VINÍCIUS PACHECO

# IMPACTO DE DADOS AUSENTES NA DESCOBERTA DE DEPENDÊNCIAS FUNCIONAIS

(*versão pré-defesa, compilada em 14 de dezembro de 2023*)

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: *Ciência da Computação*.

Orientador: Eduardo Cunha de Almeida, Marcos Sfair Sunye.

CURITIBA PR

2023

## RESUMO

Este trabalho pretende analisar o impacto de dados ausentes, também conhecidos como dados nulos, na descoberta de dependências funcionais (DFs). Para isso, o trabalho introduz diversos tópicos na área de limpeza de dados, como dependências funcionais, restrições de negações, algoritmos de descoberta de DFs e RNs, e características dos mecanismos de inserção de dados ausentes em um banco de dados relacional. O trabalho utiliza esses conceitos e ferramentas para analisar um banco de dados sobre as escolas do Brasil mantido pelo INEP. Análises são realizadas nos atributos sobre disponibilidade de água e sanitários da escola, demonstrando o impacto desses dados ausentes na criação de dependências funcionais fantasmas e na redução da métrica de cobertura, devido a atributos redundantes e a mudanças na lógica de inserção de dados ao longo dos anos. Também é feita uma análise de impacto após utilizar a estratégia de inserção de valores para reduzir o número de dependências funcionais fantasmas. O trabalho conclui resumindo as analises realizadas e sugerindo possíveis intervenções para a melhor manutenção do banco de dados para a realização de descoberta de dependências funcionais ao INEP, e também oferece com sugestões para pesquisadores que desejam executar algoritmos de descoberta neste banco de dados.



Palavras-chave: Dependência Funcional. Limpeza de Dados. Danos ausentes. Algoritmo de descoberta. INEP.

## LISTA DE FIGURAS


4.1 Gráfico retirado do artigo (Huhtala et al., 1999). As combinação em negrito são consideradas, enquanto as sem negrito foram descartadas quando B foi descartado e não são vistas pelo algoritmo . . . . . 18
4.2 Tabela DFs por valores nulos adicionados a tabela. . . . . 19
4.3 Visualização do Metanome dos DFs descobertos pelo numa tabela original, sem dados ausentes . . . . . 20
4.4 Visualização do Metanome dos DFs descobertos pelo numa tabela com 3 inserções de dados ausentes . . . . . 21
4.5 Visualização do Metanome dos DFs descobertos pelo numa tabela com 9 inserções de dados ausentes . . . . . 21
4.6 Visualização do Metanome dos DFs descobertos pelo numa tabela com 15 inserções de dados ausentes. . . . . 22
4.7 Visualização do Metanome dos DFs descobertos pelo numa tabela com 20 inserções de dados ausentes. . . . . 22
4.8 Visualização do Metanome dos DFs descobertos pelo numa tabela com 25 inserções de dados ausentes. . . . . 23

5.1 Proporção entre dados existentes e nulos ao longo dos anos para os atributos relacionados a água.. . . . . 24
5.2 Proporção entre dados existentes e nulos ao longo dos anos para os atributos relacionados à sanitários. . . . . 25

6.1 Número de dependência funcionais para cada caso apresentado nos atributos de água.. . . . . 28
6.2 Coberturas das dependências funcionais relacionadas à água descobertas, dividido em intervalos. . . . . 29
6.3 Visualização do Metanome dos DFs descobertos pela variação **mesclado** . . . . 29
6.4 Visualização do Metanome dos DFs descobertos pela variação **sem_filtrado**. . . 30
6.5 Visualização do Metanome dos DFs descobertos pela variação **sem_potavel** . . . 30

7.1 Número de dependência funcionais para cado caso apresentado nos atributos sanitários. . . . . 32
7.2 Coberturas das dependências funcionais relacionadas aos sanitários descobertas, dividido em intervalos. . . . . 33
7.3 Visualização do Metanome dos DFs descobertos pela variação **todos**. . . . . 33
7.4 Visualização do Metanome dos DFs descobertos pela variação **pre-2015**. . . . . 34

7.5 Visualização do Metanome dos DFs descobertos pela variação **pre-2015** . . . . . 34

8.1 Número de dependência funcionais para cada caso apresentado nos atributos sanitários, onde o atributo *situacao_de_funcionamento* é 1. . . . . 36
8.2 Visualização do Metanome dos DFs descobertos pela variação **todos-situação** . . 37
8.3 Visualização do Metanome dos DFs descobertos pela variação **pre 2015-situação** 37
8.4 Visualização do Metanome dos DFs descobertos pela variação **pos 2015-situação** 38

## LISTA DE TABELAS


2.1 Registros de hóspedes em um hotel. . . . . 10
2.2 Registro de hóspedes com dado sujo. . . . . 11

4.1 Tabela com exemplos de possíveis dependências funcionais encontradas após o uso de um algoritmo de descoberta em tabelas sem e com dados ausentes. . . . . 17
4.2 Tabela WDC_Planets . . . . . 19

## LISTA DE ACRÔNIMOS

| | |
|---|---|
| DF | Dependência Funcional |
| DFA | Dependência Funcional Aproximada |
| RN | Restrução de Negação |
| MCAR | Missing Completely At Random |
| MAR | Missing At Random |
| MNAR | Missing Not At Random |
| INEP | Instituto Nacional de Estudos e Pesquisas Educacionais |
| C3SL | Centro de Computação Científica e Software Livre |
| SIMCAQ | Simulador de Custo-Aluno Qualidade |

