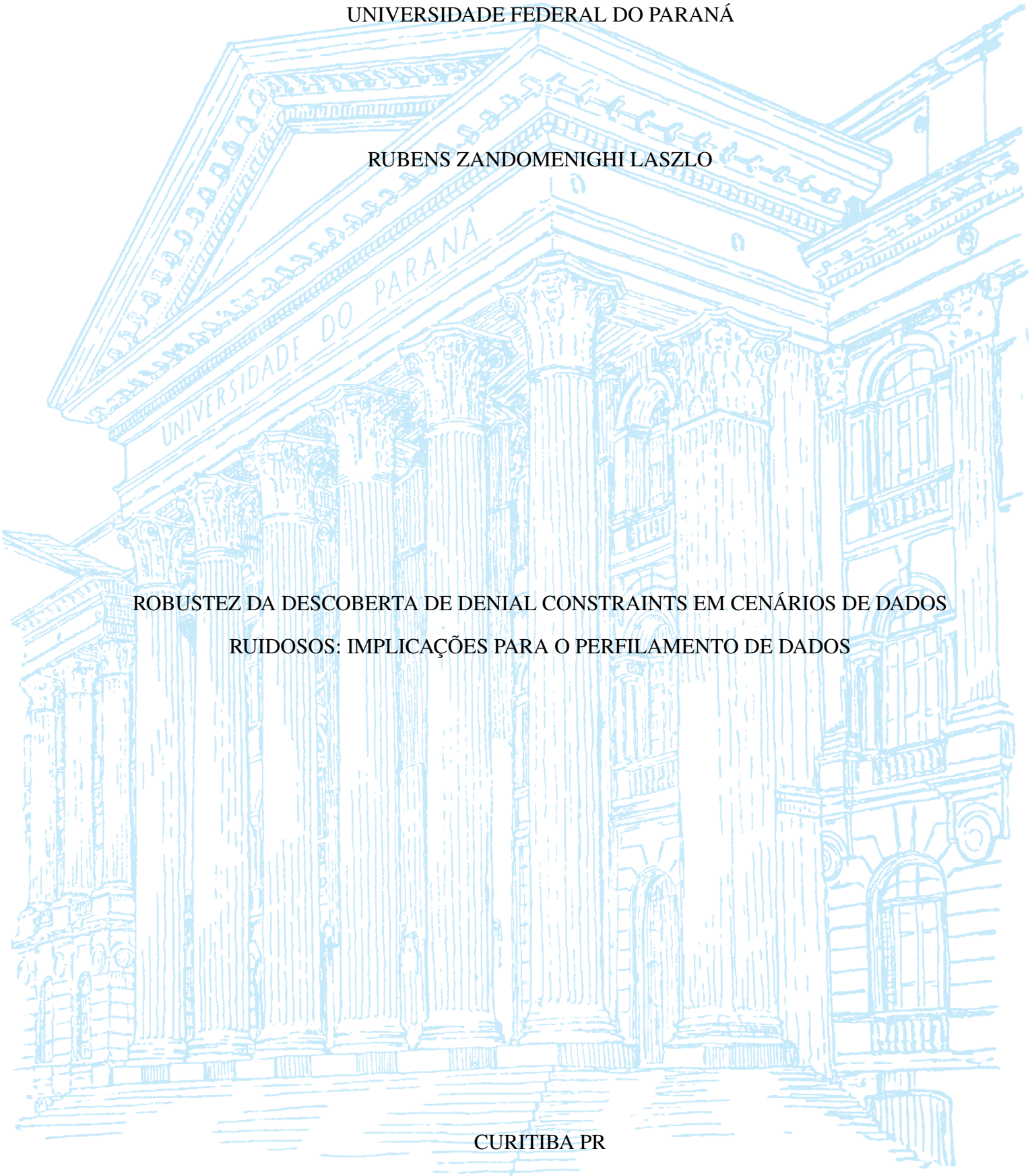
UNIVERSIDADE FEDERAL DO PARANÁ

RUBENS ZANDOMENIGHI LASZLO



ROBUSTEZ DA DESCOBERTA DE DENIAL CONSTRAINTS EM CENÁRIOS DE DADOS RUIDOSOS: IMPLICAÇÕES PARA O PERFILAMENTO DE DADOS

CURITIBA PR

2025

RUBENS ZANDOMENIGHI LASZLO

ROBUSTEZ DA DESCOBERTA DE DENIAL CONSTRAINTS EM CENÁRIOS DE DADOS RUIDOSOS: IMPLICAÇÕES PARA O PERFILAMENTO DE DADOS

Trabalho apresentado como requisito parcial à conclusão do Curso de Bacharelado em Ciência da Computação, Setor de Ciências Exatas, da Universidade Federal do Paraná.

Área de concentração: *Computação*.

Orientador: Eduardo Almeida.

CURITIBA PR

2025

Ficha catalográfica

Substituir o arquivo `0-iniciais/catalografica.pdf` pela ficha catalográfica fornecida pela Biblioteca da UFPR (PDF em formato A4).

Instruções para obter a ficha catalográfica e fazer o depósito legal da tese/dissertação (contribuição de André Hochuli, abril 2019. Links atualizados Wellton Costa, Nov 2022):

1. Estas instruções se aplicam a dissertações de mestrado e teses de doutorado. Trabalhos de conclusão de curso de graduação e textos de qualificação não precisam segui-las.
2. Verificar se está usando a versão mais recente do modelo do PPGInf e atualizar, se for necessário (`https://gitlab.c3sl.ufpr.br/maziero/tese`).
3. conferir o *checklist* de formato do Sistema de Bibliotecas da UFPR, em `https://bibliotecas.ufpr.br/servicos/normalizacao/`
4. Enviar e-mail para "`referencia.bct@ufpr.br`" com o arquivo PDF da dissertação/tese, solicitando a respectiva ficha catalográfica.
5. Ao receber a ficha, inseri-la em seu documento (substituir o arquivo `0-iniciais/catalografica.pdf` do diretório do modelo).
6. Emitir a Certidão Negativa (CND) de débito junto a biblioteca, em `https://bibliotecas.ufpr.br/servicos/certidao-negativa/`
7. Avisar a secretaria do PPGInf que você está pronto para o depósito. Eles irão mudar sua titulação no SIGA, o que irá liberar uma opção no SIGA pra você fazer o depósito legal.
8. Acesse o SIGA (`http://www.prppg.ufpr.br/siga`) e preencha com cuidado os dados solicitados para o depósito da tese.
9. Aguarde a confirmação da Biblioteca.
10. Após a aprovação do pedido, informe a secretaria do PPGInf que a dissertação/tese foi depositada pela biblioteca. Será então liberado no SIGA um link para a confirmação dos dados para a emissão do diploma.

Universidade Federal do Paraná
Setor de Ciências Exatas
Curso de Ciência da Computação

Ata de Apresentação de Trabalho de Conclusão de Curso 2

Título do Trabalho: ROBUSTEZ DA DESCOBERTA DE DENIAL CONSTRAINTS EM CENÁRIOS DE DADOS RUIDOSOS: IMPLICAÇÕES PARA O PROFILAMENTO DE DADOS

Autor(es):

GRR20206147 Nome: RUBENS ZANDOMENIGHI LASZLO

Apresentação: Data: 01/07/2025 Hora: 9h Local: Auditório do Dinf

Orientador: Eduardo Cunha de Almeida

Membro 1: Simone Dominico

Membro 2: Sergio Luiz Marques Filho

(nome) (assinatura)

AVALIAÇÃO – Produto escrito		ORIENTADOR	MEMBRO 1	MEMBRO 2	MÉDIA
Conteúdo	(00-40)				
Referência Bibliográfica	(00-10)				
Formato	(00-05)				
AVALIAÇÃO – Apresentação Oral					
Domínio do Assunto	(00-15)				
Desenvolvimento do Assunto	(00-05)				
Técnica de Apresentação	(00-03)				
Uso do Tempo	(00-02)				
AVALIAÇÃO – Desenvolvimento					
Nota do Orientador	(00-20)		**********	**********	
NOTA FINAL		**********	**********	**********	90

Os pesos indicados são sugestões.

Conforme decisão do colegiado do curso de Ciência da Computação, a entrega dos documentos comprobatório de trabalho de Conclusão de Curso 2 deve deve respeitar os seguintes procedimentos: o orientador deve abrir um processo no Sistema Eletrônico de Informações (SEI – UFPR); Selecionar o tipo: *Graduação: Trabalho Conclusão de Curso*; informar os interessados: nome do aluno e o nome do orientador; anexar esta ata escaneada e a versão final do PDF da monografia do aluno; Tramitar o processo para CCOMP (Coordenação de Ciência da Computação).

Para Marcia, Raquel e Sofia, com todo o meu carinho e gratidão.

AGRADECIMENTOS

A conclusão deste trabalho marca o fim de um ciclo importante e não seria possível sem o apoio de muitas pessoas especiais, a quem desejo expressar minha mais profunda gratidão.

Primeiramente, agradeço à minha família, por ser a base e o alicerce de toda a minha jornada acadêmica. Por todo o amor e incentivo para que eu sempre seguisse o caminho dos estudos. Cada conquista é, em grande parte, fruto do suporte e dos valores que recebi de vocês.

Ao meu orientador, Prof. Dr. Eduardo Almeida, agradeço pela disponibilidade, pela orientação precisa e pelo apoio constante em todas as fases deste trabalho. Sua experiência e seus conselhos foram essenciais para a realização desta pesquisa e para o meu desenvolvimento acadêmico. Estendo minha gratidão a todos os professores que tive a honra de ter como guias ao longo da minha graduação. Cada aula, cada ensinamento e cada desafio proposto foram peças fundamentais na construção da minha formação.

Por fim, agradeço aos professores membros da banca, Prof. Dra. Simone Dominico e Prof. Me. Sérgio Luiz Marques Filho, pela disponibilidade e pelo tempo dedicado à leitura e avaliação deste trabalho. As discussões e sugestões apresentadas foram de grande valia e, sem dúvida, contribuíram para a qualidade final desta obra.

RESUMO

A qualidade dos dados é um pilar para a tomada de decisão em ambientes corporativos e acadêmicos. O perfilamento de dados emerge como uma atividade essencial para avaliar essa qualidade, sendo a descoberta de *Denial Constraints* uma de suas técnicas mais expressivas para a detecção de inconsistências. Contudo, a eficácia e a robustez dos algoritmos de descoberta de *Denial Constraints* são desafiadas pela presença de ruídos, como erros e valores ausentes, comuns em dados do mundo real. Este trabalho apresenta, portanto, um estudo experimental que investiga o impacto da introdução de ruído na descoberta de *Denial Constraints*, com o objetivo de avaliar a robustez dos algoritmos e das métricas de qualidade. Para tal, foi desenvolvido um pipeline de avaliação sistemático onde se introduziu ruído sintético — especificamente, dados ausentes (nulos via MCAR) em diferentes níveis de poluição — em conjuntos de dados de referência. Os algoritmos DCFinder e Hydra foram então aplicados sobre os dados poluídos, e as *Denial Constraints* resultantes foram avaliadas através de métricas de qualidade consolidadas (*coverage*, *succinctness* e *interestingness*) e de uma análise de suas características estruturais, comparando-as com um conjunto ideal chamado de *"Golden Denial Constraints"*. Os resultados demonstram que o aumento do ruído leva a uma degradação consistente em todas as métricas de qualidade. Observou-se que os algoritmos tendem a gerar regras mais longas e complexas para se adaptarem aos dados imperfeitos (diminuindo a *succinctness*) e que estas perdem sua generalidade (diminuindo a *coverage*). A análise estrutural revelou que o perfil das *Denial Constraints* descobertas se desvia significativamente do perfil das *Golden Denial Constraints*, tanto no uso de operadores quanto nos atributos de foco. Conclui-se que o processo de descoberta de *Denial Constraints* é sensível à presença de ruído, e que os algoritmos, embora se adaptem, o fazem à custa da qualidade e da relevância semântica das regras. Este estudo evidencia a importância crítica de se considerar a robustez dos algoritmos de perfilamento e de se interpretar seus resultados com cautela em cenários de dados imperfeitos.

Palavras-chave: Qualidade de Dados, Perfilamento de Dados, Denial Constraints, Robustez de Algoritmos, Dados Ruidosos.

ABSTRACT

Data quality is a cornerstone for decision-making in both corporate and academic environments. Data Profiling emerges as an essential activity to assess this quality, with the discovery of Denial Constraints (DCs) being one of its most expressive techniques for detecting inconsistencies. However, the effectiveness and robustness of DC discovery algorithms are challenged by the presence of noise, such as errors and missing values, which are common in real-world data. This work, therefore, presents an experimental study that investigates the impact of noise introduction on DC discovery, aiming to evaluate the robustness of the algorithms and their quality metrics. To this end, a systematic evaluation pipeline was developed where synthetic noise—specifically, missing values (nulls through MCAR)—was introduced at different pollution levels into reference datasets. The DCFinder and Hydra algorithms were then applied to the polluted data, and the resulting DCs were evaluated using established quality metrics (*coverage*, *succinctness*, and *interestingness*), as well as through a structural analysis comparing them against an ideal set of "Golden DCs". The results demonstrate that increasing noise leads to a consistent degradation across all quality metrics. It was observed that the algorithms tend to generate longer and more complex rules to adapt to the imperfect data (decreasing *succinctness*), and that the rules lose their generality (decreasing *coverage*). The structural analysis revealed that the profile of the discovered DCs deviates significantly from the Golden DC baseline, both in operator usage and attribute focus. It is concluded that both algorithms are vulnerable to noise, albeit with distinct adaptive behaviors, highlighting the importance of considering the robustness of profiling algorithms in practical scenarios.

Keywords: Data Quality, Data Profiling, Denial Constraints, Algorithm Robustness, Noisy Data

LISTA DE FIGURAS

2.1 Hiper Grafo Conflito 22
2.2 Fluxo Execução FACET. 23
2.3 GUI BART 25

4.1 Pipeline experimental para avaliação do impacto de ruído na descoberta de DCs.. 29

5.1 Média de *Coverage* das DCs (Hydra) no conjunto de dados Hospital em função do ruído MCAR. 37
5.2 Média de *Coverage* das DCs (Hydra) no conjunto de dados Airport em função do ruído MCAR. 37
5.3 Média de *Coverage* das DCs (DCFinder) no conjunto de dados Hospital em função do ruído MCAR. 38
5.4 Média de *Coverage* das DCs (DCFinder) no conjunto de dados Airport em função do ruído MCAR. 38
5.5 Média de *Succinctness* das DCs (Hydra) no conjunto de dados Hospital em função do ruído MCAR. 39
5.6 Média de *Succinctness* das DCs (Hydra) no conjunto de dados Airport em função do ruído MCAR. 39
5.7 Média de *Succinctness* das DCs (DCFinder) no conjunto de dados Hospital em função do ruído MCAR. 40
5.8 Média de *Succinctness* das DCs (DCFinder) no conjunto de dados Airport em função do ruído MCAR. 40
5.9 Média de *Interestingness* das DCs (Hydra) no conjunto de dados Hospital em função do ruído MCAR, para diferentes valores de α 41
5.10 Média de *Interestingness* das DCs (Hydra) no conjunto de dados Airport em função do ruído MCAR, para diferentes valores de α 41
5.11 Média de *Interestingness* das DCs (DCFinder) no conjunto de dados Hospital em função do ruído MCAR, para diferentes valores de α. 42
5.12 Média de *Interestingness* das DCs (DCFinder) no conjunto de dados Airport em função do ruído MCAR, para diferentes valores de α. 42
5.13 Perfil de operadores (DCFinder) no conjunto de dados Airport. 45
5.14 Perfil de operadores (DCFinder) no conjunto de dados Hospital. 45
5.15 Perfil de operadores (Hydra) no conjunto de dados Airport. 45
5.16 Perfil de operadores (Hydra) no conjunto de dados Hospital. 45
5.17 Distribuição de atributos (DCFinder) no conjunto de dados Airport. 46
5.18 Distribuição de atributos (DCFinder) no conjunto de dados Hospital. 46
5.19 Distribuição de atributos (Hydra) no conjunto de dados Airport. 47

5.20 Distribuição de atributos (Hydra) no conjunto de dados Hospital. 47

LISTA DE TABELAS

2.1 Tabela de Funcionários 17
2.2 Conjunto de DCs Manualmente 17
2.3 Métricas de Coverage, Succinctness e Interestingness para o conjunto de DCs . . 21

4.1 Resumo dos conjuntos de dados Utilizados no Estudo 31
4.2 Amostra da tabela `PROFILING_METADATA` com métricas de Coverage e Succinctness para *Golden DCs 'Airport'* 34

5.1 Especificações do ambiente de execução dos experimentos 36

LISTA DE ACRÔNIMOS

ALADIN	ALmost Automatic Data Integration
BART	Benchmarking Algorithms for Data Repairing and Translation
Cov	Métrica de *Coverage*
CSV	Comma-Separated Values
CTE	Common Table Expression
DC	Denial Constraint
DBMS	Database Management System (Sistema de Gerenciamento de Banco de Dados)
DINF	Departamento de Informática
ECP	Evidence Context Pipeline
FACET	Fast Detection of Denial Constraint Violations
FK	Foreign Key (Chave Estrangeira)
GUI	Graphical User Interface (Interface Gráfica do Usuário)
INCS	Indexed Negative Cover Search
Int	Métrica de *Interestingness*
JSON	JavaScript Object Notation
MAR	Missing At Random
MCAR	Missing Completely At Random
MNAR	Missing Not At Random
ODBC	Open Database Connectivity
PK	Primary Key (Chave Primária)
RDBMS	Relational Database Management System (Sistema de Gerenciamento de Banco de Dados Relacional)
SPIDER	Single Pass Inclusion Dependency Recognition
SQL	Structured Query Language
Succ	Métrica de *Succinctness*
UFPR	Universidade Federal do Paraná
XML	eXtensible Markup Language

LISTA DE SÍMBOLOS

α	Alfa, parâmetro de ponderação na métrica de *Interestingness*.
β	Beta, letra grega usada para identificar uma tupla.
ϕ	Representa uma DC.
$\forall$	Quantificador universal, "para todo".
$\in$	Relação de pertencimento, "pertence a".
$\subseteq$	Relação de subconjunto, "está contido em".
$\neg$	Operador de negação lógica.
$\wedge$	Operador de conjunção lógica ("E").
$\mathbb{A}$	Alfabeto de símbolos que compõem uma DC.
$\mathbb{B}$	Conjunto de operadores de comparação (e.g., =, <, >).
$\mathbb{U}$	Conjunto de atributos de uma relação.
$\|S\|$	Cardinalidade de um conjunto S.
$=$	Operador de igualdade.
$\neq$	Operador de desigualdade.
$<$	Operador "menor que".
$>$	Operador "maior que".
$\leq$	Operador "menor ou igual a".
$\geq$	Operador "maior ou igual a".
$\sum$	Somatório.
$\times$	Operador de multiplicação.
$\approx$	Aproximadamente igual a.
t, t'	Representam tuplas (registros) de uma tabela.
p_i	Representa o i-ésimo predicado em uma DC.
$w(k)$	Função de peso utilizada no cálculo de *Coverage*.
kE	Conjunto de evidências com k predicados satisfeitos.

