrito na forma de um modelo de processos hierárquico, consiste em conjuntos de fases detalhadas em níveis de abstração do genérico ao mais específico, guiando o processo de mineração de dados [17]. As fases do CRISP-DM são: entendimento do negócio, entendimento dos dados, preparação dos dados, modelagem, avaliação e implantação, conforme visualizado na Figura 2.2, adaptada de Chapman et al. [17]. Nesta figura, as setas internas indicam as dependências e interações entre as fases, enquanto o círculo externo define o processo como cíclico.

Tabela 2.1: Fases e Atividades do CRISP-DM.

Fase	**Atividades**
1. Entendimento do Negócio	1.1 Determinação dos Objetivos de Negócio 1.2 Avaliação da Situação 1.3 Determinação dos Objetivos da Mineração 1.4 Produção do Plano do Projeto
2. Entendimento dos Dados	2.1 Coleta dos Dados 2.2 Descrição dos Dados 2.3 Exploração dos Dados 2.4 Verificação dos Dados
3. Preparação dos Dados	3.1 Seleção dos Dados 3.2 Limpeza dos Dados 3.3 Construção dos Dados 3.4 Integração dos Dados 3.5 Formatação dos Dados
4. Modelagem	4.1 Seleção da Técnica de Modelagem 4.2 Geração dos Cenários de Teste 4.3 Criação do Modelo 4.4 Validação do Modelo
5. Avaliação	5.1 Avaliação dos Resultados 5.2 Revisão do Processo 5.3 Determinação dos Próximos Passos
6. Implementação	6.1 Planeja a Implementação 6.2 Planeja o Monitoramento e Manutenção 6.3 Produção do Relatório Final 6.4 Revisão do Projeto

Cada fase do CRISP-DM é composta por uma série de atividades. As atividades do CRISP-DM, por fase, são listadas na Tabela 2.1 [17].

A revisão do estado da arte é apresentada no próximo capítulo.

Capítulo 3

Revisão do Estado da Arte

Este capítulo apresenta a revisão do estado da arte das soluções encontradas que realizam a comparação de esquemas de bancos de dados. Para encontrar os trabalhos relacionados, foram realizadas buscas de Janeiro de 2017 a Maio de 2018 no Web of Science [18], Google Acadêmico [19] e Google [20]. Termos pesquisados: *metadata, data mining, machine learning, unsupervised learning, supervised learning, schema, database integration, deduplication, schema deduplication, duplicate detection, database quality, data quality, data cleaning, duplicate data, unused data, data integration, metadata mining, data catalog, schema similarity, data matching,* e *schema matching*.

Foram encontradas diversas soluções e ferramentas para a comparação de esquemas de banco de dados. Elas foram divididas entre soluções de pesquisas acadêmicas e ferramentas de mercado. As soluções de pesquisas acadêmicas são decorrentes de pesquisa, encontradas através da literatura e artigos científicos publicados. As ferramentas de mercado foram encontradas através de sites de empresas e de comunidades de *software*.

3.1 Soluções de Pesquisas Acadêmicas

Existem ferramentas que fazem a comparação de esquemas de bancos de dados relacionais, como a COMA [21]. A COMA, como mostra na Figura 3.1, faz o comparativo entre pares de esquemas, comparando cada tabela e cada coluna da estrutura de ambos os esquemas. Neste caso, os mais similares são marcados em verde e a cor da ligação muda para amarelo ou vermelho à medida que aumenta as diferenças entre os esquemas.

Além da COMA, existem outras como a CUPID [22], ARTEMIS [23], SEMINT [24], GeRoMe [25] e SMART [26]. Elas foram avaliadas quanto às funcionalidades de comparar diferentes tipos de informação dos bancos relacionais como nomes, instância e estrutura. Também são avaliadas as métricas de similaridade, se faz tratamento de ontologia e se faz uso de alguma técnica automática ou semiautomática para a classificação, como a

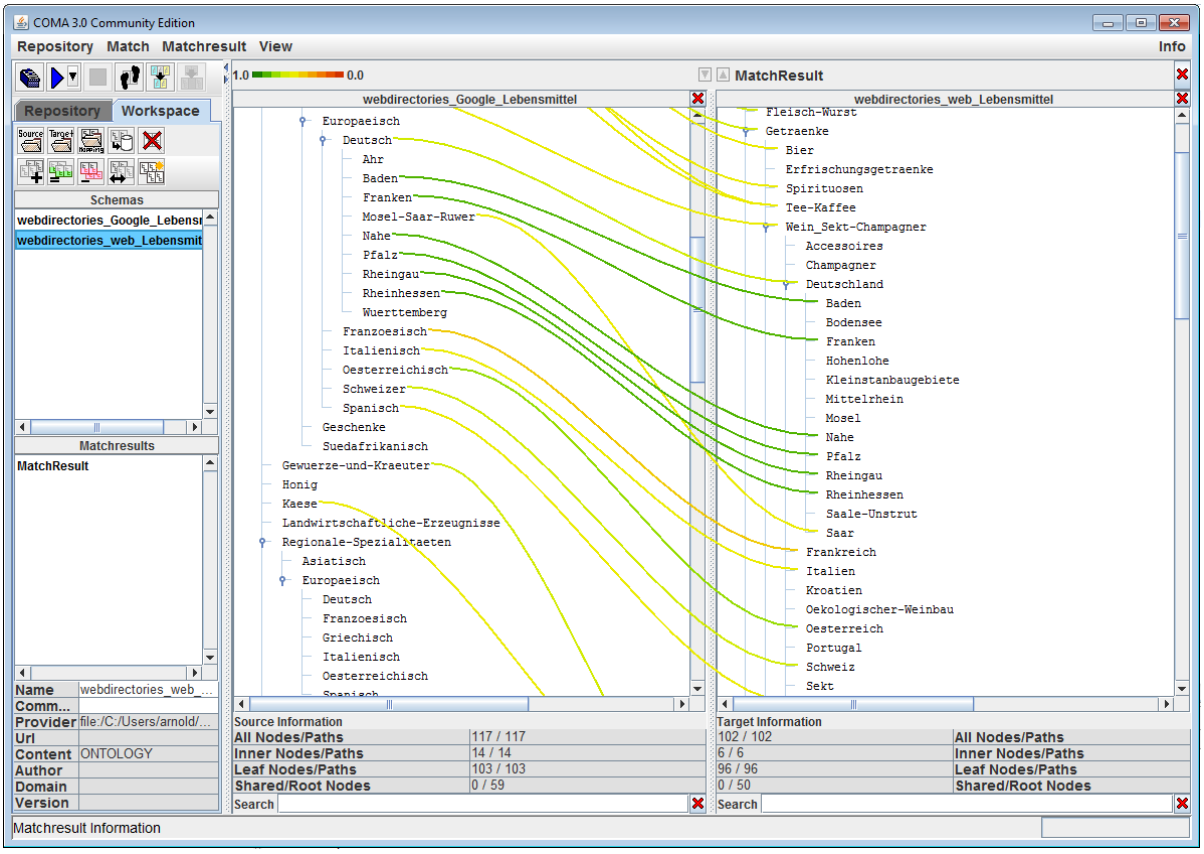


Figura 3.1: Exemplo da Ferramenta COMA.

mineração de dados. O quadro comparativo destas ferramentas é visualizado na Tabela 3.1. Dentre as ferramentas analisadas, o COMA [21], CUPID [22] e ARTEMIS [23] tem funcionalidades para comparar nomes, esquema e ontologia, se mostrando as alternativas mais completas. Porém, elas não fazem o comparativo de forma automática, o que torna sua utilização inviável face ao número elevado de estruturas de dados relacionais existentes no Ministério. Por isso, optou-se por não utilizar estas ferramentas. Por outro lado, as ferramentas SEMINT e SMART fazem uso de técnicas de mineração de dados que automatizam a classificação. Porém, a SEMINT e SMART avaliam apenas as instâncias e não as estruturas. Utilizar as instâncias para a comparação de esquemas demandam uma grande capacidade computacional para comparar grandes esquemas de bancos de dados, e, por isso, as ferramentas que utilizam as instâncias não foram utilizadas. Isto porque este estudo tem a necessidade de comparar centenas de esquemas e a maioria dos esquemas analisados nesta pesquisa possuem de milhões a bilhões de registros. Devido ao alto volume de dados, optou-se por usar e extrair apenas os metadados técnicos que compõem os detalhes físicos dos objetos de bancos de dados.

Além das ferramentas acadêmicas, foram encontrados outros dois trabalhos na literatura, propostos por Bozovic and Vassalos [27] e por Dal Bianco et al. [28], são os que mais

Tabela 3.1: Comparativo de Ferramentas Acadêmicas.

-		COMA	CUPID	ARTEMIS	SEMINT	GeRoMe	SMART
Baseado em		Grafo	Modelo	Modelo	Atributos	Atributos	Grafo
Cardinalidade		1:1	1:1 e n:1	1:1	1:1	1:1	n:m
Compara	**Nomes**	Sim	Sim	Sim	-	Sim	-
	Instância	-	-	-	Sim	-	Sim
	Estrutura	Sim	Sim	Sim	-	Sim	-
Ontologias		Sim	Sim	Sim	-	-	-
Compara por		Similaridade	Subárvores	Vizinhos	Valores	Valores, Relacionamentos	Relacionamentos
Mineração de Dados		-	-	-	Redes Neurais	-	Clusterização

se parecem com esta pesquisa. Bozovic and Vassalos [27] descreveu um novo sistema para comparar esquemas de bases relacionais chamado ASID. Este sistema usa a técnica de classificação para executar a comparação sem precisar de base de treinamento prévia. Eles fizeram um experimento e avaliaram a precisão da comparação de esquemas de tamanho de pequeno a médio. Para realizar a comparação de esquemas, sua abordagem exigiu o dicionário de dados completo. Como não existe o dicionário de dados completo de todas as bases analisadas, considerando o tempo e a dedicação para criar todos os dicionários faltantes, optou-se por não utilizar este sistema e buscar uma alternativa mais flexível que pudesse realizar a classificação com os dados existentes.

Com foco em classificação, Dal Bianco et al. [28] propuseram uma estratégia de seleção de amostra em dois estágios que seleciona um conjunto reduzido de pares para realizar o processo de deduplicação. Eles selecionam os pares mais representativos e fazem uma seleção incremental para remover as redundâncias dos subconjuntos de dados para produzir um conjunto cada vez mais reduzido de dados de treino. O primeiro estágio é focado na estrutura do esquema e o segundo estágio é focado na instância. O experimento deles mostrou efetividade para identificar quais eram os pares mais ambíguos para pequenos bancos de dados. Dessa forma, pretende-se seguir este caminho neste estudo para realizar a comparação entre pares. Porém, como a solução deles comparou pouca quantidade de esquemas pequenas e traz a incerteza sobre a escalabilidade para lidar com grande volume de esquemas de grandes bases de dados, não é uma solução que atende às necessidades deste estudo.

Ao não encontrar soluções se mostraram completas para o problema proposto, continuou-se a pesquisa por ferramentas de mercado que executam a comparação da estrutura dos esquemas.

3.2 Ferramentas de Mercado

Foram encontradas diversas ferramentas de mercado que comparam pares que estruturas de esquemas, como pode ser visualizado na Tabela 3.2. Seus preços variam, algumas são

gratuitas e outras cobram de acordo com a quantidade de usuários ou de esquemas analisados. Além do preço, foram analisadas outras características, como a compatibilidade com diversos SGBDs, se a ferramenta é capaz de analisar sinônimos e se a ferramenta é capaz de realizar a comparação de esquemas de forma automática. Dentre as 15 ferramentas analisadas, apenas 4 eram compatíveis com uma variedade de SGBDs, apenas 2 delas analisavam sinônimos e nenhuma delas executava as comparações de diversos esquemas de forma automática.

Tabela 3.2: Tabela Comparativa de Ferramentas de Mercado.

Ferramenta	**É compatível com vários SGBDs?**	**Analisa sinônimos**	**Automática**	**Gratuita**
DTM Schema Comparer[29]	Não	Não	Não	Não
DBA xPress [30]	Não	Não	Não	Não
dbForge Schema Compare [31]	Não	Não	Não	Não
SQL Examiner Suite [32]	Sim	Sim	Não	Não
ALTOVA [33]	Sim	Não	Não	Não
DB Comparer [34]	Não	Sim	Não	Sim
XSQL Software Schema Compare [35]	Não	Não	Não	Não
Apex SQL Diff [36]	Não	Não	Não	Não
RedGate [37]	Não	Não	Não	Não
MySQL Workbench [38]	Não	Não	Não	Não
SQLyog's Schema Synchronization [39]	Não	Não	Não	Não
Acqua Data Studio [40]	Sim	Não	Não	Não
Visual Studio [41]	Não	Não	Não	Não
IDERA [42]	Não	Não	Não	Não
DBSolo [43]	Sim	Não	Não	Não

Com base na análise da documentação das ferramentas de comparação de esquemas do mercado, a DTM Schema Comparer [29], DBA xPress [30], dbForge Schema Compare [31], XSQL Software Schema Compare [35], Apex SQL Diff [36], RedGate [37], Visual Studio [41], IDERA [42], MySQL Workbench [38] e a SQLyog's Schema Synchronization [39] são ferramentas pagas que comparam pares de esquemas. É gratuita a ferramenta DB Comparer [34]. Porém, o principal problema é que estas ferramentas pagas ou gratuitas são construídas para apenas um SGBD, não suportando uma variedade de SGBDs coomo Oracle, PostgreSQL, SQLServer, DB2 e MySQL. As ferramentas mais flexíveis por suportar mais de um SGBD foram a SQL Examiner Suite [32], ALTOVA [33], Acqua Data

Studio [40] e a DBSolo [43]. Porém, infelizmente nenhuma das ferramentas de mercado encontradas comparam vários esquemas de forma automática ou em grupos.

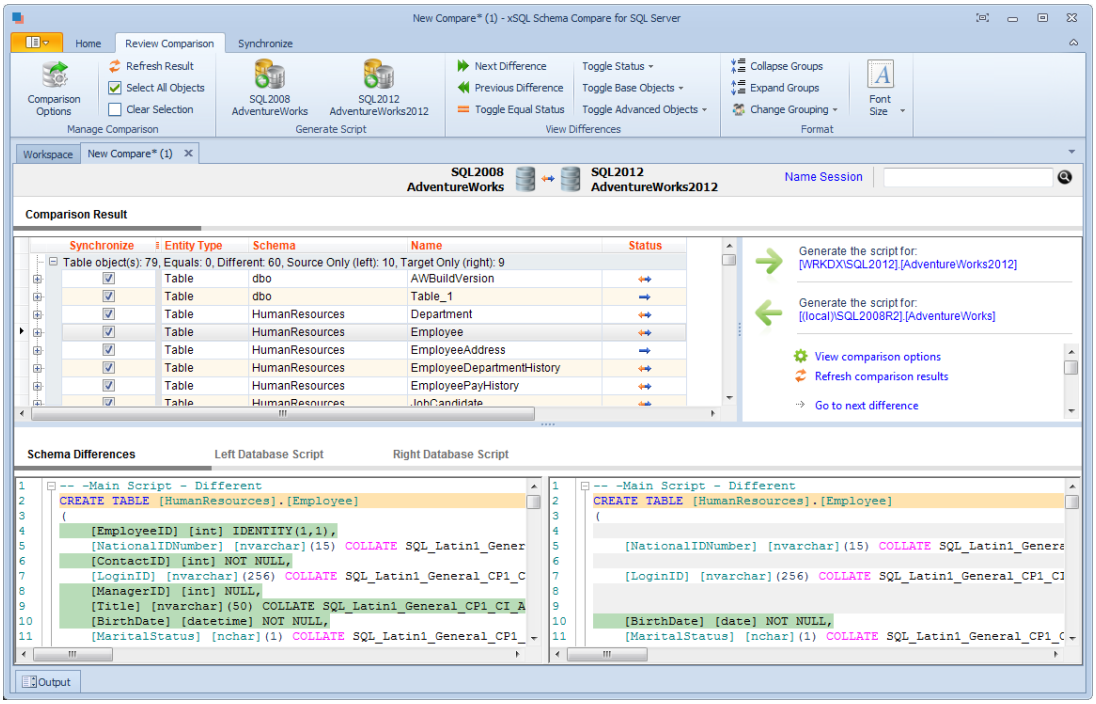


Figura 3.2: Exemplo da Ferramenta *XSQL Software Schema Compare*.

As ferramentas de mercado, como é o caso por exemplo da *XSQL Software Schema Compare* [35] que possui uma interface como a exibida na Figura 3.2, compara pares de esquemas. As interfaces das ferramentas de mercado são parecidas em suas funcionalidades e modalidades de comparação, comparando sempre pares de esquemas de acordo com sua construção, não permitindo a comparação de mais de 2 esquemas de forma automática.

Se a comparação fosse realizada por um administrador de dados utilizando alguma das ferramentas de mercado listadas na 3.2, considerando também um determinado tempo para a comparação de cada par de esquemas, digamos que 20 minutos no melhor dos casos, as 62.481 comparações resultantes dos 354 esquemas levaria 1.249.620 minutos ou 20.827 horas, o que daria mais de 2 anos de trabalho dedicado sem pausas. Se considerado um horário de expediente de 8 horas de trabalho, esta análise de esquemas similares com as ferramentas existentes levaria mais de 10 anos para ser concluída.

Destes trabalhos já realizados, tanto das soluções acadêmicas quanto das ferramentas de mercado, muitos não são flexíveis quanto ao SGBD, ou não são automáticos, ou não consideraram grandes conjuntos de dados. Alguns deles precisam de conjunto de dados treinados e outros precisam do dicionário de dados completo dos bancos de dados para executar todas as comparações, que muitas vezes não estão disponíveis. Dessa forma, é proposto o uso de técnicas de mineração de dados para executar de forma automática

ou semiautomática a comparação de esquemas em grandes conjuntos de dados, através da extração os metadados para calcular automaticamente as similaridades dos pares de esquemas com base nas suas estruturas de bancos de dados, objetivando uma solução flexível capaz de lidar com bancos de dados de estruturas de tamanhos e tecnologias variadas.

A metodologia aplicada e a solução proposta são apresentadas no próximo capítulo.

Capítulo 4

Metodologia e Solução Proposta

Este capítulo define a metodologia e descreve a solução proposta desta pesquisa. Esta pesquisa, quanto à sua natureza, é categorizada como pesquisa aplicada. Segundo Silva and Menezes [44], a pesquisa aplicada na metodologia científica tem o objetivo de gerar conhecimento para aplicação prática e é dirigido à solução de problemas específicos. Para a mineração de metadados de objetos de bancos de dados, optou-se por uma adaptação do modelo de referência CRISP-DM.

Por ser uma versão adaptada do CRISP-DM[45], algumas atividades foram removidas por não se aderirem ao escopo do projeto. No lugar destas, foram adicionadas outras atividades que são mais relevantes, que são: levantamento do estado da arte, criação do repositório de metadados, criação do banco de metadados, alteração da política e padrões de normas, divulgação do resultado da pesquisa. Foi adicionada também uma fase para a comparação dos esquemas, com as atividades de definição das abordagens de comparação, seleção das variáveis de comparação, comparação dos esquemas e comparação dos resultados das abordagens. Dessa forma, a metodologia adaptada do modelo de referência CRISP-DM para a nossa pesquisa contém 7 fases e 25 atividades, conforme a Tabela 4.1.

A metodologia aplicada está detalhada na Tabela 4.1, a qual tem o objetivo de direcionar quais estruturas internas de tabelas, colunas, tipos e tamanhos tem maiores tendências a serem realmente similares.

1. Entendimento do negócio: os objetivos de negócio são divididos entre objetivos gerais e objetivos específicos. Nesta atividade são determinados os objetivos de negócio e também são definidas a contribuição e os resultados esperados que se pretende obter ao final da pesquisa. A situação atual é avaliada para detalhar o contexto da pesquisa, descrever a justificativa, definir o problema, suas causas e consequências. O levantamento do estado da arte busca por trabalhos mais recentes

Tabela 4.1: Metodologia aplicada.

Fases	**Atividades**
1. Entendimento do negócio	1.1 Determina os objetivos de negócio 1.2 Avaliação da situação 1.3 Levantamento do estado da arte
2. Entendimento dos metadados	2.1 Coleta dos metadados 2.2 Descrição dos metadados 2.3 Exploração dos metadados 2.4 Verificação dos metadados
3. Preparação dos metadados	3.1 Seleção dos metadados 3.2 Limpeza e construção dos metadados 3.3 Integração e formatação dos metadados 3.4 Criação do banco de metadados
4. Comparação dos esquemas	4.1 Definição das abordagens de comparação 4.2 Seleção das variáveis de comparação 4.3 Comparação dos esquemas 4.4 Comparação dos resultados das abordagens
5. Mineração de metadados	5.1 Clusterização 5.2 Visualização
6. Avaliação	6.1 Avaliação dos resultados 6.2 Determina os próximos passos
7. Implantação	7.1 Criação do repositório de metadados 7.2 Alterações na política e padrões de normas 7.3 Divulgação do resultado

relacionados à mineração de metadados e à comparação de esquemas de bancos de dados relacionais.

2. Entendimento dos metadados: inicialmente realiza manualmente a listagem dos nomes e IPs de servidores de bancos de dados a serem analisados. Em seguida, realiza a extração dos metadados. Usou-se a linguagem *Structured Query Language (SQL)* para extrair os metadados dos servidores selecionados. Os metadados coletados são descritos, explorados para verificar sua completude e qualidade, antes de seguir para as próximas atividades.

3. Preparação dos metadados: são selecionados os bancos de dados de interesse e de maior importância. Os metadados selecionados estão presentes na maioria dos SGBDs relacionais, o que torna esta pesquisa flexível em termos de tecnologia de banco de dados relacional. Em seguida, realiza-se a limpeza, o tratamento, o pré-processamento dos metadados extraídos para serem padronizados e normalizados. Também é realizada a construção de novas variáveis com base nas variáveis extraídas para serem usadas na atividade de comparação de esquemas. Depois, define-se

o formato dos metadados e integra os metadados formatados num único banco de dados que irá compor o banco de metadados. Na criação do banco de metadados é definido o modelo de dados do banco de metadados e é feita a sua implementação com os metadados que foram extraídos, selecionados, limpos, formatados e integrados nas atividades anteriores. A partir deste banco de dados de metadados é possível criar a primeira versão centralizada do repositório de metadados.