**SUMÁRIO**


**1 INTRODUÇÃO . . . . . 8**
**2 FUNDAMENTOS . . . . . 10**
2.1 DEPENDÊNCIAS FUNCIONAIS . . . . . 10
2.1.1 Restrições de negação . . . . . 10
2.1.2 Exemplo do uso das restrições de negação para impedir inserção de dados sujos ao banco de dados. . . . . 11
2.2 MÉTRICAS DE DESCOBERTA DE RN . . . . . 12
2.2.1 Concisão . . . . . 12
2.2.2 Cobertura . . . . . 13
**3 DADOS AUSENTES. . . . . 14**
3.1 MECANISMOS DE PROBABILIDADE DE DADOS AUSENTES. . . . . 14
3.1.1 MISSING COMPLETELY AT RANDOM - MCAR. . . . . 14
3.1.2 MISSING AT RANDOM - MAR. . . . . 15
3.1.3 MISSING NOT AT RANDOM - MNAR. . . . . 15
**4 DESCOBERTA DE DEPENDÊNCIAS FUNCIONAIS COM DADOS AUSENTES . . . . . 16**
4.1 METANOME . . . . . 17
4.2 ALGORITMO TANE EM BANCO DE DADOS PEQUENO COM MECANISMO MCAR. . . . . 17
4.3 TANE . . . . . 18
**5 ANÁLISE EM DADOS DO INEP . . . . . 24**
5.1 AMOSTRAGEM DA TABELA . . . . . 25
**6 ANÁLISE EM ATRIBUTOS RELACIONADOS À ÁGUA . . . . . 27**
**7 ANÁLISE EM ATRIBUTOS RELACIONADOS À SANITÁRIOS . . . . . 31**
**8 RELAÇÃO ENTRE SITUAÇÃO DA ESCOLA, ANO DO CENSO E DADOS AUSENTES DOS ATRIBUTOS SANITÁRIOS . . . . . 35**
**9 CONCLUSÃO . . . . . 39**
9.1 HIPÓTESES . . . . . 39
9.2 SUGESTÕES AO GERIR BANCO DE DADOS. . . . . 39
9.3 SUGESTÕES A PESQUISADORES. . . . . 40
**REFERÊNCIAS . . . . . 41**

# 1 INTRODUÇÃO

Durante a utilização de um banco de dados relacional (BDR) é possível que por erro humano ou de equipamento, um dado incorreto seja inserido no BDR. Esse dado sujo pode desabilitar certas funcionalidades do sistema atrelado ao BDR e pode dificultar a análise de dados em cima do BDR.

Para corrigir esse problema é realizado um processo chamado limpeza de dados, onde os dados sujos são removidos ou alterados para se adequar aos requisitos e para o bom funcionamento do BDR e seu sistema.

Para esta limpeza de dados ser realizada é necessário utilizar ferramentas que definem as condições ideais e não-ideais dentro do BDR. Por conseguirem ser representadas computacionalmente, dependências funcionais e restrição de negação são algumas das ferramentas utilizadas para realizar a limpeza de dados.

Essas ferramentas podem representar regras de negócios para o banco de dados desde a sua criação. No entanto, um BDR maturo pode sofrer inúmeras alterações durante seu funcionamento, e as dependências funcionais e restrições de negação podem acabar ficando desatualizadas.

Por essa razão, diversos algoritmos de descobertas de dependências funcionais (DF) e restrições de negações (RN) foram criados. Utilizando esses algoritmos é possível descobrir quais DFs e RCs são adequadas para o uso atual do banco de dados.

No entanto, esses algoritmos não conseguem realizar suas descobertas sem um tratamento específico para dados ausentes. Dados ausentes são um tipo de dado sujo onde por algum motivo um valor de dados não é preenchido. Segundo (Berti-Équille et al., 2018), em bancos de dados utilizados em produção esse número costuma oscilar entre 20% a 80% dos dados.

Portanto, para o funcionamento desses algoritmos em casos práticos é necessário que tratem os dados ausentes de maneira em que o seu resultado não fique muito prejudicado com informações que não condizem com a realidade.

Cada algoritmo de busca precisa escolher o melhor método de tratamento para um determinado banco de dados, algoritmo e mecanismo de inserção de dados ausentes.

O Tane ((Huhtala et al., 1999)) é um algoritmo de descoberta de dependências funcionais que utiliza uma estrutura de dados em formato de látice para gastar menos recursos computacionais ao executar em grandes bancos de dados. O algoritmo utiliza dependências funcionais aproximadas (DFAs), uma variante específica de dependências funcionais que comporta a descoberta das mesmas em bancos de dados com dados ausentes, desde que abaixo do coeficiente de aproximação dado pelo próprio algoritmo.

O banco de dados utilizado para realizar as análises nesse trabalho é um banco de dados disponibilizado pelo INEP contendo informações de milhares de escolas. As análises utilizam

a ferramenta Metanome para a execução do algoritmo de descoberta Tane. As análises são realizadas nos atributos sobre disponibilidade de água e sanitários das escolas, demonstrando o impacto desses dados ausentes na criação de dependências funcionais fantasmas e na redução na métrica de cobertura, devido a atributos redundantes e a mudanças na lógica de inserção de dados ao longo dos anos.

Outra análise é feita sobre o impacto da estratégia de inserção de valores onde existem dados ausentes criado por um mecanismo MAR("Missing at random"), com o intuito de reduzir o número de dependências fantasmas (DFs não encontradas na amostra original sem dados ausentes) descobertos pelo algoritmo Tane. Está análise demonstra que este tratamento reduz significativamente o numero de dependências funcionais fantasmas encontrada pelo Tane.

A conclusão deste trabalho oferece sugestões para os administradores do banco de dados do INEP com possíveis mudanças para facilitar a descoberta de dependências funcionais no banco de dados e sugestões para pesquisadores de como tratar os dados para reduzir o número de dependências funcionais fantasmas encontradas por algoritmos de descoberta de DF's como o Tane.

# 2 FUNDAMENTOS

## 2.1 DEPENDÊNCIAS FUNCIONAIS

Um dos problemas existentes na área de banco de dados é a inconsistência dos dados. Essa inconsistência ocorre quando um valor numa tabela é inserido ou alterado de uma maneira que quebra as regras de negócio que definem as relações entre os dados do banco. Essas relações podem ser definidas através de um instrumento conhecido como dependências funcionais. Segundo Navathe (Elmasri e Shamkant, 1989), "uma dependência funcional descrita por X $\rightarrow$ Y, entre dois conjuntos de atributos X e Y que são subconjuntos de R que especificam uma restrição nas tuplas que podem formar uma relação r em R. Esta restrição indica que para quaisquer duas tuplas que pertençam a relação r onde [X] = [X], então também temos que [Y] = [Y]. Isso quer dizer que os valores do componente Y da tupla em r dependem do valores do componente X."