SUMÁRIO

1 INTRODUÇÃO 14
2 FUNDAMENTAÇÃO 16
2.1 DESCOBERTA DE REGRAS 16
2.1.1 *Denial Constraints* 16
2.1.2 Descoberta Automática de DCs. 16
2.1.3 Avaliação da Qualidade das DCs 18
2.1.4 Ferramenta Metanome: Uma Plataforma Extensível para Perfilamento de Dados . 21
2.2 DESCOBERTA DE VIOLAÇÕES 21
2.2.1 Solução Holística 22
2.2.2 FAST CONSTRAINT-BASED ERROR DETECTOR 22
2.3 FONTES DE RUÍDO E INCONSISTÊNCIAS EM DADOS 24
2.3.1 Problemas em Origem Única 24
2.3.2 Problemas em Múltiplas Origens 24
2.3.3 A Importância da Geração Controlada de Erros em Dados 25
3 PROPOSTA 27
3.1 OBJETIVO 27
3.2 HIPÓTESE 27
3.3 PERGUNTAS DE PESQUISA 27
3.4 CONTRIBUIÇÃO 28
4 *PIPELINE* DE AVALIAÇÃO DO IMPACTO DE RUÍDO 29
4.1 CONJUNTO DE DADOS ORIGINAIS E *GOLDEN DCS* 29
4.2 GERAÇÃO DE RUÍDO. 31
4.3 DESCOBERTA DCS 31
4.3.1 DCFinder 31
4.3.2 Hydra 31
4.4 CONJUNTOS DE DCS DESCOBERTAS 31
4.5 CÁLCULO DAS MÉTRICAS 32
4.5.1 *Succinctness*. 32
4.5.2 *Coverage* 33
4.6 ARMAZENAMENTO DE RESULTADOS 34
4.7 ANÁLISE COMPARATIVA 34
5 EXPERIMENTOS. 36
5.1 ANÁLISE DA MÉTRICA DE *COVERAGE* 36
5.1.1 Comportamento da *Coverage* para o Algoritmo Hydra. 37

5.1.2 Comportamento da *Coverage* para o Algoritmo DCFinder 37
5.2 ANÁLISE DA MÉTRICA DE *SUCCINCTNESS* 38
5.2.1 Comportamento da *Succinctness* para o Algoritmo Hydra 39
5.2.2 Comportamento da *Succinctness* para o Algoritmo DCFinder 40
5.3 ANÁLISE DA DISTRIBUIÇÃO DA MÉTRICA DE *INTERESTINGNESS* 41
5.3.1 *Denial Constraints* Descobertas 41
5.4 ANÁLISE DA ESTRUTURA INTERNA DAS *DENIAL CONSTRAINTS* DESCOBERTAS SOB POLUIÇÃO 44
5.4.1 Impacto da Poluição no Perfil dos Operadores 44
5.4.2 Impacto da Poluição na Distribuição Percentual de Envolvimento de Atributos . . 45
5.4.3 Síntese da Análise Estrutural Comparativa 47
6 CONCLUSÃO 48
6.1 TRABALHOS FUTUROS 48
REFERÊNCIAS 49
APÊNDICE A – CÁLCULO INTERESTINGNESS 50
A.1 CÁLCULO COVERAGE 50
APÊNDICE B – RESULTADOS ALGORITMOS DESCOBERTA DCS . . 54
B.1 DC FINDER 54
B.2 HYDRA 54

1 INTRODUÇÃO

Com o crescimento exponencial do volume de dados, a sua qualidade tornou-se um fator crítico para a eficácia das análises e da tomada de decisão. Contudo, dados do mundo real são frequentemente imperfeitos, contendo uma variedade de erros e inconsistências — coletivamente referidos como **ruído** — que podem comprometer a validade das conclusões (Rahm, 2000). O **perfilamento de dados** surge como o processo fundamental para analisar e compreender a qualidade de um conjunto de dados (Abedjan et al., 2015). Dentro deste processo, as ***Denial Constraints* (DCs)** se destacam como um formalismo poderoso e expressivo para detectar inconsistências complexas (X. Chu, 2013). Como robustez sendo a qualidade das métricas obtidas pelos algoritmos, o grande desafio, no entanto, é que os próprios algoritmos de descoberta de DCs precisam operar sobre esses dados ruidosos, levantando uma questão central: quão robusto é o processo de descoberta de DCs na presença de erros?

A **motivação** para este trabalho reside na lacuna existente sobre a compreensão do impacto de dados imperfeitos na descoberta de DCs. Embora algoritmos como o Hydra (Bleifuß et al., 2017) e o DCFinder (Pena et al., 2019) sejam conhecidos por sua eficiência em dados limpos (sem ruídos), sua robustez e o comportamento das regras que eles descobrem em cenários ruidosos são pouco explorados. É fundamental para analistas e engenheiros de dados entender se as DCs descobertas nesses cenários são confiáveis e como a qualidade delas, medida por métricas como *coverage*, *succinctness* e *interestingness*, é afetada. A dificuldade de se avaliar essa robustez, devido à falta de *benchmarks* com erros controlados, reforça a necessidade de estudos experimentais sistemáticos, como o proposto por Arocena et al. (Arocena et al., 2015).

A **proposta** deste trabalho é, portanto, uma avaliação experimental e sistemática da robustez do processo de descoberta de DCs. Para isso, desenvolveu-se um pipeline que introduz níveis controlados de ruído — com foco na inserção de **dados ausentes** (nulos via MCAR) — em conjuntos de dados de referência. Em seguida, os algoritmos DCFinder e Hydra são aplicados sobre os dados poluídos para se descobrir conjuntos de DCs. O impacto do ruído é então quantificado através da análise das métricas de qualidade e das características estruturais (como perfil de operadores e atributos envolvidos) das DCs resultantes, comparando-as com um conjunto de referência de "Golden DCs".

As principais **contribuições** deste trabalho são:

- A aplicação de um pipeline experimental sistemático para avaliar a robustez de algoritmos de descoberta de DCs frente a dados ruidosos.
- Uma análise quantitativa e estrutural detalhada dos efeitos de dados ausentes nas métricas de *coverage*, *succinctness* e *interestingness*.
- Uma discussão aprofundada sobre o comportamento adaptativo de algoritmos com diferentes estratégias (exata vs. aproximada) em dados imperfeitos.
- A geração de percepções práticos para profissionais de perfilamento de dados sobre a confiabilidade e as características das DCs descobertas em cenários realistas.

A **organização deste documento** segue a seguinte estrutura. O Capítulo 2 apresenta a fundamentação teórica sobre Denial Constraints, métricas de qualidade e as fontes de ruído em dados. O Capítulo 3 detalha a proposta, a hipótese e as perguntas de pesquisa que guiam o estudo. O Capítulo 4 descreve o pipeline experimental desenvolvido para a condução das análises. O

Capítulo 5 apresenta e discute em detalhe os resultados obtidos. Por fim, o Capítulo 6 sumariza as conclusões do trabalho, responde às perguntas de pesquisa e aponta direções para pesquisas futuras.

2 FUNDAMENTAÇÃO

2.1 DESCOBERTA DE REGRAS

Com o crescimento exponencial da quantidade de dados gerados e armazenados, a qualidade dos dados segue sendo o diferencial para a eficácia das análises e para a tomada de decisão em organizações de todos os portes. Muitos desses dados gerados não são utilizados, podendo ser devido a dados mal estruturados ou incorretos, o que pode levar a interpretações errôneas, afetando desde processos operacionais até estratégias de longo prazo.

Para a utilização desses dados necessita-se que tenham sido estruturados com qualidade, assim *perfilamento de dados* (Abedjan et al., 2015) é o processo de examinar, analisar e coletar estatísticas sobre dados em um conjunto de dados. Essa prática busca compreender melhor as características intrínsecas dos dados, como distribuição, qualidade, conformidade com regras de negócio e relações entre as variáveis, sendo uma destas *Denial Constraints*. O objetivo principal do perfilamento de dados é garantir a confiabilidade dos dados, com o intuito de entendimento do modelo semântico para identificação de padrões.

2.1.1 *Denial Constraints*

As *Denial Constraints (DCs)* (X. Chu, 2013) surgem como uma forma de analisar as dependências dos dados. São restrições de integridade que descrevem um conjunto de predicados, para o qual a coexistência entre tuplas não deve ocorrer. São utilizadas para generalizar uma série de outras restrições, como chaves únicas, dependências funcionais e dependências de ordem, e são altamente expressivas para identificar inconsistências complexas nos dados. Cada violação de uma DC representa uma potencial anomalia ou erro de qualidade no banco de dados, sendo necessária, posteriormente, uma etapa de detecção e limpeza desses dados no fluxo de *Data Cleaning*, esta detecta e efetua a limpeza desses dados.

A representação delas pode ser feita como representação em SQL para a execução direta em bancos de dados. Para este trabalho será utilizada a representação matemática das DCs, tal como no formalismo matemático descrito em (Pena et al., 2019) , onde:

- t e t' representam tuplas distintas da mesma tabela R.
- p_i são predicados de comparação entre valores de colunas de t e t', podendo incluir operadores como $=, \neq, <, >, \leq, \geq$.
- ϕ: representa uma DC, podendo ser representada pela definição:

$$\phi : \forall t, t` \in r, \neg(p_1 \wedge \cdots \wedge p_m)$$

Para exemplificação do formalismo utilizado, será apresentado um possível caso de uso da constante descrita. Para tal, foram gerados possíveis casos manualmente, sendo um possível conjunto de DCs, com sua semântica equivalente, representado como 2.2.

2.1.2 Descoberta Automática de DCs

Conforme a necessidade de perfilamento de dados para múltiplos conjuntos de dados crescentes, o tempo para a descoberta de regras torna-se cada vez maior, conforme a necessidade

ID	Nome	IDDepartamento	Salário	IDGerente	IDCargo	DataInício
1	Alexa	101	5000	3	1	2018-01-15
2	Bruno	101	4000	3	1	2017-01-15
3	Carol	101	7000	3	10	2015-02-01
4	Deyverson	102	8000	4	4	2022-03-10
5	Eduardo	102	5000	4	5	2017-01-01
6	Felix	102	5000	4	5	2017-01-01
7	Guilherme	102	12000	4	5	2017-01-01

Tabela 2.1: Tabela de Funcionários

Identificador	DC	Semântica
ϕ_1	$\forall t_x, t_y \in R, \neg(t_x.\text{IDGerente} = t_y.\text{ID} \wedge t_x.\text{Salário} > t_y.\text{Salário})$	Nenhum funcionário pode ter um salário maior que o de seu gerente.
ϕ_2	$\forall t_x, t_y \in R, \neg(t_x.\text{IDCargo} = t_y.\text{IDCargo} \wedge t_x.\text{DataInício} > t_y.\text{DataInício} \wedge t_x.\text{Salário} > t_y.\text{Salário})$	Entre funcionários com o mesmo cargo, aquele que começou mais recentemente não deve ter um salário maior que o do funcionário mais antigo no cargo.
ϕ_3	$\forall t, t' \in R, \neg(t.\text{ID} \neq t'.\text{ID})$	Cada funcionário deve ter uma identificação única.
ϕ_4	$\forall t, t' \in R, \neg(t.\text{IDCargo} = t'.\text{IDCargo} \wedge t.\text{IDDepartamento} \neq t'.\text{IDDepartamento})$	Cada cargo deve ser referente a um departamento.
ϕ_5	$\forall t_x, t_y \in R, \neg(t_x.\text{Salário} < 10 \times t_y.\text{Salário})$	Nenhum funcionário pode ter um salário 10x maior que o de outro funcionário.

Tabela 2.2: Conjunto de DCs Manualmente

de entendimento desses dados. O artigo de (X. Chu, 2013) demonstra ser um problema com custo computacional exponencial no número de predicados considerados.

Visando-se à otimização desta etapa, foram desenvolvidas técnicas para descoberta automatizada de DCs. No Apêndice B, foram utilizados os algoritmos do DCFinder e Hydra, utilizando a ferramenta Metanome para o conjunto de dados de entrada definido na Tabela 2.1 para detecção de DCs.

Conforme estudo (Pena et al., 2019), esses algoritmos trabalham em um *pipeline* com três fases sequenciais, **construção do espaço de predicados** em que são construídos os conjuntos de possíveis predicados, **construção do conjunto de evidências** em que são validados os predicados previamente encontrados a fim de qualificar os mais semanticamente válidos, mediante evidências que satisfaçam os predicados iniciais e **enumeração das DCs** em que, através das evidências encontradas anteriormente para os predicados, são enumeradas apenas as DCs que possuam alguma evidência que atenda ao predicado completo da DC candidata.

2.1.3 Avaliação da Qualidade das DCs

Utilizando ferramentas para a detecção de DCs em dados, o espaço de DCs encontrados pode se tornar inviável para a execução de algoritmos de Detecção de violações. Para assegurar que as DCs descobertas sejam úteis e representativas, é fundamental estabelecer critérios de qualidade. No estudo de Chu, Ilyas e Papotti (X. Chu, 2013), é proposta uma função de pontuação para ranquear DCs chamada *interestingness score*, que utiliza duas métricas principais para avaliar a qualidade de uma DC, sendo utilizado um limite aceito conforme os resultados obtidos das métricas para realizar a poda dos resultados com menor *Interestingness*.

2.1.3.1 Succinctness

Supõe-se que a conforme mais simples a definição maior a concisão da regra. Nesse contexto, DCs com menos predicados são consideradas mais sucintas, pois são mais fáceis de interpretar e validar.

A succinctness é calculada como o comprimento mínimo possível de uma DC dividido pelo comprimento da DC atual, definida pela equação 2.3, em que se utiliza o alfabeto da DC definido por 2.1, sendo $\mathbb{U}$ o conjunto de atributos, $\mathbb{B}$ o conjunto de todos os operadores e $Cons$ são as constantes. Assim, o $\text{Len}(\varphi)$ é dado pelo número de símbolos de $\mathbb{A}$ que aparecem na DC φ, conforme a equação 2.2.

$$\mathbb{A} = \{t\alpha, t\beta, \mathbb{B}, \mathbb{U}, Cons\} \tag{2.1}$$

$$Len(\varphi) = |\{a | a \in \mathbb{A}, a \in \varphi\}| \tag{2.2}$$

$$\text{Succ}(\varphi) = \frac{\min(\{\text{Len}(\psi) \mid \forall \psi\})}{\text{Len}(\varphi)} \tag{2.3}$$

2.1.3.2 Coverage

Esta métrica mede o suporte estatístico da DC nos dados. A cobertura avalia a quantidade de pares de tuplas que não violam a DC, indicando que a restrição representa um padrão consistente nos dados. Quanto maior a cobertura, mais relevante é a DC para a qualidade dos dados. Sendo representada pela equação 2.5, sendo utilizado peso como w(k) para medir a cobertura da DC completa, sendo φ a quantidade de tuplas satisfeitas pela DC completa e k o número de predicados, representada pela equação 2.4 .

$$w(k) = 1 - \frac{|\varphi.\text{Pres}| - k}{|\varphi.\text{Pres}|} \tag{2.4}$$

$$\text{Coverage}(\varphi) = \frac{\sum_{k=0}^{|\varphi|-1} |kE| \times w(k)}{\sum_{k=0}^{|\varphi|-1} |kE|} \tag{2.5}$$

2.1.3.3 Interestingness

A função de pontuação é, então, definida como uma combinação linear das duas métricas, nota-se que a partir da equação 2.6 possibilita-se um maior peso para a métrica de *Coverage* ou de *succinctness*, conforme a definição do peso utilizado.

2.6:

$$\text{Interestingness}(\varphi) = \alpha \times \text{Coverage}(\varphi) + (1 - \alpha) \times \text{Succ}(\varphi) \tag{2.6}$$

onde α é um parâmetro que pondera a importância relativa de cada métrica. Esse sistema de avaliação permite selecionar e priorizar as DCs mais relevantes, serão utilizados nos testes dois pesos diferentes para avaliar o interestingness das DCs encontradas anteriormente.

Dada a Tabela 2.1 , a avaliação dessas métricas, utilizando o conjunto de DCs definidas por 2.2, resulta na Tabela 2.3:

- Para a DC: $\phi_1 = \forall t_x, t_y \in R, \neg(t_x.\text{IDGerente} = t_y.\text{ID} \wedge t_x.\text{Salário} > t_y.\text{Salário})$

  Tuplas relevantes:

  - $(t_1 = \text{Alexa}, t_3 = \text{Carol}) : t_1.\text{Salário} = 5000 \leq t_3.\text{Salário} = 7000$ satisfaz
  - $(t_7 = \text{Guilherme}, t_4 = \text{Deyverson}) : t_7.\text{Salário} = 12000 > t_4.\text{Salário} = 8000$ não satisfaz

  Total de predicados: $|\phi_1.\text{Pres}| = 2$

- $k = 1$ (1 predicado satisfeito)

  Cálculo de $w(k)$:

- $w(1) = 1 - \frac{2-1}{2} = 0.5$

  $\mathbb{A} = \{t\alpha, t\beta, \mathbb{B}, \mathbb{U}, Cons\} = Len\{t\alpha, t\beta, (>, =), (\text{IDGerente, ID, Salário}), ()\}$
  $\text{Succ}(\varphi) = \frac{\min(\{\text{Len}(\psi)|\forall\psi\})}{\text{Len}(\varphi)} = \frac{4}{8} = 0.5$

  $\text{Coverage}(\varphi) = \frac{\sum_{k=0}^{|\varphi|-1} |kE| \times w(k)}{\sum_{k=0}^{|\varphi|-1} |kE|} = \frac{14 \times 0.5 + 16 \times 0.5}{14+16} = 0.53$

- Para a DC: $\phi_2 = \forall t_x, t_y \in R, \neg(t_x.\text{IDCargo} = t_y.\text{IDCargo} \wedge t_x.\text{DataInício} > t_y.\text{DataInício} \wedge t_x.\text{Salário} > t_y.\text{Salário})$

  Tuplas relevantes

  - $(t_1 = \text{Alexa}, t_2 = \text{Bruno}) : t_1.\text{DataInício} = 2018 > t_2.\text{DataInício} = 2017$ $t_1.\text{Salário} = 5000 \leq t_2.\text{Salário} = 4000$ satisfaz
  - $(t_6 = \text{Felix}, t_7 = \text{Guilherme}) : t_6.\text{DataInício} = 2017 \leq t_7.\text{DataInício} = 2017$ $t_6.\text{Salário} = 5000 < t_7.\text{Salário} = 12000$ satisfaz parcialmente

  Total de predicados: $|\phi_2.\text{Pres}| = 3$

- $k = 2$ (2 predicados satisfeitos)

  Cálculo de $w(k)$:

- $w(2) = 1 - \frac{3-2}{3} = \frac{2}{3} \approx 0.67$

  $\mathbb{A} = \{t\alpha, t\beta, \mathbb{B}, \mathbb{U}, Cons\} = \{t\alpha, t\beta, (=, >), (\text{IDCargo, DataInício, Salário}), ()\}$
  $\text{Succ}(\varphi) = \frac{\min(\{\text{Len}(\psi)|\forall\psi\})}{\text{Len}(\varphi)} = \frac{4}{7} = 0.57$

  $\text{Coverage}(\varphi) = \frac{\sum_{k=0}^{|\varphi|-1} |kE| \times w(k)}{\sum_{k=0}^{|\varphi|-1} |kE|} = \frac{7 \times 0.67 + 17 \times 0.67 + 5 \times 0.67}{7+17+5} = \frac{19.43}{29} = 0.67$

- Para a DC $\phi_3 = \forall t, t' \in R, \neg(t.\text{ID} \neq t'.\text{ID})$

  Tuplas relevantes: Não há tuplas com IDs duplicados. Todos os IDs na tabela são únicos.