4. Comparação de esquemas: é proposta uma abordagem para comparar os esquemas que compara todos os esquemas contra apenas um esquema para descobrir a distância entre eles até o ponto de início. Esta abordagem é confrontada com outras duas abordagens. A primeira compara todos os esquemas com todos os outros, exceto a comparação inversa e exceto a comparação com ele mesmo. A segunda compara os grupos dos esquemas que tem tamanhos similares na quantidade de metadados, isto é, o número de tabelas e colunas. No experimento, estas abordagens são aplicadas e testadas, anotando-se a duração do tempo de execução e uso de recursos para a avaliação do resultado da comparação de esquemas. A inserção dos metadados extraídos no banco de metadados é realizada através de processos de *Extract, Transform, and Load* (ETL) que fazem a extração, transformação e carga dos metadados de forma automatizada e periódica.

5. Mineração de metadados: o resultado da comparação dos esquemas é clusterizado para facilitar a visualização dos esquemas mais similares. Nesta atividade, são utilizadas as funções de clusterização hierárquica. Para a visualização, são utilizados principalmente os dendrogramas.

6. Avaliação e implantação: Os metadados extraídos e preparados são usados na criação do repositório de metadados. No repositório de metadados, utilizamos o *QlikView*, que é uma ferramenta já implantada no Ministério, que receberá dados do banco de metadados que foi criado nas fases anteriores. São sugeridas atualizações na política de padrões e normas para a inclusão de exigência de buscar no repositório de metadados se já existe uma estrutura similar de banco de dados, antes da criação de um novo banco de dados. Por fim, são divulgados os resultados da análise exploratória, da identificação dos esquemas similares e também da criação do repositório de metadados.

Na verificação, limpeza, construção e formatação dos metadados e na comparação de esquemas foi usando a linguagem R, especificamente o R na versão 3.4.3 e o RStudio 1.1.419. A solução foi construída utilizando um computador de 64 bits com 4 cores de processadores e 16 GB de RAM.

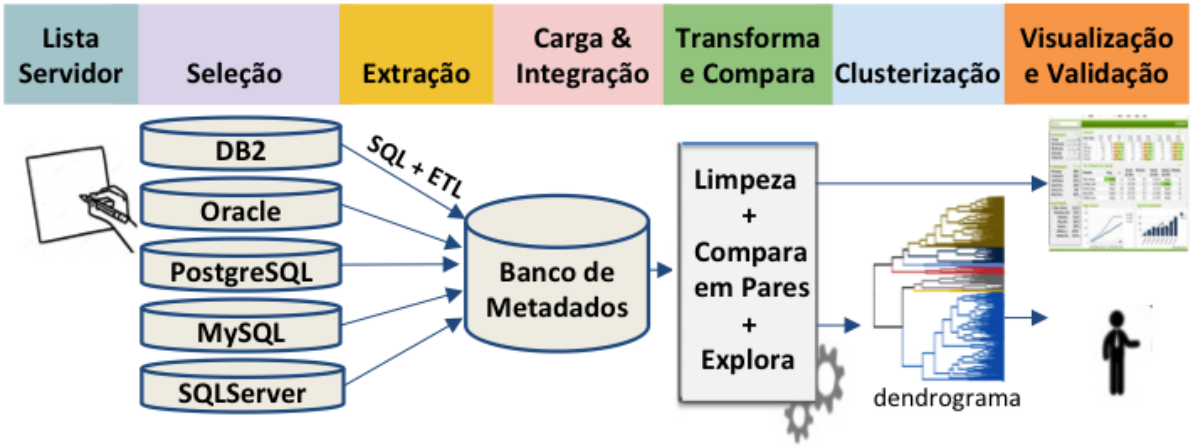


Figura 4.1: Visão Geral da Solução.

De uma forma geral, uma visão geral da solução proposta, na Figura 4.1, ilustra a realização das fases e atividades na metodologia aplicada. Para validar a solução proposta, é realizado um estudo de caso de abordagens de comparação de esquemas com técnicas de mineração de metadados para identificar, em uma parte dos bancos de dados, as estruturas similares de bancos de dados relacionais.

Os experimentos realizados são apresentados no próximo capítulo.

Capítulo 5

Experimentos

Neste capítulo, são descritos os experimentos que incluem o entendimento dos dados, a preparação dos dados, a comparação de esquemas e a avaliação de similaridade. O experimento descrito aqui foi realizado utilizando o processamento em modo sequencial, tanto na preparação dos dados quanto na comparação de esquemas e na clusterização para visualização da similaridade.

5.1 Entendimento dos Dados

O entendimento dos dados contém a coleta dos dados, a descrição e exploração dos dados e, por fim, a verificação dos dados.

5.1.1 Coleta dos Dados

A coleta dos dados iniciou a partir de um levantamento dos servidores de banco de dados, realizada primeiro manualmente, com o objetivo de listar os nomes e *Internet Protocol* (IP)s de servidores de bancos de dados do órgão. Esta listagem depois foi aumentada e automatizada através de uma varredura em toda a rede do Ministério para procurar por todas as máquinas que respondiam por alguma porta conhecida como porta de servidor de banco de dados. As portas buscadas foram: 50000, 50001, 50002, 60000, 50070, 50010, 21050, 1521, 1433, 5432, 3306, 3050, 2483 e 8020.

Como resultado desta busca inicial, foram encontrados 109 servidores de Banco de Dados (BD) em SGBDs variados, como pode ser visualizado na Figura 5.1. Destes, foram destacados os servidores de BD sob a gestão da equipe de TI, um total de 60 servidores, que são distribuídos da seguinte forma: 27 servidores de DB2, 13 servidores PostgreSQL, 10 servidores Oracle, 6 servidores MySQL e 4 servidores SQLServer, como pode ser visualizado na Figura 5.2.

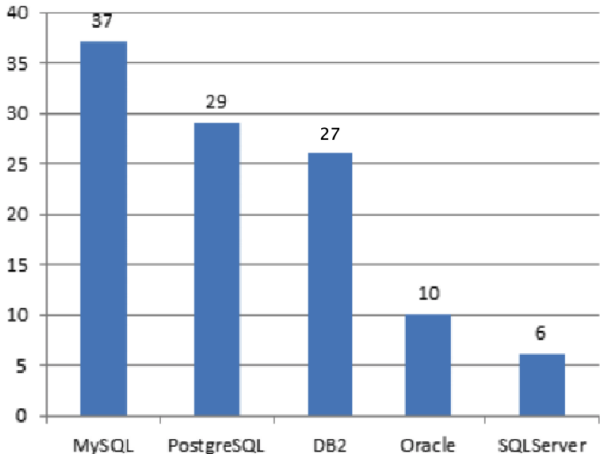


Figura 5.1: Quantidade de Servidores por SGBD - Varredura na Rede por Porta de BD.

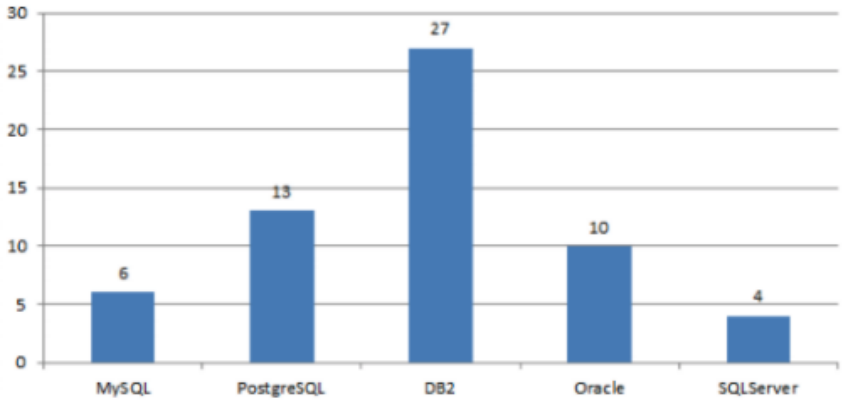


Figura 5.2: Quantidade de Servidores por SGBD - Sob Domínio da Equipe de TI.

5.1.2 Descrição dos Dados

No Ministério, os servidores de bancos de dados são organizados em: servidores de produção, servidores de homologação e os servidores de carga de dados históricos. A maioria dos outros servidores de BD que não estão sob os cuidados da equipe de TI são MySQL e PostgreSQL, tecnologias livres que na maioria dos casos do Ministério representam apenas bancos de dados de desenvolvimento, testes ou bancos de dados auxiliares para análises de dados. Isto ocorre porque algumas equipes possuem computadores para desenvolvimento, testes locais ou laboratório de dados. Porém, estes não serão analisados neste momento porque não representam bancos de dados críticos para o órgão. Também não serão analisados os dados históricos ou de homologação. Isto porque os dados dos servidores de produção são os mais importantes para o Ministério. Além disso, os dados de produção são os que já passaram por desenvolvimento e homologação, são os que teriam o melhor nível de qualidade, são os que representam os dados reais do Ministério.

Para cada servidor de banco de dados de produção foram coletados os seus metadados.

Os metadados contém os nomes dos servidores de banco de dados, dos esquemas, das tabelas e das colunas, o tipo de dado da coluna, o tamanho em quantidade de *bytes* reservados para a coluna, o dicionário de dados da coluna e da tabela, se a coluna permite valores nulos, se a coluna é uma *Foreign Key (FK)*, qual a quantidade de linhas da tabela e qual o tamanho da tabela em *gigabytes*. A lista dos metadados extraídos como variáveis é apresentado na Tabela 5.1. A consulta realizada através de *SQL* para extrair os metadados dos bancos de dados relacionais selecionados também está disponível sob demanda para tornar esta pesquisa reprodutível.

Tabela 5.1: Variáveis Extraídas.

1. Nome do servidor
2. Nome do *tablespace*
3. Nome do banco
4. Nome do esquema
5. Nome da tabela
6. Nome da coluna
7. Tipo da coluna
8. Quantidade de *bytes* reservados para o tamanho da coluna
9. Se é do tipo *null* ou *not null*
10. Se é uma *Foreign Key* (FK)
11. Dicionário da tabela
12. Dicionário da coluna
13. Número de registros
14. Tamanho da tabela (em GB)

A maioria dos esquemas analisados bilhões de registros, eles somam juntos mais de 15 TB de dados. Devido ao volume e variedade de dados, neste projeto não será incluído, por enquanto, como metadados o domínio dos dados.

5.1.3 Exploração e Verificação dos Dados

Na exploração de dados, verificou-se algumas características descritivas como a quantidade, tamanho, variedade. A quantidade de esquemas em cada servidor varia entre 1 a 47 esquemas. O maior esquema tem 1.591 tabelas. A maior tabela tem 179 colunas. Ainda sem qualquer tratamento no nome dos esquemas, tabelas e colunas, verificou-se que dos 354 esquemas analisados, 159 tinham o mesmo nome, representando cerca de 40% de esquemas com suspeita de similaridade.

A verificação dos dados buscou analisar a qualidade e completude dos dados. Nisso, verificou-se que o dicionário de dados não está presente em todas as estruturas devido aos bancos de dados legados. Menos de 30% das tabelas analisadas dos bancos de dados de produção e homologação possuem dicionário de dados.

Um caso interessante observado foi que aproximadamente 35% das tabelas estavam vazias, isto é, sem registros. Em uma pesquisa interna, descobriu-se que esta alta quantidade de tabelas vazias era decorrente de funcionalidades ainda não implementadas de sistemas.

5.2 Preparação dos Dados

A preparação dos dados realiza a seleção dos dados, a limpeza dos dados, a construção dos dados, a integração dos dados, a formatação e a criação do banco de metadados.

5.2.1 Seleção dos Dados

Selecionou-se para a comparação de esquemas os bancos de dados de produção, descartando os dados de desenvolvimento, teste, homologação e os esquemas que são criados na instalação padrão de um SGBD. A seleção dos servidores com bases de dados considerou os servidores que possuem os dados mais importantes para o Ministério, de forma a incluir os dados de produção e descartar os dados de testes ou de bases não-relacionais. Por isso, os dados escolhidos para esta análise são os dados dos servidores de produção, que estão sob a gestão da TI e que possuem informações do Ministério.

Tabela 5.2: Quantitativo de Objetos de Banco de Dados.

-	**Repositório de Metadados**	**Identificação de Similaridade**
Ambiente	homologacão e produção	produção
Servidores	36	26
Bancos	193	107
Esquemas	734	354
Tabelas	43.959	21.340
Colunas	500.994	318.839

O quantitativo total de objetos de bancos de dados é mostrado na Tabela 5.2, divididos entre quantitativo de objetos de banco de dados para o repositório de metadados e o quantitativo de objetos de banco de dados para a identificação de similaridade dos esquemas. Para o repositório de metadados, foram extraídos e selecionados os dados de homologação e de produção para a análise exploratória dos dados. Nem todos os metadados extraídos foram selecionados para serem usados na identificação de esquemas similares. Dos 734 esquemas extraídos, 414 esquemas foram selecionados por serem esquemas do ambiente de produção. Destes 414 esquemas, 60 foram removidos da análise por serem esquemas da instalação padrão de um SGBD, que são os esquemas com os nomes: *operationsmanager, opm, sys, dbaudit, explain, quest, db2mon, performance_schema,* e *trace.* Assim,

selecionamos 354 esquemas reais e de maior importância, mantidos pelo Ministério. Eles representam bancos de dados relacionais do ambiente de produção apenas, vindos de uma diversidade de SGBDs como Oracle, PostgreSQL, MySQL, SQLServer e DB2. O tamanho dos esquemas variam de 2 a cerca de 20 mil metadados. Todos os esquemas juntos somam um total de 21.340 tabelas e 318.839 colunas. Desta forma, ao se referir ao repositório de metadados são apresentados os esquemas tanto do ambiente de produção quanto de homologação. Enquanto que para a identificação de similaridade de esquemas foram utilizados apenas os esquemas de produção.

5.2.2 Limpeza e Construção dos Dados

Várias atividades de limpeza foram realizadas, como: tratar os valores faltantes, padronizar o *encoding* dos dados extraídos, remover as múltiplas ocorrências de espaços em branco antes e depois de cada palavra, remover os caracteres especiais, converter todas as letras em letras minúsculas, converter os números para o tipo numérico. O idioma Português Brasileiro possui muitas palavras com acentos e caracteres especiais, como o *ç*, por exemplo. Estes foram substituídos pelas mesmas letras, porém sem acentos ou cedilha.

Para realizar a comparação entre pares, o conjunto de dados foi expandido de forma que cada variável fosse comparada a todas as demais, exceto a si mesma. Por exemplo, o nome da *tabela1* comparada com o nome da *tabela2*. Dessa forma, a cada par de variáveis será possível calcular a sua distância. Por isso, foram criadas variáveis para calcular a distância entre os pares de metadados extraídos, conforme listadas na Tabela 5.3. Além das variáveis de distância, foram calculados também a quantidade de tabelas do esquema, a quantidade de colunas e o tamanho do esquema. Estas variáveis criadas também estão incluídas na Tabela 5.3.

Na comparação par a par das variáveis para calcular sua distância, é importante notar que as variáveis textuais de nome dos metadados geralmente possuem apenas uma palavra, e não uma frase ou texto completo. Por essa razão, as variáveis textuais foram comparadas usando distância por cosseno para medir a distância entre as palavras. Tata and Patel [46] explicam que a distância cosseno é uma medida baseada em vetor que mede a similaridade entre duas palavras transformando cada palavra em um vetor onde o ângulo cosseno entre os dois vetores é uma medida de quanto eles estão próximos. A distância cosseno vai de 0 a 1, onde 0 significa completamente similar e 1 completamente diferente.

Para comparar as variáveis numéricas, aplicou-se a distância euclidiana, definida na Equação 5.1, que calcula a distância entre dois pontos p e q.

$$\sqrt{(p_x - q_x)^2} = |p_x - q_x| \tag{5.1}$$

Tabela 5.3: Variáveis Criadas a partir das Variáveis Extraídas.

1. Distância entre os nomes dos bancos
2. Distância entre os nomes dos esquemas
3. Distância entre os nomes das tabelas
4. Distância entre os nomes das colunas
5. Distância entre os nomes dos tipos das colunas
6. Distância entre as quantidades de *bytes* reservados para o tamanho das colunas
7. Distância entre as quantidades de colunas do tipo *null*
8. Distância entre as quantidades de colunas FK
9. Quantidade de tabelas do esquema
10. Distância entre as quantidades de tabelas dos esquemas
11. Quantidade de colunas do esquema
12. Distância entre as quantidades de colunas dos esquemas
13. Tamanho do esquema em GB

5.2.3 Integração e Formatação dos Dados

Primeiro, os metadados foram extraídos de cada servidor de banco de dados no mesmo formato de valores separados por vírgulas ou *Comma Separated Value (CSV)*, gerando um arquivo. Todos os arquivos extraídos foram reunidos num único arquivo para integrar os metadados de todos os servidores de banco de dados analisados. O formato da extração foi padronizado para facilitar a integração dos dados, uma vez que todos os SGBD possuíam os mesmos conjuntos de metadados, mesmo com a variação da tecnologia de SGBD. O recurso usado para a padronização do formato foi usar a SQL para formatar a extração dos metadados para que todos os metadados fossem exportados no mesmo formato. A ordem dos campos deste formato pode ser visualizado na Tabela 5.4.

Tabela 5.4: Formato da Extração dos Metadados via SQL para CSV.

no_servidor, nu_ip, ds_sgbd, ds_sistema_operacional, st_ativo, tp_ambiente, no_banco, nu_tamanho_banco, ds_dicionario_banco, dt_importacao, nu_porta, ds_versao_sgbd, no_tabela, ds_dicionario_tabela, nu_tamanho_tab, dt_criacao_tab, no_esquema, no_tablespace, qt_registro, qt_coluna, no_coluna, tp_coluna, nu_tamanho_col, st_null, st_fk, ds_dicionario_coluna, dt_criacao_col

Para descobrir a similaridade entre esquemas, seus metadados foram comparados por pares de esquemas por meio de uma função de comparação. Esta função foi criada para agregar em uma tabela os dados de um esquema e em outra tabela os dados de outro

esquema. Então, as duas tabelas são comparadas uma contra a outra, considerando todas as variáveis.

Padronização e Normalização

A nomenclatura usada em bancos de dados relacionais usualmente é abreviada ou reduzida para cumprir o limite de caracteres de nomenclatura de cada SGBD. Os metadados de bancos de dados relacionais também tem uma mistura de palavras em Português e Inglês. Foi aplicada a padronização e normalização nas variáveis textuais para diminuir a variedade de termos, ou seja, o número de palavras diferentes que tinham o mesmo significado. Neste estudo, as variáveis padronizadas e normalizadas foram o tipo de dados da coluna, o nome do esquema, o nome da tabela e o nome da coluna.

Tabela 5.5: Exemplo da Padronização dos Tipos de Dados da Coluna.

Tipo	**Variações**
char	char, character, character varying, nchar
timestamp	datetime, timestamp, timestamp with time zone, timestamp without time zone, timestamp(0), timestamp(6), timestamp with time zone, timestamp(9)
time	time, time with time zone, time without time zone
int	int, integer,mediumint, bigint, tinyint, smallint
varchar	varchar, varchar2, long varchar, nvarchar, nvarchar2
double	double, double precision
clob	clob, blob,longblob, mediumblob,tinyblob
text	text, longtext, mediumtext, tinytext, ntext
xml	xml, xmltype
jms aq_jms_text_message	jms, aq_jms_message, aq_jms_object_message,
number	number, numeric
decimal	decimal, decfloat
binary	binary, binary_double, varbinary
raw	raw, long raw
edn	edn, edn_event_data, edn_oaoo_delivery

É comum que exista uma variância na nomenclatura entre SGBD para representar o mesmo tipo de dado. Originalmente existiam 85 tipos de dados diferentes. Depois da padronização conforme exibido na tabela, restaram apenas 45 tipos de dados diferentes. Por exemplo, as ocorrências de *varchar2, long varchar, nvarchar* e *nvarchar2* foram substituídas por *varchar*. Uma lista não exaustiva destas substituições realizadas com o objetivo de padronizar os tipos de dados dos SGBDs estão na Tabela 5.5. A lista completa de substituições e o código utilizado para esta padronização pode ser obtido sob demanda. Além da variável do tipo de dado da coluna como visto na Tabela 5.5, este tratamento de

normalização e padronização para diminuir a variância foi realizado também na variável de nome do esquema.

Tabela 5.6: Exemplo de Padronização de Nomes dos Esquemas.

Termo	**Variações**
teste	teste, test, teste_kadu
cadunico	cad_v6_20080701, cad13122014, cad15122012, cad16092017, cad17052014, cad17062017, cad17122016, cad18022017, cad18032017, cad18042015, cad18062016, cad19082017, cad19112016, cad19122015, cad20052017, cad20082016, cad20122013, cad21012017, cad21032015, cad21042017, cad21102017, cad22072017, cad30122011, cadastro, cargav6, tab_cad_16092017, tab_cad_17062017, tab_cad_18112017, tab_cad_19082017, tab_cad_20052017, tab_cad_20062015, tab_cad_21042017, tab_cad_21102017, tab_cad_22072017, cadunico
bpm	bpm, bpm_iau, bpm_mds, bpm_opss, bpm_soainfra, bpm_stb, bpm_ums, bpm_wls, bpmstd
sicnas	sicnas, sicnasdataprev, sicnasdp
sigsuas	sigsuas, sigsuas2
siafas	siafas, siafasv2
rais	rais, rais_caged
plandem	plandem, plandem12h
auditoria	auditoria, auditor, auditoria_acesso
sispraticas	sispraticas, sispraticas_old
dados	dados, dadosold, dadosoriginais
sgbf	sgbf, sgbfmigracao, sgbftmp, sgbfhst
sicnas	sicnas, sicnasdataprev, sicnasdp

Os nomes dos esquemas refletem informações do negócio ao que o banco de dados se refere, trazendo muitas siglas ou palavras específicas que não existem num dicionário ou vocabulário padrão. Por exemplo, a palavra *rais* que corresponde a base de dados de pessoas empregadas. Apenas os que trabalham com este tipo de informação saberá o significado de muitas palavras usadas na nomenclatura. Nestas situações, é necessária a padronização manual dos analistas de dados que conhecem estas bases de dados. Para isso, usamos a técnica de *bag of words* para descobrir a frequência e iniciar a normalização e padronização a partir das palavras que apareciam mais vezes. Para o nome do esquema, foram padronizados manualmente conforme o contexto de bancos de dados do órgão. A Tabela 5.6 mostra os detalhes de padronização dos nomes dos esquemas. Antes da padronização, existiam 254 nomes diferentes de esquemas. Depois da padronização, a quantidade de diferentes nomes de esquemas diminuiu para 194 nomes distintos.