No exemplo da tabela 2.1, pode-se criar uma dependência funcional que afirma que o nome e a nacionalidade de um registro dependem do seu CPF. Pode-se expressar a dependência funcional da seguinte maneira:

- CPF $\rightarrow$ Nome
- CPF $\rightarrow$ Nacionalidade

Embora as dependências funcionais sejam úteis, elas não são expressivas, e portanto outros instrumentos derivados delas acabam sendo mais úteis para garantir a integridade dos dados.

### 2.1.1 Restrições de negação

Segunda Pena (Pena et al., 2019), "uma restrição de negação (RN) $\phi$ sobre um relação r é a afirmação da forma:

- $\phi$: $\forall$tx,ty $\in$ r, $\neg$(p1 $\wedge$ . . . $\wedge$ pm)

onde $\phi$ é satisfeito por r se e somente se para qualquer par de tuplas tx, ty $\in$ r para pelo menos um dos predicados p1, ..., pm é falso."

| TID | Nome | CPF | Passaporte | Nacionalidade | Data de hospedagem |
|---|---|---|---|---|---|
| 1 | João da Silva | 749.992.870-56 | | Brasileiro | 01/01/2005 |
| 2 | Mariana Gomes | 766.990.260-46 | | Brasileiro | 01/01/2005 |
| 3 | Rui Alves | | IM003023 | Argentino | 03/01/2005 |
| 4 | João da Silva | 749.992.870-56 | ZD110332 | Brasileiro | 02/02/2008 |

Tabela 2.1: Registros de hóspedes em um hotel

| TID | Nome | CPF | Passaporte | Nacionalidade | Data de hospedagem |
|---|---|---|---|---|---|
| 1 | João da Silva | 749.992.870-56 | | Brasileiro | 01/01/2005 |
| 2 | Mariana Gomes | 766.990.260-46 | | Brasileiro | 01/01/2005 |
| 3 | Rui Alves | | IM003023 | Argentino | 03/01/2005 |
| 4 | João da Silva | 749.992.870-56 | ZD110332 | Brasileiro | 02/02/2008 |
| 5 | Mariana Gomes da Silva | 766.990.260-46 | | Brasileiro | 05/05/2012 |

Tabela 2.2: Registro de hóspedes com dado sujo.

Utilizando os dados da tabela 2.1, e com o apoio das dependências funcionais previamente demonstradas, pode-se criar as seguintes dependências funcionais:

- c1: $\forall$ tA,tB $\in$ 1.1: $\neg$(tA.CPF = tB.CPF $\wedge$ tA.Nome $\neq$ tB.Nome)
- c2: $\forall$ tA,tB $\in$ 1.1: $\neg$(tA.CPF = tB.CPF $\wedge$ tA.Nacionalidade $\neq$ tB.Nacionalidade)

A restrição de negação c1 implica que caso o CPF de duas tuplas forem idênticas e seus nomes forem diferentes, então existe uma inconsistência de dados. A restrição de negação c2 implica que caso o CPF de duas tuplas forem idênticas e suas nacionalidades forem diferentes, então existe uma inconsistência de dados. Outra razão a favor da utilização das restrições de negação é que elas são facilmente portadas para a linguagem SQL, possibilitando o uso delas para impedir uma alteração do banco que possa gerar uma inconsistência ou para encontrar inconsistências e remover "dados sujos" em bancos já existentes, este último processo sendo conhecido como limpeza de dados.

### 2.1.2 Exemplo do uso das restrições de negação para impedir inserção de dados sujos ao banco de dados.

Expandido no exemplo dado pela tabela 2.1, pode-se imaginar um caso onde a cliente Mariana Gomes se casa e altera seu nome para Mariana Gomes da Silva e se hospeda no mesmo hotel, agora com seu nome de casada. Caso a inserção do novo registro for bem-sucedida, tem-se a tabela 2.2.

No entanto, nesse caso a inserção falhará devido ao fato que esse novo registro gera um inconsistência de acordo com a restrição de negação c1:

c1: $\forall$ t2,t5 $\in$ T2: $\neg$(t2.CPF = t5.CPF $\wedge$ t2.Nome = t5.Nome).

Como t2.Nome é "Mariana Gomes" e t5.Nome é "Mariana Gomes da Silva", essa restrição aponta a inconsistência. Limpeza de dados e descoberta de restrições de negação.

Outro caso de uso para restrições de negação é na limpeza de dados. Nesse caso tem-se um banco de dados com "dados sujos" (que geram inconsistências) já existentes, como a tabela 2.2 Neste caso, podemos usar as restrições para separar os dados limpos dos sujos, podendo assim tratá-los, "limpando" o banco de dados. No entanto, nem todos os bancos de dados têm restrições de negações bem formuladas e acabam com dados sujos que precisam ser limpos, mas sem as

restrições necessárias para encontrá-los. Foi por essa razão que algoritmos como o DCFinder foram criados: descobrir as restrições que são necessárias para a limpeza e manutenção do banco de dados.

## 2.2 MÉTRICAS DE DESCOBERTA DE RN

Para analisar quais são as melhores restrições de negação descobertas pelo algoritmo de descoberta é necessário utilizar certas métricas para a avaliação da relevância de cada resultado. Os resultados dessas métricas serão utilizados por um função “interesse” para ordenar as restrições de negação devolvidas pelo algoritmo. Utiliza-se as seguintes métricas:

### 2.2.1 Concisão

Seguindo a heurística da Navalha de Ockham, também conhecida como Lei da Parcimônia, damos preferência a hipóteses que menos pressupõem fatos. No contexto da descoberta de restrições de negação, utiliza-se a métrica de concisão para garantir que as restrições com ordenamento favorável são aquelas cujo tamanho é o menor proporcionalmente as outras restrições descobertas. Para chegar no valor de concisão de um restrição de negação, divide-se o menor o tamanho da menor restrição encontrada pelo tamanho da restrição atual.