  Total de predicados: $|\phi_3.\text{Pres}| = 1$

- $k = 1$ (todos os predicados são satisfeitos)

  Cálculo de $w(k)$:

- $w(1) = 1 - \frac{1-1}{1} = 1$

  $\mathbb{A} = \{t\alpha, t\beta, \mathbb{B}, \mathbb{U}, Cons\} = \{t\alpha, t\beta, (\neq), (\text{ID}), ()\}$

  $\text{Succ}(\varphi) = \frac{\min(\{\text{Len}(\psi)|\forall\psi\})}{\text{Len}(\varphi)} = \frac{4}{4} = 1$

  $\text{Coverage}(\varphi) = \frac{\sum_{k=0}^{|\varphi|-1} |kE| \times w(k)}{\sum_{k=0}^{|\varphi|-1} |kE|} = \frac{1}{1} = 1$

- Para a DC $\phi_4 = \forall t, t' \in R, \neg(t.\text{IDCargo} = t'.\text{IDCargo} \wedge t.\text{IDDepartamento} \neq t'.\text{IDDepartamento})$

  Tuplas relevantes: Não há cargos com IDs iguais associados a departamentos diferentes.

  Total de predicados: $|\phi_4.\text{Pres}| = 1$

- $k = 1$ (todos os predicados são satisfeitos)

  Cálculo de $w(k)$:

- $w(1) = 1 - \frac{1-1}{1} = 1$

  $\mathbb{A} = \{t\alpha, t\beta, \mathbb{B}, \mathbb{U}, Cons\} = \{t\alpha, t\beta, (=, \neq), (\text{IDCargo, IDDepartamento}), ()\}$

  $\text{Succ}(\varphi) = \frac{\min(\{\text{Len}(\psi)|\forall\psi\})}{\text{Len}(\varphi)} = \frac{4}{6} = 0.66$

  $\text{Coverage}(\varphi) = \frac{\sum_{k=0}^{|\varphi|-1} |kE| \times w(k)}{\sum_{k=0}^{|\varphi|-1} |kE|} = 1$

- Para a DC $\phi_5 = \forall t_x, t_y \in R, \neg(t_x.\text{Salário} < 10 \times t_y.\text{Salário})$

  - $(t_7 = \text{Guilherme}, t_5 = \text{Eduardo}) : t_7.\text{Salário} = 12000 < 10 \cdot t_5.\text{Salário} = 50000$ satisfaz
  - $(t_6 = \text{Felix}, t_1 = \text{Alexa}) : t_6.\text{Salário} = 5000 < 10 \cdot t_1.\text{Salário} = 50000$ satisfaz

  Total de predicados: $|\phi_5.\text{Pres}| = 1$

- $k = 1$ (todos os predicados são satisfeitos)

  Cálculo de $w(k)$:

- $w(1) = 1 - \frac{1-1}{1} = 1$

  $\mathbb{A} = \{t\alpha, t\beta, \mathbb{B}, \mathbb{U}, Cons\} = \{t\alpha, t\beta, (<), (\text{Salário}), (2)\}$

  $\text{Succ}(\varphi) = \frac{\min(\{\text{Len}(\psi)|\forall\psi\})}{\text{Len}(\varphi)} = \frac{5}{5} = 1$

  $\text{Coverage}(\varphi) = \frac{\sum_{k=0}^{|\varphi|-1} |kE| \times w(k)}{\sum_{k=0}^{|\varphi|-1} |kE|} = 1$

DC	Coverage	Succinctness	Interestingness (α = 1)	Interestingness (α = 0.5)
ϕ_1	0.53	0.5	0.53	0.515
ϕ_2	0.67	0.57	0.67	0.62
ϕ_3	1	1	1	1
ϕ_4	1	0.66	1	0.83
ϕ_5	1	1	1	1

Tabela 2.3: Métricas de Coverage, Succinctness e Interestingness para o conjunto de DCs

Caso seja determinado um limite para poda dos resultados parametrizado como 0.65, seriam podadas as DCs: { $\phi 1$ }, utilizando-se ênfase em succinctness(α = 0.5) e as DCS: $\{\phi 1, \phi 2\}$ caso seja dada ênfase em Coverage (α = 1), sendo assim, para a continuidade do trabalho de verificação de violações, o conjunto de DCs compreenderia as DCs: $\{\phi 2, \phi 3, \phi 4, \phi 5\}$ e $\{\phi 3, \phi 4, \phi 5\}$

O uso de DCs na etapa de perfilamento de dados adiciona uma valiosa forma de representação para serem realizadas verificações em um conjunto de dados, podendo ser utilizado em todos os estágios de representação dos dados trabalhados. A aplicação dessas constantes oferece uma abordagem estruturada para capturar inconsistências, a qual, em conjunto com técnicas tradicionais de perfilamento de dados, promove uma melhor compreensão e qualidade dos dados. Essa representação fornece uma base sólida para a posterior etapa de Limpeza dos Dados.

2.1.4 Ferramenta Metanome: Uma Plataforma Extensível para Perfilamento de Dados

A descoberta de metadados complexos, como as DCs, requer ferramentas especializadas que implementem algoritmos do estado da arte. Nesse contexto, destaca-se a plataforma **Metanome**, um sistema de código aberto projetado para o perfilamento de dados (Papenbrock et al., 2015). Conforme descrito no artigo (Papenbrock et al., 2015), o objetivo do Metanome é servir como uma plataforma extensível que incorpora diversos algoritmos de perfilamento, focando na descoberta automática de metadados complexos, como dependências funcionais, combinações de colunas únicas e, crucialmente para este trabalho, DCs.

Uma das principais contribuições do Metanome é fornecer um ambiente unificado para a execução e avaliação comparativa de diferentes algoritmos de descoberta (Papenbrock et al., 2015). Ele oferece uma infraestrutura que abstrai as fontes de dados, permitindo que os algoritmos operem sobre diferentes sistemas de banco de dados e arquivos. Além disso, a plataforma inclui funcionalidades para parametrizar os resultados de perfilamento com base em várias métricas e para visualizar os conjuntos de metadados descobertos (Papenbrock et al., 2015).

Com o uso da ferramenta, para o utilizador, a execução de algoritmos distintos segue um fluxo de trabalho semelhante, com comandos e configurações padronizados. Ao abstrair as complexidades de execução, permite que o pesquisador se concentre na análise dos resultados, em vez de se preocupar com os pormenores de implementação de cada algoritmo utilizado individualmente.

2.2 DESCOBERTA DE VIOLAÇÕES

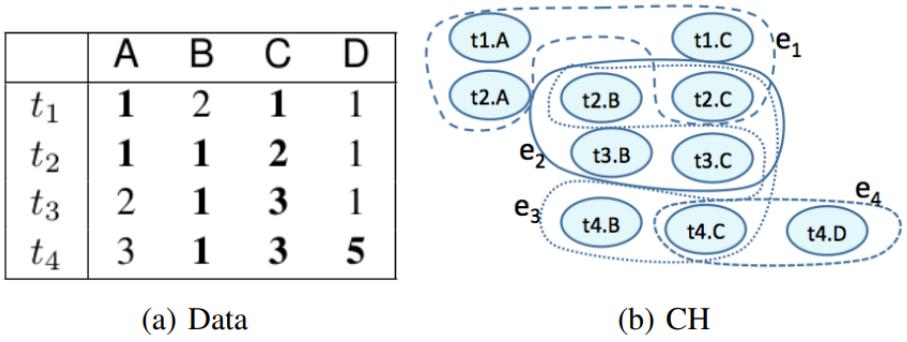
Após a etapa de perfilamento de dados, para as DCs avaliadas como ideais para o conjunto de dados trabalhado, é necessária a verificação de violações das constantes definidas.

Para tal, diferentes estratégias na construção de detectores de violações foram desenvolvidas para realizar esse serviço.

2.2.1 Solução Holística

No estudo de (Chu et al., 2013), apresenta-se uma solução holística para limpeza de dados, sendo utilizado para a representação de violações de DCs um hipergrafo de conflitos.

Nesta abordagem, conforme o conjunto de dados utilizado, é gerado um hipergrafo de conflitos, tal que os nodos são as células que violam uma DC e as arestas relacionam os nodos violadores de uma mesma DC. Porém, o tempo de execução torna-se polinomial, por serem testados todos os predicados dada uma DC, como na Figura 2.1 extraída do artigo.



(a) Data (b) CH

Figura 2.1: Hiper Grafo Conflito

2.2.2 FAST CONSTRAINT-BASED ERROR DETECTOR

O processamento de verificação de violação das DCs é um processo custoso, e essa etapa consome a maioria do tempo de processamento. Utilizando uma abordagem de planejamento de consulta híbrido, o algoritmo é baseado em *column sketches*, representação de resultados intermediários flexível à constante trabalhada, conforme proposto no estudo de (Pena et al., 2021). O FACET (Fast Constraint-Based Error Detector) é um algoritmo especialista em detecção de violação dado um conjunto de dados e um conjunto de DCs, utiliza técnicas para otimização de operações de *Self Join* em um Banco de Dados.

O algoritmo do FACET organiza a ordem de planejamento de consulta, através da separação por classes de predicado, sendo respectivamente executa-se em ordem as classes de Equalidades, Não-Equalidades e Inequalidades. Com essa estratégia, busca-se a utilização de predicados que são estatisticamente mais seletivos, para que, conforme o fluxo de execução, sejam trabalhados com uma menor quantidade de resultados intermediários.

Para a organização entre predicados de uma mesma classe, são processados anteriormente predicados que lidam com colunas únicas, e caso lidem com a mesma quantidade, então é utilizado o algoritmo de HyperLogLog para, com base em *column sketches*, realizar predição da cardinalidade da coluna a ser trabalhada, optando-se pela coluna com menor cardinalidade para ser processada anteriormente. A representação intermediária é realizada através da identificação de padrões de computação para escolher entre bitmaps e arrays, utiliza representações compactas de pares de tuplas para reduzir o processamento, sendo um diferencial da maioria dos outros operadores de *Self Join* que expressam relacionamento de pares de tuplas, o que tende a ter baixa seletividade.

Para a classe com menor seletividade, sendo a Inequalidade, realiza-se a execução híbrida do algoritmo a ser executado conforme o volume de dados a ser trabalhado, utilizando-se o algoritmo de HyperLogLog para que, conforme a cardinalidade da coluna do predicado sendo executado, opta-se entre a execução dos algoritmos *IEJoin*(otimizado para alta cardinalidade), *Binary-Hash-Sort-Merge*(otimizado para média cardinalidade), *Hash-Sort-Merge*(otimizado para baixa cardinalidade). Utiliza-se o algoritmo de HyperLogLog para a predição da cardinalidade, devido a, ainda que possa ocorrer erro na predição, como tem um custo baixo de processamento, então mesmo com possíveis erros o custo de processamento é compensado durante a execução.

Utilizando-se a Tabela 2.1 , será representado o fluxo de execução da detecção de violações com o algoritmo do FACET. Dado o conjunto de DCs dada pela Tabela 2.2, utilizando-se a DC $\phi 2 : \neg(t_x.\text{IDCargo} = t_y.\text{IDCargo} \wedge t_x.\text{DataInício} < t_y.\text{DataInício} \wedge t_x.\text{Salário} > t_y.\text{Salário})$ encontrada sobre a Tabela 2.1 definida anteriormente.

Durante o planejamento da consulta, após a divisão de predicados por classes, as operações são realizadas em ordem, respectivamente, por igualdades, não igualdades e desigualdades. O primeiro refinamento a ser executado seria:

$$p1 : t_x.\text{IDCargo} = t_y.\text{IDCargo}$$

Já para a ordenação dos predicados restantes, devido a estarem na mesma classe (inequalidade) e como em ambos os casos estarem se tratando de inequalidades com apenas uma coluna, portanto se utiliza o algoritmo HyperLogLog, o qual realiza predição sobre a cardinalidade das colunas envolvidas na junção, tal que, neste caso é predito que a cardinalidade da coluna DataInício = 4 e que a cardinalidade da coluna Salário = 5. Devido a isso, como se priorizam colunas com menor cardinalidade, a ordem para a classe de inequalidade seria por:

$$p2 : t_x.\text{DataInício} > t_y.\text{DataInício}$$

$$p3 : t_x.\text{Salário} > t_y.\text{Salário}$$

E o fluxo do funcionamento do FACET para o predicado, como na Figura 2.2

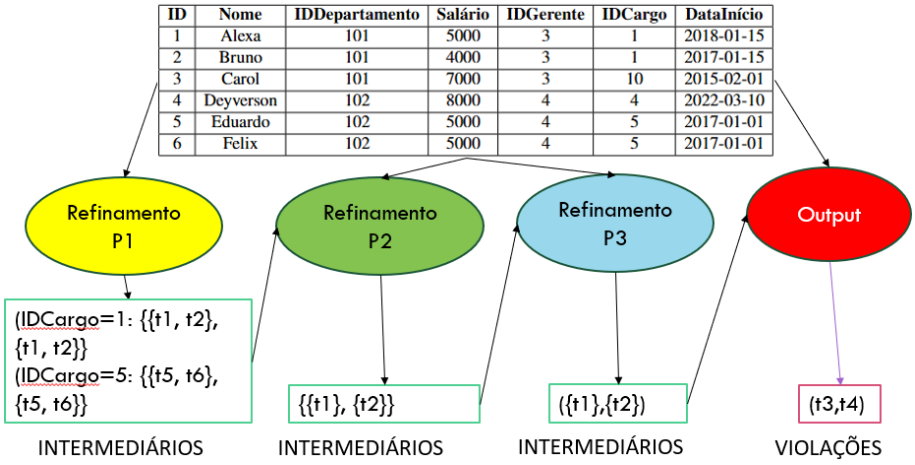


Figura 2.2: Fluxo Execução FACET

Introduziram-se duas formas de detecção de violações dado um conjunto de dados e um conjunto de DCs, tendo sido exemplificado a execução do FACET, o qual é o estado da arte para a detecção de violações. Os resultados obtidos nessa fase serão posteriormente utilizados para soluções de limpeza de dados e construção de semânticas íntegras.

2.3 FONTES DE RUÍDO E INCONSISTÊNCIAS EM DADOS

A qualidade dos dados em sistemas reais é frequentemente comprometida por uma variedade de erros e inconsistências, coletivamente referidos como 'ruído'. Este ruído pode surgir de diversas fontes, incluindo entrada de dados manual, falhas em processos de integração ou a evolução natural dos sistemas. Para investigar o impacto desses erros de forma sistemática, este trabalho adota um processo de **poluição de dados**, que consiste na introdução sintética e controlada de ruído sobre um conjunto de dados original. Para contextualizar o estudo do impacto de tais problemas, esta seção descreve algumas classes comuns de erros. Muitos desses erros são frequentemente encontrados em ambientes de integração, como em projetos de *Data Warehousing*, e podem ser modelados e detectados por DCs. Conforme destacado por Inmon (Inmon, 2005), a ideia de esperar que os dados operacionais estejam limpos antes de utilizá-los para análise é teoricamente atraente, mas impraticável.

Na realidade, os problemas de qualidade de dados devem ser gerenciados ativamente. Para contextualizar o estudo do impacto de tais problemas na descoberta de DCs, esta seção descreve algumas classes comuns de erros. Com base na literatura sobre qualidade de dados (Rahm, 2000), os ruídos podem ser categorizados de diversas formas. Para os propósitos deste trabalho, são agrupados em problemas que podem ser observados tanto em uma única fonte de dados quanto no resultado da integração de múltiplas fontes.

2.3.1 Problemas em Origem Única

Os dados carregados em uma *Data Warehouse* podem ser provindos de diferentes formas, tal como ODBC (tabelas com esquemas definidos e relações) ou de diferentes formatos como XML, CSV, JSON (sendo estes com menor definição de constantes/correlações a serem respeitadas), assim são definidos domínios de problemas.

- **Atributo**: Para este tipo de problema, referem-se às instâncias em que os valores são ilegais para o domínio do atributo, como data com mês maior que 12 ou menor que 1, dia maior que 31.
- **Registro**: Em nível de registro, a violação refere-se à violação da dependência entre atributos de um mesmo registro por outro atributo deste mesmo registro. Como dado uma data de nascimento e a idade, a idade ser incompatível.
- **Tipo de registro**: A violação desse registro sugere-se a violação de unicidade de um atributo como de uma tabela de cadastro existirem dois registros com CPFs iguais e nomes distintos.
- **Violação de Integridade Referencial**: A violação de integridade referencial refere-se a registros referenciados como de Foreign Keys serem violados por registros inexistentes, como um registro de compra conter um atributo contendo um CPF de compra, porém na tabela com compradores este CPF não existir.

2.3.2 Problemas em Múltiplas Origens

Ao realizar a carga utilizando como origem múltiplas fontes, necessita-se a identificação de correlação entre os dados, como a identificação de registros que se interseccionam. Porém, um obstáculo para isso é que cada fonte serve a um propósito próprio, seguindo não necessariamente as mesmas regras. Para que esses dados sejam integrados, é necessário identificar duplicação

para a eliminação, possibilidades de realizar junções entre os dados por meio de relacionamentos entre as origens.

- **Conflito de nome:** Cada origem pode seguir um padrão de nomenclatura para as colunas, um possível exemplo disso seria tendo duas origens, em uma estar definida a coluna CD_CPF_PESSOA e em outra COD_CPF_PESSOA , ambos os atributos são referentes ao Código do CPF da Pessoa.

- **Representação:** Mesmo com colunas com mesmo nome, é possível terem representações diferentes, como representação de valores monetários para uma fonte estar sendo utilizado Dólares e em outro estar sendo utilizado Real, ou como representação de gêneros estar sendo representado por valores inteiros e em outro como carácter.