Tabela 5.7: Exemplo da Padronização dos Nomes das Tabelas.

Remove caracteres que começam com	del_, rl_, carga_, ib_, xxx_, bp_, cp_, old_, tb_, tab_, tbd_, tbf_, tbl_, vw_, view_, views_, mv_, tmp_, temp_
Remove caracteres que terminam com	_tab, _tb, _vw, _view, _views, _tmp, _temp, _carga, _rmv, _del, _v
Remove caracteres que contém	bkp, backup, _old, nova, fora, _2
Remove números	de 0 a 9
Remove caracteres especiais	underline (_) espaço antes e depois da palavra

Já para as variáveis de nome da tabela e nome da coluna foi aplicado um tratamento diferente. Nestas duas variáveis, o primeiro passo foi remover as *stopwords*. Foram consideradas como *stopwords* o padrão de nomenclatura utilizado nos bancos de dados para nomear tabelas e colunas, regida pelo documento de Padrões de Normas de Banco de Dados do Ministério, como por exemplo, incluir *tb_* antes de qualquer nome de tabela e *vw_* antes de qualquer nome de *view*. Também foram consideradas como *stopwords* as nomenclaturas inadequadas que estavam fora do padrão. Eles foram incluídos na lista de *stopwords* porque estes padrões foram implementados apenas há poucos anos. Assim, muitos bancos de dados antigos não usavam um padrão de nomenclatura definido. Ao se juntar num banco de metadados os esquemas antigos e os novos a variação na nomenclatura era alta. Para diminuir a variância, reunimos nesta análise ambos os períodos de usos de nomenclatura. Além das palavras usadas no padrão de nomenclatura, usamos também a técnica de *bag of words* para descobrir outros usos de nomenclatura fora do padrão que apareciam mais vezes.

Tabela 5.8: Exemplo da Padronização de Nome das Colunas.

Remove caracteres que começam com	co_, cd_, dt_, hr_, dh_, ds_, ge_, no_, nu_, rf_, sg_, st_, tp_, qt_, vl_, pc_, sk_, id_, nr_, cp_, old_, tmp_, temp_, carga_, del_
Remove caracteres que terminam com	_v, _id, _temp, _carga, _del
Remove caracteres que contém	_old, nova, fora, _2, _1, backup, bkp
Remove números	de 0 a 9
Remove caracteres especiais	que começam com underline (_) que terminam com underline (_) que começam com traço (-) espaço antes e depois da palavra

Para padronizar o nome das tabelas fizemos a remoção apenas dos seus caracteres considerados como *stopwords*, tanto dentro e fora do padrão de BD. Com a remoção das *stopwords*, a variação nos nomes de tabelas diminuiu de 15.311 para 12.890 nomes de tabelas distintos depois da padronização. A lista das *stopwords* ou ocorrências nos nomes das tabelas que foram removidas para padronização estão listadas na Tabela 5.7.

Já no nome das colunas, a remoção apenas dos seus caracteres considerados como *stopwords*, tanto dentro e fora do padrão de BD do fez diminuir a variação nos nomes de 50.404 para 41.274 nomes de colunas distintos. A lista das *stopwords* dos nomes das colunas que foram removidas para padronização estão na Tabela 5.8.

5.2.4 Criação do Banco de Metadados

Foi criado um banco de dados para armazenar os metadados extraídos de forma centralizada num único local. Este banco, por guardar apenas os metadados ocupou pouco espaço em disco, cerca de 35 MB. O banco de dados utilizado para o banco de metadados foi o PostgreSQL, um banco de dados relacional de tecnologia livre. A modelagem do banco de metadados, visualizada na Figura 5.3, possui as 4 tabelas que são a TB_SERVIDOR, a TB_BANCO, a TB_TABELA e a TB_COLUNA, que representam os metadados dos servidores, do banco de dados, das tabelas de bancos de dados e das colunas das tabelas de banco de dados, respectivamente.

O modelo do banco de metadados, mostrado na Figura 5.3 tem a cardinalidade definida de forma que um servidor pode ter um ou vários bancos de dados, onde um banco de dados é proveniente de apenas um servidor de banco de dados. Um banco pode ter nenhuma ou várias tabelas e uma tabela é proveniente de um e apenas um banco de dados. Uma tabela pode ter uma ou várias colunas e a coluna é proveniente de uma tabela.

A tabela TB_SERVIDOR armazena os dados do nome do servidor, endereço IP, nome do SGBD, nome do sistema operacional, se o servidor está ativo ou desativo, se o servidor é um servidor de produção ou de teste. A TB_BANCO tem um relacionamento com a tabela TB_SERVIDOR, armazena os dados do nome do banco de dados, tamanho, dicionário do banco de dados, a data de importação, o número da porta de rede usada pelo banco de dados e a versão do SGBD. A TB_TABELA tem um relacionamento com a TB_BANCO e armazena o nome da tabela, o dicionário da tabela, o tamanho da tabela em GB, a data e criação da tabela, o nome do esquema, o nome do *tablespace*, a quantidade de registros e a quantidade colunas de cada tabela. Por fim, a TB_COLUNA tem um relacionamento com a TB_TABELA e guarda informações como o nome da coluna, o tipo de dados da coluna, o tamanho em *bytes* reservados para o tipo da coluna, se a coluna aceita valores do tipo nulos ou não, se a coluna é uma chave estrangeira ou não, o dicionário de dados da coluna e a data de criação da coluna, vide Figura 5.3.

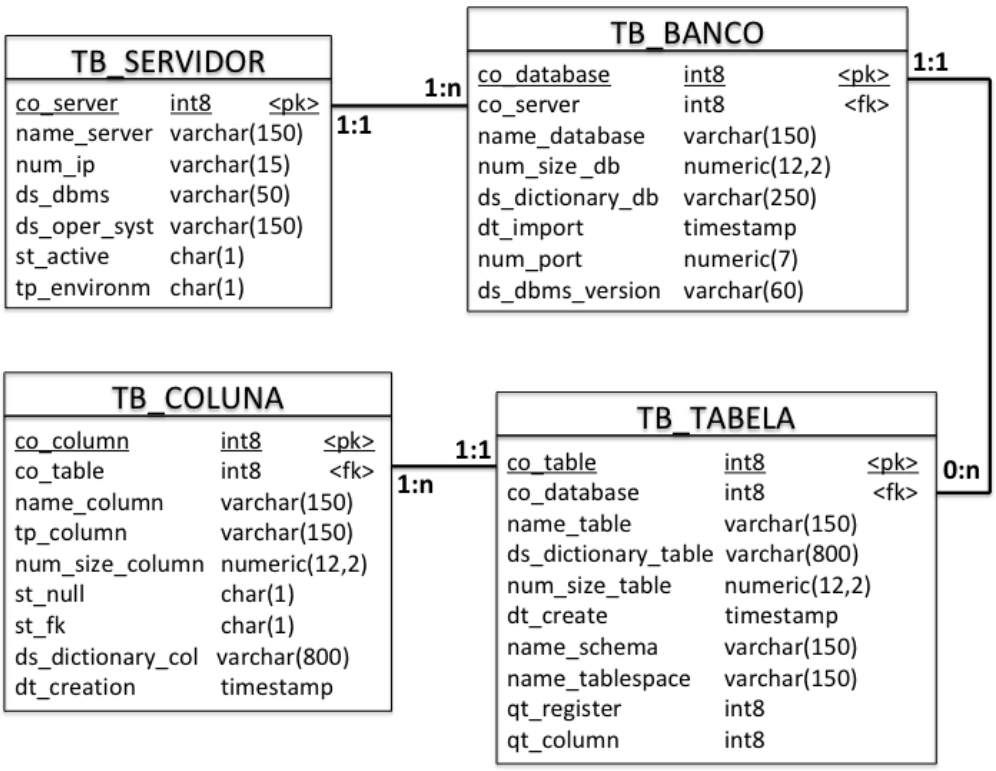


Figura 5.3: Banco de Metadados.

Esta estratégia de criar um banco de metadados foi importante para diminuir a necessidade das contínuas conexões nos demais bancos de dados para extração dos seus metadados. O uso de um banco de dados de metadados também facilitou a extração e a integração dos dados de metadados de uma forma automatizada através do uso do ETL. O banco de metadados é a principal fonte de origem de dados para o repositório de metadados. Se construído de forma a guardar o histórico, é possível visualizar as evoluções de crescimento do ambiente de banco de dados, da quantidade de esquemas e do crescimento dos esquemas de uma forma global. Sendo um importante aliado na futura estimativa de recursos de infraestrutura do órgão, dentre outras utilidades.

5.2.5 ETL

Para extrair os metadados, transformar no modelo de dados do banco de metadados e carregar os metadados de objetos de bancos de dados dentro do banco de metadados, foram criados processos de *Extract, Transform, and Load (ETL)*, que são responsáveis pela extração, transformação e carga dos dados.

Foram criados 4 processos de ETL, um para cada tabela do banco de metadados. Sendo assim, existe um processo de ETL para a extração, transformação e carga dos metadados de bancos de dados do servidor, do banco, da tabela e da coluna. Eles são

orquestrados para serem executados nesta ordem, obrigatoriamente. Os processos de ETL foram criados usando a versão 11.5 da ferramenta Datastage da IBM.

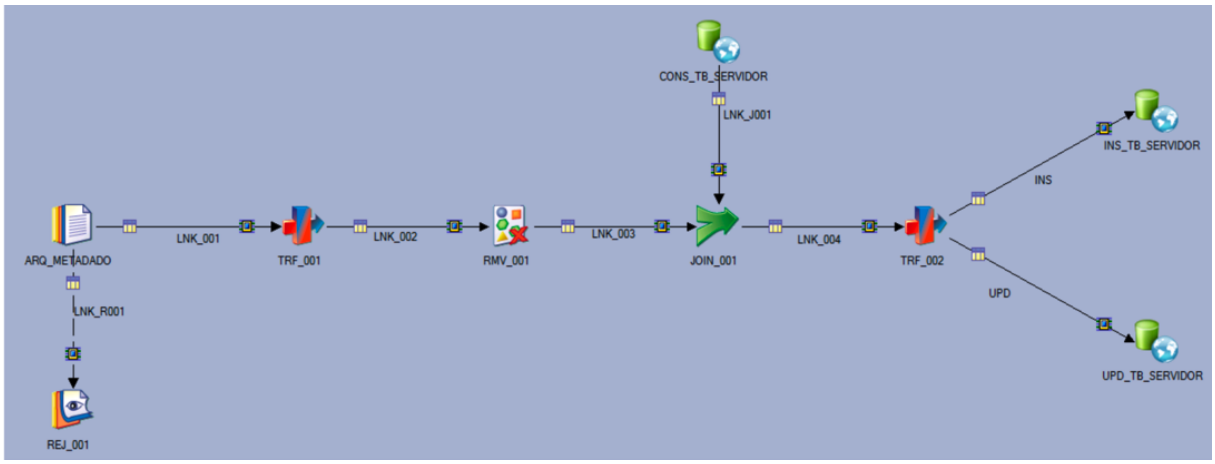


Figura 5.4: ETL de Carga da TB_SERVIDOR.

O processo de ETL para a carga dos metadados de servidor de banco de dados na tabela TB_SERVIDOR do banco de metadados é exibido na Figura 5.4. Este processo é iniciado a partir da extração destes metadados através do SQL que exportam os dados no formato CSV dentro do arquivo chamado ARQ_METADADO. Este arquivo tem a estrutura validada e se tiver erros é rejeitado pela tarefa REJ_001. Caso o arquivo esteja íntegro e dentro da estrutura definida, ele passa por uma leitura através da tarefa TRF_001 que tem o objetivo de ler os dados do arquivo CSV e identificar se existem estruturas duplicadas e remove as duplicadas através da tarefa RMV_001. Em seguida, a tarefa JOIN_001 consulta os dados já existentes no banco de metadados na tabela TB_SERVIDOR e compara os dados advindos do arquivo CSV. Caso os dados sejam inexistentes eles são inseridos como novo registro na tabela TB_SERVIDOR através da tarefa INS_TB_SERVIDOR. Caso os dados sejam existentes eles são atualizados na tabela TB_SERVIDOR através da tarefa UPD_TB_SERVIDOR.

Para a carga automática dos metadados do banco de dados na tabela TB_BANCO do banco de metadados, o processo de ETL é exibido na Figura 5.5. Este processo é iniciado a partir da extração destes metadados através do SQL que exportam os dados no formato CSV dentro do arquivo chamado ARQ_METADADO. Este arquivo tem a estrutura validada e se tiver erros é rejeitado pela tarefa REJ_001. Caso o arquivo esteja íntegro e dentro da estrutura definida, ele passa por uma leitura através da tarefa TRF_001 que tem o objetivo de ler os dados do arquivo CSV. O próximo passo consiste em identificar se existem estruturas duplicadas e remove as duplicadas através da tarefa RMV_001. Em seguida, a tarefa JOIN_001 consulta os dados já existentes no banco de metadados na tabela TB_SERVIDOR e a tarefa JOIN_002 consulta os dados já existentes

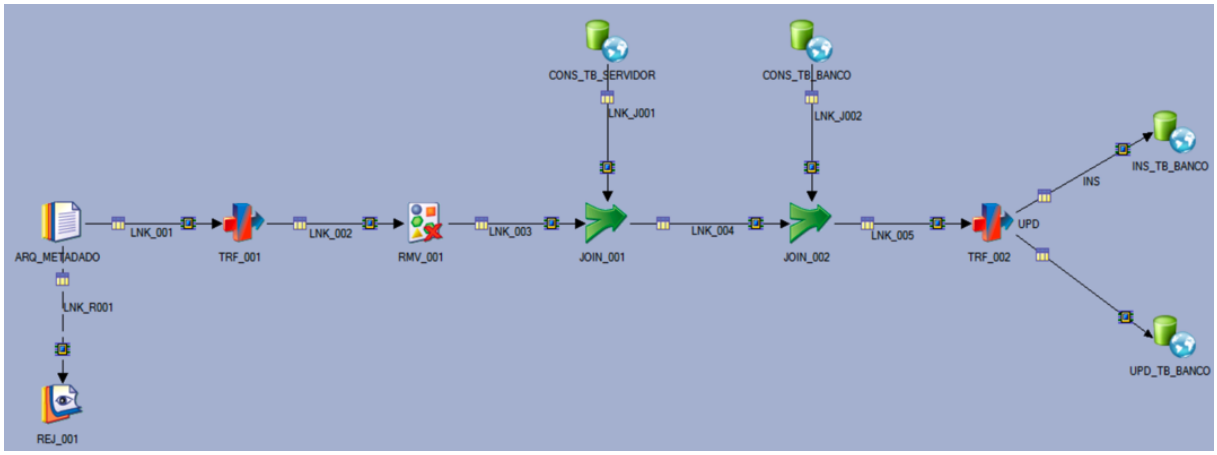


Figura 5.5: ETL de Carga da TB_BANCO.

no banco de metadados na tabela TB_BANCO. Caso os dados sejam inexistentes eles são inseridos como novo registro na tabela TB_BANCO através da tarefa INS_TB_BANCO. Caso os sejam existentes eles são atualizados na tabela TB_SERVIDOR através da tarefa UPD_TB_BANCO.

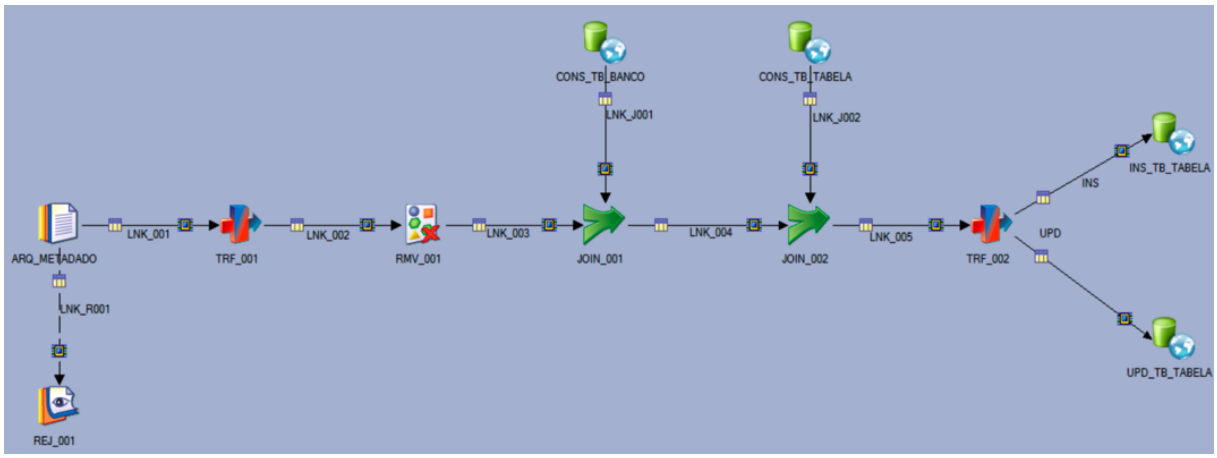


Figura 5.6: ETL de Carga da TB_TABELA.

As mesmas tarefas do processo descrito para a extração, transformação e carga da TB_BANCO da Figura 5.5 são utilizadas para popular e atualizar os dados da TB_TABELA, conforme mostra a Figura 5.6, bem como para popular e atualizar os dados da TB_COLUNA, conforme mostra a Figura 5.7, alterando, é claro, em cada caso, as origens de consultas de dados e destino de gravação de dados.

Os ETLs são *scripts* que podem ser agendados para serem executados automaticamente na periodicidade desejada, mantendo assim o banco de metadados atualizado. Para este experimento, foi agendada uma atualização mensal realizada no primeiro final de semana de cada mês. No caso de novos servidores de banco de dados ainda não mape-

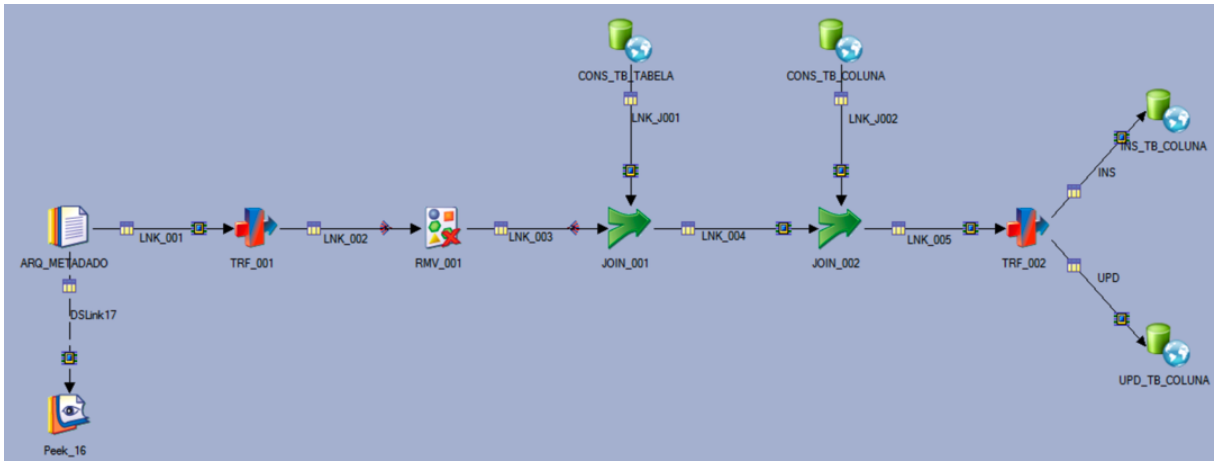


Figura 5.7: ETL de Carga da TB_COLUNA.

ados, existe o trabalho adicional de mapear a conexão para este novo servidor para que seja possível agendar também deste novo servidor a extração, transformação e carga dos seus metadados.

5.3 Comparação de Esquemas

A primeira atividade da fase de comparação de esquemas é a definição das abordagens de comparação, depois a seleção das variáveis para comparação e, em seguida, a execução da comparação dos esquemas usando as abordagens e variáveis definidas.