Formalmente, Papotti (Chu et al., 2013) define a concisão de um RN como "o tamanho viável mínimo de um RN dividido pelo seu tamanho". É possível expressar essa definição com a seguinte fórmula:

$$Succ(\varphi) = \frac{Min(\{Len(\phi)|\forall\phi\})}{Len(\varphi)}$$

Para calcular o tamanho de uma restrição, utilizaremos a quantidade de símbolos únicos pertencentes ao alfabeto da restrição. O alfabeto da restrição é definido como sendo o conjunto de suas tuplas, seus atributos, seus operadores e suas constantes.

Para demonstrar o cálculo da concisão será necessário adicionar mais uma restrição de negação, que será chamada c3. Dessa forma, teremos o seguinte conjunto de restrições:

- c1: $\forall$ tA,tB $\in$ 2.1: $\neg$(tA.CPF = tB.CPF $\wedge$ tA.Nome = tB.Nome)
- c2: $\forall$ tA,tB $\in$ 2.1: $\neg$(tA.CPF = tB.CPF $\wedge$ tA.Nacionalidade = tB.Nacionalidade)
- c3: $\forall$ tA,tB $\in$ 2.1: $\neg$(tA.CPF = tB.CPF $\wedge$ tA.Passaporte = tB.Passaporte $\wedge$ tA.CPF $\neq$ NULL $\wedge$ tA.Passaporte $\neq$ NULL)

Para essas restrições, podemos montar o seguinte alfabeto:

$$\mathbb{A} = \{tA, tB, CPF, Nome, Nacionalidade, Passaporte, NULL, =, \neq\}$$

Para calcular a concisão de cada restrição de negação, precisamos calcular o tamanho de cada uma delas. Para isso, conferimos quantos símbolos pertencem a cada:

- Símbolos de c1 = tA, tB, CPF , Nome, =. Portanto, Len(c1) = 5
- Símbolos de c2 = tA, tB, CPF , Nacionalidade, =. Portanto, Len(c2) = 5
- Símbolos de c3 = tA, tB, CPF , Passaporte, NULL, =, ≠ . Portanto, Len(c3) = 7

Com esses valores, sabemos que Min(Len(Ø)) = 5. Portanto:

- Succ(c1) = 5 / 5 = 1
- Succ(c2) = 5 / 5 = 1
- Succ(c3) = 5 / 7 = 0.714

### 2.2.2 Cobertura

A métrica de cobertura de uma restrição de negação mede a importância daquela restrição no banco de dados. Esta métrica é representada pela quantidade de evidências que satisfazem a restrição, onde uma evidência é um conjunto de tuplas que satisfazem pelo menos um predicado da restrição. Quanto maior o número de predicados satisfeitos na evidência, mais forte ela é, aumentando o valor da métrica de cobertura.

Formalmente, Papotti (Chu et al., 2013) define a evidência-k (kE) para cobertura de $\phi$ como "um par de tuplas onde k é o número de predicados em $\phi$ que satisfazem a tupla e k ≤ |$\phi$.Pres| - 1.", onde $\phi$.Pres é a quantidade de predicados da RC $\phi$ e define o peso da evidência-k com a seguinte fórmula:

$$w(k) = \frac{k+1}{|\varphi.Pres|}$$

Com essas definições, Papotti calcula a cobertura de $\phi$ com a seguinte fórmula:

$$Coverage(\varphi) = \frac{\sum_{k=0}^{|\varphi.Pres|-1} |kE| * w(k)}{\sum_{k=0}^{|\varphi.Pres|-1} |kE|}$$

# 3 DADOS AUSENTES

Quando um determinado valor dentro de um banco de dados está faltando ele é considerado ausente. Essa ausência pode ser causada por diversos fatores, como falha na inserção de dados pelo usuário, falha no equipamento que coleta os dados (por exemplo, um sensor desligado pois sua bateria acabou), ou até mesmo por falha do hardware que armazena os dados. A partir de um certo tamanho e maturidade, a existência de dados ausentes em um banco é inevitável.

A análise dos dados de um banco, como por exemplo na descoberta de dependências funcionais pelo algoritmo Tane, pode escolher uma de diversas estratégias para tratar de dados ausentes, como por exemplo:

1. Ignorar a tupla inteira da análise.
2. Ignorar a coluna inteira para a análise.
3. Inserir um valor para todos os dados ausentes.

Utilizando as estratégias 1 e 2 é possível realizar a análise apenas com dados completos, no entanto, já que elas não consideram dados que podem ser relevantes de conjuntos que contenham algum dado ausente, essas estratégias podem criar análises falhas. Esse problema é acentuado quando a ausência dos dados seguem algum tipo de padrão. Em casos mais graves, é possível que a maioria das tuplas ou colunas contenham algum dado ausente. Nesses casos, essas estratégias podem ignorar a grande maioria dos dados, impossibilitando qualquer tipo de análise.

A estratégia de inserção de dados nos lugares ausentes pode ser realizada por diversos métodos, mas a inserção de valores arbitrários diminui a qualidade da análise. Um método mais preciso considera os dados não-ausentes para prever um valor a ser inserido no dado ausente que não cause uma grande perda de acurácia para a análise.

Para a escolher a melhor estratégia para o tratamento dos dados ausentes, eles são classificados em três mecanismo de probabilidade distintos.

## 3.1 MECANISMOS DE PROBABILIDADE DE DADOS AUSENTES

Dependendo da forma que os dados ausentes foram inseridos, eles podem ser classificados por um dos três mecanismo de probabilidade:

### 3.1.1 MISSING COMPLETELY AT RANDOM - MCAR

O mecanismo de probabilidade MCAR ("missing completely at random") é caracterizado pela probabilidade de inserção de dados ausentes não ser afetada por nenhuma relação de

causalidade ou correlação com qualquer outro dado do banco. A possibilidade que dados naturalmente sigam esse comportamento é implausível, portanto esse mecanismo é utilizado principalmente quando os dados ausentes tem como sua causa falha de equipamento.