- **Duplicadas:** Para a identificação de valores duplicados podem ser verificado a variância entre os valores dos atributos das fontes distintas, após a identificação necessita-se a concatenação dos valores e posterior remoção da duplicação.Um possível caso seriam duas origens em que para uma delas é utilizado na coluna NOME o {Nome da pessoa + Último nome} de uma pessoa e em outro ser representado pelo {Nome completo}.

2.3.3 A Importância da Geração Controlada de Erros em Dados

A avaliação da robustez de algoritmos de perfilamento de dados, como os de DCs, e das métricas de qualidade associadas, é fundamental para garantir sua eficácia em cenários práticos, onde os dados frequentemente contêm erros e inconsistências. No entanto, a criação de *benchmarks* com erros realistas e um *ground truth* conhecido para avaliação é um desafio considerável.

Nesse contexto, o trabalho de (Arocena et al., 2015) apresenta uma contribuição relevante ao introduzir o BART (*Benchmarking Algorithms for Data Repairing and Translation*). Este sistema propõe uma metodologia para a geração sistemática e controlada de diversos tipos de erros em conjuntos de dados limpos, permitindo a criação de conjuntos de dados ruidosos para a avaliação rigorosa de algoritmos de limpeza e, por extensão, de técnicas de perfilamento de dados. A Figura 2.3 demonstra a GUI do sistema, o qual permite a pré-configuração de relações previamente encontradas, além da coleta de estatísticas referentes à introdução de erros sinteticamente posteriormente.

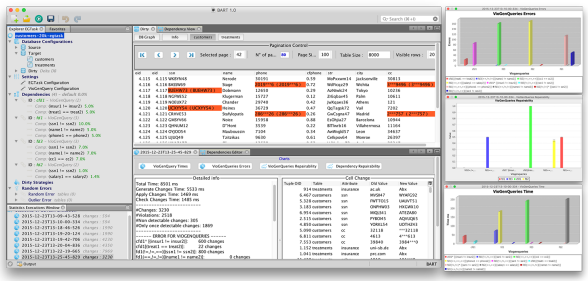


Figura 2.3: GUI BART

O artigo classifica os erros que podem ser gerados em duas categorias principais:

- **Valores Ausentes (*Missing Values*):** Simulam a falta de informação, podendo ser introduzidos seguindo diferentes mecanismos, como MCAR (*Missing Completely At Random*), MAR (Missing At Random) ou MNAR (Missing Not At Random).

- **Valores Errôneos (*Erroneous Values*):** Esta categoria abrange uma ampla gama de corrupções de dados, incluindo:
  - *Erros de Digitação (Typos):* Pequenas alterações nos valores, como inserções, deleções, substituições ou transposições de caracteres, permite configurar a quantidade de caracteres a serem afetados.
  - *Outliers e Inliers:* Valores atípicos que se desviam significativamente do esperado (outliers) ou valores inesperados que ainda se encontram dentro do intervalo de domínio (inliers).
  - *Troca de Valores (Swapping):* Troca de valores entre colunas diferentes ou dentro da mesma coluna para tuplas distintas.
  - *Valores Espúrios/Falsos (Bogus Values):* Introdução de valores inteiramente novos no conjunto de dados, que não existiam previamente. Estes podem ser gerados com base no domínio conhecido do atributo ou de forma aleatória, simulando a inserção de dados completamente incorretos ou fabricados.
  - *Contradições:* Geração de violações a regras de negócio ou restrições de integridade conhecidas.

A capacidade de simular uma gama tão variada e realista de problemas de qualidade de dados, incluindo a criação de *Bogus Values*, é fundamental para testar a robustez e a eficácia das abordagens de tratamento e análise de dados.

3 PROPOSTA

3.1 OBJETIVO

O objetivo geral deste trabalho é analisar o impacto da introdução de poluição sintética em conjuntos de dados sobre as métricas estado da arte na avaliação de *Denial Constraints* — em especial, *coverage*, *succinctness* e *interestingness* — a fim de compreender a robustez e o comportamento dos algoritmos de descoberta de DCs em cenários com diferentes níveis de ruído. Para tal intuito, serão utilizados os conjuntos de dados: Airport, Hospital, e para a inclusão de erros utilizando a técnica MCAR(*Missing Completely at Random*) para a escolha de registros dentro do conjunto de dados que serão utilizados para inserir dados inválidos.

3.2 HIPÓTESE

Considerando a diversidade de fontes de ruído e os tipos de erros em dados descritos na Seção 2.3, que vão desde valores ausentes até inconsistências semânticas e valores espúrios, para investigar esta questão de forma controlada e aprofundada, este estudo foca em um dos tipos de ruído mais prevalentes em cenários reais: a **ausência de dados**. A hipótese central deste trabalho é que a introdução de valores nulos em um conjunto de dados, em níveis crescentes de poluição, irá degradar de forma mensurável e progressiva a qualidade das DCs descobertas, além de alterar as suas características estruturais fundamentais.

3.3 PERGUNTAS DE PESQUISA

Para validar a hipótese e guiar a investigação experimental, este trabalho foi estruturado para responder ao seguinte conjunto de perguntas de pesquisa.

- Como os dados ausentes alteram a estrutura interna das DCs descobertas? Além das métricas de qualidade, a geração de ruído por nulos induz mudanças na própria estrutura das DCs que os algoritmos conseguem descobrir? Especificamente:
  - O perfil dos operadores lógicos (e.g., `EQUAL` vs. `UNEQUAL`) utilizados nos predicados se altera em resposta à ausência de valores?
  - O foco dos algoritmos muda em relação aos atributos mais frequentemente envolvidos nas DCs? As regras descobertas em dados com nulos se desviam estruturalmente de um conjunto de referência ideal (*Golden DCs*)?
- De que maneira a presença de valores ausentes, em diferentes níveis de poluição, afeta as métricas consolidadas utilizadas para avaliar a qualidade das DCs descobertas? Esta pergunta se desdobra em três investigações específicas:
  - Como a *coverage* das DCs é degradada, e o que isso implica sobre a generalidade das regras em dados com informação faltante?
  - Qual o efeito na *succinctness*, e isso sugere que os algoritmos geram regras mais longas ou complexas para se adaptar aos dados incompletos?
  - Como a *interestingness*, uma métrica combinada, se comporta sob diferentes ponderações α entre cobertura e concisão neste cenário de ruído por omissão?

- Os algoritmos de descoberta de DCs mantêm robustez consistente diante de diferentes níveis de ruído com valores ausentes?

3.4 CONTRIBUIÇÃO

- Proposição de um *pipeline* para avaliar a robustez de algoritmos de descoberta de DCs frente à presença de poluição sintética com dados ausentes nos dados.
- Análise dos efeitos da poluição com dados ausentes sobre métricas fundamentais de avaliação de DCs, fornecendo subsídios para a seleção de algoritmos mais adequados em contextos de ruído de valores ausentes.
- Discussão sobre o comportamento de *coverage*, *succinctness* e *interestingness* em cenários de ruído sintético de valores ausentes dos dados.
- Apoio à construção de processos de qualidade de dados mais resilientes, com base na identificação de restrições mais informativas mesmo em ambientes com informações corrompidas.

4 *PIPELINE* DE AVALIAÇÃO DO IMPACTO DE RUÍDO

Este capítulo detalha o *pipeline* experimental desenvolvido para investigar a robustez da descoberta de DCs em cenários de dados ruidosos. O objetivo deste *pipeline* é fornecer uma metodologia sistemática e reprodutível para avaliar como diferentes níveis de poluição afetam os algoritmos de descoberta e as métricas de qualidade das DCs resultantes. O *pipeline* experimental, ilustrado na Figura 4.1, é composto por quatro etapas sequenciais principais: (1) Geração de Ruído, (2) Descoberta de DCs, (3) Cálculo de Métricas, e (4) Análise Comparativa. Cada etapa foi projetada para isolar e avaliar aspectos específicos do impacto do ruído. As seções seguintes detalham cada uma dessas etapas.

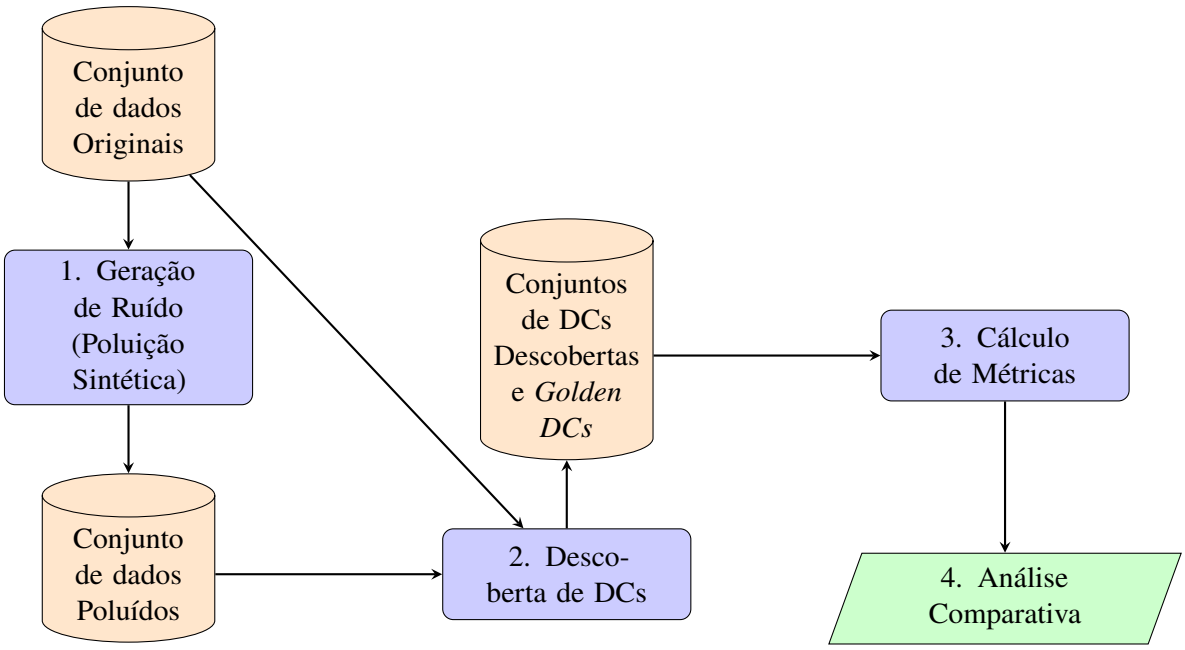


Figura 4.1: Pipeline experimental para avaliação do impacto de ruído na descoberta de DCs.

4.1 CONJUNTO DE DADOS ORIGINAIS E *GOLDEN DCS*

Um componente central da nossa avaliação experimental é o uso de um conjunto de referência de *Golden Denial Constraints* (Golden DCs). O conceito de *Golden DCs*, conforme discutido e utilizado no trabalho de (Martin et al., 2025), refere-se a um conjunto de restrições de integridade conhecidas como verdadeiras e semanticamente relevantes para um determinado conjunto de dados.

Para avaliar a robustez dos algoritmos de descoberta de DCs nos cenários apresentados, foram selecionados dois conjuntos de dados públicos e amplamente utilizados na literatura de qualidade de dados e perfilamento de dados: *Hospital* e *Airport*.

A escolha destes conjuntos de dados se baseou nos seguintes critérios:

- **Uso em Trabalhos Anteriores:** Ambos os conjuntos de dados são frequentemente utilizados para avaliar algoritmos de descoberta de restrições de integridade, como nos trabalhos de (X. Chu, 2013) e (Bleifuß et al., 2017), o que facilita a contextualização dos resultados.

- **Diversidade de Domínios:** Os conjuntos de dados abrangem domínios distintos (saúde e aviação), com diferentes tipos de dados (numéricos, categóricos, códigos, etc.), permitindo uma avaliação mais abrangente do comportamento dos algoritmos.
- **Disponibilidade de Regras de Referência:** A existência de um conjunto de *Golden DCs* para estes conjuntos de dados é crucial para a nossa metodologia de avaliação, permitindo uma análise da diferença estrutural das métricas.

O conjunto de dados **Airport** agrega dados sobre aeroportos de todo o mundo, compilados a partir de diversas fontes públicas. Este conjunto de dados é interessante por conter uma mistura de códigos padronizados, coordenadas geográficas e informações textuais. A diversidade de seus tipos de dados inclui:

- **Códigos Padronizados:** Atributos como `ident`, `gps_code`, `iso_country` e `iso_region` utilizam códigos internacionais (IATA, ISO 3166), um cenário comum em integração de dados.
- **Dados Categóricos:** A coluna `type` classifica os aeroportos em categorias como *small_airport* ou *heliport*.
- **Dados Textuais:** Colunas como `name` e `municipality` contêm nomes próprios, que são suscetíveis a variações e erros.
- **Dados Numéricos e Geográficos:** O atributo `elevation_ft` representa um valor numérico (elevação), enquanto `coordinates` contém informação numérica estruturada, sendo úteis para avaliar predicados de comparação.

O conjunto de dados **Hospital** contém informações sobre hospitais nos Estados Unidos. Ele é um *benchmark* clássico para tarefas de perfilamento e limpeza de dados.

A amostragem aleatória de 1000 registros, necessária para os experimentos, foi realizada mediante um *script* em Python que utiliza a função *sample()* da biblioteca *Pandas* após a conversão dos dados de entrada para uma estrutura de *dataframe*.

Apesar de algumas colunas representarem valores categóricos (como Emergency Service), todos os atributos estão codificados no formato de *string* no arquivo. A sua relevância reside na mistura de atributos que são típicos de sistemas de informação em saúde:

- **Identificadores e Códigos:** Atributos como `Provider Number`, `ZIP Code` e `Measure Code` representam códigos e identificadores que deveriam seguir padrões estritos.
- **Dados Categóricos e Textuais:** Colunas como `Hospital Name`, `City`, `State`, `Hospital Type` e `Condition` contêm dados textuais e categóricos, ideais para testar predicados de igualdade e desigualdade sobre *strings*.

Para este trabalho, o conjunto de *Golden DCs* foi definido com base na metodologia de validação de restrições proposta (Martin et al., 2025). A utilização das *Golden DCs* é o que nos permite ter um limiar válido comparativo. Os conjuntos de dados escolhidos possuem as propriedades conforme a Tabela 4.1.

Tabela 4.1: Resumo dos conjuntos de dados Utilizados no Estudo

Conjunto de dados	**n.º de Colunas**	**n.º de Registros**	***Golden DCs***
Airport	18	55.113	2
Hospital	15	114.919	35

4.2 GERAÇÃO DE RUÍDO

Para simular cenários com dados de baixa qualidade, foi realizada uma estratégia de poluição sintética nos conjuntos de dados. A abordagem adotada seguiu o mecanismo MCAR (*Missing Completely At Random*), no qual os valores são corrompidos de forma totalmente aleatória, sem depender de outros atributos ou da distribuição dos dados. Essa técnica é amplamente utilizada em simulações de dados ausentes estudos na literatura como nos trabalhos (Little e Rubin, 2019) e (Arocena et al., 2015), sendo relevante ao estudo de coerência das relações obtidas em ambientes reais. Essa estratégia é interessante para o estudo de robustez dos algoritmos

A geração dos arquivos poluídos foi feita em diferentes taxas de corrupção (0%, 5%, 10%, 15%), permitindo observar o impacto progressivo nas métricas de *coverage*, *succinctness* e *interestingness* das DCs extraídas.

4.3 DESCOBERTA DCS

Para os propósitos deste trabalho, a plataforma Metanome foi utilizada como o ambiente de execução para os algoritmos de descoberta de DCs, Hydra e DCFinder.

4.3.1 DCFinder

O *DCFinder* (Pena et al., 2019) é um algoritmo projetado para descobrir DCs tanto exatas quanto aproximadas. Ele combina estruturas de dados chamadas *position list indexes* com técnicas baseadas na seletividade de predicados para validar candidatos a DCs de forma eficiente. Essa abordagem permite a descoberta de DCs que podem conter exceções, tornando-o adequado para conjuntos de dados que apresentam inconsistências ou erros.

4.3.2 Hydra

O *Hydra* (Bleifuß et al., 2017) é um algoritmo eficiente para a descoberta de DCs exatas em conjuntos de dados relacionais. Ele supera a complexidade quadrática de tempo de execução de métodos anteriores, apresentando um crescimento de tempo quase linear em relação ao número de tuplas. Essa eficiência é alcançada por meio de técnicas de amostragem focada e estruturas de dados otimizadas, permitindo a descoberta de DCs em grandes volumes de dados em um tempo significativamente reduzido.

4.4 CONJUNTOS DE DCS DESCOBERTAS

Para a etapa de armazenamento dos resultados e para a execução dos cálculos das métricas de qualidade de *coverage* e *succinctness*, foi escolhida a ferramenta ***DuckDB***. O *DuckDB* é um sistema de gerenciamento de banco de dados (SGBD) analítico, de código aberto e projetado para ser executado *in-process*, ou seja, embutido diretamente na aplicação que o utiliza, sem a necessidade de um servidor externo.

A escolha do *DuckDB* para este trabalho se justifica por três características principais que o tornam ideal para pipelines de análise de dados:

- **Desempenho em Cargas de Trabalho Analíticas:** Diferente de bancos de dados transacionais como o SQLite, o *DuckDB* utiliza uma arquitetura de armazenamento colunar e uma execução de consultas vetorizada. Isso o torna extremamente eficiente para consultas analíticas, que envolvem agregações e varreduras de grandes volumes de dados, como as necessárias para calcular a métrica de *coverage* sobre milhares de pares de tuplas.
- **Integração com o Ecossistema Python:** O *DuckDB* possui uma integração nativa e de alta performance com a biblioteca *Pandas*. Ele permite executar consultas SQL diretamente sobre *DataFrames* do *Pandas* sem a necessidade de importar os dados, facilitando a interação entre a manipulação de dados em Python e a execução de consultas complexas em SQL.
- **Simplicidade e Portabilidade:** Por ser um banco de dados *in-process* e sem dependências externas, o *DuckDB* simplifica a configuração do ambiente experimental. Os dados e os resultados podem ser armazenados em um único arquivo, o que garante a portabilidade e a reprodutibilidade do *pipeline* de análise.

As métricas adotadas neste trabalho foram *Coverage*, *Succinctness* e *Interestingness*. As métricas de *Coverage* e *Succinctness* foram calculadas diretamente no ambiente *DuckDB*, com base nos predicados extraídos dos algoritmos de perfilamento de dados aplicados aos conjuntos de dados.

A partir desses valores, foram derivadas as variações da métrica de *Interestingness* conforme diferentes valores do parâmetro α, permitindo analisar o impacto da poluição sintética sobre a qualidade das DCs extraídas. Para cada valor de α, foi também calculada a média do *interestingness*, com o intuito de observar tendências de *overfitting* à medida que a porcentagem de corrupção dos dados aumenta.