5.3.1 Abordagens de Comparação

Tem-se o objetivo de encontrar uma abordagem rápida e econômica em recurso de infraestrutura, capaz de realizar comparações de pares de esquemas inclusive entre numerosas e grandes bases de dados. Isto porque se a comparação dos esquemas for realizada usando força bruta, que consiste em comparar todos os esquemas com todos os outros esquemas e inclusive ele mesmo, é gerada uma complexidade quadrática. Com isso, o cálculo para realizar a comparação da quantidade de esquemas (n) é definido na Equação 5.2. O uso de força bruta cria um alto número de comparações, impactando na duração do processamento de comparação e também no uso de infraestrutura para executar os comparativos.

$$n * n \tag{5.2}$$

Por isso, para comparar as estruturas dos esquemas, são propostas outras 3 abordagens de comparação, diferentes de força bruta, que são:

1. Abordagem 1: comparar todos os esquemas com todos os outros, exceto ele mesmo e exceto a comparação inversa. O cálculo da quantidade de comparações é exibido na Equação 5.3, onde n é a quantidade de esquemas.

$$\frac{(n * n) - n}{2} \tag{5.3}$$

2. Abordagem 2: primeiro agrupar os esquemas por tamanho de acordo com a quantidade de metadados. A quantidade de metadados é proporcional ao tamanho da estrutura do esquema em quantidade de tabelas e colunas. Uma pequena quantidade de metadados corresponde um esquema com estrutura pequena, enquanto que uma grande quantidade de metadados corresponde a esquemas com estruturas grandes. Dessa forma, agrupando por tamanho de esquemas, os esquemas pequenos são comparados com esquemas pequenos e os esquemas grandes são comparados com esquemas grandes. Depois de dividir em grupos, dentro de cada grupo são comparados todos os esquemas com todos os outros esquemas daquele grupo, exceto a comparação inversa e exceto a comparação com ele mesmo. O cálculo da quantidade de comparações é exibido na Equação 5.4, onde n é a quantidade de esquemas e i o grupo ao qual pertence o esquema.

$$\frac{(n_i * n_i) - n_i}{2} \tag{5.4}$$

3. Abordagem 3: comparar apenas um esquema com todos os outros para mensurar a distância entre um esquema até todos os outros esquemas. Dessa forma, os esquemas mais próximos tendem a ser mais similares entre si, enquanto que os esquemas mais distantes tendem a ser mais diferentes uns dos outros à medida que a distância entre eles aumenta. O cálculo da quantidade de comparações é realizado subtraindo 1 da quantidade de esquemas n, como mostrado na Equação 5.5.

$$n - 1 \tag{5.5}$$

A Figura 5.8 mostra um exemplo de comparação de 4 esquemas de banco de dados aplicando tanto força bruta quanto as 3 abordagens propostas. Nesta figura, os números representam os esquemas e as linhas com setas representam as comparações. Veja que a quantidade de comparações diminui ao se comparar a abordagem de força bruta com qualquer outra das 3 abordagens. Esta diferença na quantidade de comparações ainda é mais expressiva quando se tem uma grande quantidade de esquemas de banco de dados. Por necessitar de uma grande quantidade de comparações, tempo e recursos, não será

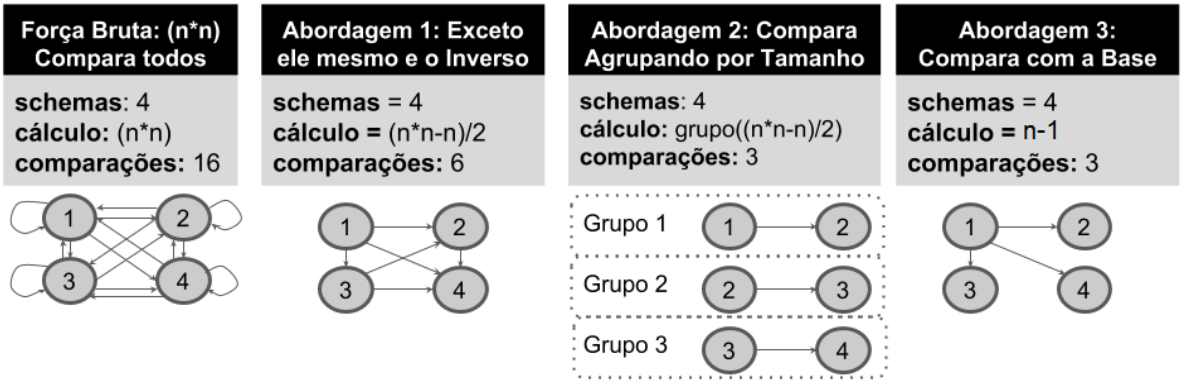


Figura 5.8: Abordagens de Comparação de Esquemas.

executada a abordagem de força bruta. Assim, será executado o comparativo apenas das 3 abordagens propostas.

5.3.2 Variáveis de Comparação

As variáveis usadas na comparação de pares dos esquemas medem a distância entre os seus respectivos metadados. São usadas 10 medidas de distância como variáveis de comparação, conforme mostra a Tabela 5.9. Estas variáveis foram calculadas na construção de variáveis a partir das variáveis extraídas dos metadados, conforme explica a Seção 5.2.2.

Tabela 5.9: Variáveis usadas na Comparação de Esquemas.

1. Similaridade entre os nomes dos bancos
2. Similaridade entre os nomes dos esquemas
3. Similaridade entre os nomes das tabelas
4. Similaridade entre os nomes das colunas
5. Similaridade entre os nomes dos tipos das colunas
6. Similaridade entre as quantidades de *bytes* reservados para o tamanho das colunas
7. Similaridade entre as quantidades de colunas do tipo *null*
8. Similaridade entre as quantidades de colunas FK
9. Similaridade entre as quantidades de tabelas dos esquemas
10. Similaridade entre as quantidades de colunas dos esquemas

Todas as variáveis possuem valor numérico e foram padronizadas para representar uma régua de distância que vai de 0 a 100. Onde 0 significa que as variáveis estão tão distantes ao ponto de serem totalmente distintas entre si. Enquanto que, 100 significa que não existe nenhuma distância e as duas variáveis comparadas são totalmente similares. Esta escala representa a oposta da escala calculada pela distância de textos por cosseno. Por isso, foi feito um cálculo do complemento das variáveis de distância cosseno para se igualarem ao

cálculo da distância das variáveis numéricas. Assim, todas as variáveis possuem a mesma escala de similaridade de 0 a 100, conforme proposto na Figura 5.9.

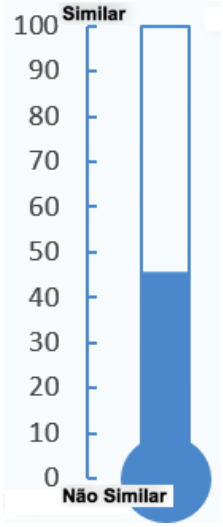


Figura 5.9: Escala de Similaridade.

Por exemplo, o cálculo da distância por cosseno dos pares de nomes de esquema *pesq versus cadp* resulta 0.75. O complemento representa o valor absoluto de 100 - (0.75*100) que é 25. Após executar o mesmo cálculo de complemento em todas as variáveis de similaridade de texto calculadas pela distância cosseno resultou num *dataset* equalizado onde todas as variáveis que calculam a distância tanto de variáveis textuais quanto variáveis numéricas possuem o mesmo intervalo numa mesma escala padronizada. Assim, 0 significa que os metadados daquela variável texto ou numérica não são similares, ou seja, são diferentes. Enquanto que 100 significa que os seus metadados são totalmente similares, ou seja, são idênticos. O valor entre 0 e 100 representa alguma similaridade entre os valores comparados de forma que se estiver mais próximo de 0 os metadados são menos similares entre si, ao mesmo tempo que os valores mais próximos de 100 representa os metadados mais similares entre si. Com isso, uma variável não tem mais peso sobre a outra e todas são fatores que juntos compõem o cálculo de similaridade entre pares dos esquemas.

Um exemplo do *dataset* resultante da comparação de pares de esquemas está na Tabela 5.10, onde as duas primeiras colunas representam a identificação dos pares dos esquemas a serem comparados. A identificação do esquema é composta pelo nome do SGBD, pelo nome do servidor, pelo nome do banco de dados e pelo nome do esquema, respectivamente. As demais colunas representam o resultado da similaridade das variáveis comparadas em pares. Os resultados foram normalizados de 0 a 100, onde 0 significa totalmente diferente e 100 significa totalmente igual, seguindo a escala de similaridade da Figura 5.9.

Analisando os resultados da Tabela 5.10, a primeira linha é um exemplo da comparação entre dois esquemas com nomes totalmente diferentes, porém com estruturas iguais

Tabela 5.10: Exemplo do Formato da Comparação dos Pares de Esquemas.

Esquema 1	Esquema 2	Nome banco	Nome esquema	Nome tabela	Nome coluna	Qtd coluna *null*	Qtd coluna FK	Tipo coluna	Qtd coluna	Qtd tabela	Qtd *bytes*
MySQL srv1 datab1 cad	PostgreSQL srv11 db7 public	41	24	100	100	100	100	100	100	33	31
DB2 srv2 db8 aud	DB2 srv4 dtb4 aud	58	100	100	100	100	100	100	100	100	100
SQLServer srv2 dbo pesq	Oracle srv3 ora3 cadp	29	25	0	56	0	0	44	55	14	0

de tabelas e colunas, exceto pelo número de tabelas e *bytes* reservados para o tamanho da coluna, o que representa que um esquema é uma cópia parcial do outro esquema. Este resultado é importante, porque ao simplesmente procurar por esquemas com nomes similares sem analisar a estrutura não seria possível descobrir estes dois esquemas parcialmente similares.

A segunda linha da Tabela 5.10 apresenta dois esquemas idênticos que estão duplicados em dois servidores diferentes. Mesmo que eles tenham o mesmo nome, não seria fácil encontrar se a análise não reunisse todos os metadados de todos os bancos de dados em apenas um *dataset*. Este é um caso de deduplicação de esquema dentro de dois servidores que estavam em dois endereços diferentes da organização.

A terceira e última linha da Tabela 5.10 mostra o último exemplo do resultado da comparação, onde existem 2 esquemas diferentes com composições diferentes e certamente não são candidatos a serem considerados similares ou duplicados. Eles são diferentes em todos os aspectos.

Para ter este resultado de comparação de todos os 354 esquemas analisados, a seção seguinte realiza a comparação aplicando as 3 abordagens propostas.

5.3.3 Comparação dos Esquemas

Nesta atividade são realizados os comparativos dos pares de esquemas de forma automática em cada uma das 3 abordagens propostas. A primeira abordagem compara todos os esquemas, a segunda compara por grupo de esquemas e a terceira abordagem compara com um esquema base.

Abordagem 1: Compara Todos os Esquemas

A primeira abordagem compara todos os esquemas com todos os outros, exceto a comparação inversa e exceto a comparação com ele mesmo. O cálculo da quantidade de comparações é exibido na Equação 5.3. A comparação de 354 esquemas resulta, é calculada em (354*354-354)/2, o que resulta em 62.481 comparações.

Apenas a comparação de 13 dos 354 esquemas foi concluída. O tempo de processamento ultrapassou de 3 dias e foi abortado por estouro de memória. Nesta abordagem não foi considerado o tamanho dos esquemas. Portanto, comparou-se esquemas grandes e esquemas pequenos ao mesmo tempo. Uma abordagem mais aprimorada e de menor complexidade será necessária para realizar todas as comparações.

Abordagem 2: Compara por Grupos de Esquemas

A segunda abordagem compara apenas pares de esquemas dentro de um grupo de esquemas com tamanhos similares de acordo com a quantidade de metadados. Deste modo, esquemas de estruturas pequenas são comparados com outros esquemas de estruturas pequenas. Enquanto que esquemas de estruturas grandes são comparados com outros esquemas de estruturas grandes. O cálculo da quantidade de comparações é exibido na Equação 5.4.

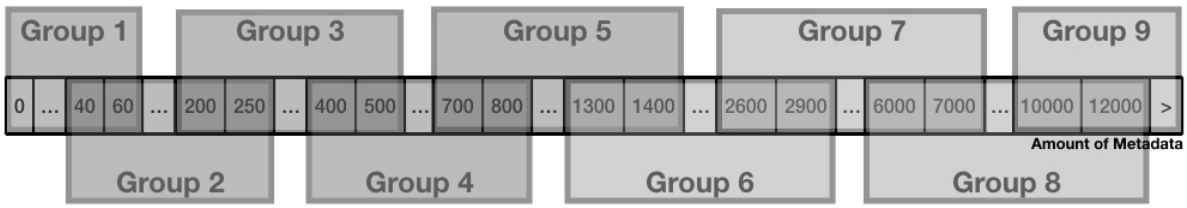


Figura 5.10: Agrupamento de Esquemas por Quantidade de Metadados.

A comparação dos 354 esquemas nesta segunda abordagem primeiro dividiu os esquemas em 9 grupos de acordo com a quantidade de metadados, considerando uma janela deslizante para incluir também os esquemas que ficaram entre um limiar de um grupo e outro, assim como mostra a Figura 5.10.

Os detalhes da divisão dos grupos e o resultado da comparação dos esquemas em cada grupo é exibido na Tabela 5.11. Esta tabela contém, para cada grupo, a quantidade de metadados dos esquemas, a quantidade de esquemas com aquela quantidade de metadados, a quantidade de comparações e a duração do processamento da comparação. Os últimos dois grupos, *Grupo 8* e *Grupo 9*, depois de horas de execução abortaram o processamento por estouro de memória e não completaram a execução com sucesso.

Os esquemas com estruturas menores, ou seja, os que possuíam a menor quantidade de metadados, executaram em menor tempo. Por outro lado, à medida que cresce o tamanho

Tabela 5.11: Comparação de Esquemas por Grupos.

Grupo	Metadados	Esquema	Comparações	Duração
1	0 - 60	97	4.656	14min
2	40 - 250	112	6.216	1h30
3	200 - 500	78	3.003	4h12
4	400 - 800	50	1.225	4h24
5	700 - 1.400	35	595	6h00
6	1.300 - 2.900	21	210	11h40
7	2.600 - 7.000	11	55	14h50
8	6.000 - 12.000	9	36	-
9	>10.000	7	21	-

das estruturas dos esquemas em quantidade de metadados, aumenta-se também o tempo de processamento, mesmo com a decrescente quantidade de esquemas em cada grupo.

Abordagem 3: Compara com um Esquema Base

A terceira abordagem compara todos os esquemas contra apenas um esquema, chamado de esquema base que representa o ponto de partida. Os resultados mostram a distância entre um esquema até todos os outros. Ao mesmo tempo, mostra que esquemas similares tendem a ficar mais próximos entre si, enquanto que esquemas diferentes tendem a ficar distantes.

A escolha do esquema base para a comparação de esquemas não influencia no resultado. O resultado ainda será proporcional, considerando a distância ou similaridade entre os esquemas analisados. Isto porque ao escolher um ponto de início, os demais pontos terão a mesma distância entre eles, e, mesmo que o ponto de início seja alterado, os demais pontos continuam com a mesma distância entre eles.

Tabela 5.12: Execuções da Comparação com o Esquema Base.

Quantidade de metadados do esquema base	Comparações	Duração
2	353	1min22s
60	353	15min
1.079	353	6h07min

A Tabela 5.12 contém a quantidade de metadados, a quantidade de comparações e a duração da execução das comparações. São apresentadas como exemplo a escolha de 3 esquemas diferentes como sendo o esquema base. Independente da escolha do esquema base, a quantidade de comparações continua a mesma. São geradas 354 - 1 totalizando 353 comparações.

A escolha do esquema base tem um impacto importante na duração do processamento das comparações. Escolher um esquema base com mais metadados aumenta a duração da execução das comparações. Por esta razão, o melhor é sempre escolher o menor esquema para servir como esquema base para a comparação com os outros esquemas. Para a escolha do esquema base, primeiro os esquemas foram ordenados pela quantidade de metadados e o esquema base escolhido foi aquele com a menor quantidade de metadados.

5.4 Avaliação da Similaridade

A similaridade compreende a mineração de metadados com o uso das técnicas de clusterização e de visualização. Ao final, a avaliação é feita por um especialista.

5.4.1 Clusterização

A modelagem aplicada neste capítulo tem a intenção de facilitar a visualização e descrição do resultado da comparação de esquemas. Neste sentido, a técnica de mineração de dados escolhida foi a de clusterização. Para Tan et al. [47], a clusterização ou agrupamento encontra grupos similares, define uma hierarquia através de subcategorias de *clusters*, encontra padrões, identifica os grupos que fogem dos padrões, seleciona grupos de interesse, realiza sumarização, realiza compressão ou até mesmo encontra o vizinho mais próximo através do cálculo de distância entre todos os pontos.

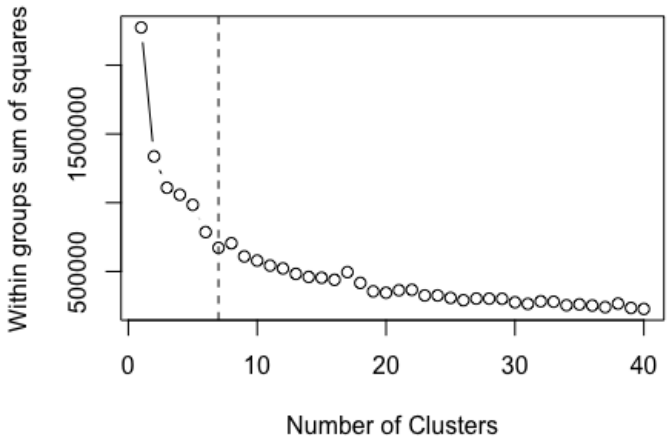


Figura 5.11: Número Ideal de *Clusters* com a Curva de Cotovelo.

A função usada para a criação da clusterização hierárquica se chama *HClust*. A *Hclust* é uma função que se refere a uma coleção de técnicas de clusterização que produz um

cluster hierárquico, onde cada ponto inicia um *cluster* e depois é agregado a outro *cluster* mais próximo até que todos estejam agrupados [47]. Segundo Tan et al.[47], o método *WARD* usa a estratégia de mesclar dois *clusters*, de forma que os *clusters* que tem os centros mais próximos são escolhidos e mesclados para resultar em um *cluster* e diminuir a distância entre os pontos do *cluster*. Neste sentido, usaremos a técnica de clusterização, especificamente a que usa clusterização hierárquica, com método *WARD* através do uso da função *HClust*.

Antes de criar o modelo, buscamos descobrir o número ideal de *clusters* para o nosso conjunto de dados. Para isso, usamos a curva de cotovelo que calcula o menor número de *clusters* usando *Kmeans* e a soma dos quadrados dentro de um *cluster*. Seu resultado, exibido na Figura 5.11, mostrou que 7 seria o número ideal de clusters. Ao realizar a clusterização com 7 *clusters* a melhor visualização do resultado foi através do dendrograma. O dendrograma é um gráfico que permite a visualização dos *clusters* de forma hierárquica e o resultado completo que mostra a distância entre os 354 esquemas pode ser visualizada no Apêndice B.

5.4.2 Visualização

Os 7 grupos de esquemas foram marcados com cores diferentes para facilitar a visualização da divisão e a proporção do tamanho dos grupos de esquemas similares, conforme mostra a Figura 5.12.

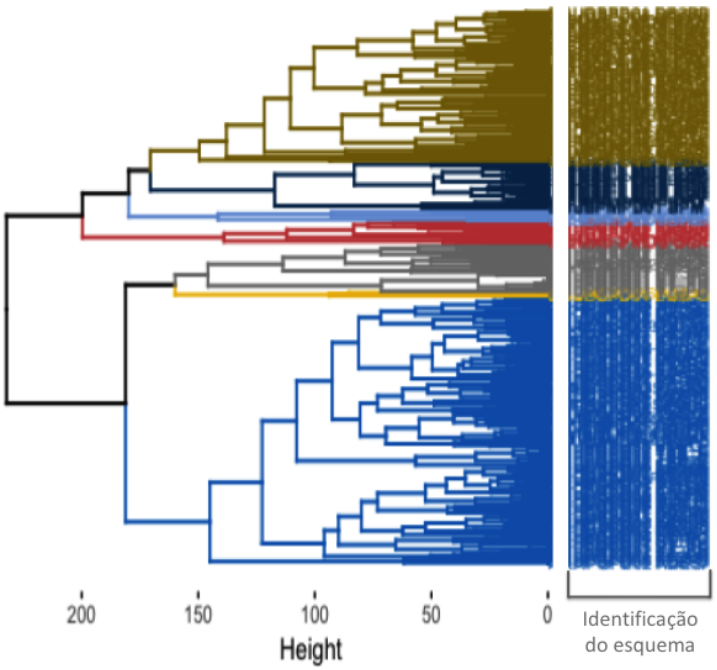


Figura 5.12: Clusterização Hierárquica.

Como são muitos esquemas a serem visualizados, foi incluído no Apêndice B a visualização completa. Aqui, focou-se numa subárvore do dendrograma, exibido na Figura 5.13 para mostrar com maiores detalhes como é o resultado final da identificação de uma parte dos esquemas com estruturas similares.

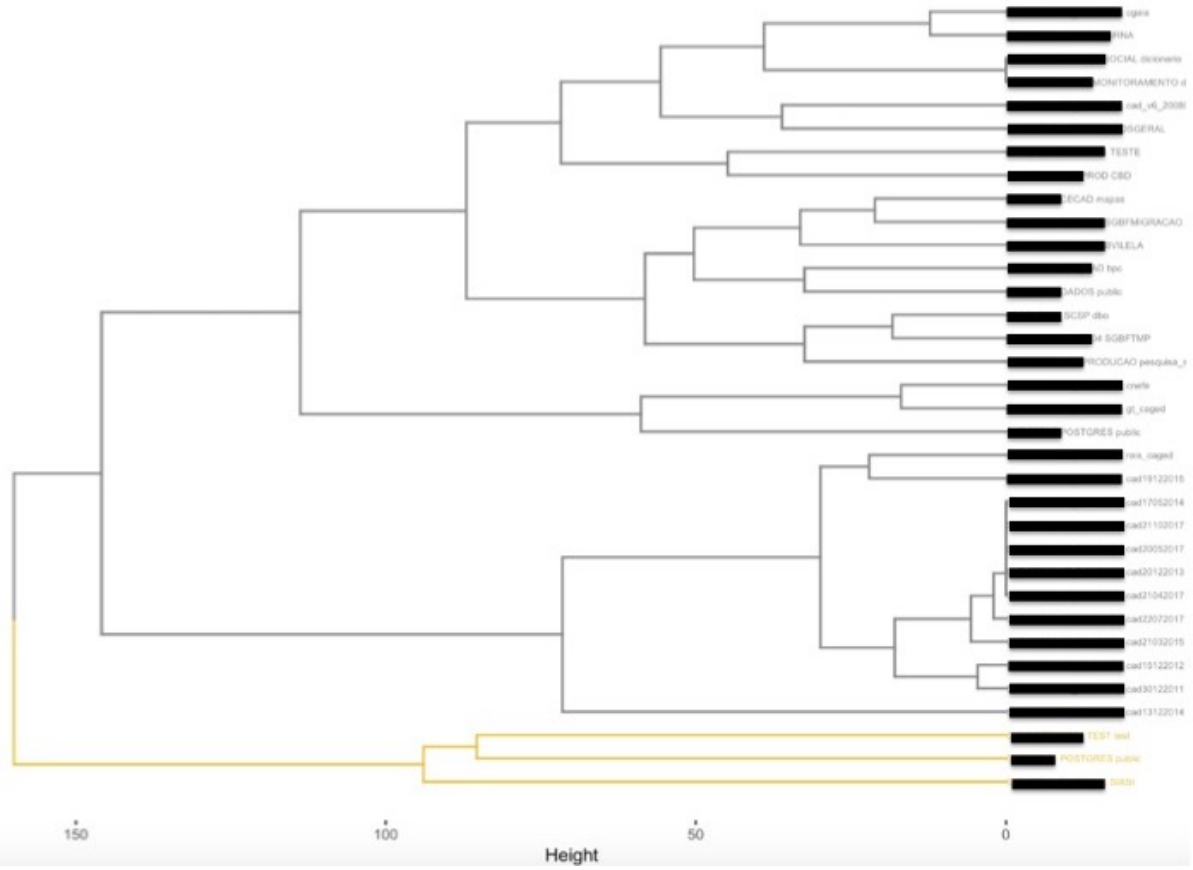


Figura 5.13: Visualização de uma Subárvore.