Em 2.2 um exemplo de dado ausente por MCAR seria na hipótese onde na tupla 1 o hóspede já tinha um passaporte, mas sua inserção na tabela não ocorreu devido a um erro no sistema.

### 3.1.2 MISSING AT RANDOM - MAR

O mecanismo de probabilidade MAR ("missing at random") é caracterizado pela probabilidade de inserção do dado ausente não ser afetada pelo valor do do próprio dado ausente. No entanto, a possibilidade de um valor ser ausente pode ter relações de causalidade e correlação com os valores não-ausentes inseridos na mesma tupla.

Devido a estas relações, a estratégia de inserção de novos valores para o tratamento de dados ausentes se torna viável. As relações permitem observar possíveis valores para o dado ausente baseando em tuplas com dados similares.

É importante ressaltar que as relações observadas em dados inseridos por esse mecanismo podem ter apenas uma relação de correlação, não tendo uma relação direta de causalidade com os outros dados da tupla.

Em 2.2 um exemplo de dado ausente por MAR seria na hipótese que o CPF ausente na tupla 3 é estatisticamente mais provável, já que o hóspede não é brasileiro, enquanto os hóspedes brasileiros no restantes das tuplas não apresentam esse atributo como dado faltante.

### 3.1.3 MISSING NOT AT RANDOM - MNAR

O mecanismo de probabilidade MNAR ("missing not at random") é caracterizada pela probabilidade de inserção do dado ausente ser afetada pelo próprio valor do dado ausente. Existe uma possibilidade da probabilidade da ocorrência do dado ausente também ser afetada pelas relações do dado ausente com outros dados da tupla.

Este é o mecanismo que causa a maior dificuldade para a predição de um possível valor para substituir o dado ausente. Caso seja necessário realizar uma análise num banco que tenha dados inseridos por mecanismos MNAR, frequentemente o problema é abstraído e tratado como um problema MAR.

Em 2.2 um exemplo de dado ausente por MNAR seria na hipótese onde hóspedes de certas nacionalidade tem uma probabilidade maior de ter o atributo Nacionalidade em suas tuplas como um dado ausente.

# 4 DESCOBERTA DE DEPENDÊNCIAS FUNCIONAIS COM DADOS AUSENTES

Algoritmos de descoberta de DFs podem ser utilizados em banco de dados reais para diversas aplicações, entre elas a limpeza de dados. No entanto, os dados ausentes que precisam ser limpos exigem tratamento durante a descoberta de DFs. Existem diferentes métodos para realizar este tratamento:

- **Ignorar a tupla inteira para a descoberta de DFs:** método mais simples de tratamento, no entanto é inviável em bancos de dados com quantidades significativas de dados ausentes.
- **Tratar cada valor ausente como um valor único (NULL-NOT-EQ) ou idêntico (NULL-EQ):** esse método altera as definições das DFs para tratar cada nulo com sendo de um semântica específica: Na semântica NULL-NOT-EQ todos os nulos são considerados únicos, enquanto em NULL-EQ eles são considerados idênticos.
- **Utilizar Dependências Funcionas Aproximadas (DFAs):** Uma Dependência Funcional normalmente exige que todos os dados de um banco se comportem a ela. Uma Dependência Funcional Aproximada admite que um pequeno número de tuplas não se comportam no DFA. A proporção do número de tuplas que podem quebrar as condições delas é chamado de coeficiente de aproximação.
- **Inserção de dados:** É feita uma predição de um possível valor substituto para cada dado ausente. Dependendo do mecanismo de inserção de nulos, é impossível utilizar este método.

No entanto, após realizar estes tratamentos as DFs não serão os mesmas das DFs do mesmo banco de dados sem dados ausentes. Idealmente, o nosso método escolhido deve tornar as DFs criadas pelo banco com dados ausentes o mais semelhante possível aos DFs sem dados ausentes.

Para melhor analisar a descoberta de DFs com dados ausentes pode-se comparar as DF resultantes do tratamento com as DFs resultante da descoberta sem dados ausentes. Assim, pode-se classificar os DFs em três grupos:

- **Dependência Funcionais Idênticas:** uma DF idêntica a uma DF encontrada no banco de dados sem dados ausentes.
- **Dependência Funcionais Falsas:** uma DF que não é encontrada na descoberta de DFs sem dados ausentes.

- **Dependência Funcionais Fantasma:** uma DF que é encontrada na descoberta sem dados ausentes mas não está presente no conjunto de DFs encontradas com dados ausentes.

| DFs descobertas a tabela original | DFs descobertas na tabela com dados ausentes |
|---|---|
| CPF $\rightarrow$ Nome | CPF $\rightarrow$ Nome |
| CPF $\rightarrow$ Nacionalidade | CPF $\rightarrow$ Passaporte |

Tabela 4.1: Tabela com exemplos de possíveis dependências funcionais encontradas após o uso de um algoritmo de descoberta em tabelas sem e com dados ausentes.

Utilizando a tabela 4.1 como exemplo, pode-se classificar as dependências funcionais da seguinte maneira:

- CPF $\rightarrow$ Nome: DF Idêntica
- CPF $\rightarrow$ Nacionalidade: DF Falsa
- CPF $\rightarrow$ Nome: DF Fantasma

A escolha do tratamento deve se basear em qual método cria mais Dependências Funcionais Idênticas e evita Dependências Funcionais Falsas e Dependências Funcionais Fantasma.

## 4.1 METANOME

As seguintes análises utilizaram o software Metanome para executar os algoritmos de Descoberta de Dependência Funcional. A ferramenta também disponibiliza metadados da execução e uma visualização do resultado final. Ambos foram utilizados na formação das análises,

## 4.2 ALGORITMO TANE EM BANCO DE DADOS PEQUENO COM MECANISMO MCAR

O algoritmo de descoberta de Dependência Funcional utilizado por essa análise se chama Tane. Ele utiliza uma estrutura de dados em formato de látice para melhorar o desempenho da busca. Também utiliza o método de Dependência Funcionais Aproximadas para quando um banco de dados contém dados ausentes. Foi utilizada a versão 1.2 do Tane nesta análise.