4.5 CÁLCULO DAS MÉTRICAS

Nesta seção são descritas as abordagens utilizadas para o cálculo das métricas de *Succinctness* e *Coverage* para cada DC identificada nos experimentos. As consultas das métricas foram implementadas em Python, e posteriormente executadas sobre os dados armazenados no banco de dados DuckDB.

4.5.1 *Succinctness*

A métrica de *succinctness* visa medir o grau de concisão de uma DC, considerando a quantidade de símbolos distintos necessários para sua representação. Conforme definido por (Pena et al., 2022), o valor mínimo possível de uma DC é dado pela menor quantidade de elementos distintos em sua definição (atributos, tabelas, operadores e constantes), sendo este valor de referência igual a 4.

Para o cálculo da métrica, são percorridos os arquivos resultantes das execuções dos algoritmos da etapa 4.3, percorrendo os predicados de uma DC e constrói um alfabeto contendo os identificadores únicos de colunas, tabelas, operadores e constantes envolvidas, utilizando estrutura de conjuntos, para tal. Após isso, o cálculo é então realizado conforme a fórmula 2.3.

4.5.2 *Coverage*

A métrica de *Coverage* definida pela Equação 2.5 mede o quanto uma DC cobre os pares de tuplas não violadores, levando em consideração o grau de satisfação parcial dos predicados. Utilizando a metodologia descrita em (Pena et al., 2019), a cobertura é calculada com base nos pares de tuplas que não violam a DC completamente, atribuindo pesos proporcionais ao número de predicados satisfeitos por par.

Assim, utilizou-se uma função em linguagem Python para, dado os arquivos de saída dos algoritmos de descoberta de regras, converte do formato JSON para SQL, subdividindo os predicados pertencentes a cada DC, seguindo o seguinte procedimento:

1. Constrói, para cada predicado da DC, uma expressão SQL que avalia se ele é satisfeito entre pares de tuplas distintos (t_0, t_1) da tabela de entrada.

2. Soma, para cada par de tuplas, a quantidade de predicados satisfeitos.

3. Calcula a cobertura utilizando a fórmula 2.5

A execução do cálculo da métrica *Coverage* é realizada diretamente no banco de dados DuckDB, por meio de consultas SQL. Essa abordagem garante reprodutibilidade e facilidade na integração com os dados experimentais. Vale destacar que a consulta envolve um *CROSS JOIN* entre todas as tuplas da tabela — operação que, por natureza, possui complexidade quadrática, uma vez que gera $n \times (n-1)$ pares para um conjunto de dados com n linhas. Como consequência, o tempo de execução cresce rapidamente conforme o volume de dados aumenta, o que pode tornar o processamento inviável em conjunto de dados muito grandes sem a aplicação de técnicas de amostragem ou otimização.

Como exemplo, o cálculo para uma das *Goldens DC* do conjunto de dados Airport, em que é obtido um coverage máximo de 1:

$$\neg\big(t_0.\text{ident} = t_1.\text{ident} \wedge t_0.\text{latitude} \neq t_1.\text{latitude}\big)$$

A consulta SQL para cálculo da métrica *coverage* segue a estrutura abaixo, considerando os predicados do DC:

```sql
WITH PARES AS (
    SELECT T0.row_num AS ID0, T1.row_num AS ID1,
          CASE WHEN T0."ident" = T1."ident" THEN 1 ELSE 0 END AS P0,
          CASE WHEN T0."latitude" <> T1."latitude" THEN 1 ELSE 0 END AS P1
    FROM airport T0, airport T1
    WHERE T0.row_num <> T1.row_num
),
CLASSIFICACAO AS (
    SELECT *, P0 + P1 AS predicados_satisfeitos
    FROM PARES
),
RESUMO AS (
    SELECT predicados_satisfeitos AS k, COUNT(*) AS kE,
          (CAST(k AS FLOAT) + 1) / 2 AS w_k
    FROM CLASSIFICACAO
    WHERE k < 2
    GROUP BY k
),
COVERAGE_CALC AS (
    SELECT SUM(kE * w_k) AS numerador, SUM(kE) AS denominador
    FROM RESUMO
```

```
22 )
23 SELECT ROUND(numerador * 1.0 / NULLIF(denominador, 0), 6) AS coverage FROM
       COVERAGE_CALC;
```

Listing 4.1: Exemplo de consulta para cálculo da métrica coverage

4.6 ARMAZENAMENTO DE RESULTADOS

Foi escolhida uma estratégia *in-database* para realizar os experimentos, tal que, após a execução dos algoritmos de perfilamento de dados, foram armazenados os resultados em uma base de Metadados do conjunto de dados, utilizando o SGBD DuckDB. Os dados consolidados encontram-se na tabela *PROFILING_METADATA*, composta pelas seguintes colunas:

- **DS_DATA_PROFILER**: Nome do algoritmo de perfilamento utilizado;
- **PERC_DATASET_CORRUPTION**: Porcentagem de corrupção aplicada ao conjunto de dado;
- **ID_PROFILING_DATASET_NUMBER**: Identificador da relação de perfilamento obtida;
- **DS_DATASET**: Nome do conjunto de dados utilizado como base para o estudo;
- **DS_PROFILING_INFO**: Expressão matemática da relação de perfilamento detectada;
- **VL_COVERAGE**: Valor da métrica *coverage* calculado;
- **VL_SUCCINCTNESS**: Valor da métrica *succinctness* calculado.

DS_DATA_PROFILER	PERC_DATASET_CORRUPTION	ID_PROFILING_INFO	DS_DATASET	DS_PROFILING_INFO	VL_COVERAGE	VL_SUCCINCTNESS
Golden DC	0.0	1	Airport	$\neg[t_0.ident = t_1.ident \wedge t_0.latitude \neq t_1.latitude]$	1.000	1.000
Golden DC	0.0	2	Airport	$\neg[t_0.iso\_country = t_1.iso\_country \wedge t_0.gps\_code \neq t_1.gps\_code]$	0.911	0.667

Tabela 4.2: Amostra da tabela `PROFILING_METADATA` com métricas de Coverage e Succinctness para *Golden DCs 'Airport'*

4.7 ANÁLISE COMPARATIVA

A etapa final do pipeline, cujos resultados detalhados são apresentados no Capítulo 5, consiste na análise comparativa dos dados de métricas obtidos. Nesta fase, foram investigadas as tendências e os padrões para responder às perguntas de pesquisa, comparando o impacto da poluição entre os diferentes níveis, algoritmos, conjuntos de dados, contrastando os resultados com o conjunto de referência de *Golden DCs*.

Os algoritmos de descoberta de DCs podem gerar um volume expressivo de regras para um único conjunto de dados, com cada regra possuindo valores de métricas distintos. Para viabilizar uma análise comparativa clara e objetiva entre os diferentes cenários experimentais (combinações de algoritmo, conjunto de dados e nível de poluição), optou-se por agregar os resultados utilizando a **média aritmética** para as métricas de *coverage*, *succinctness* e *interesstingness*. Dado que os valores unitários das DCs para as métricas estão carregados

conforme descrito na seção 4.5, para o cálculo médio foi utilizado a função de agregação *MEAN* da linguagem SQL agrupando conforme o cenário analisado.

A utilização da média como medida de tendência central permite sumarizar a distribuição dos valores de cada métrica em um único indicador representativo. Dessa forma, a `MÉDIA COVERAGE` indica a média aritmética da métrica coverage de uma DC descoberta em um determinado cenário, enquanto a `MÉDIA SUCCINCTNESS` reflete a sua succinctness média. A métrica de `MÉDIA INTERESTINGNESS` representa o cálculo médio da métrica, conforme o cálculo ponderando entre as métricas succinctness e coverage. Esta abordagem facilita a identificação de tendências gerais e a comparação direta do impacto da poluição na robustez de cada algoritmo, conforme apresentado nas análises subsequentes.

Além da avaliação quantitativa das métricas, foi realizada uma análise estrutural para investigar como os algoritmos se adaptam ao ruído. Para isso, o perfil das DCs descobertas em cada cenário — especificamente a distribuição de operadores lógicos e a frequência de envolvimento dos atributos — foi comparado com a estrutura de um conjunto ideal de *Golden DCs*. Esta análise comparativa foi fundamental para avaliar se as adaptações induzidas pelo ruído aproximam ou distanciam as regras geradas de um padrão semanticamente relevante, revelando as estratégias e os vieses de cada algoritmo.

5 EXPERIMENTOS

Os experimentos descritos neste trabalho foram conduzidos em ambiente local, conforme especificações detalhadas na tabela 5.1.

Tabela 5.1: Especificações do ambiente de execução dos experimentos

Componente	**Especificação**
Processador (CPU)	AMD Ryzen 5 5500U with Radeon Graphics
Arquitetura	x86_64 (64 bits)
Núcleos físicos	6
Threads	12
Clock (máx / mín)	4.05 GHz / 0.40 GHz
Memória RAM	16 GB DDR4
Sistema Operacional	Ubuntu 22.04 LTS (64 bits)

5.1 ANÁLISE DA MÉTRICA DE *COVERAGE*

A métrica de *Coverage* quantifica a proporção de pares de tuplas em um conjunto de dados que satisfazem uma determinada DC. Em outras palavras, ela mede o quão bem uma DC representa um padrão consistente e generalizável nos dados. Uma *Coverage* elevada indica que a DC é amplamente respeitada no conjunto de dados, enquanto uma *Coverage* baixa sugere que a regra é frequentemente violada ou se aplica a um subconjunto muito específico dos dados. Esta métrica é fundamental para responder à primeira pergunta de pesquisa deste trabalho, que investiga como a poluição sintética afeta a capacidade de cobertura das DCs identificadas.

É crucial ressaltar que o cálculo exato da métrica de *Coverage*, conforme detalhado na Seção 4.5.2, envolve a avaliação de todos os pares de tuplas possíveis em um conjunto de dados, resultando em uma complexidade computacional quadrática ($O(n^2)$ em relação ao número de registros n). Para conjuntos de dados volumosos, frequentemente encontrados em ambientes reais, essa abordagem pode se tornar proibitiva em termos de tempo de processamento, inviabilizando a análise em tempo hábil. Diante dessa limitação prática, e para viabilizar a condução dos experimentos deste trabalho em um escopo exequível, optou-se pela utilização de amostras representativas dos conjuntos de dados para o cálculo e análise da *Coverage* e demais métricas. Embora a amostragem introduza uma aproximação, ela permite investigar as tendências e o impacto da poluição de forma eficiente, refletindo uma abordagem pragmática que pode ser necessária em cenários de grande escala.

No contexto deste estudo, que avalia o impacto da introdução de ruído sobre estas amostras, espera-se que um aumento na percentagem de poluição nos conjuntos de dados resulte em uma diminuição da *Coverage* média das DCs descobertas. Isso pode ocorrer por duas razões principais: (1) os valores corrompidos, ao serem introduzidos aleatoriamente (MCAR), podem quebrar predicados que antes eram satisfeitos, tornando a DC inválida para um número maior de pares de tuplas; ou (2) os algoritmos de descoberta, na tentativa de manter a validade das regras, podem gerar DCs menos generalizáveis (mais específicas) para se adequar aos dados corrompidos, resultando em menor suporte estatístico e, consequentemente, menor cobertura. As subseções seguintes analisam o comportamento da *coverage* para os algoritmos Hydra e DCFinder, utilizando os dados experimentais apresentados.

5.1.1 Comportamento da *Coverage* para o Algoritmo Hydra

As Figuras 5.1 e 5.2 ilustram o impacto do aumento progressivo da poluição (0%, 5%, 10% e 15% de valores nulos) na média de *Coverage* das DCs identificadas pelo algoritmo Hydra nos conjuntos de dados Hospital e Airport, respectivamente. Os dados numéricos agregados são:

- **Conjunto de dados Airport:** A `MÉDIA_COVERAGE` inicia em `0.611069` (0% de poluição) e decai consistentemente para `0.388427` (5%), `0.361746` (10%), e `0.325848` (15%).
- **Conjunto de dados Hospital_1000:** A `MÉDIA_COVERAGE` começa em `0.562225` (0%) e apresenta uma queda também consistente para `0.524336` (5%), `0.489918` (10%), e `0.450665` (15%).

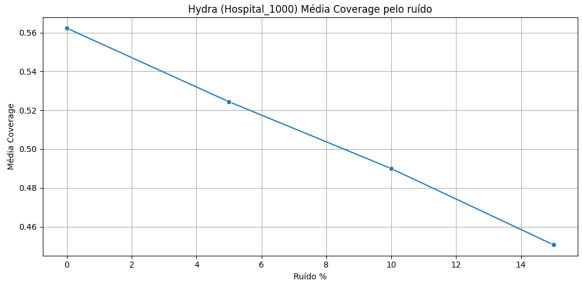


Figura 5.1: Média de *Coverage* das DCs (Hydra) no conjunto de dados Hospital em função do ruído MCAR.

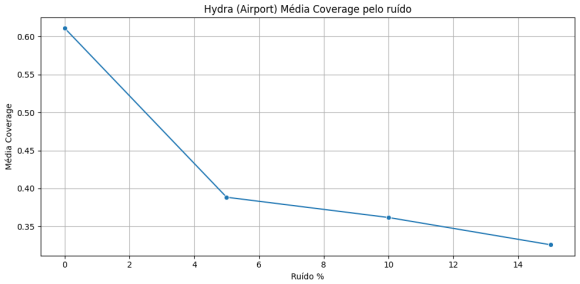


Figura 5.2: Média de *Coverage* das DCs (Hydra) no conjunto de dados Airport em função do ruído MCAR.

A queda observada na *Coverage* média para o Hydra é um resultado esperado, dadas as características do algoritmo e a natureza da poluição. O Hydra é projetado primariamente para a descoberta eficiente de DCs exatas (Bleifuß et al., 2017). A introdução de valores nulos, mesmo que completamente aleatória (MCAR), tem o potencial de invalidar predicados que compunham DCs anteriormente válidas. Por exemplo, um predicado $t.A = t'.B$ não será satisfeito se $t.A$ ou $t'.B$ for nulo. Como o Hydra busca regras que não possuam violações no conjunto de evidências analisado (inicialmente uma amostra, depois complementado)(Bleifuß et al., 2017), qualquer par de tuplas que, devido a um nulo, deixe de satisfazer a conjunção de predicados de uma DC candidata (ou passe a satisfazê-la, no caso da negação da DC) altera o conjunto de evidências. O artigo (Bleifuß et al., 2017) não detalha extensivamente o tratamento de nulos, mas a prática comum em descoberta de DCs exatas é que predicados envolvendo nulos frequentemente não são considerados satisfeitos no sentido estrito. Assim, as DCs que permanecem válidas (exatas) nos dados poluídos tendem a ser aquelas que ou não envolvem os atributos corrompidos, ou são suficientemente específicas para evitar as instâncias onde os nulos "quebram"o padrão.

Isso naturalmente leva a uma redução na *Coverage* média, pois o espaço de padrões universalmente consistentes diminui. A estratégia de amostragem do Hydra, seguida pela completude do conjunto de evidências, visa encontrar todas as DCs exatas; se os padrões generalizados são fragmentados por nulos, a *Coverage* das DCs resultantes reflete essa nova realidade dos dados corrompidos, indicando que as regras descobertas possuem menor aplicabilidade geral.

5.1.2 Comportamento da *Coverage* para o Algoritmo DCFinder

De forma análoga, as Figuras 5.3 e 5.4 apresentam os resultados da *Coverage* média para as DCs descobertas pelo algoritmo DCFinder. Os dados numéricos agregados são:

- **Conjunto de dados Hospital_1000:** A `MÉDIA_COVERAGE` inicia em `0.748503` (0%), mantém-se relativamente estável em `0.744250` (5%), e então decai para `0.596226` (10%), e `0.548877` (15%).
- **Conjunto de dados Airport:** A `MÉDIA_COVERAGE` começa em `0.864066` (0%) e decai progressivamente para `0.724660` (5%), `0.646771` (10%) e `0.540813` (15%).

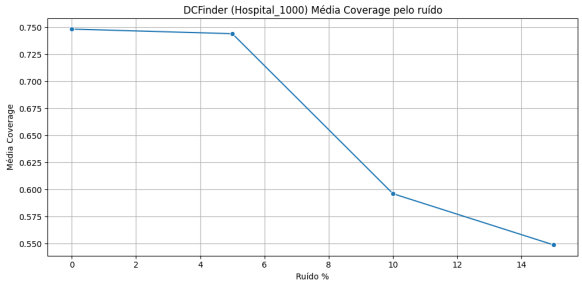


Figura 5.3: Média de *Coverage* das DCs (DCFinder) no conjunto de dados Hospital em função do ruído MCAR.

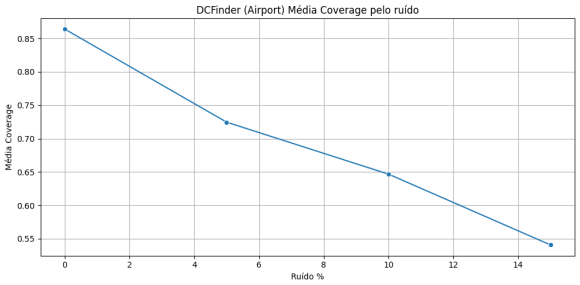


Figura 5.4: Média de *Coverage* das DCs (DCFinder) no conjunto de dados Airport em função do ruído MCAR.

O DCFinder, e a família de algoritmos descrita por (Pena et al., 2022) (incluindo ECP para construção do conjunto de evidências e INCS para enumeração), é projetado para descobrir tanto DCs exatas quanto aproximadas (Pena et al., 2019). Essa capacidade de lidar com aproximações é fundamental para entender seu comportamento sob ruído. A introdução de nulos afeta a validade dos predicados de forma similar ao Hydra. No entanto, a possibilidade de descobrir DCs aproximadas, utilizando a multiplicidade das evidências capturadas pelo ECP (Evidence Context Pipeline)(Pena et al., 2022), pode explicar a manutenção de uma *Coverage* relativamente alta em níveis baixos de poluição (e.g., Hospital_1000 com 5% de ruído, onde a queda é mínima).

O DCFinder pode identificar DCs que são violadas por uma pequena fração de pares de tuplas – potencialmente aqueles afetados pelos nulos – mas que ainda cobrem bem o restante dos dados, desde que o limiar de violação (definido pela função de aproximação g_1 (Pena et al., 2022)) não seja excedido. Contudo, à medida que a percentagem de poluição aumenta, a estrutura dos dados é mais significativamente perturbada. Mesmo para DCs aproximadas, o número de violações pode exceder os limiares aceitáveis, ou as regras que ainda se sustentam podem precisar se tornar mais específicas para evitar as áreas ruidosas. Essa especialização inerentemente reduz a *coverage*. A robustez na construção do conjunto de evidências pelo ECP (Pena et al., 2022) garante que a qualidade das DCs aproximadas seja bem avaliada. Se as DCs descobertas (exatas ou aproximadas) em dados poluídos se tornam mais específicas para contornar os nulos, sua *coverage* naturalmente diminui, indicando uma perda de generalidade.