Nesta subárvore da Figura 5.13, é fácil visualizar a similaridade entre os esquemas e a distância de um esquema à outro. Partes dos nomes dos esquemas onde continha o nome do servidor e nome do banco de dados foram mascarados para garantir a segurança dos bancos de dados, deixando apenas o nome do esquema visível. Os esquemas mais similares tem a distância menor, medida pelo eixo x com nome de *Height*, como é o caso dos esquemas *_MISOCIAL dicionario* e *BDPG_MONITORAMENTO dicionario* que tem estruturas exatamente iguais. Elas são iguais porque a distância entre eles é nula, como pode ser visto em maiores detalhes no *zoom* da Figura 5.14. Neste caso, estes dois esquemas estão no mesmo servidor e, apesar de terem nomes de banco de dados diferentes *BDPG_SOCIAL* e *BDPG_MONITORAMENTO*, possuem nomes de esquemas e estruturas iguais.

Figura 5.14: Esquemas com Estruturas Semelhantes.

Um caso interessante visualizado em outra subárvore do dendrograma, visualizado na Figura 5.15, é o caso de dois esquemas que estão no mesmo servidor, possuem o mesmo nome do banco e possuem nome de esquema muito parecido *SIAFAS* e *SIAFASV2*, porém possuem estruturas diferentes como pode ser visualizado pela distância da ligação entre eles.

Figura 5.15: Esquemas com Estruturas Diferentes.

Outro caso também interessante observado em outra subárvore do dendrograma da Figura 5.12, com o foco mostrado na Figura 5.16, onde é possível visualizar esquemas que estão no mesmo servidor, possuem o mesmo nome de banco de dados e nomes parecidos de esquemas que mudam apenas a data de referência no final do nome de cada esquema. Estes representam esquemas grandes, possuem em sua estrutura além de um alto volume de dados, centenas de tabelas e milhares de colunas. Comparar e descobrir a similaridade destes esquemas de forma manual ou mesmo com as ferramentas de mercado seria trabalhoso pelo tamanho destes esquemas. Mas aqui, em apenas uma visualização é fácil identificar a similaridade entre a estrutura deles. É possível ver, por exemplo, que o esquema *tab_cad_22072017* possui estrutura idêntica ao *tab_cad_16092017*, se diferenciam um pouco do *tab_cad_19082017* e mais ainda do *tab_cad_21102017* e *tab_cad_18112017*.

Figura 5.16: Esquemas com Estruturas Variadas.

Threshold

A curva de cotovelo usada para encontrar o número ideal de clusters não é tão importante para a visualização dos resultados. O que agrega valor neste caso é o uso do *threshold*.

Um *threshold* neste caso estabelece um limite do que é considerado similar ou não, estabelecendo uma distância máxima dentro de uma régua de similaridade. Uma distância, também chamada de *heigh*, representa neste caso a distância de similaridade entre um esquema em outro. O limite ou *threshold* pode ser escolhido pelo usuário ou pela organização para que eles estabeleçam um limite dos esquemas que são considerados similares ou não.

Neste experimento, os resultados da distância (*heigh*) de similaridade de todos os esquemas analisados vai de 0 a 250, como pode ser visualizado na 5.12. O valor mais próximo a 0 representa os esquemas mais similares, enquanto que os valores mais distantes de 0 representa os esquemas mais diferentes entre si.

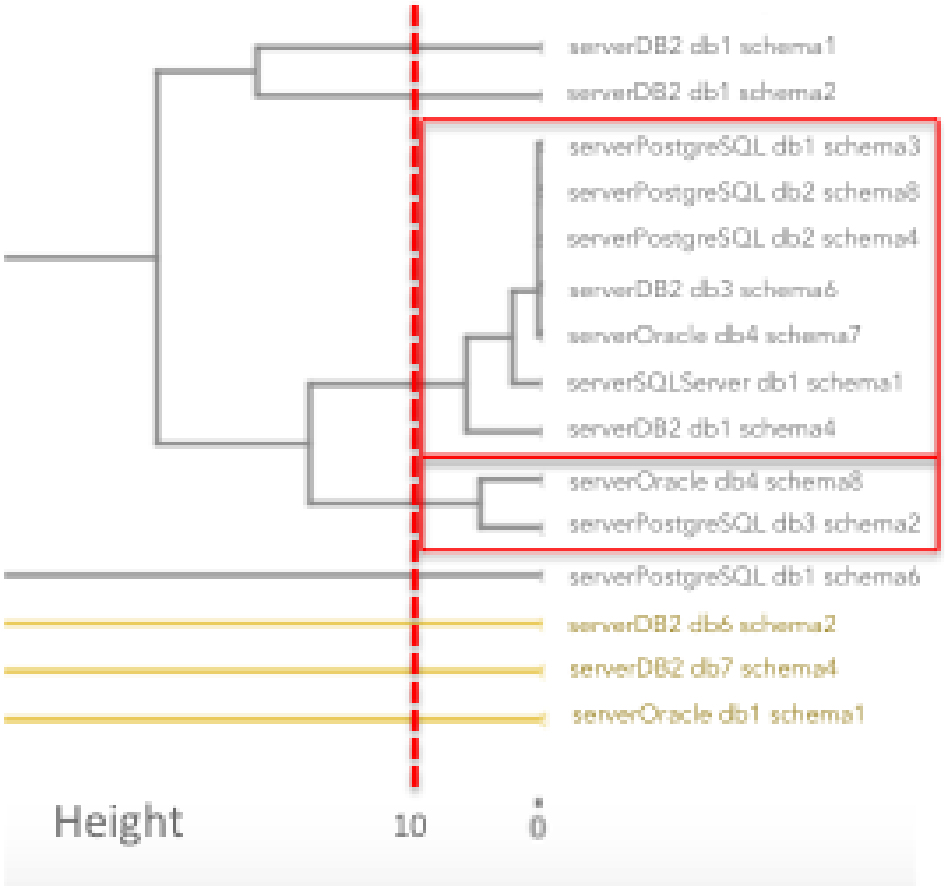


Figura 5.17: *Threshold*.

Foi aplicado um limite ou *threshold* de distância ou peso 10, delimitado por uma linha vermelha pontilhada, e, para melhor visualização de uma parte do resultado, foi aplicado um *zoom* ou aproximação no dendrograma que pode visualizada na Figura 5.17. Os nomes representam a identificação dos esquemas, contendo em cada linha o servidor, SGBD, nome do banco e nome do esquema. A identificação original dos esquemas foi mascarada para salvaguarda da segurança da informação do ambiente de banco de dados

do ministério. O *threshold* aplicado serviu como um filtro para trazer os esquemas mais similares dentro daquele limite definido. Assim, são contabilizados todos os grupos com mais de 1 esquema que estão dentro daquele limite, desenhados pela linha contínua em vermelho. No recorte da Figura 5.17, existem dois grupos de esquemas similares que reúnem 9 esquemas similares. Consideramos o peso 10 como *threshold*, uma medida conservadora, uma vez que o alcance máximo neste caso pode ser até 250. Ao analisar toda a base do experimento, dentro deste limite de peso 10, identificou-se 30 grupos de esquemas similares que reúnem 75 esquemas similares do total de 354 esquemas. Os esquemas similares identificados foram validados por um especialista em administração de dados do ministério que acessou os bancos de dados, comparou amostras de dados e considerou que os resultados representam com sucesso a similaridade dos esquemas.

Os resultados obtidos são apresentados no próximo capítulo.

Capítulo 6

Resultados Obtidos

Neste capítulo são descritos os resultados das três abordagens de comparação de esquemas, é feita a avaliação dos resultados de forma geral, a criação e os resultados do repositório de metadados, a sugestão de alteração na política de padrões e normas de BD do Ministério. Também são descritos os trabalhos publicados e apresentações decorrentes de trabalhos derivados desta pesquisa.

6.1 Comparativo das Abordagens de Comparação

De acordo com os resultados de cada abordagem, a Tabela 5.10 mostra o resultado da comparação em pares dentre os esquemas, sendo o mesmo formato do resultado de todas as 3 abordagens. Todas as abordagens utilizaram o mesmo conjunto de variáveis. Todas as execuções em cada uma das 3 abordagens utilizaram o mesmo computador, ou seja, usaram os mesmos recursos de infraestrutura. O que muda de uma abordagem para outra é a quantidade de comparações que são realizadas, o que impacta diretamente no custo da duração e da necessidade de recursos computacionais para processar a comparação de esquemas, sendo uma característica importante para viabilizar ou escolher o uso da abordagem.

A primeira abordagem tem uma estratégia de comparar todos os esquemas, exceto a comparação inversa e exceto a comparação com ele mesmo, gerando para 354 esquemas uma quantidade de mais de 62 mil comparações. Sendo que dos 354 esquemas, conseguiu processar apenas a comparação de 13 esquemas que geram 78 comparações. O tempo de processamento ultrapassou 3 dias e abortou por estouro de memória.

A segunda abordagem tem uma estratégia de primeiro agrupar os esquemas por tamanho de acordo com a quantidade de metadados e depois realizar a comparação de todos os esquemas do grupo, exceto a comparação inversa e exceto a comparação com ele mesmo, gerando 9 grupos de esquemas numa janela deslizante que inclui os esquemas de tamanho

que estão entre numa fronteira entre um grupo e outro. Nesta abordagem, foi possível processar o comparativo de 338 esquemas do total de 354 esquemas, gerando cerca de 16 mil comparações. O tempo de processamento total de todos os grupos foi de quase 43 horas. Os demais esquemas não tiveram o processamento do comparativo de esquemas concluído com sucesso porque comparavam os grupos de esquemas grandes e estouravam a memória do computador.

A terceira abordagem tem uma estratégia de comparar todos os esquemas com um esquema base e exceto a comparação com ele mesmo, gerando 353 comparações. Dos 354 esquemas, foi possível realizar a comparação de todos os 354 esquemas analisados. No melhor caso, a duração foi de 1 minuto e 22 segundos. Nesta abordagem a escolha do esquema base influencia no tempo de processamento. Se escolhido um esquema pequeno, ou seja, com pouca quantidade de metadados, a duração da comparação é pequena. Já a escolha de um esquema grande, com muitos metadados, pode inviabilizar o uso desta abordagem. A comparação das 3 abordagens consta na Tabela 6.1.

Tabela 6.1: Comparativo das Abordagens.

Abordagem	**Quantidade de esquemas a serem comparados**	**Quantidade de esquemas comparados com sucesso**	**Quantidade de comparações**	**Duração**
1. Compara todos	354	13	62.481	3 dias
2. Compara em grupos	354	338	15.960	42h50min
3. Compara com a base	354	354	353	1min22s

Por fim, a clusterização hierárquica facilitou a visualização do resultado da identificação dos esquemas similares. A visualização gráfica dos resultados em formato de dendrograma, que identificam os esquemas similares, ajudaram a reduzir o escopo e à direcionar para onde se encontram os esquemas mais similares.

6.2 Economia de Recursos

Com a análise exploratória, foi possível identificar um servidor de banco de dados totalmente duplicado, com bancos de dados em desuso. Ao desabilitá-lo, gerou-se uma economia de 10% no custo de serviços de monitoramento e backup. No início deste trabalho, o custo mensal total com este serviço era de R$69.065,34 e agora passou a ser R$62.034,43. Desta forma, o custo anual passou a ser de R$828.784,17 para R$744.413,16.

6.3 Repositório de Metadados

A visualização do Repositório de Metadados foi criada com a ferramenta *QlikView* porque era a disponível na instituição. Esta visualização mostra a quantidade de diferentes SGBDs mantidos pelo órgão, a quantidade de bancos de dados, esquemas, tabelas, colunas, tamanho, quantos possuem dicionário de dados, quantas tabelas vazias e também é possível pesquisar por determinado nome e ter como resultado todos os metadados que correspondem aquele nome, como exibido na Figura 6.2 e na Figura 6.1. Partes dos nomes dos servidores de banco de dados, dos IPs e das portas de rede usadas pelos bancos de dados foram mascaradas para garantir a segurança da informação e evitar possíveis ataques à estes bancos dados. Consolidar e visualizar as informações do banco de metadados em *dashboard* auxiliou na análise exploratória dos metadados.

Figura 6.1: Visualização do Repositório de Metadados - Aba Inicio.

Este repositório foi criado com o mecanismo de atualização automática de forma que ser for criado um novo banco ou tabela ou coluna de dados dentro dos servidores de banco de dados já mapeados e conectados, estas informações irão aparecer no repositório de metadados com a execução programada do processo de ETL. Agora, para novos servidores de banco de dados, é necessário configurar a extração dos seus metadados para o banco de metadados, para que assim sejam exibidos seus metadados no repositório.

Figura 6.2: Visualização do Repositório de Metadados - Aba Schemas.

6.4 Alterações na Política de Padrões e Normas de Banco de Dados

A sugestão de atualização da Política de Padrões e Normas de Banco de Dados do Ministério tem a intenção de incluir a exigência de buscar no repositório de metadados se já existe uma estrutura similar de banco de dados, antes da criação de um novo banco de dados para evitar que sejam criados novos bancos de dados duplicados. Assim, foram sugeridas alterações no documento de Padrões e Normas de Banco de Dados, que são:

1. Inclusão de novo item chamado de *Regras para Criação de Esquema ou de Banco de Dados*, com a seguinte orientação: Antes da criação de um novo banco de dados ou de um novo esquema, deve-se realizar uma consulta prévia com o objetivo de verificar se o mesmo já existe, mesmo que possua outro nome parecido ou que tenha sido criado em outro SGBD ou servidor de banco de dados, evitando assim a criação de bancos ou esquemas repetidos. Esta consulta prévia pode ser realizada através no repositório de metadados.

6.5 Divulgação dos Resultados

Resultados parciais desta pesquisa foram publicados e apresentados no Seminário de Dados Abertos do Ministério, em formato de artigo e pôster numa conferência de mineração de dados e também foi apresentado para a equipe do Ministério, como detalhado a seguir.

6.5.1 Apresentação no Seminário de Dados Abertos

Em novembro de 2017 foi apresentado no Seminário de Dados Abertos do Ministério [48] para o público em geral quais eram As Bases de Dados do Ministério [49], como resultado parcial da exploração dos metadados extraídos. A apresentação realizada para um público aberto interessado em dados abertos do governo mostrou como foi realizada a varredura interna em todos os computadores do Ministério que respondiam por portas de banco de dados para identificar servidores de bancos de dados que não eram mapeados pela equipe de TI. Junto com a varredura, foi apresentado um sumário do quantitativo e da diversidade de SGBDs encontrados no Ministério, a quantidade de tabelas vazias, esquemas sem dicionário de dados e esquemas que pareciam ser duplicados por terem nomes iguais. Nesta oportunidade, foi apresentado também um breve histórico de como foi criado o Ministério e os problemas causados pela falta de mapeamento das bases de dados mantidas pelo órgão e como isso impactava diretamente no programa de dados abertos.

6.5.2 Artigo e Poster

Em dezembro de 2017 foi publicado no *International Conference on Machine Learning and Applications* (ICMLA) um artigo e também apresentado um poster na conferência IEEE ICMLA possui Qualis B1. O trabalho publicado é entitulado *Aprendizado Paralelo em Duas Fases para Identificar Estruturas Similares entre Bancos de Dados Relacionais* [50]. Esta publicação relata estudo realizado para aplicar técnicas de classificação para identificar esquemas similares de bancos de dados. Neste estudo, foi proposta uma nova abordagem de aprendizado em paralelo em duas fases para identificar rapidamente as estruturas similares de bancos de dados relacionais. Cada fase representa um nível de agregação dos metadados. Para testar a abordagem, realizamos um experimento com algumas bases de dados massivas do Ministério para classificar quais bancos de dados relacionais tinham estruturas similares de tabelas e colunas baseadas nos seus metadados. A medida de similaridade considerou *Levenshtein* e cosseno. Foram aplicadas as técnicas de *GLM*, *RF* e *GBM* no modelo de classificação. Cada modelo teve a performance compara entre a execução em modo paralelo e sequencial. Como resultado, a execução

em paralela do *GBM* foi 10 vezes mais rápida do que o processamento sequencial. Este estudo buscou o uso de técnicas mais eficientes para identificar esquemas similares de bases de dados massivas. No Apêndice C é possível visualizar o resumo, data, autores e link de acesso.

6.5.3 Apresentação no Ministério

Em fevereiro de 2018 foi apresentada à equipe da TI do Ministério o resultado da análise exploratória, o repositório dos metadados e também as sugestões de alterações da Política de Padrões e Normas de BD. Nesta oportunidade foi apresentado à equipe da TI a primeira versão do repositório de metadados com dados dos bancos de dados de produção e de homologação com atualização periódica. Foi apresentado principalmente as funcionalidades de busca de bases de dados ou de campos de informações no repositório de metadados. Adicionalmente, foram apresentadas as sugestões de alteração na Política de Padrões e Normas de BD para sempre consultar o repositório de metadados antes da criação de um novo banco de dados e, assim, evitar futuros esquemas duplicados. Também foi apresentado o resultado da identificação dos esquemas similares.

As limitações, a conclusão e o direcionamento para futuros trabalhos são apresentados no próximo capítulo.

Capítulo 7

Conclusões

Este capítulo conclui este trabalho, faz um comparativo das soluções analisadas com esta pesquisa, lista as limitações e dá direcionamentos para os trabalhos futuros.

Existem várias soluções para comparar dois esquemas de banco de dados relacionais. Porém, não foram encontradas alternativas para comparar de forma automática vários esquemas de tamanhos e SGBDs variados. A Tabela 7.1 mostra a lista das soluções acadêmicas ou ferramentas de mercado encontradas para comparar esquemas, comparando características como a compatibilidade com diversos SGBDs, se a solução ou ferramenta faz o comparativo da estrutura do esquema, se faz a comparação de forma automática ou semiautomática e se possui escalabilidade para comparar muitos esquemas de tamanhos variados, incluindo esquemas de tamanhos pequeno a grande de acordo com a quantidade de objetos de banco de dados.

Tabela 7.1: Comparativo com a Abordagem Proposta.

Solução acadêmica ou ferramenta de mercado	Compatível com diversos SGBDs	Compara estrutura do esquema	Comparação automática ou semiautomática	Escalabilidade para muitos esquemas de tamanhos diversos
1. COMA [21]	✓	✓	✗	✗
2. CUPID [22]	✓	✓	✗	✗
3. ARTEMIS [23]	✓	✓	✗	✗
4. SEMINT [24]	✓	✗	✓	✗
5. GeRoMe [25]	✓	✓	✗	✗
6. SMART [26]	✓	✗	✓	✗
7. ASID [27]	✓	✗	✓	✗

Table 7.1 continued from previous page

Solução acadêmica ou ferramenta de mercado	Compatível com diversos SGBDs	Compara estrutura do esquema	Comparação automática ou semiautomática	Escalabilidade para muitos esquemas de tamanhos diversos
8. Dal Bianco [28]	✓	✓	✓	✗
9. DTM [29]	✗	✓	✗	✗
10. DBA xPress [30]	✗	✓	✗	✗
11. dbForge [31]	✗	✓	✗	✗
12. SQL Examiner [32]	✓	✓	✗	✗
13. ALTOVA [33]	✓	✓	✗	✗
14. DB Comparer [34]	✗	✓	✗	✗
15. XSQL Software [35]	✗	✓	✗	✗
16. Apex SQL Diff [36]	✗	✓	✗	✗
17. RedGate [37]	✗	✓	✗	✗
18. MySQL Workbench [38]	✗	✓	✗	✗
17. SQLyog's [39]	✗	✓	✗	✗
18. Acqua Data Studio [40]	✓	✓	✗	✗
19. Visual Studio [41]	✗	✓	✗	✗
20. IDERA [42]	✗	✓	✗	✗
21. DBSolo [43]	✓	✓	✗	✗
Abordagem proposta	✓	✓	✓	✓

Este estudo propôs o uso dos metadados para criar a primeira versão de um catálogo de dados e apresentou 3 diferentes abordagens para comparar automaticamente os esquemas e descobrir estruturas similares de esquemas a partir de seus metadados. Para validar as abordagens, foi realizado um experimento com 354 esquemas reais de diferentes tamanhos e de uma diversidade de SGBDs.

No experimento, a primeira abordagem teve a maior complexidade e não alcançou o objetivo de comparar todos os esquemas. A segunda abordagem teve um bom resultado para esquemas de tamanho pequeno a médio e pode ser uma boa opção para os casos em que os tamanhos dos esquemas não são maiores do que a memória para processar o comparativo, mesmo que a quantidade de esquemas seja elevada. Finalmente, a terceira abordagem teve uma ótima performance e concluiu todas as comparações de esquemas, até mesmo dos esquemas grandes. Porém, na terceira abordagem deve-se ter o cuidado na escolha do esquema base porque esta escolha influencia na duração do processamento

do comparativo dos esquemas e pode-se tornar inviável se a escolha do esquema base for um esquema grande.

Os resultados facilitam a comparação de esquemas de tamanho pequeno a grande, se tornando uma alternativa para corporações que precisam iniciar um catálogo de dados ou que precisam descobrir se mantém esquemas similares de bancos de dados relacionais. Os metadados extraídos são utilizados também para a criação do repositório de metadados que representa a versão inicial de um catálogo de dados da organização.

Esta pesquisa tem aplicabilidade em instituições governamentais e privadas que armazenam vários esquemas de bancos de dados, sendo útil por exemplo na integração de dados em situações como junção de ministérios ou compra de uma empresa privada por outra com negócios semelhantes.