A execução do algoritmo Tane é realizada em 3 etapas:

- **Criação de partições:** para consumir menos recursos computacionais, a primeira parte da execução do algoritmo é a criação de múltiplas partições, onde cada uma representa as tuplas que apresentam o mesmo valor para certo atributo. Essas partições são a unidade utilizada para a descoberta dos DFs, diminuindo a quantidade de recursos utilizados para a execução do algoritmo.

- **Poda em estrutura de dados de látice:** Para garantir o que certos conjuntos de partições não criem DFs não concisas, o algoritmo poda combinações entre diferentes partições que podem ser descritas por um por combinações mais simples. Para executar essa poda o algoritmo utiliza uma estrutura de dados em formato de látice, onde a poda é realizada através de um algoritmo de busca em largura, garantindo que as primeiras podas em combinações com menos partições evitem que o algoritmo precise buscar todas as combinações derivadas por ela, evitando o processamento desnecessário de dados. É possível ver essa poda em ação no gráfico 4.1, retirado do artigo (Huhtala et al., 1999), onde a poda de B faz com que seja desnecessário realizar a poda em combinações derivadas.
- **Descoberta das DFs:** A partir das combinações resultantes da poda é possível realizar a descoberta das DFs das tuplas, criando o resultado final da execução do algoritmo.

## 4.3 TANE

O algoritmo Tane trata nulos com a semântica NULL-EQ e cria DFAs com um coeficiente de aproximação definido pelo algoritmo.

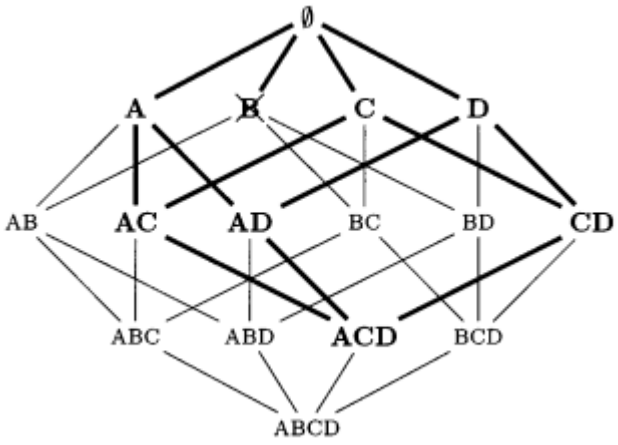


Figura 4.1: Gráfico retirado do artigo (Huhtala et al., 1999). As combinação em negrito são consideradas, enquanto as sem negrito foram descartadas quando B foi descartado e não são vistas pelo algoritmo

Para demonstrar o funcionamento do algoritmo Tane e da ferramenta Metanome será utilizado o banco de dados provido pela instalação padrão do Metanome chamado WDC_Planets, que contém informações dos diversos planetas do sistema solar. O banco tem 90 valores, divididos em 10 colunas e 9 tuplas.

A inserção de dados ausentes foi feita utilizando um algoritmo que utiliza o mecanismo de probabilidade MCAR, onde uma coordenada aleatória da tabela cujo valor não está como dado ausente é transformada em dado ausente em cada iteração do algoritmo. Dessa forma, o algoritmo permite que o usuário facilmente insira a quantidade de dados que quiser dentro da tabela.

| Name | Type | EquatorialDiameter | Mass | OrbitalRadius | OrbitalPeriod | RotationPeriod | ConfirmedMoons | Rings | Atmosphere |
|---|---|---|---|---|---|---|---|---|---|
| Mercury | Terrestrial | 0.382 | 0.06 | 0.47 | 0.24 | 58.64 | 0 | no | minimal |
| Venus | Terrestrial | 0.949 | 0.82 | 0.72 | 0.62 | -243.02 | 0 | no | CO2 N2 |
| Earth | Terrestrial | 1 | 1 | 1 | 1 | 1 | 1 | no | N2 O2 Ar |
| Mars | Terrestrial | 0.532 | 0.11 | 1.52 | 1.88 | 1.03 | 2 | no | CO2 N2 Ar |
| Jupiter | Giant | 11.209 | 317.8 | 5.2 | 11.86 | 0.41 | 67 | yes | H2 He |
| Satum | Giant | 9.449 | 95.2 | 9.54 | 29.46 | 0.43 | 62 | yes | H2 He |
| Uranus | Giant | 4.007 | 14.6 | 19.22 | 84.01 | -0.72 | 27 | yes | H2 He |
| Neptune | Giant | 3.883 | 17.2 | 30.06 | 164.8 | 0.67 | 14 | yes | H2 He |

Tabela 4.2: Tabela WDC_Planets

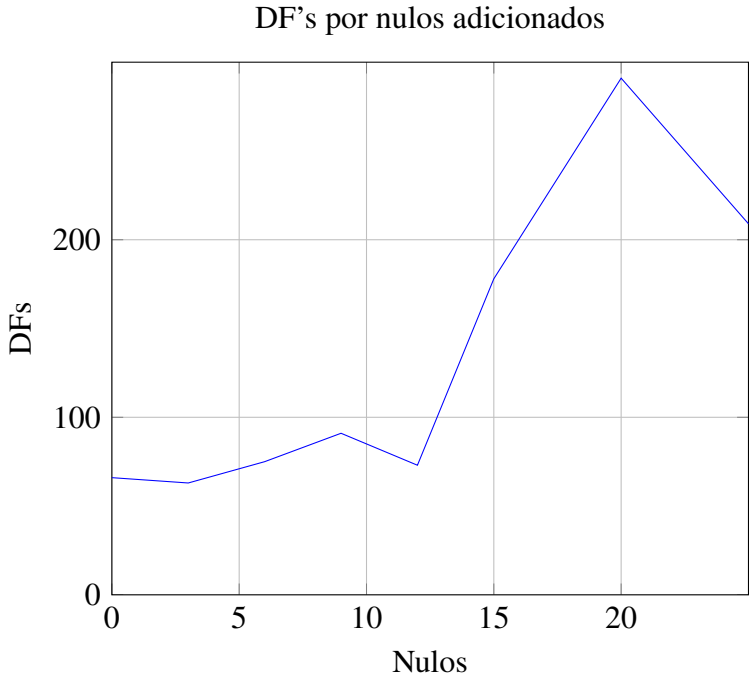


Figura 4.2: Tabela DFs por valores nulos adicionados a tabela.