5.2 ANÁLISE DA MÉTRICA DE *SUCCINCTNESS*

A métrica de *Succinctness* avalia o grau de concisão ou simplicidade de uma DC. Conforme definido por (Pena et al., 2022) e referenciado na Seção 2.1.3.1 deste trabalho, valores mais altos (próximos de 1) indicam DCs mais simples, com menos predicados, facilitando sua interpretação e validação. Esta seção investiga como a presença de ruído (valores nulos MCAR) nos dados afeta a *Succinctness* das DCs descobertas, buscando responder à segunda pergunta de pesquisa.

A hipótese subjacente é que, com o aumento da poluição, os algoritmos podem ser forçados a gerar DCs mais longas e complexas (com mais predicados) na tentativa de se ajustar às exceções e inconsistências introduzidas pelos dados ruidosos. Tal comportamento resultaria em uma diminuição do valor médio de *Succinctness*. Uma queda observada na *Succinctness* pode, portanto, sugerir uma tendência ao *overfitting* ao ruído, onde a DC perde generalidade ao incorporar predicados específicos para lidar com as instâncias corrompidas, tornando-se menos aplicável a dados limpos ou a um contexto mais amplo.

5.2.1 Comportamento da *Succinctness* para o Algoritmo Hydra

As Figuras 5.5 e 5.6 apresentam a variação da média de *Succinctness* das DCs descobertas pelo algoritmo Hydra nos conjuntos de dados Hospital e Airport, respectivamente, em função do aumento percentual de ruído. Os dados numéricos agregados são:

- **Conjunto de dados Airport:** A `MÉDIA_succinctness` começa em `0.596973` (0%) e diminui progressivamente para `0.473065` (5%), `0.452395` (10%), e `0.419728` (15%).
- **Conjunto de dados Hospital_1000:** A `MÉDIA_succinctness` inicia em `0.59` (0%) e também decai consistentemente para `0.474` (5%), `0.452` (10%), e `0.4125` (15%).

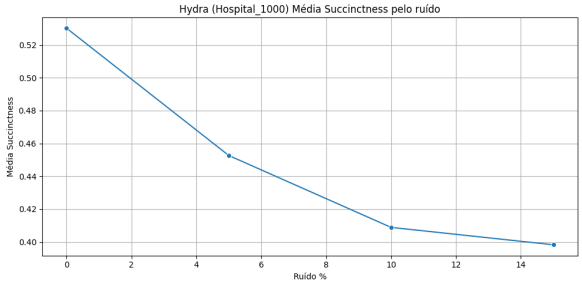


Figura 5.5: Média de *Succinctness* das DCs (Hydra) no conjunto de dados Hospital em função do ruído MCAR.

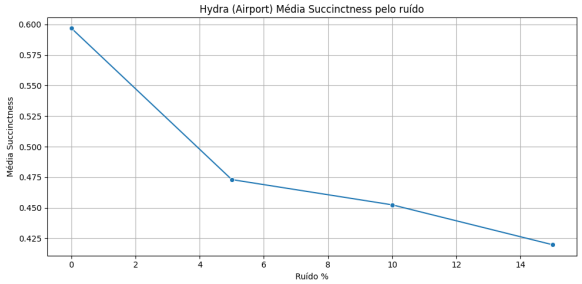


Figura 5.6: Média de *Succinctness* das DCs (Hydra) no conjunto de dados Airport em função do ruído MCAR.

A diminuição observada na *Succinctness* média com o aumento da poluição sugere que o Hydra tende a descobrir DCs mais longas (com mais predicados) nos conjuntos de dados ruidosos. Este fenômeno pode ser explicado pela necessidade do Hydra de encontrar DCs que sejam perfeitamente válidas (exatas) (Bleifuß et al., 2017). Quando valores nulos são introduzidos, padrões que antes eram capturáveis por DCs simples podem ser quebrados. Para que uma DC ainda seja considerada exata, ela pode precisar incorporar predicados adicionais que a tornem mais específica, "contornando"as instâncias afetadas pelos nulos. Por exemplo, uma regra simples $\neg(A = B \wedge C = D)$ pode ser violada por um nulo em A. O algoritmo pode, então, descobrir uma regra mais complexa como $\neg(A = B \wedge C = D \wedge E = F)$ que ainda se mantém exata. A fase de "Evidence Inversion"do Hydra (Bleifuß et al., 2017), que constrói DCs iterativamente, ao encontrar evidências que violam DCs candidatas simples (devido aos nulos), irá especializá-las adicionando predicados, resultando em DCs finais menos sucintas.

5.2.2 Comportamento da *Succinctness* para o Algoritmo DCFinder

As Figuras 5.7 e 5.8 ilustram o comportamento da *Succinctness* média das DCs descobertas pelo algoritmo DCFinder sob as mesmas condições de poluição. Os dados numéricos agregados são:

- **Conjunto de dados Hospital_1000:** A `MÉDIA_succinctness` começa em `0.795414` (0%), com leve queda para `0.791887` (5%), depois para `0.764848` (10%), e apresentando uma ligeira subida para `0.773333` (15%).
- **Conjunto de dados Airport:** A `MÉDIA_succinctness` inicia em `0.863988` (0%) e decai consistentemente para `0.819048` (5%), `0.784719` (10%), `0.770702` (15%).

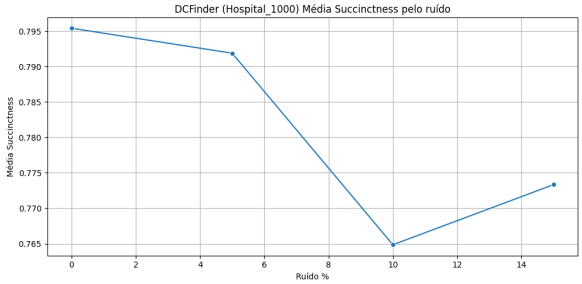


Figura 5.7: Média de *Succinctness* das DCs (DCFinder) no conjunto de dados Hospital em função do ruído MCAR.

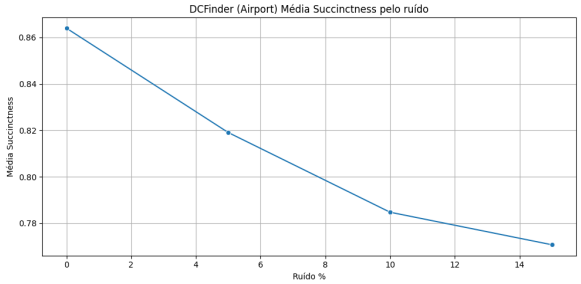


Figura 5.8: Média de *Succinctness* das DCs (DCFinder) no conjunto de dados Airport em função do ruído MCAR.

A tendência geral para o DCFinder também é de diminuição da *Succinctness* com o aumento da poluição, indicando que este algoritmo, de forma similar ao Hydra, tende a descobrir DCs mais complexas em dados ruidosos. Mesmo com a sua capacidade de encontrar DCs aproximadas(Pena et al., 2019), se os nulos quebram muitos padrões simples, as DCs resultantes (sejam elas exatas ou aproximadas que ainda satisfazem o limiar de violação) podem necessitar de predicados adicionais para definir corretamente o escopo onde ainda são consideradas válidas. O algoritmo INCS (*Indexed Negative Cover Search*), descrito por (Pena et al., 2022) e associado ao DCFinder, realiza uma busca por *"negative covers"* e aplica uma verificação de minimalidade tardia (Pena et al., 2022).

Se as DCs se tornam mais longas para se ajustar aos dados ruidosos, isso pode ser interpretado como um sinal de *overfitting* ao ruído: a DC perde generalidade (refletida na menor *Succinctness*) para se conformar a um conjunto de dados específico e corrompido. A ligeira subida na *succinctness* para o conjunto de dados Hospital_1000 com 15% de poluição (`0.773333` versus `0.764848` com 10%) é um comportamento atípico que merece atenção. Uma possível explicação é que, neste ponto específico de ruído, a distribuição das evidências pode ter levado o algoritmo INCS a encontrar um conjunto de DCs mínimas que, em média, eram ligeiramente mais curtas. Isso poderia ocorrer se o ruído mais elevado tornasse muitas DCs complexas candidatas insustentáveis (mesmo como aproximadas), e o processo de poda e minimalidade do INCS (Pena et al., 2019) favorecesse um conjunto remanescente de regras inerentemente mais simples, embora potencialmente com cobertura reduzida. Esta anomalia justificaria uma investigação mais aprofundada em trabalhos futuros, analisando possivelmente a distribuição dos comprimentos das DCs individuais e o número de DCs descobertas nesse ponto.

5.3 ANÁLISE DA DISTRIBUIÇÃO DA MÉTRICA DE *INTERESTINGNESS*

A métrica de *Interestingness*, conforme definida pela Equação 2.6), é amplamente utilizada para avaliar a qualidade global das DCs descobertas. Ela permite um balanceamento, através do parâmetro α, entre as métricas de *Coverage* da DC nos dados e a *Succinctness*. Valores de α próximos de 1 conferem maior peso à cobertura da DC, enquanto valores próximos de 0 valorizam mais a concisão dos predicados. Um valor de $\alpha = 0.5$ indica um peso equilibrado. Esta seção, respondendo à terceira pergunta de pesquisa, analisa como a *interestingness* média das DCs é impactada pela poluição por nulos MCAR.

5.3.1 *Denial Constraints* Descobertas

5.3.1.1 Algoritmo Hydra

A análise do comportamento da *interestingness* para as DCs descobertas pelo algoritmo Hydra é apresentada a seguir, com base nos resultados obtidos para os conjuntos de dados 'Airport' e 'Hospital_1000'. As Figuras 5.9 e 5.10 ilustram graficamente essas tendências.

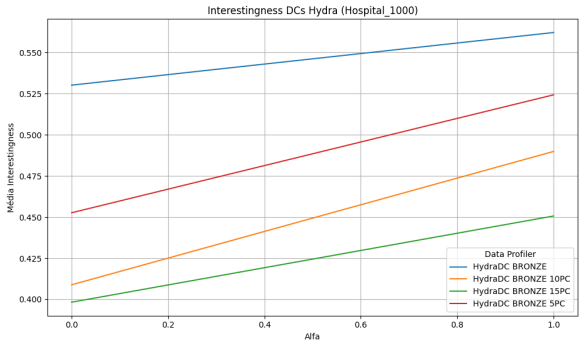


Figura 5.9: Média de *Interestingness* das DCs (Hydra) no conjunto de dados Hospital em função do ruído MCAR, para diferentes valores de α

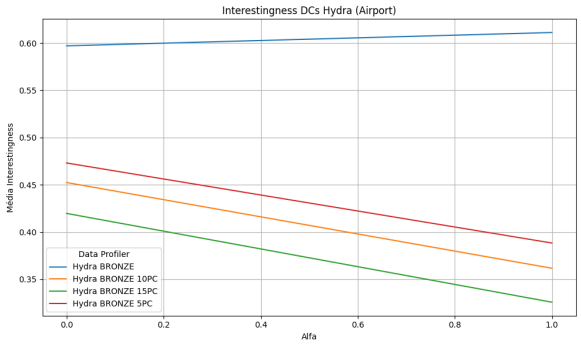


Figura 5.10: Média de *Interestingness* das DCs (Hydra) no conjunto de dados Airport em função do ruído MCAR, para diferentes valores de α

Conjunto de dados 'Airport' (Hydra): Com 0% de poluição, a *Coverage* média (`0.611069`) supera ligeiramente a *Succinctness* média (`0.596973`), resultando em *Interestingness* crescente com α (de `0.596973` a `0.611069`). Contudo, a partir de 5% de poluição, a *Coverage* (Cov) torna-se consistentemente menor que a *Succinctness* ($Succ$):

- 5% Poluição: $Cov \approx$ `0.388`, $Succ \approx$ `0.473`. I varia de `0.473065` ($\alpha = 0$) a `0.388427` ($\alpha = 1$).
- 10% Poluição: $Cov \approx$ `0.362`, $Succ \approx$ `0.452`. I varia de `0.452395` ($\alpha = 0$) a `0.361746` ($\alpha = 1$).
- 15% Poluição: $Cov \approx$ `0.326`, $Succ \approx$ `0.419728`. I varia de `0.419728` ($\alpha = 0$) a `0.325848` ($\alpha = 1$).

A inversão da relação entre Cov e $Succ$ sob poluição significa que a *interestingness* passa a diminuir com o aumento de α. Isso sugere que, para o Hydra neste conjunto de dados, a poluição degrada a cobertura dos predicados de forma mais acentuada, tornando a concisão o fator que mais contribui para a métrica de *interestingness* das DCs em cenários ruidosos. A utilidade de regras que priorizam cobertura (alto α) é, portanto, severamente reduzida, e o utilizador pode

preferir regras mais concisas (baixo α) mesmo que menos abrangentes, pois estas demonstram maior valor de *interestingness* relativa.

Conjunto de dados 'Hospital_1000' (Hydra) Assumindo os valores médios de *Coverage* (Cov) e *Succinctness* (Succ) previamente analisados:

- 0% Poluição: $Covov \approx 0.562$, $Succ \approx 0.530$. Como $Cov > Succ$, *interestingness* aumenta com α (de ≈ 0.530 a ≈ 0.562).
- 5% Poluição: $Cov \approx 0.524$, $Succ \approx 0.453$. Como $Cov > Succ$, *interestingness* aumenta com α (de ≈ 0.453 a ≈ 0.524).
- 10% Poluição: $Cov \approx 0.490$, $Succ \approx 0.409$. Como $Cov > Succ$, *interestingness* aumenta com α (de ≈ 0.409 a ≈ 0.490).
- 15% Poluição: $Cov \approx 0.451$, $Succ \approx 0.398$. Como $Cov > Succ$, *interestingness* aumenta com α (de ≈ 0.398 a ≈ 0.451).

No conjunto de dados Hospital, o Hydra mantém $Cov > Succ$ mesmo sob poluição. Assim, a *Interestingness* consistentemente aumenta com α, favorecendo regras com maior cobertura. No entanto, os valores absolutos de *interestingness* decaem em todos os níveis de α com o aumento da poluição (e.g., para $\alpha = 0.5$, de ≈ 0.546 em 0% para ≈ 0.425 em 15%), indicando uma deterioração geral da qualidade. Este comportamento distinto entre os conjuntos de dados Airport e Hospital para o Hydra sugere que a estrutura intrínseca dos dados e das DCs válidas em cada um influencia qual aspecto da qualidade, sendo cobertura (através da métrica de *Coverage* ou a concisão (através da métrica de *succinctness*), é mais resiliente ao ruído quando se trata de DCs exatas.

5.3.1.2 Algoritmo DCFinder

A análise do comportamento da *Interestingness* para as DCs descobertas pelo algoritmo DCFinder é detalhada abaixo. As Figuras 5.11 e 5.12 fornecem a representação visual.

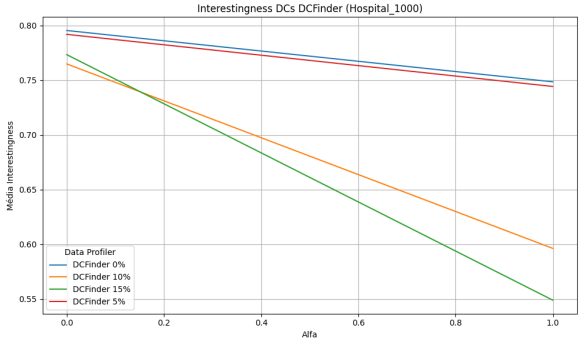


Figura 5.11: Média de *Interestingness* das DCs (DCFinder) no conjunto de dados Hospital em função do ruído MCAR, para diferentes valores de α.

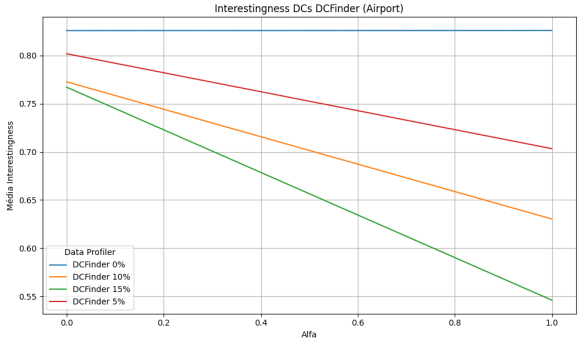


Figura 5.12: Média de *Interestingness* das DCs (DCFinder) no conjunto de dados Airport em função do ruído MCAR, para diferentes valores de α.

Conjunto de dados 'Airport' (DCFinder): Com 0% de poluição, $Cov \approx 0.826004$ e $Succ \approx 0.825905$ são quase idênticas, resultando em *interestingness* estável. Com a poluição, Cov decai mais que $Succ$:

- 5% Poluição: $Cov \approx 0.703$, $Succ \approx 0.802$. I varia de 0.801858 ($\alpha = 0$) a 0.703275 ($\alpha = 1$).
- 10% Poluição: $Cov \approx 0.630$, $Succ \approx 0.773$. I varia de 0.772698 ($\alpha = 0$) a 0.630171 ($\alpha = 1$).
- 15% Poluição: $Cov \approx 0.546$, $Succ \approx 0.767$. I varia de 0.767075 ($\alpha = 0$) a 0.545862 ($\alpha = 1$).

Similarmente ao Hydra no Airport, a partir da introdução de poluição, $Cov < Succ$, e a *Interestingness* diminui com α. A maior queda percentual da *Coverage* em relação à *Succinctness* torna as regras mais concisas as mais relevantes sob ruído, especialmente em níveis de poluição mais elevados. A *interestingness* geral para $\alpha = 0.5$ cai de ≈ 0.826 (0%) para ≈ 0.65 (15%), uma redução substancial, o que questiona a utilidade prática de DCs descobertas sob alto ruído, mesmo se a sua concisão for priorizada.

Conjunto de dados 'Hospital_1000' (DCFinder): Neste conjunto de dados, o DCFinder consistentemente produz $Succ > Cov$:

- 0% Poluição: $Cov \approx 0.749$, $Succ \approx 0.795$. I varia de 0.795414 ($\alpha = 0$) a 0.748503 ($\alpha = 1$).
- 5% Poluição: $Cov \approx 0.744$, $Succ \approx 0.792$. I varia de 0.791887 ($\alpha = 0$) a 0.744250 ($\alpha = 1$).
- 10% Poluição: $Cov \approx 0.596$, $Succ \approx 0.765$. I varia de 0.764848 ($\alpha = 0$) a 0.596226 ($\alpha = 1$).
- 15% Poluição: $Cov \approx 0.549$, $Succ \approx 0.773$. I varia de 0.773333 ($\alpha = 0$) a 0.548877 ($\alpha = 1$).