Algumas limitações foram identificadas em momentos diferentes no decorrer desta pesquisa. Para a varredura na rede do Ministério realizada para descobrir os servidores de bancos de dados foi necessário ter permissões de administrador. Para a extração dos metadados dos objetos de banco de dados foi necessário ter permissões de administração em todos os bancos de dados acessados. A extração dos metadados técnicos foi realizada fora do horário de expediente para não sobrecarregar os bancos de dados durante o horário de maior uso. Informações detalhadas dos servidores de banco de dados como o endereço IP, o nome do banco de dados e outros metadados dos objetos de bancos de dados podem ser considerados sensíveis em relação à segurança da informação. Por isso, o resultado das análises tiveram as informações sensíveis mascaradas para evitar possíveis ataques aos bancos de dados. Numa parte do tratamento dos dados, especificamente na padronização da nomenclatura, foi considerado o contexto e nomenclatura utilizada pela organização onde o experimento foi realizado. Assim, para usar esta solução em outra organização seria necessário rever as *stopwords* e nomenclatura usada para nomear os esquemas, tabelas e colunas. Por último, não foram encontradas bibliotecas em R ou em outra tecnologia de *dashboard* e visualização para criar um dendrograma interativo capaz de dinamicamente mostrar e atualizar os grupos dos esquemas mais similares de acordo com que o *threshold* fosse movido ou arrastado na régua de distância entre a similaridade dos esquemas. Por isso, em cada alteração do *threshold*, foi necessário recalcular os grupos e a quantidade de esquemas similares. Assim, existe uma oportunidade de desenvolver o gráfico de dendrograma interativo para facilitar a visualização e subgrupos de forma dinâmica à medida em que se altera o *threshold*.

Estudos futuros podem considerar a melhoria do processo de comparação para executar em processamento paralelo a preparação dos dados e usar estratégias de *BigMemory* que possibilitem o uso mais memória *RAM* do que a existente no computador para a comparação de esquemas muito grandes. Além disso, os trabalhos futuros incluem a

divulgação no Sistema de Administração dos Recursos de Tecnologia da Informação (SISP) e também a submissão de artigo para o *Workshop* de *Data Integration*.

Referências

[1] M. Temer, A. Moraes, F. C. Filho, D. Oliveira, and G. Mendonça, "Lei Nº 13.341," 2016. [Online]. Available: http://www.planalto.gov.br/ccivil_03/_Ato2015-2018/2016/Lei/L13341.htm 1

[2] P. Christen, *Data matching: concepts and techniques for record linkage, entity resolution, and duplicate detection.* Springer, 2012. 2, 8

[3] C. Ritchie, *Relational Database Principles*, 2nd ed. Thompson, 2002. 2

[4] Ministério do Desenvolvimento Social MDS, "Plano de Dados Abertos," 2017. [Online]. Available: http://www.mds.gov.br/webarquivos/arquivo/acesso_informacao/transparencia/plano_dados_abertos.pdf 3

[5] Ministério do Planejamento, Orçamento e Gestão - MPOG, "Estratégia de Governança Digital - EGD da Administração Pública Federal 2016-19," Jan. 2016. [Online]. Available: http://www.planejamento.gov.br/EGD/arquivos/Estrategia-de-Governanca-Digital.pdf 3

[6] A. Vaduva and T. Vetterli, "Metadata management for data warehousing: an overview," *International Journal of Cooperative Information Systems*, 2001. 7

[7] M. Mosley, M. Brackett, D. International, S. Earley, and D. Henderson, *The DAMA guide to the data management body of knowledge: DAMA-DMBOK Guide.* Technics Publications, 2010. 7

[8] E. Rahm and P. A. Bernstein, "A survey of approaches to automatic schema matching," *the VLDB Journal*, vol. 10, no. 4, pp. 334–350, 2001. 8, 9

[9] P. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining.* Addison-Wesley, 2013. 9

[10] N. Friedman, D. Geiger, and M. Goldszmidt, "Bayesian network classifiers," *Machine learning*, vol. 29, 1997. 9

[11] L. A. Silva, S. M. Peres, and C. Boscarioli, *Introdução à Mineração de Dados - Com Aplicações em R*, 1st ed. Rio de Janeiro, Brasil: Elsevier, 2016. 9

[12] L. Getoor and B. Taskar, *Introduction to Statistical Relational Learning.* MIT Press, 2007. 9

[13] M. Nickel, K. Murphy, V. Tresp, and E. Gabrilovich, "A review of relational machine learning for knowledge graphs," *Proceedings of the IEEE*, 2016. 10

[14] H. Khosravi and B. Bina, "A Survey on Statistical Relational Learning." in *Canadian Conference on AI*. Springer, 2010. 10

[15] C.-F. Tsai, W.-C. Lin, and S.-W. Ke, "Big data mining with parallel computing: A comparison of distributed and MapReduce methodologies," *Journal of Systems and Software*, vol. 122, pp. 83–93, 2016. 10

[16] L. Zeng, L. Li, L. Duan, K. Lu, Z. Shi, M. Wang, W. Wu, and P. Luo, "Distributed data mining: a survey," *Information Technology and Management*, 2012. 10

[17] P. Chapman, J. Clinton, K. Randy, T. Khabaza, T. Reinartz, C. Shearer, and R. Wirth, "CRISP-DM 1.0," 2000. 11

[18] "Web of Science," 2018. [Online]. Available: http://login.webofknowledge.com/ 12

[19] "Google Scholar," 2018. [Online]. Available: https://scholar.google.com.br/ 12

[20] "Google," 2018. [Online]. Available: https://www.google.com/ 12

[21] D. Aumueller, H.-H. Do, S. Massmann, and E. Rahm, "Schema and ontology matching with COMA++," in *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*. ACM, 2005. 12, 13, 54

[22] J. Madhavan, P. A. Bernstein, and E. Rahm, "Generic schema matching with cupid," in *vldb*, 2001. 12, 13, 54

[23] J. Berlin and A. Motro, "Database schema matching using machine learning with feature selection," in *Proceedings of the 14th International Conference on Advanced Information Systems Engineering*. Springer-Verlag, 2002. 12, 13, 54

[24] W.-S. Li and C. Clifton, "SEMINT: A tool for identifying attribute correspondences in heterogeneous databases using neural networks," *Data & Knowledge Engineering*, vol. 33, 2000. 12, 54

[25] D. Kensche, C. Quix, X. Li, and Y. Li, "GeRoMeSuite: A system for holistic generic model management," in *Proceedings of the 33rd international conference on Very large data bases*. VLDB Endowment, 2007. 12, 54

[26] Q. V. H. Nguyen, T. T. Nguyen, V. T. Chau, T. K. Wijaya, Z. Miklos, K. Aberer, A. Gal, and M. Weidlich, "SMART: A tool for analyzing and reconciling schema matching networks," in *Data Engineering (ICDE), 2015 IEEE 31st International Conference on*. IEEE, 2015. 12, 54

[27] N. Bozovic and V. Vassalos, "Two-phase schema matching in real world relational databases," in *24th International Conference on Data Engineering Workshop - ICDEW*. IEEE, 2008. 13, 14, 54

[28] G. Dal Bianco, R. Galante, M. A. Gonçalves, S. Canuto, and C. A. Heuser, "A practical and effective sampling selection strategy for large scale deduplication," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 9, 2015. 13, 14, 55

[29] "DTM schema comparer," 2018. [Online]. Available: http://www.sqledit.com/scmp/compare-db2.html 15, 55

[30] "DBAxPRESS," 2018. [Online]. Available: https://www.sentryone.com/products/pragmatic-workbench/dba-xpress/sql-development-monitoring-tools 15, 55

[31] "DbForge schema compare," 2018. [Online]. Available: https://mariadb.com/kb/en/library/dbforge-schema-compare/ 15, 55

[32] "SQL Examiner Suite," 2018. [Online]. Available: http://www.sqlaccessories.com/sql-examiner-suite/ 15, 55

[33] "ALTOVA DatabaseSpy," 2018. [Online]. Available: https://www.altova.com/databasespy 15, 55

[34] "DB Comparer," 2018. [Online]. Available: http://dbcomparer.com 15, 55

[35] "XSQL schema compare," 2018. [Online]. Available: https://www.xsql.com/products/sql_server_schema_compare/default.aspx 15, 16, 55

[36] "ApexSQL Diff schema compare SQL Server," 2018. [Online]. Available: https://www.apexsql.com/sql_tools_diff.aspx 15, 55

[37] "RedGate schema compare for Oracle," 2018. [Online]. Available: https://www.red-gate.com/products/oracle-development/schema-compare-for-oracle/index 15, 55

[38] "MySQL WorkBench to compare and report differences in catalogs," 2018. [Online]. Available: https://dev.mysql.com/doc/workbench/en/wb-database-diff-report.html 15, 55

[39] "SQLYOG schema synchronization," 2018. [Online]. Available: https://www.webyog.com/product/sqlyog 15, 55

[40] "AquaFold Aqua Data Studio Schema Compare and Tabs Compare," 2018. [Online]. Available: https://www.aquafold.com/aquadatastudio/schema_sql_compare 15, 16, 55

[41] "Visual Studio schema compare," 2018. [Online]. Available: http://www.oracle.com/webfolder/technetwork/tutorials/obe/db/12c/r1/appdev/dotnet/SchemaCompare/index.html 15, 55

[42] "IDERA SQL comparison toolset," 2018. [Online]. Available: https://www.idera.com/productssolutions/sqlserver/sqlcomparisontoolset 15, 55

[43] "DBSolo schema comparison tool," 2018. [Online]. Available: http://www.dbsolo.com/schema_comparison.html 15, 16, 55

[44] E. L. Silva and E. M. Menezes, *Metodologia de Pesquisa e Elaboração de Dissertação*, 4th ed. Florianópolis: UFSC. 18

[45] R. Wirth and J. Hipp, "CRISP-DM: Towards a standard process model for data mining," in *Proceedings of the 4th international conference on the practical applications of knowledge discovery and data mining*, 2000. 18

[46] S. Tata and J. M. Patel, "Estimating the selectivity of TF-IDF based cosine similarity predicates," *ACM Sigmod Record*, vol. 36, no. 2, pp. 7–12, 2007. 26

[47] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining*, 1st ed. University of Minnesota: Pearson, 2005. 42, 43

[48] MDS, "Seminário de Boas Práticas em Dados Abertos do MDS," Nov. 2017. [Online]. Available: https://aplicacoes.mds.gov.br/sagirmps/hc/index.php?grupo=352

[49] D. Reis, "As Bases de Dados do MDS," Nov. 2017. [Online]. Available: https://goo.gl/aQsQEi 52

[50] D. G. Reis, R. N. Carvalho, R. S. Carvalho, and M. Ladeira, "Two-phase Parallel Learning to Identify Similar Structures Among Relational Databases," in *Machine Learning and Applications (ICMLA), 2017 16th IEEE International Conference on*. IEEE, 2017, pp. 1020–1023. 52

Apêndice A

SQL de Extração dos Metadados

A.1 Formato da Extração dos Metadados

```
NO_SERVIDOR | NU_IP | DS_SGBD | DS_SISTEMA_OPERACIONAL |
ST_ATIVO | TP_AMBIENTE | NO_BANCO | NU_TAMANHO_BANCO |
DS_DICIONARIO_BANCO | DT_IMPORTACAO | NU_PORTA |
DS_VERSAO_SGBD | NO_TABELA | DS_DICIONARIO_TABELA |
NU_TAMANHO_TAB | DT_CRIACAO_TAB | NO_SCHEMA |
NO_TABLESPACE | QT_REGISTRO | QT_COLUNA | NO_COLUNA |
TP_COLUNA | NU_TAMANHO_COL | ST_NULL | ST_FK |
DS_DICIONARIO_COLUNA | DT_CRIACAO_COL
```

A.2 Exemplo do SQL do DB2

```
db2 "catalog tcpip node <nó> remote <servidor> server 50001"
db2 "catalog db <banco> as PD02 at node <nó> with 'Producao'"
db2 terminate
db2 connect to PD02 user <usuario> using <senha>
db2 "export to $path/metadados_<servidor>.txt of del modified by coldel| nochardel

SELECT
  '<servidor>' AS NO_SERVIDOR,
  '<ip>' AS NU_IP,
  'DB2' AS DS_SGBD,
  '<OS>' AS DS_SISTEMA_OPERACIONAL,
  'S' AS ST_ATIVO,
  'P' AS TP_AMBIENTE,
  '<banco>' AS NO_BANCO,
```

```
  tdb.nu_tamanho_banco as NU_TAMANHO_BANCO,
  'Produção dedicada as aplicações <aplicacao>' as DS_DICIONARIO_BANCO,
   to_char(SYSDATE,'YYYY-MM-DD HH24:MI:SS') as DT_IMPORTACAO,
  '50001' AS NU_PORTA,
  'LUW AESE v10.1 FP5' as DS_VERSAO_SGBD,
  CHAR(C.TABNAME,60) AS NO_TABELA,
  CASE WHEN COM_TAB.COMMENTS IS NULL THEN 'Não Informado'
   ELSE COM_TAB.COMMENTS END AS ds_dicionario_tabela,
  trunc(((SUM(A.DATA_OBJECT_P_SIZE)+SUM(A.INDEX_OBJECT_P_SIZE)+
  SUM(A.LONG_OBJECT_P_SIZE)+SUM(A.LOB_OBJECT_P_SIZE)+
  SUM(A.XML_OBJECT_P_SIZE))/1024),0) as nu_tamanho_tab,
  to_char(Q.CREATE_TIME,'YYYY-MM-DD HH24:MI:SS') as DT_CRIACAO_TAB,
  char(C.TABSCHEMA,25) NO_SCHEMA,
  TBS.TBSP_NAME AS no_tablespace,
  Q.CARD AS qt_registro,
  Q.COLCOUNT AS qt_coluna,
  CHAR(C.COLNAME,60) AS no_coluna,
  CHAR(C.TYPENAME,20) AS tp_coluna,
  C.LENGTH AS nu_tamanho_col,
  CASE WHEN C.NULLS = 'Y' THEN 'S' ELSE 'N' END AS st_null,
  CASE WHEN R.FKCOLUMN_NAME IS NOT NULL then 'S' ELSE 'N' END AS st_fk,
  CASE WHEN COM_COL.COMMENTS IS NULL THEN 'Não Informado'
   ELSE COM_COL.COMMENTS END AS ds_dicionario_coluna,
  '1970-01-01 00:00:00' as dt_criacao_col
from syscat.columns C
CROSS JOIN (SELECT trunc((SUM(TBSP_TOTAL_SIZE_KB)/1024),0) AS NU_TAMANHO_BANCO
FROM SYSIBMADM.TBSP_UTILIZATION TBSUTIL) tdb
inner join syscat.tables Q on Q.tabschema=C.tabschema and Q.tabname=C.tabname
inner join sysibmadm.ALL_TAB_COMMENTS COM_TAB on COM_TAB.TABLE_SCHEMA=Q.tabschema
and COM_TAB.TABLE_NAME=Q.tabname
inner join sysibmadm.ALL_COL_COMMENTS COM_COL on COM_COL.TABLE_SCHEMA=C.tabschema
and COM_COL.TABLE_NAME=C.tabname and COM_COL.COLUMN_NAME=C.colname
inner join SYSIBMADM.ADMINTABINFO A on A.tabschema=C.tabschema
and A.tabname=C.tabname
LEFT JOIN TABLE (MON_GET_TABLE('','',-2)) AS T ON T.TABNAME = q.TABNAME
AND T.TABSCHEMA = q.TABSCHEMA
LEFT JOIN TABLE(MON_GET_TABLESPACE('',-2)) AS TBS ON T.TBSP_ID = TBS.TBSP_ID
left join sysibm.SQLFOREIGNKEYS R on C.TABSCHEMA=R.FKTABLE_SCHEM
and C.tabname=R.FKTABLE_NAME and C.colname=R.FKCOLUMN_NAME
where Q.type in ('S','T') and Q.tabschema not like 'SYS%'
```

```
AND Q.TABSCHEMA <> 'DB2INST1' and Q.TABSCHEMA <> 'IBM_RTMON'
group by tdb.NU_TAMANHO_BANCO, C.tabname, COM_TAB.COMMENTS,
Q.create_time, C.TABSCHEMA, TBS.TBSP_NAME, Q.card, Q.colcount,
    C.colname, C.typename,c.length, c.nulls, r.FKCOLUMN_NAME, COM_COL.COMMENTS"

db2 "import from $path/metadados_<servidor>.txt of del
modified by coldel| nochardel replace into DB2INST1.TB_METADADOS"

db2 -x "select char(tabschema,20), char(tabname,60), char(tbspace,30)
         from syscat.tables where tabname
         in (select no_tabela from db2inst1.tb_metadados
         where NO_TABLESPACE is NULL) " | awk '{print "update db2inst1.tb_metadados
         set NO_TABLESPACE="$3 " where no_schema="$1 "
         and no_tabela="$2}' > $path/saida ;
sed "s/NO_TABLESPACE=/NO_TABLESPACE='/g" $path/saida > $path/saida.2 ;
sed "s/ where no_schema=/' where no_schema='/g" $path/saida.2 > $path/saida.3 ;
sed "s/ and no_tabela=/' and no_tabela='/g" $path/saida.3 > $path/saida.4 ;
sed "s/$/';/g" $path/saida.4 > $path/saida.5 ;
db2 -tvf $path/saida.5 > $path/atl_tbs_log_PD02.sql

# TAMANHO PARA TABLESPACES PARTICIONADAS
        somapd02=`db2 -x "SELECT trunc(((SUM(DATA_OBJECT_P_SIZE)+
                        SUM(INDEX_OBJECT_P_SIZE)+ SUM(LONG_OBJECT_P_SIZE)+
                        SUM(LOB_OBJECT_P_SIZE)+ SUM(XML_OBJECT_P_SIZE))/1024),0)
                        FROM
                        SYSIBMADM.ADMINTABINFO where TABSCHEMA='SISPAA' and
                        TABNAME='TB_LOG_VARCHAR' group by TABSCHEMA, TABNAME"`
        db2 "update db2inst1.tb_metadados set NU_TAMANHO_TAB=${somapd02}
         where no_schema='<nome_esquema>' and no_tabela='TB_LOG_VARCHAR'"

db2 "update db2inst1.tb_metadados set DT_CRIACAO_COL='1970-01-01 00:00:00'
WHERE DT_CRIACAO_COL is NULL"
db2 "update db2inst1.tb_metadados set DS_DICIONARIO_TABELA='Não informado'
WHERE DS_DICIONARIO_TABELA is NULL"
db2 "update db2inst1.tb_metadados set DS_DICIONARIO_COLUNA='Não informado'
WHERE DS_DICIONARIO_COLUNA is NULL"

db2 "export to $path/metadados_DB2_SUDBPD02.txt of del modified by coldel|
nochardel codepage=819 select * from DB2INST1.TB_METADADOS"
cat $path/cabecalho.txt $path/metadados_DB2_SUDBPD02.txt >
```

```
$path/log/metadados_DB2_SUDBPD02.txt
db2 connect reset

# UNCATALOG DBs E NODEs
db2 "uncatalog node <nó>"
db2 "uncatalog db <database>"
rm $path/said*
rm $path/metadados_*
rm $path/metadados_*
```

A.3 Exemplo do SQL do MySQL

```
Cria o arquivo listaServidores.txt:
# vi listaServidores.txt

O arquivo listaServidores.txt possui o seguinte conteúdo:
ID:NOME:AMBIENTE
1:<servidor:H
2:<servidor>:P

Cria arquivo SelectMetaDados.sql:
# vi SelectMetaDados.sql

O arquivo SelectMetaDados.sql possui o seguinte conteúdo:
Select  @@hostname      as no_servidor
        ,CASE WHEN @@hostname = '<servidor>' THEN '<IP>'
              WHEN @@hostname = '<servidor>' THEN '<IP>'
              ELSE 'desconhecido'
        END             AS nu_ip
        ,'MySql'        as ds_sgbd
        ,'linux'        as ds_sistema_operacional
        ,'s'            as st_ativo
        ,CASE WHEN @@hostname = '<servidor>' THEN 'h'
              WHEN @@hostname = '<servidor>' THEN 'p'
              ELSE 'X'
        END             as tp_ambiente
        ,TAB.TABLE_SCHEMA               AS no_banco
        ,TAMANHO_BANCO.nu_tamanho_banco as nu_tamanho_banco
        ,'Não Informado'                as ds_dicionario_banco
```

```
        ,NOW()                          as dt_importacao
        ,@@port                         as nu_porta
        ,@@version                      as ds_versao_sgbd
        , COL.TABLE_NAME                AS no_tabela
        ,IFNULL(IF (COL.COLUMN_COMMENT = '','Não Informado',COL.COLUMN_COMMENT),
         'Não Informado') AS ds_dicionario_tabela
        ,IFNULL(ROUND((TAB.DATA_LENGTH/1024)/1024,2),0) AS nu_tamanho_tab
        ,'1970-01-01 00:00:00'                  as dt_criacao_tab
        ,TAB.TABLE_SCHEMA AS no_schema
        ,IFNULL(IF(TAB.TABLE_CATALOG = '' , 'Não Informado',TAB.TABLE_CATALOG),
                  'Não Informado') AS no_tablespace
        ,IFNULL(TAB.TABLE_ROWS,0)       AS qt_registro
        ,IFNULL(COL_QTDE.QTDE_COL_TAB,0) AS qt_coluna
        ,COL.COLUMN_NAME                AS no_coluna
        ,COL.DATA_TYPE                  AS tp_coluna
        ,CASE WHEN COL.DATA_TYPE IN ('int','bigint' )
        THEN COL.NUMERIC_PRECISION
             WHEN COL.DATA_TYPE IN ('char','varchar' )
             THEN COL.CHARACTER_MAXIMUM_LENGTH
             ELSE 0 END              AS nu_tamanho_Col
        ,CASE WHEN COL.is_nullable = 'NO' THEN 'n' ELSE 's' END         AS st_null
      ,CASE WHEN COL_FK.COLUMN_NAME IS NULL THEN 'n' ELSE 's' END       AS st_fk
      ,IFNULL(IF (COL.COLUMN_COMMENT = '','Não Informado',COL.COLUMN_COMMENT),
          'Não Informado') AS ds_dicionario_coluna
      ,'1970-01-01 00:00:00'                    as dt_criacao_col
from information_schema.columns COL
inner JOIN information_schema.tables TAB
    ON TAB.TABLE_SCHEMA = COL.TABLE_SCHEMA AND
       TAB.TABLE_NAME = COL.TABLE_NAME
inner join (SELECT table_schema, round((sum(DATA_LENGTH)/1024)/1024,2)
as nu_tamanho_banco
FROM information_schema.tables group by table_schema) as TAMANHO_BANCO
 on TAMANHO_BANCO.table_schema = TAB.table_schema
LEFT join (select COL.TABLE_SCHEMA
                 ,COL.TABLE_NAME
                 ,COUNT(1) AS QTDE_COL_TAB
            from information_schema.columns COL
            GROUP BY COL.TABLE_SCHEMA, COL.TABLE_NAME
            ) AS COL_QTDE
    ON  COL_QTDE.TABLE_SCHEMA = TAB.TABLE_SCHEMA AND
```

```
        COL_QTDE.TABLE_NAME = TAB.TABLE_NAME

LEFT JOIN (SELECT key_col.TABLE_SCHEMA
                 ,key_col.TABLE_NAME
                 ,key_col.COLUMN_NAME
            from information_schema.key_column_usage key_col
            where key_col.REFERENCED_TABLE_SCHEMA is not null AND
                  key_col.CONSTRAINT_NAME <> 'PRIMARY' )
      AS COL_FK
    on COL_FK.TABLE_SCHEMA = COL.TABLE_SCHEMA AND
       COL_FK.TABLE_NAME = COL.TABLE_NAME AND
       COL_FK.COLUMN_NAME = COL.COLUMN_NAME
where COL.table_schema not in ('information_schema' ,'mysql')
ORDER BY COL.TABLE_SCHEMA, COL.TABLE_NAME,COL.ORDINAL_POSITION;
```