Foi realizada uma série de execuções do algoritmo com várias modificações de WDC_Planets, onde cada modificação contém um certo número de dados ausentes. Para essa análise, foram realizados testes com 0, 3, 6, 9, 12, 15, 20, e 25 dados ausentes.

Através da figura 4.2 é possível observar que o número de DFs descobertas pelo algoritmo sobe drasticamente nos testes com 15 e 20 dados ausentes, mas começa a cair nos teste com 25. Essas dependências não estão presentes no teste com 0 nulos, portanto são consideradas dependências funcionais falsas. É possível observar na 4.2 que a grande maioria das novas dependências funcionais falsas são criadas com múltiplos determinantes, enquanto a grande maioria das dependências funcionais no teste com 0 dados ausentes possui apenas um determinante.

A visualização Metanome do resultado do algoritmo Tane consiste na criação de um gráfico circular, onde cada camada indica um um determinante de uma DF criada. Cada cor desse gráfico é um atributo diferente, e múltiplas camadas sobrepostas indica que existe um DF encontrado com aqueles determinantes.

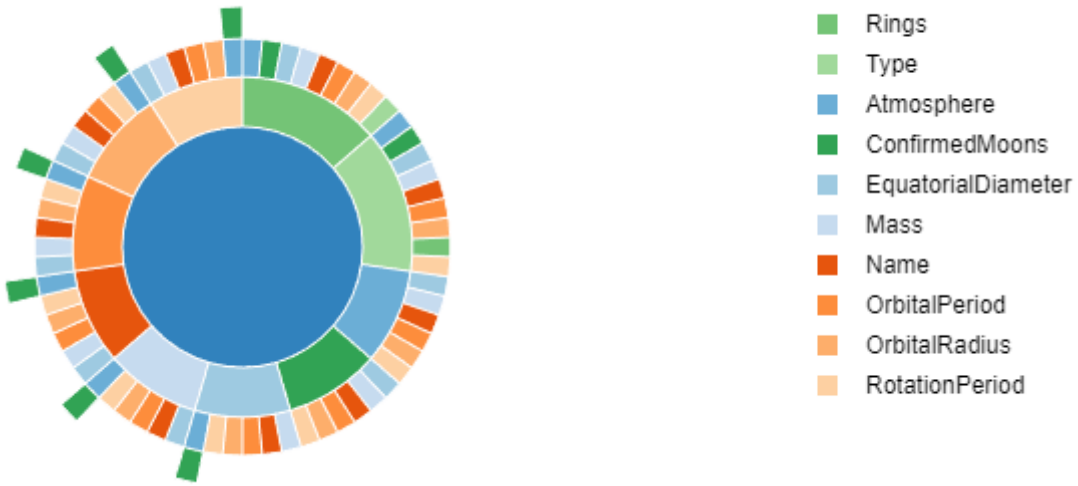


Figura 4.3: Visualização do Metanome dos DFs descobertos pelo numa tabela original, sem dados ausentes

No gráficos 4.3 e 4.4 é possível observar que a inserção de poucos nulos não afeta a quantidade de DFs em grande quantidade, e a maioria dos DFs possui 2 determinantes, com algumas exceções de DFs que também incluem o número de luas ao redor do planeta.

No entanto, os gráficos 4.5, 4.6, 4.7 e 4.8 demonstram que a inserção de novos nulos eventualmente causam que o número de DFs fantasma aumenta drasticamente, especialmente no número de DFs fantasma com 3 dependentes. Com um número de nulo suficientemente grande, é possível observar até DFs com 4 dependentes.

Também é possível observar que a queda de DFs criadas nos testes de 20 dados ausentes para 25 dados ausentes ocorre devido a falta de informação para o Tane criar as dependências funcionais. Chamaremos esse ponto de queda na quantidade de DFs como "ponto de saturação".

Concluindo, é possível afirmar que nestas condições o crescimento de dados ausentes faz com que o algoritmo Tane crie mais DFs falsas até que chegue em seu ponto de saturação, onde a falta de dados faz com que o número de DFs encontradas diminua.

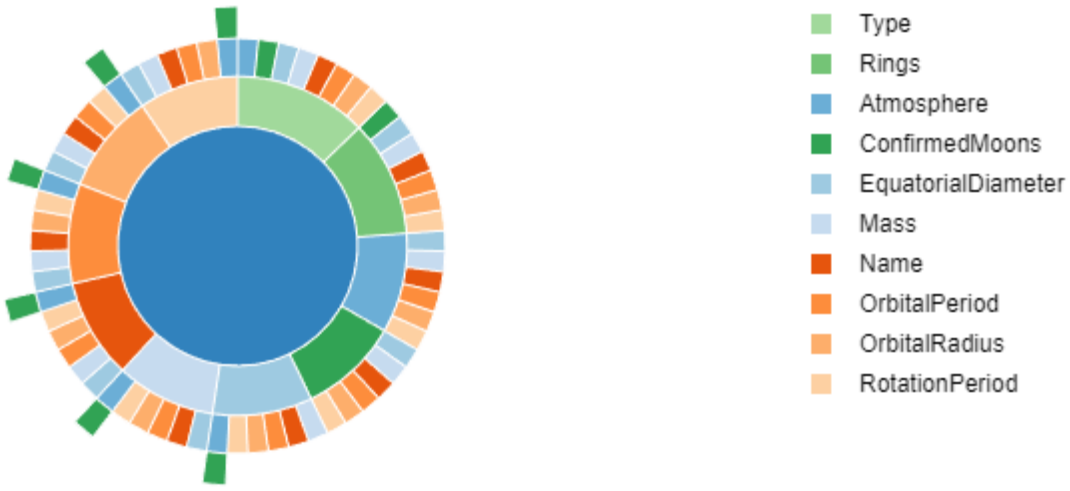


Figura 4.4: Visualização do Metanome dos DFs descobertos pelo numa tabela com 3 inserções de dados ausentes