Em todos os casos, a *interestingness* diminui à medida que α aumenta, indicando que a *succinctness* é o componente de maior valor relativo para este algoritmo neste conjunto de dado. A poluição, ao reduzir ainda mais a *coverage*, apenas reforça essa dinâmica. A queda na *interestingness* geral (e.g., para $\alpha = 0.5$, de ≈ 0.772 para ≈ 0.661) é significativa, embora a *succinctness* se mantenha em patamares relativamente mais altos em comparação com a *coverage*.

5.3.1.3 Conclusões sobre a Análise de Interestingness

A análise da métrica de *interestingness* e seu comportamento frente à poluição por nulos MCAR permite extrair as seguintes conclusões principais, que contribuem para responder à terceira pergunta de pesquisa:

1. **Impacto Negativo da Poluição:** A introdução de poluição por valores nulos (MCAR) consistentemente leva a uma redução da *interestingness* média das DCs descobertas, para ambos os algoritmos e em ambos os conjunto de dados, independentemente do valor de α escolhido. Isso indica uma degradação geral na qualidade percebida das regras, afetando o seu potencial de aplicação prática.
2. **Alteração da Relevância dos Componentes (Cov vs. $Succ$) com o Ruído:** A relação entre *Coverage* e *Succinctness* é crucial e frequentemente alterada pela poluição. Observou-se que a *Coverage* tende a ser mais penalizada pelo ruído do que a *Succinctness*.

Como consequência, em cenários ruidosos, a *Succinctness* frequentemente emerge como o fator mais robusto, levando a uma maior *interestingness* quando α é baixo (priorizando concisão). A Figura 5.10 para o Hydra no conjunto de dados Airport é um exemplo claro dessa inversão, onde a inclinação da curva de *interestingness* vs. α muda de positiva para negativa com a introdução de poluição.

3. **Implicações da Escolha de α:** A sensibilidade da *interestingness* à poluição é fortemente modulada por α. Se o objetivo do utilizador é descobrir regras com alta aplicabilidade e generalidade (alto α), a qualidade percebida (*interestingness*) dessas regras será severamente degradada pelo ruído. Por outro lado, se a preferência é por regras concisas, mesmo que menos abrangentes (baixo α), a *interestingness* pode apresentar uma queda menos abrupta, mas isso ocorre à custa da cobertura da regra. A escolha de α deve, portanto, ser informada pelo contexto da aplicação e pela tolerância à perda de cobertura contra complexidade.

4. **Comportamento dos Algoritmos:** Ambos os algoritmos, Hydra e DCFinder, demonstram uma degradação da *interestingness* com a poluição. A capacidade do DCFinder de encontrar DCs aproximadas, embora possa mitigar parcialmente a queda na *coverage* em baixos níveis de ruído, não o torna imune à deterioração da *interestingness* geral, pois tanto a *Coverage* quanto a *Succinctness* das regras (sejam exatas ou aproximadas) são, em última instância, impactadas negativamente pelo ruído.

Estes resultados reforçam a ideia de que a avaliação de DCs em contextos realistas, propensos a erros, não pode depender de uma única métrica ou de uma única ponderação de seus componentes. A análise de *interestingness* com variação de α oferece uma perspectiva mais rica sobre o ganha-perde envolvidos na descoberta de conhecimento útil a partir de dados imperfeitos.

5.4 ANÁLISE DA ESTRUTURA INTERNA DAS *DENIAL CONSTRAINTS* DESCOBERTAS SOB POLUIÇÃO

Para complementar a avaliação baseada nas métricas de qualidade agregadas, esta seção investiga o impacto da poluição por valores nulos (MCAR) nas características estruturais intrínsecas das DCs descobertas. Analisa-se o perfil dos operadores utilizados e a distribuição percentual do envolvimento de atributos-chave nas DCs. Os resultados dos algoritmos Hydra e DCFinder são comparados com um conjunto de referência de *Golden DCs* para os conjuntos de dados 'Airport' e 'Hospital', permitindo avaliar se as adaptações estruturais induzidas pelo ruído aproximam ou distanciam as DCs descobertas do perfil considerado ideal.

5.4.1 Impacto da Poluição no Perfil dos Operadores

O perfil dos operadores (`EQUAL`, `UNEQUAL`) nas DCs é um indicador das estratégias lógicas empregadas pelos algoritmos em face do ruído.

Algoritmo DCFinder: Para o algoritmo DCFinder, os gráficos 5.13 e 5.14 apresentam o perfil de operadores para os conjuntos de dados 'Airport' e 'Hospital'. No conjunto de dados 'Airport' (Figura 5.13), o perfil das *Golden DCs* (62.20% `EQUAL`) contrasta com o das DCs descobertas em dados limpos (73.08% `EQUAL`). Com o aumento da poluição, a proporção de `EQUAL` intensifica-se, alcançando 89.80% com 15% de ruído, distanciando-se do perfil *Golden*. Uma tendência similar ocorre no conjunto de dados 'Hospital' (Figura 5.14): as *Golden DCs*

(62.20% EQUAL) diferem das DCs em dados limpos (72.06% EQUAL), e a poluição eleva a proporção de EQUAL para 88.17% com 15% de ruído. Este aumento da predominância de operadores EQUAL sob poluição, afastando o perfil das DCs descobertas das *Golden DCs*, sugere uma possível simplificação da lógica das regras ou um foco em relações de igualdade mais diretas para contornar as incertezas dos nulos.

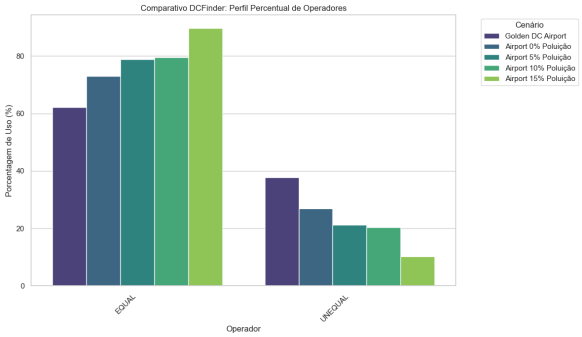


Figura 5.13: Perfil de operadores (DCFinder) no conjunto de dados Airport.

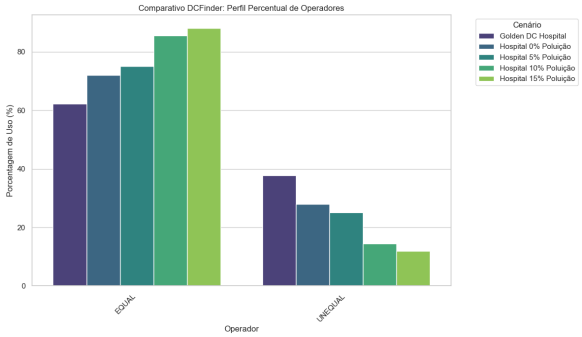


Figura 5.14: Perfil de operadores (DCFinder) no conjunto de dados Hospital.

Algoritmo Hydra: As figuras 5.15 e 5.16 mostra o perfil de operadores para o Hydra no conjunto de dados 'Airport' (Figura 5.15), o perfil das *Golden DCs* (66.67% EQUAL) é similar ao das DCs descobertas em dados limpos (61.79% EQUAL). Contudo, com poluição, o Hydra também favorece EQUAL (até 79.24% com 15% de ruído), divergindo do perfil *Golden*. Para o 'Hospital' (Figura 5.16), o perfil das *Golden DCs* (62.20% EQUAL) difere do perfil inicial do Hydra (71.07% EQUAL). Com 5% de poluição, a proporção de EQUAL cai para 57.44%, aproximando-se do perfil *Golden*. Em níveis mais altos, estabiliza em torno de 60-65%. Este comportamento do Hydra, especialmente a aproximação inicial ao perfil *Golden* no 'Hospital', sugere uma interação complexa entre sua estratégia de descoberta exata e os dados ruidosos.

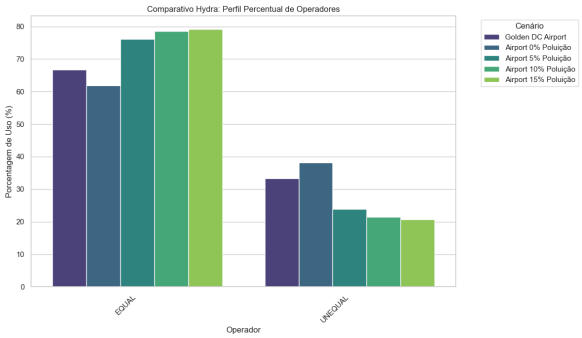


Figura 5.15: Perfil de operadores (Hydra) no conjunto de dados Airport.

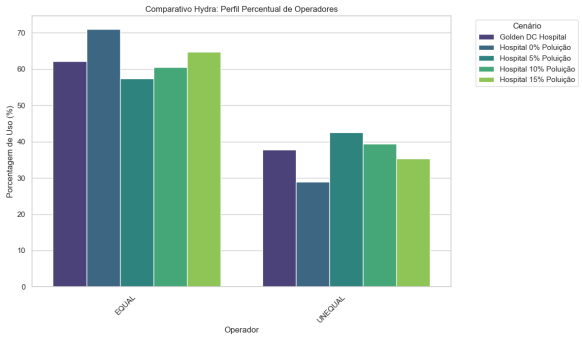


Figura 5.16: Perfil de operadores (Hydra) no conjunto de dados Hospital.

5.4.2 Impacto da Poluição na Distribuição Percentual de Envolvimento de Atributos

A análise da distribuição percentual dos atributos mais frequentemente envolvidos nas DCs indica se os algoritmos mantêm o foco nos atributos semanticamente importantes das *Golden DCs*.

Algoritmo DCFinder: No 'Airport' (Figura 5.17), as *Golden DCs* para este cenário são exclusivamente focadas nos atributos de identificação: `ident(String)`,

`iso_country(String)`, e `iso_region(String)` (cada um com 33.33% de participação). Em contraste, as DCs descobertas pelo DCFinder em dados limpos (0% de poluição) dão pouca importância a esses atributos (todos com ≈3.8%) e priorizam outros como `continent(String)`, `elevation_ft(String)` e `local_code(String)` (todos com ≈15.4%). Com o aumento da poluição, o foco do DCFinder continua a se desviar, redistribuindo a importância entre vários atributos, mas nunca se alinhando com o perfil das *Golden DCs*. Isso sugere que, para este conjunto de dados, o DCFinder não captura as restrições de referência mais importantes, e a poluição apenas modifica o perfil das regras alternativas que ele descobre.

Para o 'Hospital' (Figura 5.18), as *Golden DCs* mostram um foco claro em `Provider Number(String)` (17.07%) e `Phone Number(String)` (14.63%). Em dados limpos, o DCFinder sub-representa drasticamente estes atributos (ambos com ≈1.5%) e prioriza um conjunto diferente, como `Hospital Type` e `Measure Code/Name`. com o aumento da poluição, a participação percentual de `Provider Number` e `Phone Number` nas DCs descobertas aumenta, chegando a cerca de 7% cada com 15% de ruído. Este comportamento sugere que o ruído, ao invalidar regras mais simples, pode forçar o algoritmo a buscar padrões mais complexos que podem se alinham um pouco mais com alguns dos atributos semanticamente importantes da referência.

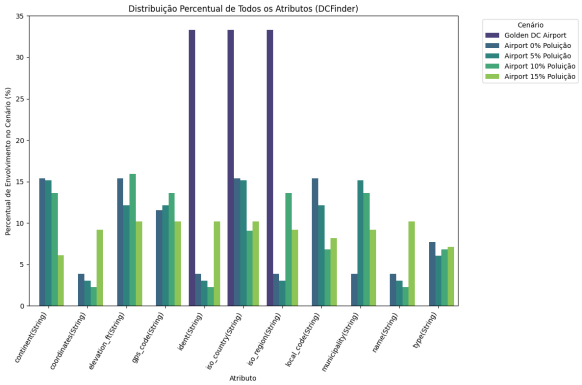


Figura 5.17: Distribuição de atributos (DCFinder) no conjunto de dados Airport.

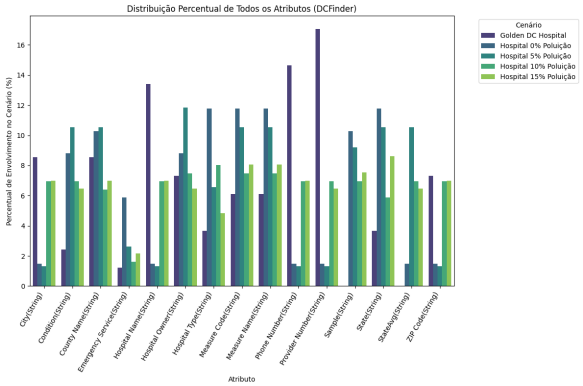


Figura 5.18: Distribuição de atributos (DCFinder) no conjunto de dados Hospital.

Algoritmo Hydra: No 'Airport' (Figura 5.19),o Hydra também demonstra um desalinhamento com o perfil *Golden* em dados limpos, priorizando o atributo `name(String)` (16.26%). Com a introdução de ruído, a distribuição percentual de envolvimento dos atributos de topo torna-se notavelmente mais **uniforme**. Com 15% de poluição, a maioria dos atributos mais frequentes apresenta uma participação em torno de 8-11%. Este espalhamento do foco, combinado com o aumento massivo nas contagens absolutas de predicados (discutido na análise de frequência), sugere que a estratégia do Hydra para manter a exatidão em dados ruidosos é gerar um volume muito grande de regras diversas e complexas. Esta proliferação de regras se afasta da estrutura mais focada e concisa das *Golden DCs*.

Para o 'Hospital' (Figura 5.20), o Hydra em dados limpos já demonstra um bom alinhamento com um dos atributos *Golden*, `Hospital Name(String)` (15.52%), mas sub-representa outros como `Provider Number` e `Phone Number`. Similarmente ao seu comportamento no conjunto de dados *Airport*, com o aumento da poluição, a distribuição percentual de envolvimento dos atributos de topo torna-se mais homogênea (a maioria com ≈7% de participação com 15% de ruído). Isso reforça a conclusão de que a estratégia de adaptação do Hydra ao ruído envolve a geração de um conjunto de regras vasto e diversificado, que dilui a

importância relativa de qualquer atributo individual, afastando-se do foco mais direcionado das *Golden DCs*.

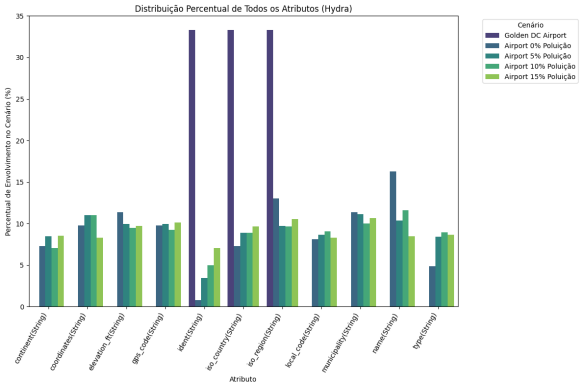


Figura 5.19: Distribuição de atributos (Hydra) no conjunto de dados Airport.

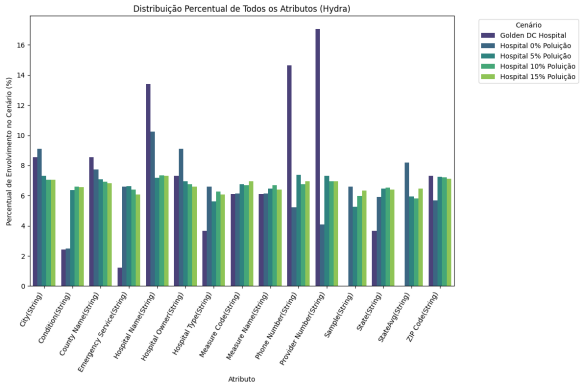


Figura 5.20: Distribuição de atributos (Hydra) no conjunto de dados Hospital.

5.4.3 Síntese da Análise Estrutural Comparativa

A inclusão das *Golden DCs* como referência na análise estrutural revela que:

1. **Perfil de Operadores:** A poluição tende a aumentar a proporção de operadores `EQUAL` nas DCs descobertas por ambos os algoritmos. Esta adaptação nem sempre alinha as DCs com o perfil das *Golden DCs*, podendo indicar uma simplificação ou um desvio da lógica ideal das regras em ambientes ruidosos.

2. **Foco nos Atributos e Alinhamento com *Golden DCs*:**

   - Em vários cenários, como para o DCFinder no dataset *Airport*, os algoritmos focaram em um conjunto de atributos quase inteiramente distinto do conjunto ideal, e a poluição apenas redistribuiu o foco entre esses atributos alternativos. Isso sugere que os algoritmos, sob as condições testadas, podem não capturar as regras semanticamente mais importantes, mesmo em dados limpos.
   - Para o Hydra, o aumento expressivo no volume de predicados (inferido pelas frequências absolutas anteriormente analisadas), mesmo que a distribuição percentual se torne mais uniforme com a poluição, aponta para uma estratégia de geração de regras mais complexas e numerosas como forma de manter a exatidão, o que pode se distanciar da estrutura mais enxuta esperada das *Golden DCs*.

3. **Implicações para Métricas de Qualidade:** As alterações estruturais observadas são fatores cruciais que explicam as variações nas métricas de *coverage*, *succinctness* e *interestingness*. Uma DC que se desvia estruturalmente do ideal (*Golden DC*) ou que se torna excessivamente complexa para contornar o ruído provavelmente apresentará menor *Coverage* de padrões relevantes e menor *Succinctness*, impactando negativamente sua *Interestingness*.

Esta análise detalhada da estrutura interna das DCs, portanto, não apenas complementa as métricas de qualidade, mas também oferece um diagnóstico mais profundo de como os algoritmos de descoberta respondem à poluição e se as DCs resultantes mantêm relevância semântica em comparação com um conjunto de regras de referência.

6 CONCLUSÃO

Este trabalho investigou a robustez da descoberta de *Denial Constraints* em cenários de dados ruidosos,visando compreender o impacto de erros na qualidade e na estrutura das regras descobertas. Mediante uma análise experimental controlada, que utilizou a inserção de dados ausentes (nulos via MCAR) em diferentes níveis de poluição, avaliamos o comportamento dos algoritmos DCFinder e Hydra e das métricas de qualidade *Coverage*, *Succinctness* e *Interestingness*. A seguir, é apresentada a recapitulação dos resultados obtidos e as considerações finais deste estudo.