Cria o arquivo metadadospost.sh:

```
# vi metadadospost.sh
```

O arquivo metadadospost.sh possui o seguinte conteúdo:

```
#/bin/bash
# Programa que conecta nos servidores e extrai o meta dado de todos os bancos.

# ----------------------Configurações ---------------------------------
SEP=:                         #define o separador de leitura dos arquivos
SQL="/home/mysql/ScriptMetaDados/SelectMetaDados.sql"
SQLBANCO="/home/mysql/ScriptMetaDados/SelectBanco.sql"
SERVIDORES="/home/mysql/ScriptMetaDados/listaServidores.txt"
USUARIO_LOGIN="<usuario>"
SENHA="<senha>"
PSQL="/usr/bin/mysql"
CABECALHO_ARQ="no_servidor|nu_ip|ds_sgbd|ds_sistema_operacional|
st_ativo|tp_ambiente|no_banco|nu_tamanho_banco|ds_dicionario_banco|
dt_importacao|nu_porta|ds_versao_sgbd|no_tabela|ds_dicionario_tabela|
nu_tamanho_tab|dt_criacao_tab|no_schema|no_tablespace|qt_registro|
qt_coluna|no_coluna|tp_coluna|nu_tamanho_col|st_null|st_fk|
ds_dicionario_coluna|dt_criacao_col"
#-----------------------Bloco Principal-------------------------------
# O arquivo com os servidores deve estar definido
[ "$SERVIDORES" ] || {
```

```
    echo "O arquivo com a lista de servidores deve ser definodo.
          Use a variável SERVIDOR."
    return 1
}

# O arquivo com os servidores pode ser lido e gravado
[ -r "$SERVIDORES" -a -w "$SERVIDORES" ] || {
    echo "Arquivo com a lista de servidores esta travado.
          Confira as permissoes de leitura e escrita."
    return 1
}

#Pega a quantidade de servidores que deve ser lido
QTD_SERVIDORES=$(tail "$SERVIDORES" -n 1 | cut -d $SEP -f 1)
for X in $(seq "$QTD_SERVIDORES")
do
        # Pega o nome do servidor
        SERVIDOR=$(grep -i "^$X$SEP" "$SERVIDORES" | cut -d $SEP -f 2)

        #Descobrindo a versão do Postgres
        #VERSAO=$(echo "$(echo "$("$PSQL" -h "$SERVIDOR" -U
          "$USUARIO_LOGIN"  -t -c "select version();")" |
          cut -d " " -f 3)" | cut -d "." -f 1)

        #Monta o nome do arquivo
        ARQUIVO_META_DADOS="${CAMINHO_ARQUIVO_MD}${SERVIDOR}_METADADOS.TXT"

        #Carrega o cabeçalho do arquivo e zera o arquivo se ele ja existir.
        echo "$CABECALHO_ARQ" > $ARQUIVO_META_DADOS
        echo "Servidor $SERVIDOR Iniciado!"
        # Pega todos os bancos de cada servidor
        #faz a consulta de meta dados na versao 9
        #echo "$("$PSQL" -h "$SERVIDOR" -U "$USUARIO_LOGIN" -d "$database"
           -t -A -F"|" -f "$SQL")" >> $ARQUIVO_META_DADOS
        echo "$("$PSQL" -u "$USUARIO_LOGIN" -p"$SENHA" -h "$SERVIDOR" <
           "$SQL" | sed 's/\t/\|/g')" > $ARQUIVO_META_DADOS
        echo "Servidor $SERVIDOR concluido!"
done
echo "Done!!"
```

A.4 Exemplo do SQL do Oracle

```
Cria o arquivo listaServidores.txt:
# vi listaServidores.txt

O arquivo listaServidores.txt possui o seguinte conteúdo:
ID:NOME:AMBIENTE
1:<servidor>:P
2:<servidor>:H

Cria arquivo SelectBanco.sql:
# vi SelectBanco.sql

O arquivo SelectBanco.sql possui o seguinte conteúdo:
set feedback off
set heading off
set termout off
Set echo off
set verify off
Set Pagesize 0
Set linesize 1000
Set Trimspool on
set colsep '|'
spool &1
SELECT NAME, LOG_MODE FROM V$DATABASE;
spool off
exit;

Cria arquivo SelectMetaDados.sql:
# vi SelectMetaDados.sql

O arquivo SelectMetaDados.sql possui o seguinte conteúdo:
set feedback off
set echo off
set headsep off
set heading off
set termout off
set verify off
set pagesize 0
set trimspool on
```

```
set linesize 1000
Alter Session Set nls_language='BRAZILIAN PORTUGUESE';
Alter Session Set NLS_TERRITORY = 'BRAZIL';
Alter Session Set NLS_NUMERIC_CHARACTERS=',';
ALTER SESSION SET NLS_DATE_FORMAT = 'YYYY-MM-DD HH24:MI:SS';
ALTER SESSION SET NLS_CHARACTERSET = 'WE8ISO8859P1';
spool &1
SELECT TRIM(TRIM(no_servidor) || '|'||
TRIM(nu_ip) || '|'||
TRIM(ds_sgbd) || '|'||
TRIM(ds_sistema_operacional) || '|'||
TRIM(st_ativo) || '|'||
TRIM(tp_ambiente) || '|'||
TRIM(no_banco) || '|'||
TRIM(nu_tamanho_banco) || '|'||
TRIM(ds_dicionario_banco) || '|'||
TRIM(dt_importacao) || '|'||
TRIM(nu_porta) || '|'||
TRIM(ds_versao_sgbd) || '|'||
TRIM(no_tabela) || '|'||
translate(TRIM(ds_dicionario_tabela),chr(10)||chr(13)||'|',' ') || '|'||
TRIM(nu_tamanho_tab) || '|'||
TRIM(dt_criacao_tab) || '|'||
TRIM(no_schema) || '|'||
TRIM(no_tablespace) || '|'||
TRIM(qt_registro) || '|'||
TRIM(qt_coluna) || '|'||
TRIM(no_coluna) || '|'||
TRIM(tp_coluna) || '|'||
TRIM(nu_tamanho_Col) || '|'||
TRIM(st_null) || '|'||
TRIM(st_fk) || '|'||
translate(TRIM(ds_dicionario_coluna),chr(10)||chr(13)||'|',' ') || '|'||
TRIM(dt_criacao_col)) AS META_DADOS
FROM(
SELECT
        (SELECT  UPPER(HOST_NAME) FROM V$INSTANCE)      AS no_servidor
        ,UTL_INADDR.get_host_address()                  AS nu_ip
        ,'ORACLE'                                       AS ds_sgbd
        ,'Linux'                                        AS ds_sistema_operacional
```

```
        ,'s'                                            AS st_ativo
        ,'p'                                            AS tp_ambiente
        --
        ,(SELECT NAME FROM V$DATABASE)                  AS no_banco
        ,(SELECT SUM(SG.BYTES)/1024/1024  AS nu_tamanho_banco FROM DBA_SEGMENTS SG)
         AS nu_tamanho_banco
        ,'Não Informado'                                AS ds_dicionario_banco
        ,SYSDATE                                        AS dt_importacao
        ,NVL((select substr(value, instr(VALUE, 'PORT=')+5, 4)
         FROM V$LISTENER_NETWORK where type ='REMOTE LISTENER'),
            (select substr(value, instr(VALUE, 'PORT=')+5, 4)
             FROM V$LISTENER_NETWORK where type ='LOCAL LISTENER'))  AS nu_porta
        ,(SELECT VERSION FROM V$INSTANCE)               AS ds_versao_sgbd
        --
        ,TABELA.TABLE_NAME                              AS no_tabela
        ,NVL(TAB_COMM.COMMENTS,'Não Informado')         AS ds_dicionario_tabela
        ,TAB_TAMANHO.BYTES /1024/1024                   AS nu_tamanho_tab
        ,DATA_TABELA.CREATED                            AS dt_criacao_tab
        ,TABELA.OWNER                                   AS no_schema
        ,NVL(TABELA.TABLESPACE_NAME,'SYSTEM')           AS no_tablespace
        ,TABELA.NUM_ROWS                                AS qt_registro
        ,COL_TAB.QTDE_COLUNAS                           AS qt_coluna
        --
        ,COLUNA.COLUMN_NAME                             AS no_coluna
        ,COLUNA.DATA_TYPE                               AS tp_coluna
        ,COLUNA.DATA_LENGTH                             AS nu_tamanho_Col
        ,CASE WHEN COLUNA.NULLABLE = 'Not Null' THEN 's' ELSE  'n' END  AS st_null
        ,CASE WHEN COL_FK.COLUMN_NAME IS NULL THEN 'n' ELSE 's' END     AS st_fk
        ,NVL(COLUNA_COMM.COMMENTS,'Não Informado')      AS ds_dicionario_coluna
        ,'1970-01-01 00:00:00' as dt_criacao_col
FROM            ALL_TABLES      TABELA
INNER JOIN      ALL_TAB_COLUMNS COLUNA
                ON      TABELA.OWNER            =       COLUNA.OWNER
                AND     TABELA.TABLE_NAME       =       COLUNA.TABLE_NAME
LEFT JOIN       ALL_TAB_COMMENTS TAB_COMM
                ON      TAB_COMM.OWNER          =       TABELA.OWNER
                AND     TAB_COMM.TABLE_NAME     =       TABELA.TABLE_NAME
--tamnho da Tabela
INNER JOIN      (SELECT OWNER, SEGMENT_NAME,  SUM(BYTES) AS BYTES FROM  DBA_SEGMENTS
                GROUP BY OWNER, SEGMENT_NAME)    TAB_TAMANHO
```

```
                ON      TAB_TAMANHO.OWNER       =       TABELA.OWNER
                AND     TAB_TAMANHO.SEGMENT_NAME =      TABELA.TABLE_NAME
INNER JOIN      DBA_OBJECTS     DATA_TABELA
                ON      DATA_TABELA.OBJECT_NAME = TABELA.TABLE_NAME
                AND     OBJECT_TYPE             = 'TABLE'
                AND     DATA_TABELA.OWNER       = TABELA.OWNER
-- IDENTIFICAR QTDE COLUNAS NA TABELA
INNER JOIN       (SELECT COL.OWNER ,COL.TABLE_NAME , COUNT(1) AS QTDE_COLUNAS
                        FROM ALL_TAB_COLUMNS COL
                        GROUP BY COL.OWNER ,COL.TABLE_NAME) COL_TAB
                ON      COL_TAB.OWNER           =       TABELA.OWNER
                AND     COL_TAB.TABLE_NAME      =       TABELA.TABLE_NAME
-- IDENTIFICAR SE COLUNA É UMA FK
LEFT JOIN      (SELECT CC.OWNER, CC.CONSTRAINT_NAME, CC.TABLE_NAME, CC.COLUMN_NAME
                        FROM ALL_CONS_COLUMNS CC
                        WHERE CC.CONSTRAINT_NAME IN (SELECT CTR.CONSTRAINT_NAME
                        FROM ALL_CONSTRAINTS CTR
                        WHERE CTR.CONSTRAINT_TYPE = 'R'
                        AND CTR.TABLE_NAME = CC.TABLE_NAME )
                )               COL_FK
                ON      COL_FK.OWNER            = COLUNA.OWNER
                AND     COL_FK.TABLE_NAME       = COLUNA.TABLE_NAME
                AND     COL_FK.COLUMN_NAME      = COLUNA.COLUMN_NAME
INNER JOIN      ALL_COL_COMMENTS        COLUNA_COMM
                ON      COLUNA_COMM.OWNER       = COLUNA.OWNER
                AND     COLUNA_COMM.TABLE_NAME  = COLUNA.TABLE_NAME
                AND     COLUNA_COMM.COLUMN_NAME = COLUNA.COLUMN_NAME
);
spool off
exit;
```

A.5 Exemplo do SQL do PostgreSQL

```
Cria o arquivo listaServidores.txt:
# vi listaServidores.txt

O arquivo listaServidores.txt possui o seguinte conteúdo:
ID:NOME:AMBIENTE
1:<servidor>:H
```

```
2:<servidor>:P

Cria arquivo SelectBanco.sql:
# vi SelectBanco.sql

O arquivo SelectBanco.sql possui o seguinte conteúdo:
select dbs.datname from pg_database dbs
        where dbs.datistemplate = false
order by dbs.datname

Cria arquivo SelectMetaDados.sql:
# vi SelectMetaDados.sql

O arquivo SelectMetaDados.sql possui o seguinte conteúdo:
select
        (CASE inet_server_addr()
                WHEN '<IP>' then '<servidor>'
                WHEN '1<IP>' then '<servidor>'
                ELSE 'NAO TEM'  END)
                AS no_servidor
        , inet_server_addr()
        AS nu_ip
        , 'Não Informado'
        AS ds_sgbd
        , 'linux'
        as ds_sistema_operacional
        , 'S'
        as st_ativo
        , (CASE inet_server_addr()
                WHEN '<IP>' then 'H'
                WHEN '<IP>' then 'H'
                ELSE 'X'        END)
                as tp_ambiente

        ,current_database()
        as no_banco
        ,round((cast(pg_database_size(current_database())
        as numeric(30,2))/1024)/1024,2)
        as nu_tamanho_banco
        ,'Não Iinformado'
```

```
       as ds_dicionario_banco
      ,current_timestamp
      as dt_importacao
      ,inet_server_port()
      as nu_porta
      ,regexp_replace(replace(coalesce(version(), 'Não informado'),'|',' '),
      E'[\\n\\r]+', ' ', 'g')   as ds_versao_sgbd
       ,tabela.relname   as no_tabela
       ,regexp_replace(replace(coalesce(pg_catalog.obj_description(tabela.oid,
       'pg_class'),'Não informado'),'|',' '), E'[\\n\\r]+',' ', 'g')
                      as ds_dicionario_tabela
      ,round((cast(pg_total_relation_size(tabela.oid)as numeric(30,2))/1024)/1024,2)
                      as nu_tamanho_tab
      ,'1970-01-01 00:00:00'   as dt_criacao_tab
      , coalesce(coluna.table_schema           ,'Não informado')   as no_schema
      , coalesce(tablespace.spcname            ,'Não informado')   as no_tablespace
      , coalesce(tabela.reltuples::BIGINT      ,'0')               as qt_registro
      , (SELECT count(column_name) FROM information_schema.columns
       WHERE table_name = tabela.relname
                                          ) as qt_coluna
       ,coluna.column_name                   as no_coluna
       ,coalesce(coluna.data_type,'Não informado') as tp_coluna
       ,coalesce(
       (case  when coluna.data_type in ('character varying','character','boolean')
       then coluna.character_maximum_length
        when coluna.data_type in ('timestamp without time zone','time','date')
        then coluna.datetime_precision
        when coluna.data_type in ('numeric', 'smallint', 'bigint', 'integer')
        then coluna.numeric_precision
        when coluna.data_type in ('text')
        then coluna.character_octet_length
                      end)
              ,'0')                                              as nu_tamanho_col
       ,(CASE WHEN coluna.Is_Nullable = 'YES' THEN 'S' else 'N' end)  as st_null
       ,(CASE WHEN FK.conkey is null THEN 'N' else 'S' end)         as st_fk
     ,regexp_replace(replace(coalesce(pg_catalog.col_description(tabela.oid,
       coluna.ordinal_position), 'Não informado'),'|',' '),
       E'[\\n\\r]+', ' ', 'g' ) as ds_dicionario_coluna
       ,'1970-01-01 00:00:00' as dt_criacao_col
from          pg_catalog.pg_class             as tabela
```

```
  left join    information_schema.columns      as coluna
  on tabela.relname = coluna.table_name
  left join    pg_catalog.pg_tablespace
  as tablespace   on tablespace.oid = tabela.reltablespace
  left join    pg_constraint    as FK
  on FK.conrelid = tabela.oid
  and tabela.relname = coluna.table_name
  and coluna.ordinal_position = FK.conkey[1]
  and FK.contype = 'f'
where tabela.relkind = 'r'
and coluna.table_schema NOT IN ('pg_catalog', 'information_schema')

Cria arquivo SelectMetaDados8.sql:
# vi SelectMetaDados8.sql
O arquivo SelectMetaDados8.sql possui o seguinte conteúdo:
select (CASE inet_server_addr()
                WHEN '<IP>' then '<servidor>'
                WHEN '<IP>' then '<servidor>'
                ELSE 'NAO TEM'  END) AS no_servidor
        , inet_server_addr() AS nu_ip
        , 'Não Informado' AS ds_sgbd
        , 'linux' as ds_sistema_operacional
        , 'S' as st_ativo
        , (CASE inet_server_addr()
                WHEN '<IP>' then 'H'
                WHEN '<IP>' then 'H'
                ELSE 'X'        END)as tp_ambiente
      ,col.table_catalog as no_banco
      ,round((cast(pg_database_size(col.table_catalog)
      AS NUMERIC(20,2))/1024)/1024,2) as nu_tamanho_banco
      ,'Não informado' AS ds_dicionario_banco
      ,current_timestamp as dt_importacao
      , inet_server_port() as nu_porta
      , replace(version(),'|','') as ds_versao_sgbd
      ,col.table_name as no_tabela
      ,regexp_replace(replace(coalesce(pg_catalog.obj_description(tab.oid,
       'pg_class'), 'Não informado'),'|',' '),
        E'[\\n\\r]+', ' ', 'g' ) as ds_dicionario_tabela
      ,round(cast(tab_size.table_size as NUMERIC(20,2))/1024,2)
       As nu_tamanho_tab   -- Tamanho calculado Versao inferior a versao 9.0
```

```
       ,'1970-01-01 00:00:00' as dt_criacao_tab
       ,col.table_schema as no_schema
       ,coalesce((SELECT spcname FROM pg_catalog.pg_tablespace pt
       WHERE pt.oid=tab.reltablespace),'Não Informado') as no_tablespace
       ,tab.reltuples::BIGINT    As qt_registro
       ,Qtde_col.Qtde_Colunas    As qt_coluna

       ,col.column_name as no_coluna
       ,case when col.data_type = 'ARRAY' then (COALESCE(view_type.data_type,
        col.udt_name) || ' ' || col.data_type)
             else col.data_type end
        as tp_coluna
       ,coalesce(
         (case when col.data_type in ('character varying','character','boolean')
         then col.character_maximum_length
             when col.data_type in ('timestamp without time zone','time') then
             col.datetime_precision
             when col.data_type in ('numeric', 'smallint', 'bigint', 'integer')
             then col.numeric_precision
             when col.data_type in ('text') then col.character_octet_length
                 --when col.data_type in ('numeric', 'smallint','integer',
                 'bigint','double precision','money') then numeric_precision
        end),'0') as nu_tamanho_col
       --,case when col.data_type in ('numeric', 'smallint','integer')
       then col.numeric_scale end as Tamanho_Precisao
       --,case when col.data_type in ('numeric', 'smallint','integer','bigint','money')
       then numeric_scale end as Tamanho_Precisao
       ,CASE WHEN col.Is_Nullable = 'YES' THEN 'S' else 'N' end  as st_null
       ,(Case when fk.column_name is null then 'N' else 'S' End) as st_fk
      ,regexp_replace(replace(coalesce(pg_catalog.col_description(
        tab.oid, col.ordinal_position), 'Não informado'),'|',' '),
         E'[\\n\\r]+', ' ', 'g' ) as ds_dicionario_coluna
       ,'1970-01-01 00:00:00' as dt_criacao_col
from information_schema.columns col
inner join pg_class tab
     on tab.relname =col.table_name
inner join (select sub_col.table_schema, sub_col.table_name, count(1) as Qtde_Colunas
             from information_schema.columns sub_col
             where sub_col.TABLE_SCHEMA NOT IN ('pg_catalog', 'information_schema')
             group by sub_col.table_schema,sub_col.table_name) as Qtde_col
```

```
      on Qtde_col.table_schema = col.table_schema and
         Qtde_col.table_name =  col.table_name
left join (SELECT tc.table_name, kcu.column_name
                  FROM information_schema.table_constraints AS tc
              JOIN information_schema.key_column_usage AS kcu
                     ON tc.constraint_name = kcu.constraint_name
              JOIN information_schema.constraint_column_usage AS ccu
                     ON ccu.constraint_name = tc.constraint_name
              GROUP BY tc.table_name, kcu.column_name  ) fk
     on fk.table_name = col.table_name and fk.column_name = col.column_name
-- calcula tamanho da tabela em KB - Superior a VersÃ£o 9.0
/*
left join ( SELECT Nome_Schema, Nome_Tabela, all_tables.table_name,
pg_table_size(all_tables.table_name)/1024 AS table_size
            FROM ( SELECT ('"' || table_schema || '"."'
             || table_name || '"') AS table_name,
                    table_schema as Nome_Schema, table_name as Nome_Tabela
                      FROM information_schema.tables
                      WHERE table_schema
                      NOT IN ('pg_catalog', 'information_schema')
                     ) AS all_tables
          ) as tab_size
     on tab_size.Nome_Schema = col.table_schema and
        tab_size.Nome_Tabela = col.table_name
*/
-- calcula tamanho da tabela em KB - Inferior a VersÃ£o 9.0
left join ( SELECT ('"' || tabela.table_schema || '"."' ||
tabela.table_name || '"') AS table_name
                                       , tabela.table_schema as Nome_Schema
                                       , tabela.table_name as Nome_Tabela
                                       , (tab_class.relpages*8) AS table_size
                    FROM information_schema.tables as tabela
                        inner join pg_class tab_class
                                on tab_class.relname =tabela.table_name
                        WHERE tabela.table_schema
                        NOT IN ('pg_catalog', 'information_schema')
                ) as tab_size
     on tab_size.Nome_Schema = col.table_schema and
        tab_size.Nome_Tabela = col.table_name
-- Retorna data type de ARRAY
```

```
left join information_schema.element_types view_type
     on view_type.object_type = 'TABLE'        and
        view_type.object_schema = col.table_schema and
        view_type.object_name = col.table_name and
        view_type.collection_type_identifier = col.dtd_identifier
where tab.relkind = 'r' AND col.TABLE_SCHEMA
NOT IN ('pg_catalog', 'information_schema')
order by col.table_catalog, col.table_name, col.ordinal_position;
```