A análise experimental confirmou a hipótese central de que a presença de ruído nos dados influencia de maneira distinta e mensurável a descoberta de DCs. Ao analisar as perguntas de pesquisa, observou-se que a introdução de dados ausentes resultou em uma degradação consistente das métricas de qualidade para ambos os algoritmos. A *coverage* das DCs diminuiu progressivamente, indicando que as regras perdem sua generalidade e suporte estatístico em dados imperfeitos. Similarmente, a *succinctness* também apresentou uma tendência geral de queda, revelando que os algoritmos tendem a gerar regras mais longas e complexas para se adaptarem às exceções criadas pelo ruído, um comportamento que aponta para um *overfitting*. Consequentemente, a *interestingness*, que combina ambos os fatores, também foi negativamente afetada, e a relação de importância entre cobertura e concisão frequentemente se inverteu em cenários poluídos.

Adicionalmente, a investigação da estrutura interna das DCs revelou que essas mudanças nas métricas de qualidade estão associadas a alterações na composição das próprias regras. A poluição induziu uma preferência por operadores de igualdade (`EQUAL`) e alterou o foco dos atributos mais envolvidos, muitas vezes desalinhando o perfil das DCs descobertas de um conjunto de referência ideal (*Golden DCs*). Por fim, foi possível constatar que os algoritmos DCFinder e Hydra exibem diferentes níveis de robustez. Enquanto o DCFinder, com sua capacidade de lidar com DCs aproximadas, demonstrou uma degradação de qualidade mais suave em certos cenários, o Hydra, focado em exatidão, respondeu ao ruído com um aumento mais drástico na complexidade e no número de regras para manter a validade.

6.1 TRABALHOS FUTUROS

Para **trabalhos futuros**, sugerem-se os trabalhos:

- Expansão desta metodologia para avaliar o impacto de outras classes de erro, como *Typos* (erros de digitação) e *Bogus Values* (valores espúrios), conforme proposto por (Arocena et al., 2015).
- Expansão dos estudos deste trabalho com comparativo entre mais algoritmos de descoberta de DCs, como o FASTDC (Pena et al., 2022) com mais conjuntos de dados com perfis de operadores e tipos de dados distintos.
- Como para o cálculo de interesstingness, na etapa do cálculo de coverage é necessário um produto cartesiano no conjunto de dados trabalhados, e pela técnica de amostragem empregada nesse estudo, seria interessante conduzir um estudo sobre as diferenças estatísticas causadas devido à amostragem.

REFERÊNCIAS

Abedjan, Z., Golab, L. e Naumann, F. (2015). Profiling relational data: A survey. *The VLDB Journal*, 24(4):557–581.

Arocena, P. C., Glavic, B., Mecca, G., Miller, R. J., Papotti, P. e Santoro, D. (2015). Messing up with BART: Error generation for evaluating data-cleaning algorithms. *Proc. VLDB Endow.*, 9(2):36–47.

Bleifuß, T., Kruse, S. e Naumann, F. (2017). *Efficient Denial Constraint Discovery with Hydra*. PVLDB.

Chu, X., Ilyas, I. F. e Papotti, P. (2013). *Holistic Data Cleaning: Putting Violations Into Context*. In Proceedings of the International Conference on Data Engineering (ICDE).

Inmon, W. H. (2005). *Building the Data Warehouse, Third Edition*, página 285. Robert Ipsen.

Little, R. J. e Rubin, D. B. (2019). *Statistical Analysis with Missing Data*. John Wiley & Sons.

Martin, A., de Almeida, E. C., Romero, O. e Queralt, A. (2025). How and why false denial constraints are discovered. *Proceedings of the VLDB Endowment*, 18(10).

Papenbrock, T., Bergmann, T., Finke, M., Zwiener, J. e Naumann, F. (2015). Data profiling with metanome. *Proc. VLDB Endow.*, 8(12):1860–1863.

Pena, E. H., de Almeida, E. C. e Naumann, F. (2021). *Fast detection of denial constraint violations*. Proceedings of the VLDB Endowment.

Pena, E. H., Porto, F. e Naumann, F. (2022). *Fast algorithms for denial constraint discovery*. Proceedings of the VLDB Endowment.

Pena, E. H. M., de Almeida, E. C. e Naumann, F. (2019). *Discovery of Approximate (and Exact) Denial Constraints*. PVLDB.

Rahm, E. (2000). *Data Cleaning: Problems and Current Approaches*. DBLP.

X. Chu, I. F. Ilyas, P. P. (2013). *Discovering Denial Constraints*. Proceedings of the VLDB Endowment.

APÊNDICE A – CÁLCULO INTERESTINGNESS

A.1 CÁLCULO COVERAGE

Para o cálculo de coverage, inicialmente foi criado o dataset conforme a tabela 2.1 sendo convertido para o padrão SQL Ansi conforme código A.1 , utilizando o DBMS PostgresSQL, foi convertido em Common Table Expression (CTE) A.2 utilizando SQL padrão ANSI, os predicados das Denial Constraints, através dos quais foram agrupados utilizando o código SQL incluído em A.3 , sendo a saída utilizada para os cálculos em

```sql
CREATE TABLE Funcionarios (
    ID SERIAL PRIMARY KEY,
    Nome VARCHAR(100) NOT NULL,
    IDDepartamento INT NOT NULL,
    Salario NUMERIC(10, 2) NOT NULL,
    IDGerente INT NOT NULL,
    IDCargo INT NOT NULL,
    DataInicio DATE NOT NULL
);

-- Inserir os dados fornecidos
INSERT INTO Funcionarios (ID, Nome, IDDepartamento, Salario, IDGerente,
    IDCargo, DataInicio)
VALUES
    (1, 'Alexa', 101, 5000, 3, 1, '2018-01-15'),
    (2, 'Bruno', 101, 4000, 3, 1, '2017-01-15'),
    (3, 'Carol', 101, 7000, 3, 10, '2015-02-01'),
    (4, 'Deyverson', 102, 8000, 4, 4, '2022-03-10'),
    (5, 'Eduardo', 102, 5000, 4, 5, '2017-01-01'),
    (6, 'Felix', 102, 5000, 4, 5, '2017-01-01'),
    (7, 'Guilherme', 102,12000, 4, 5, '2017-01-01');
```

Listing A.1: Inicialização Dataset

```sql


-- ============================================================
-- DC 1:
-- Predicados:
-- P1: t_x.IDGerente = t_y.ID
-- P2: t_x.Salário > t_y.Salário
-- ============================================================
WITH CTE_DC1 AS (
    SELECT
        CASE
            WHEN f1.IDGerente = f2.ID
            AND f1.Salario > f2.Salario THEN '2'
            WHEN f1.IDGerente = f2.ID
            OR f1.Salario > f2.Salario THEN '1'
            ELSE '0'
        END AS Categoria,
        2 AS N_PREDICADOS
    FROM
        Funcionarios f1
        LEFT JOIN Funcionarios f2 ON 1 = 1
```

```
22        AND F1.ID <> F2.ID
23  ) -- ============================================================
24  -- DC 2:
25  -- Predicados:
26  -- P1: t_x.IDCargo = t_y.IDCargo
27  -- P2: t_x.DataInicio > t_y.DataInicio
28  -- P3: t_x.Salario > t_y.Salario
29  -- ============================================================
30  ,
31  CTE_DC2 AS (
32     SELECT
33        CASE
34           WHEN f1.IDCargo = f2.IDCargo
35           AND f1.DataInicio > f2.DataInicio
36           AND f1.Salario > f2.Salario THEN '3'
37           WHEN (
38              f1.IDCargo = f2.IDCargo
39              AND f1.DataInicio > f2.DataInicio
40           )
41           OR (
42              f1.IDCargo = f2.IDCargo
43              AND f1.Salario > f2.Salario
44           )
45           OR (
46              f1.DataInicio > f2.DataInicio
47              AND f1.Salario > f2.Salario
48           ) THEN '2'
49           WHEN f1.IDCargo = f2.IDCargo
50           OR f1.DataInicio > f2.DataInicio
51           OR f1.Salario > f2.Salario THEN '1'
52           ELSE '0'
53        END AS Categoria,
54        3 AS N_PREDICADOS
55     FROM
56        Funcionarios f1
57        LEFT JOIN Funcionarios f2 ON 1 = 1
58        AND F1.ID <> F2.ID
59  ) -- ============================================================
60  -- DC 3:
61  -- Predicados:
62  -- P1: IDs sejam únicos
63  -- ============================================================
64  ,
65  CTE_DC3 AS (
66     SELECT
67        CASE
68           WHEN f1.ID <> f2.ID THEN '1'
69           ELSE '0'
70        END AS Categoria,
71        1 AS N_PREDICADOS
72     FROM
73        Funcionarios f1
74        LEFT JOIN Funcionarios f2 ON 1 = 1
75        AND F1.ID <> F2.ID
76  ) -- ============================================================
77  -- DC 4:
78  -- Predicados:
79  -- P1: t_x.IDCargo = t_y.IDCargo
```

```
-- P2: t_x.IDDepartamento <> t_y.IDDepartamento
-- ============================================================
,
CTE_DC4 AS (
   SELECT
      CASE
         WHEN f1.IDCargo = f2.IDCargo
         AND f1.IDDepartamento <> f2.IDDepartamento THEN '2'
         WHEN f1.IDCargo = f2.IDCargo
         OR f1.IDDepartamento <> f2.IDDepartamento THEN '1'
         ELSE '0'
      END AS Categoria,
      2 AS N_PREDICADOS
   FROM
      Funcionarios f1
      LEFT JOIN Funcionarios f2 ON 1 = 1
      AND F1.ID <> F2.ID
) -- ============================================================
-- DC 5:
-- Predicados:
-- P1: t_x.Salário > 10 * t_y.Salário
-- ============================================================
,
CTE_DC5 AS (
   SELECT
      CASE
         WHEN f1.Salario > 10 * f2.Salario THEN '1'
         ELSE '0'
      END AS Categoria,
      1 AS N_PREDICADOS
   FROM
      Funcionarios f1
      LEFT JOIN Funcionarios f2 ON 1 = 1
      AND F1.ID <> F2.ID
);
```

Listing A.2: Common Table Expressions Cálculo Coverage

```
SELECT 'DC 1' AS N_DC,N_PREDICADOS,Categoria AS K_EVIDENCIAS, COUNT(*) AS
    Registros
FROM CTE_DC1
GROUP BY K_EVIDENCIAS, N_DC, N_PREDICADOS

UNION ALL

SELECT 'DC 2' AS N_DC,N_PREDICADOS,Categoria AS K_EVIDENCIAS, COUNT(*) AS
    Registros
FROM CTE_DC2
GROUP BY K_EVIDENCIAS, N_DC, N_PREDICADOS

UNION ALL

SELECT 'DC 3' AS N_DC,N_PREDICADOS,Categoria AS K_EVIDENCIAS, COUNT(*) AS
    Registros
FROM CTE_DC3
GROUP BY K_EVIDENCIAS, N_DC, N_PREDICADOS

UNION ALL
```

```
18
19 SELECT 'DC 4' AS N_DC,N_PREDICADOS,Categoria AS K_EVIDENCIAS, COUNT(*) AS
       Registros
20 FROM CTE_DC4
21 GROUP BY K_EVIDENCIAS, N_DC, N_PREDICADOS
22
23 UNION ALL
24
25 SELECT 'DC 5' AS N_DC,N_PREDICADOS,Categoria AS K_EVIDENCIAS, COUNT(*) AS
       Registros
26 FROM CTE_DC5
27 GROUP BY K_EVIDENCIAS, N_DC, N_PREDICADOS
28
29 ORDER BY 1, 3;
```

Listing A.3: Agrupamento Cálculo Coverage

```
1  n_dc | n_predicados | k_evidencias | registros
2  ------+--------------+--------------+-----------
3   DC 1 |            2 | 0            |        14
4   DC 1 |            2 | 1            |        16
5   DC 2 |            3 | 0            |         7
6   DC 2 |            3 | 1            |        17
7   DC 2 |            3 | 2            |         5
8   DC 2 |            3 | 3            |         1
9   DC 3 |            1 | 1            |        30
10  DC 4 |            2 | 0            |         8
11  DC 4 |            2 | 1            |        22
12  DC 5 |            1 | 0            |        30
```

Listing A.4: Resultados Cálculo Coverage

APÊNDICE B – RESULTADOS ALGORITMOS DESCOBERTA DCS

B.1 DC FINDER

- $\forall t_0 \in \text{funcionarios.csv}, t_1 \in \text{funcionarios.csv} : \neg[t_0.\text{Salário} = t_1.\text{Salário} \wedge t_0.\text{IDGerente} \geq t_1.\text{IDGerente} \wedge t_0.\text{IDCargo} < t_1.\text{IDCargo}]$
- $\forall t_0 \in \text{funcionarios.csv}, t_1 \in \text{funcionarios.csv} : \neg[t_0.\text{IDGerente} < t_1.\text{IDGerente} \wedge t_0.\text{IDCargo} \geq t_1.\text{IDCargo} \wedge t_0.\text{Salário} = t_1.\text{Salário}]$
- $\forall t_0 \in \text{funcionarios.csv}, t_1 \in \text{funcionarios.csv} : \neg[t_0.\text{IDDepartamento} \geq t_1.\text{IDDepartamento} \wedge t_0.\text{ID} \leq t_1.\text{ID} \wedge t_0.\text{IDGerente} \neq t_1.\text{IDGerente}]$
- $\forall t_0 \in \text{funcionarios.csv}, t_1 \in \text{funcionarios.csv} : \neg[t_0.\text{IDDepartamento} \neq t_1.\text{IDDepartamento} \wedge t_0.\text{ID} \leq t_1.\text{ID} \wedge t_0.\text{IDGerente} \geq t_1.\text{IDGerente}]$
- $\forall t_0 \in \text{funcionarios.csv}, t_1 \in \text{funcionarios.csv} : \neg[t_0.\text{ID} \leq t_1.\text{ID} \wedge t_0.\text{Salário} = t_1.\text{Salário} \wedge t_0.\text{IDGerente} \geq t_1.\text{IDGerente}]$
- $\forall t_0 \in \text{funcionarios.csv}, t_1 \in \text{funcionarios.csv} : \neg[t_0.\text{ID} \leq t_1.\text{ID} \wedge t_0.\text{IDCargo} \geq t_1.\text{IDCargo} \wedge t_0.\text{Salário} = t_1.\text{Salário}]$
- $\forall t_0 \in \text{funcionarios.csv}, t_1 \in \text{funcionarios.csv} : \neg[t_0.\text{Nome} = t_1.\text{Nome}]$
- $\forall t_0 \in \text{funcionarios.csv}, t_1 \in \text{funcionarios.csv} : \neg[t_0.\text{Salário} = t_1.\text{Salário} \wedge t_0.\text{IDGerente} = t_1.\text{IDGerente} \wedge t_0.\text{DataInício} \neq t_1.\text{DataInício}]$
- $\forall t_0 \in \text{funcionarios.csv}, t_1 \in \text{funcionarios.csv} : \neg[t_0.\text{IDDepartamento} < t_1.\text{IDDepartamento} \wedge t_0.\text{IDGerente} \geq t_1.\text{IDGerente}]$
- $\forall t_0 \in \text{funcionarios.csv}, t_1 \in \text{funcionarios.csv} : \neg[t_0.\text{IDDepartamento} \geq t_1.\text{IDDepartamento} \wedge t_0.\text{Salário} = t_1.\text{Salário} \wedge t_0.\text{IDGerente} \neq t_1.\text{IDGerente} \wedge t_0.\text{IDCargo} \leq t_1.\text{IDCargo}]$

B.2 HYDRA

- $\forall t_0 \in \text{funcionarios.csv}, t_1 \in \text{funcionarios.csv} : \neg[t_0.\text{Salário} = t_1.\text{Salário} \wedge t_0.\text{IDCargo} \neq t_1.\text{IDCargo} \wedge t_0.\text{IDDepartamento} = t_1.\text{IDDepartamento}]$
- $\forall t_0 \in \text{funcionarios.csv}, t_1 \in \text{funcionarios.csv} : \neg[t_0.\text{IDCargo} > t_1.\text{IDCargo} \wedge t_0.\text{Salário} = t_1.\text{Salário} \wedge t_0.\text{IDDepartamento} \leq t_1.\text{IDDepartamento}]$
- $\forall t_0 \in \text{funcionarios.csv}, t_1 \in \text{funcionarios.csv} : \neg[t_0.\text{IDDepartamento} \neq t_1.\text{IDDepartamento} \wedge t_0.\text{Salário} = t_1.\text{Salário} \wedge t_0.\text{ID} \geq t_1.\text{ID} \wedge t_0.\text{IDCargo} \leq t_1.\text{IDCargo}]$
- $\forall t_0 \in \text{funcionarios.csv}, t_1 \in \text{funcionarios.csv} : \neg[t_0.\text{Salário} = t_1.\text{Salário} \wedge t_0.\text{IDCargo} \neq t_1.\text{IDCargo} \wedge t_0.\text{ID} \geq t_1.\text{ID} \wedge t_0.\text{IDDepartamento} \leq t_1.\text{IDDepartamento}]$
- $\forall t_0 \in \text{funcionarios.csv}, t_1 \in \text{funcionarios.csv} : \neg[t_0.\text{IDGerente} \neq t_1.\text{IDGerente} \wedge t_0.\text{DataInício} = t_1.\text{DataInício}]$

- $\forall t_0 \in \text{funcionarios.csv}, t_1 \in \text{funcionarios.csv} : \neg[t_0.\text{IDCargo} > t_1.\text{IDCargo} \wedge t_0.\text{IDGerente} \leq t_1.\text{IDGerente} \wedge t_0.\text{Salário} = t_1.\text{Salário}]$
- $\forall t_0 \in \text{funcionarios.csv}, t_1 \in \text{funcionarios.csv} : \neg[t_0.\text{Salário} = t_1.\text{Salário} \wedge t_0.\text{IDGerente} > t_1.\text{IDGerente} \wedge t_0.\text{IDCargo} \leq t_1.\text{IDCargo}]$
- $\forall t_0 \in \text{funcionarios.csv}, t_1 \in \text{funcionarios.csv} : \neg[t_0.\text{Salário} = t_1.\text{Salário} \wedge t_0.\text{IDGerente} \neq t_1.\text{IDGerente} \wedge t_0.\text{ID} \geq t_1.\text{ID} \wedge t_0.\text{IDCargo} \leq t_1.\text{IDCargo}]$
- $\forall t_0 \in \text{funcionarios.csv}, t_1 \in \text{funcionarios.csv} : \neg[t_0.\text{IDGerente} \leq t_1.\text{IDGerente} \wedge t_0.\text{Salário} = t_1.\text{Salário} \wedge t_0.\text{IDCargo} \neq t_1.\text{IDCargo} \wedge t_0.\text{ID} \geq t_1.\text{ID}]$
- $\forall t_0 \in \text{funcionarios.csv}, t_1 \in \text{funcionarios.csv} : \neg[t_0.\text{IDGerente} \leq t_1.\text{IDGerente} \wedge t_0.\text{IDDepartamento} > t_1.\text{IDDepartamento}]$