Cria arquivo metadadospost.sh:

```
# vi metadadospost.sh
```

O arquivo metadadospost.sh possui o seguinte conteúdo:

```
#/bin/bash
# Programa que conecta nos servidores e extrai o meta dado de todos os bancos.

# ----------------------Configurações ---------------
SEP=:
SQL="/home/postgres/ScriptMetaDados/SelectMetaDados.sql"
SQL8="/home/postgres/ScriptMetaDados/SelectMetaDados8.sql"
SQLBANCO="/home/postgres/ScriptMetaDados/SelectBanco.sql"
SERVIDORES="/home/postgres/ScriptMetaDados/listaServidores.txt"
USUARIO_LOGIN="postgres"
PSQL="/usr/bin/psql"
CAMINHO_ARQUIVO_MD="/home/postgres/ScriptMetaDados/log/"
CABECALHO_ARQ="no_servidor|nu_ip|ds_sgbd|ds_sistema_operacional|
st_ativo|tp_ambiente|no_banco|nu_tamanho_banco|ds_dicionario_banco|
dt_importacao|nu_porta|ds_versao_sgbd|no_tabela|ds_dicionario_tabela|
nu_tamanho|dt_criacao|no_schema|no_tablespace|qt_registro|qt_coluna|
no_coluna|tp_coluna|nu_tamanho|st_null|st_fk|ds_dicionario_coluna|
dt_criacao"

#-----------------------Bloco Principal-----------------
# O arquivo com os servidores deve estar definido
[ "$SERVIDORES" ] || {
    echo "O arquivo com a lista de servidores deve ser definodo.
    Use a variável SERVIDOR."
    return 1
}
# O arquivo com os servidores pode ser lido e gravado
[ -r "$SERVIDORES" -a -w "$SERVIDORES" ] || {
```

```
    echo "Arquivo com a lista de servidores esta travado.
    confira as permissoes de leitura e escrita."
    return 1
}
#Pega a quantidade de servidores que deve ser lido
QTD_SERVIDORES=$(tail "$SERVIDORES" -n 1 | cut -d $SEP -f 1)
for X in $(seq "$QTD_SERVIDORES")
do
        # Pega o nome do servidor
        SERVIDOR=$(grep -i "^$X$SEP" "$SERVIDORES" | cut -d $SEP -f 2)
        #Descobrindo a versão do Postgres
        VERSAO=$(echo "$(echo "$("$PSQL" -h "$SERVIDOR" -U "$USUARIO_LOGIN"
        -t -c "select version();")" | cut -d " " -f 3)" | cut -d "." -f 1)
        #Monta o nome do arquivo
  ARQUIVO_META_DADOS="${CAMINHO_ARQUIVO_MD}${SERVIDOR}_METADADOS.TXT"
        #Carrega o cabeçalho do arquivo e zera o arquivo se ele ja existir.
        echo "$CABECALHO_ARQ" > $ARQUIVO_META_DADOS
        echo "Servidor $SERVIDOR Iniciado!"
        # Pega todos os bancos de cada servidor
        for database in $("$PSQL" -h "$SERVIDOR" -U
                          "$USUARIO_LOGIN"  -t -f "$SQLBANCO")
        do
                echo "Base  $database iniciada!"
                if [ "$VERSAO" = "9" ]
                then
                   #faz a consulta de meta dados na versao 9
                   echo "$("$PSQL" -h "$SERVIDOR" -U "$USUARIO_LOGIN"
                   -d "$database" -t -A -F"|" -f "$SQL")" >> $ARQUIVO_META_DADOS
                else
                   #faz a consilta de meta dados na versão 8
                   echo "$("$PSQL" -h "$SERVIDOR" -U "$USUARIO_LOGIN"
                   -d "$database" -t -A -F"|" -f "$SQL")" >> $ARQUIVO_META_DADOS
                fi
                echo "Base $database finalizada!"
        done
        echo "Servidor $SERVIDOR concluido!"
done
echo "Done!!"
```

A.6 Exemplo do SQL do SQLServer

```
Cria o arquivo listaServidores.txt:
# vi listaServidores.txt

O arquivo listaServidores.txt possui o seguinte conteúdo:
ID:NOME:AMBIENTE
1:<servidor>:P

Cria arquivo SelectMetaDados.sql:
# vi SelectMetaDados.sql

O arquivo SelectMetaDados.sql possui o seguinte conteúdo:
DECLARE @tabelas TABLE( no_banco     varchar(200),
--nu_tamanho_banco              NUMERIC(12,2),
--ds_dicionario_banco           VARCHAR(200),
--dt_importacao                 DATE,
--nu_porta                              INT,
--ds_versao_sgbd                        VARCHAR(60),
no_tabela                               varchar(200),
ds_dicionario_tabela    VARCHAR(200),
nu_tamanho_tab                  numeric(12,2),
dt_criacao_tab                  varchar(50),
no_schema                               varchar(200),
no_tablespace                   varchar(200),
qt_registro                     INT,
qt_coluna                               INT,
no_coluna                               varchar(200),
tp_coluna                               varchar(200),
nu_tamanho_col                  INT,
st_null                                 CHAR(1),
st_fk                                   CHAR(1),
ds_dicionario_coluna    varchar(200),
dt_criacao_col                  varchar(50)
);
DECLARE @database SYSNAME;
DECLARE @PORTA int;
SET NOCOUNT ON;
SET @PORTA = (SELECT TOP 1 local_tcp_port FROM sys.dm_exec_connections
WHERE local_tcp_port IS NOT NULL);
```

```
DECLARE bases CURSOR LOCAL FAST_FORWARD FOR
  SELECT d.name
    FROM sys.databases d where database_id > 4;
OPEN bases
FETCH NEXT FROM bases INTO @database
WHILE @@FETCH_STATUS = 0
BEGIN
  INSERT INTO @tabelas(
                                          no_banco,
                                          --nu_tamanho_banco,
                                          --ds_dicionario_banco,
                                          --dt_importacao,
                                          --nu_porta,
                                          --ds_versao_sgbd,
                                          no_tabela,
                                          ds_dicionario_tabela,
                                          nu_tamanho_tab,
                                          dt_criacao_tab,
                                          no_schema,
                                          no_tablespace,
                                          qt_registro,
                                          qt_coluna,
                                          no_coluna,
                                          tp_coluna,
                                          nu_tamanho_col,
                                          st_null,
                                          st_fk,
                                          ds_dicionario_coluna,
                                          dt_criacao_col
                                        )
  exec(
                ,
                WITH Sizing AS
                  (SELECT
                      '''+ @database +''' AS no_banco
                      , t.NAME AS no_tabela
                      , convert(decimal(12,2),(convert(decimal(12,2),
                       SUM(a.total_pages))* 8)/ 1024) AS nu_tamanho_tab
                      , p.rows AS RowCounts
                      , t.create_date as dt_criacao_tab
```

```
  , s.name as no_schema
  FROM '+ @database +'.sys.tables t
  INNER JOIN '+ @database +'.sys.indexes i
   ON t.OBJECT_ID = i.object_id
  INNER JOIN '+ @database +'.sys.partitions p
   ON i.object_id = p.OBJECT_ID
   AND i.index_id = p.index_id
  INNER JOIN '+ @database +'.sys.allocation_units a
   ON p.partition_id = a.container_id
  LEFT OUTER JOIN '+ @database +'.sys.schemas s
   ON t.schema_id = s.schema_id
  INNER JOIN '+ @database +'.sys.columns c
   ON t.object_id = c.object_id
 WHERE
    t.NAME NOT LIKE ''dt%''
    AND t.is_ms_shipped = 0
    AND i.OBJECT_ID > 255
GROUP BY
    t.Name, p.Rows ,t.create_date ,s.name
)
 --INSERT INTO ##CONSULTAVOLUMETRIA
 SELECT distinct SZ.no_banco AS no_banco,
 SZ.no_tabela AS no_tabela,
 ''Não Informado'' AS ds_dicionario_tabela,
 SZ.nu_tamanho_tab AS nu_tamanho_tab,
 CONVERT(varchar(23), SZ.dt_criacao_tab, 121) as dt_criacao_tab,
 SZ.no_schema as no_schema,
 ''Não Informado'' as no_tablespace,
 SZ.RowCounts AS qt_registro,
 COUNT(C.name) OVER (PARTITION BY t.object_id) AS qt_coluna,
 c.name AS no_coluna,
 ISNULL(TYPE_NAME(c.user_type_id),''guid'') AS tp_coluna,
 C.max_length AS nu_tamanho_col
 --, C.precision as TAM_PRECISAO
 , CASE
 WHEN C.is_nullable = 0 THEN ''N''
 WHEN C.is_nullable = 1 THEN ''S''
 END AS st_null
 , CASE
 WHEN FK.ReferenceColumnName IS NULL THEN ''N''
```

```
                    WHEN FK.ReferenceColumnName IS NOT NULL THEN ''S''
                    END AS st_fk
                    ,''Não Informado'' as ds_dicionario_coluna
                    ,''1970-01-01 00:00:00.000'' as dt_criacao
                    --INTO ##CONSULTAVOLUMETRIA
                              FROM Sizing SZ
                              INNER JOIN '+ @database +'.sys.schemas S
                                        ON SZ.no_schema = S.[name]
                              INNER JOIN '+ @database +'.sys.tables T
                                        ON T.schema_id = S.schema_id
                                        AND T.name = SZ.no_tabela
               INNER JOIN '+ @database +'.sys.columns C
               ON T.object_id = C.object_id
               LEFT JOIN (
                 SELECT f.name AS ForeignKey
                 , OBJECT_SCHEMA_NAME(f.parent_object_id) AS SchemaName
                 , OBJECT_NAME(f.parent_object_id) AS TableName
                 , COL_NAME(fc.parent_object_id, fc.parent_column_id) AS ColumnName
                 , OBJECT_SCHEMA_NAME(f.referenced_object_id) AS ReferenceSchemaName
                 , OBJECT_NAME (f.referenced_object_id) AS ReferenceTableName
                 , COL_NAME(fc.referenced_object_id, fc.referenced_column_id)
                    AS ReferenceColumnName
                 FROM    '+ @database +'.sys.foreign_keys AS f
                 INNER JOIN '+ @database +'.sys.foreign_key_columns AS fc
                  ON f.OBJECT_ID = fc.constraint_object_id
               ) AS FK
               ON SZ.no_schema = FK.ReferenceSchemaName
               AND SZ.no_tabela = FK.ReferenceTableName
               AND C.name = FK.ReferenceColumnName
               ORDER BY no_tabela
            ')
  FETCH NEXT FROM bases INTO @database;
END;
CLOSE bases;
DEALLOCATE bases;
select  CONVERT( sysname, SERVERPROPERTY('servername')) as no_servidor,
               CONNECTIONPROPERTY('local_net_address') as nu_ip,
               'Não Informado' as ds_sgbd,
               'Windows Server'ds_sistema_operacional,
               'S' as st_ativo,
```

```
                'P' as tp_ambiente,
                banco.no_banco,
                banco.nu_tamanho_banco,
                'Não Informado' as ds_dicionario_banco,
                CONVERT(varchar(23), getdate(), 121)  as  dt_importacao,
                (SELECT TOP 1 local_tcp_port FROM sys.dm_exec_connections
                 WHERE local_tcp_port IS NOT NULL ) as nu_porta,
                SUBSTRING(@@version,1,25) as ds_versao_sgbd,
                tabelas.no_tabela,
                tabelas.ds_dicionario_tabela,
                tabelas.nu_tamanho_tab,
                tabelas.dt_criacao_tab,
                tabelas.no_schema,
                tabelas.no_tablespace,
                tabelas.qt_registro,
                tabelas.qt_coluna,
                tabelas.no_coluna,
                tabelas.tp_coluna,
                tabelas.nu_tamanho_col,
                tabelas.st_null,
                tabelas.st_fk,
                tabelas.ds_dicionario_coluna,
                tabelas.dt_criacao_col
from (
        SELECT DB.name as no_banco ,
        (SUM(size) * 8) / 1024 AS nu_tamanho_banco
        FROM sys.databases DB
        INNER JOIN sys.master_files
         ON DB.database_id = sys.master_files.database_id
        GROUP BY DB.name
) as banco
inner join @tabelas as tabelas
on banco.no_banco = tabelas.no_banco
```

Cria arquivo metadadospost.sh:

```
# vi metadadospost.sh
```

O arquivo metadadospost.sh possui o seguinte conteúdo:

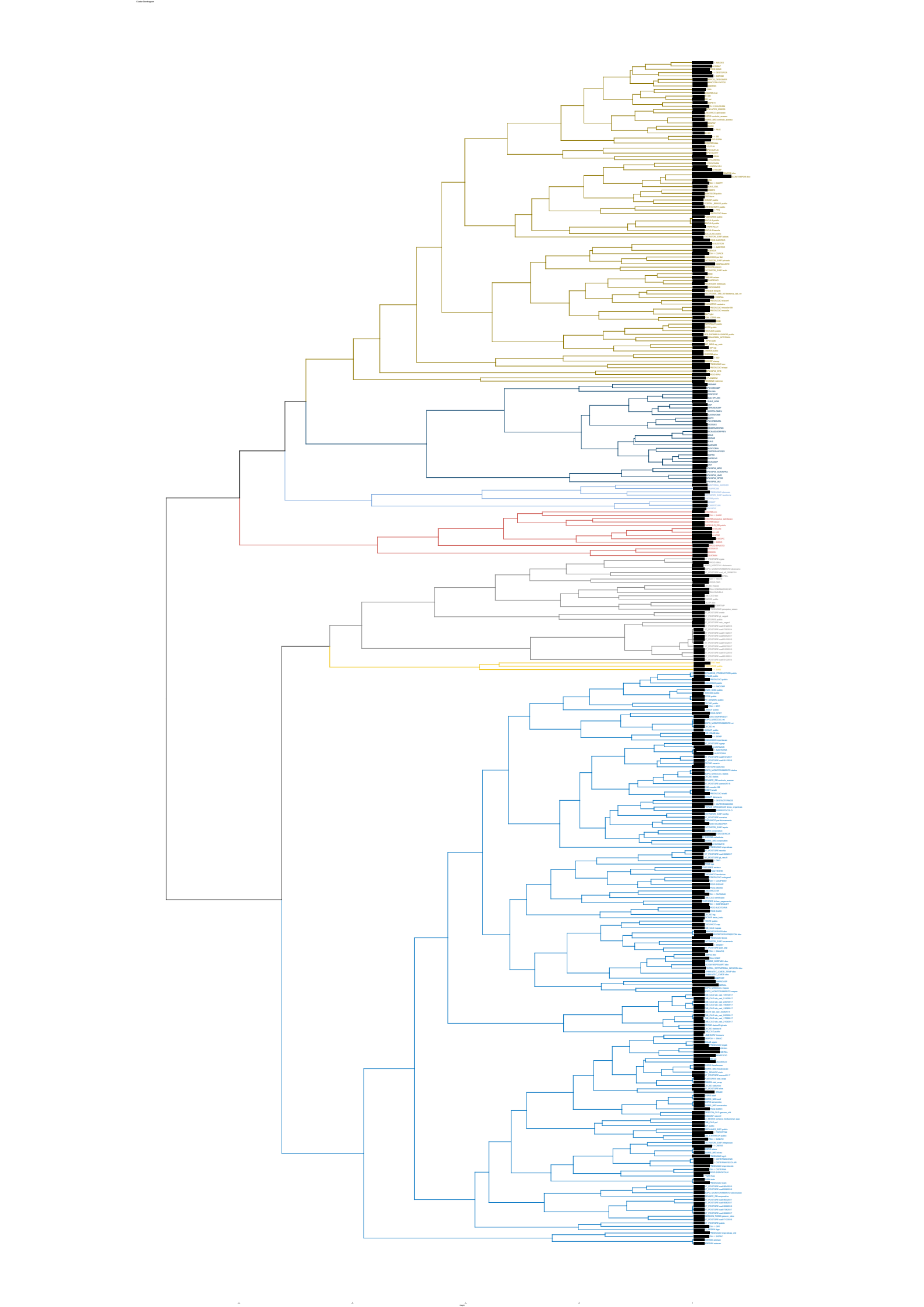
```
#/bin/bash
# Programa que conecta nos servidores e extrai o meta dado de todos os bancos.
```

```
# ----------------------Configurações ---------------------------------
SEP=:
SQL="/home/sqlserver/ScriptMetaDados/SelectMetaDados.sql"
SQLBANCO="/home/sqlserver/ScriptMetaDados/SelectBanco.sql"
SERVIDORES="/home/sqlserver/ScriptMetaDados/listaServidores.txt"
USUARIO_LOGIN="sa"
SENHA="Mdsdticgss01"
PSQL="sqlcmd"
CAMINHO_ARQUIVO_MD="/home/sqlserver/ScriptMetaDados/log/"
CABECALHO_ARQ="no_servidor|nu_ip|ds_sgbd|ds_sistema_operacional|
st_ativo|tp_ambiente|no_banco|nu_tamanho_banco|ds_dicionario_banco|
dt_importacao|nu_porta|ds_versao_sgbd|no_tabela|ds_dicionario_tabela|
nu_tamanho_tab|dt_criacao_tab|no_schema|no_tablespace|qt_registro|
qt_coluna|no_coluna|tp_coluna|nu_tamanho_col|st_null|st_fk|
ds_dicionario_coluna|dt_criacao_col"
#-----------------------Bloco Principal-------------------------------
# O arquivo com os servidores deve estar definido
[ "$SERVIDORES" ] || {
    echo "O arquivo com a lista de servidores deve ser definodo.
     Use a variável SERVIDOR."
    return 1}

# O arquivo com os servidores pode ser lido e gravado
[ -r "$SERVIDORES" -a -w "$SERVIDORES" ] || {
    echo "Arquivo com a lista de servidores esta travado.
    Confira as permissoes de leitura e escrita."
    return 1}

#Pega a quantidade de servidores que deve ser lido
QTD_SERVIDORES=$(tail "$SERVIDORES" -n 1 | cut -d $SEP -f 1)
for X in $(seq "$QTD_SERVIDORES")
do
        # Pega o nome do servidor
        SERVIDOR=$(grep -i "^$X$SEP" "$SERVIDORES" | cut -d $SEP -f 2)

        #Monta o nome do arquivo
        ARQUIVO_META_DADOS="${CAMINHO_ARQUIVO_MD}${SERVIDOR}_METADADOS.TXT"

        #Carrega o cabeçalho do arquivo e zera o arquivo se ele ja existir.
```

```
        echo "$CABECALHO_ARQ" > $ARQUIVO_META_DADOS
        echo "Servidor $SERVIDOR Iniciado!"
        # Pega todos os bancos de cada servidor
        echo "$("$PSQL" -U "$USUARIO_LOGIN" -P"$SENHA" -S "$SERVIDOR"
         -i "$SQL" -W -h -1 -s \| -o $ARQUIVO_META_DADOS)"
        echo "Servidor $SERVIDOR concluido!"
done
```

Apêndice B

Visualização Completa do Dendrograma


Cluster Dendrogram
Height

Apêndice C

Artigo Publicado no ICMLA

Two-phase Parallel Learning to Identify Similar Structures Among Relational Databases

4
Author(s)

Debora G. Reis ; Rommel N. Carvalho ; Ricardo S. Carvalho ; Marcelo Ladeira

View All Authors

Abstract:
The need for efficient techniques for dealing with large databases increases as the number of large databases grows. We propose a new two-phase parallel learning approach to identify similar structures of relational databases fast. Each phase represents a level of relational metadata aggregation. To test the approach, we realized an experiment in with several large databases of Ministry of Social Development of Brazil to classify which relational database have a similar structure of tables and columns, based on its metadata. The measure of similarity considered Levenshtein and cosine. Generalized Linear Model, Random Forest, and Gradient Boost Machines (GBM) techniques are applied to develop the model. Each model was executed in sequential and parallel processing and had performance compared. As results, the parallel execution of GBM was at least ten times faster than the sequential processing. The results encourage further applications of the propositional parallel learning in relational databases.

Published in: 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)

Date of Conference: 18-21 Dec. 2017

Date Added to IEEE *Xplore*: 18 January 2018

ISBN Information:

INSPEC Accession Number: 17521923

DOI: 10.1109/ICMLA.2017.00-17

Publisher: IEEE

Conference Location: Cancun, Mexico