Figura 4.5: Visualização do Metanome dos DFs descobertos pelo numa tabela com 9 inserções de dados ausentes

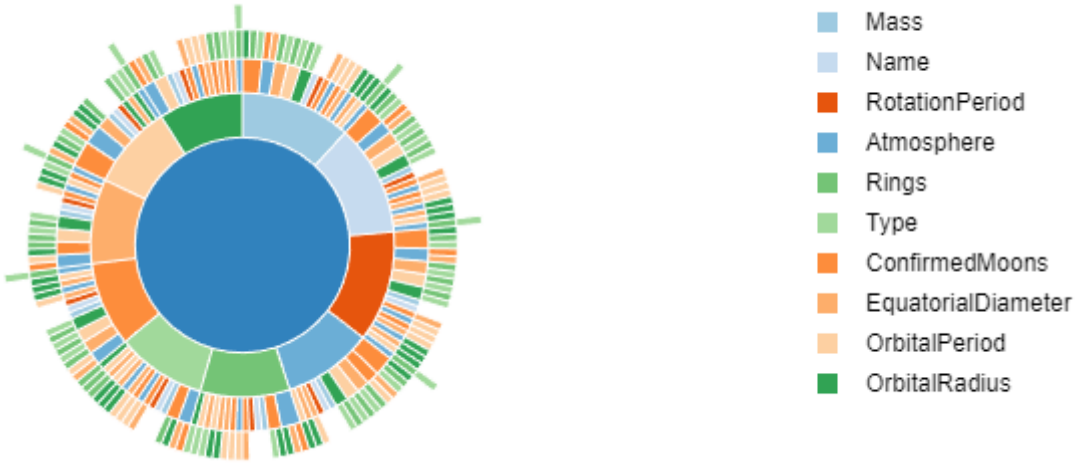


Figura 4.6: Visualização do Metanome dos DFs descobertos pelo numa tabela com 15 inserções de dados ausentes


Figura 4.7: Visualização do Metanome dos DFs descobertos pelo numa tabela com 20 inserções de dados ausentes


Figura 4.8: Visualização do Metanome dos DFs descobertos pelo numa tabela com 25 inserções de dados ausentes

# 5 ANÁLISE EM DADOS DO INEP

Desde 2007, o Instituto Nacional de Estudos e Pesquisas Educacionais (INEP) realiza censos anuais e coleta dados das milhares de escolas no país com o intuito de fornecer essas informações para uso da sociedade civil. O Centro de Computação Científica e Software Livre (C3SL) utiliza esse dados para o projeto Simulador de Custo-Aluno Qualidade (SIMCAQ). Ao longo dos anos dois problemas surgiram neste banco:

- A criação de novos atributos redundantes as tabelas onde algoritmos como o Tane consideram todas as tuplas de anos anteriores a criação da tabela como sendo nulos.
- Mudanças na quantidade de nulos em uma tupla ao longo dos anos. A consistência da razão entre valores nulos e não-nulos entre diferentes atributos sugere que a categoria do mecanismo de probabilidade de dado ausente neste caso se trata do mecanismo MAR ("missing at random").

O primeiro problema pode ser visto na visualização entre a proporção entre os dados existentes e nulos ao longo dos anos. É possível observá-lo na tabela *escola* do SIMCAQ ao reparar nos atributos relacionados a água nas escolas presentes no banco do censo. A mapa de calor 5.1 demonstra que o atributo *agua_filtrada* (representado por um número 0 ou 1 na tabela *escola*) foi substituído por *agua_potavel* (representado pelos booleanos "verdadeiro"e "falso"na tabela *escola*) em 2019.

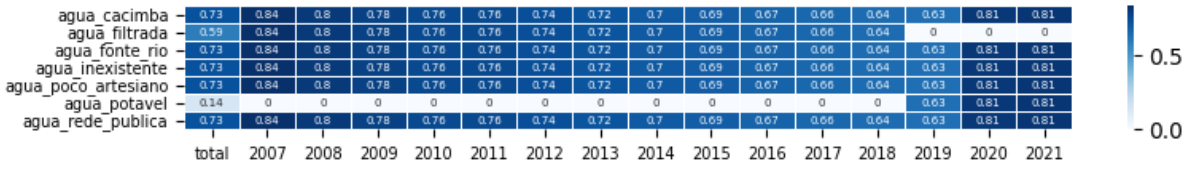


Figura 5.1: Proporção entre dados existentes e nulos ao longo dos anos para os atributos relacionados a água.

A adição tardia deste atributo na tabela *escola* cria um problema para algoritmos de descoberta de dependência funcionais: como visto no capítulo 4.3 a adição de dados ausentes numa tabela acaba criando DFs fantasmas, e com o novo atributo todos os resultados de *agua_potavel* entre os anos 2007 e 2018 e os resultados de *agua_filtrada* entre 2009 e 2021 são considerados ausentes. Portanto, uma hipótese a ser testada é se realmente existe um aumento no número de dependências funcionais devido a adição de dados ausentes devido a um atributo redundante.

Para testar esta hipótese, pode ser feita uma mesclagem entre as colunas dos dois atributos e realizar uma comparação entre as DFs descobertas entre essa versão e a versão original com os atributos redundantes.

O segundo problema também pode ser visto na visualização entre a proporção entre os dados existentes e nulos ao longo dos anos. É possível observá-lo na tabela *escola* do SIMCAQ ao reparar nos atributos relacionados ao sanitários nas escolas presentes no banco do censo. A partir do ano 2015, o número de nulos aumentou drasticamente. No período entre 2015 e 2018 cerca de 30% dos dados são nulos nesses atributos sobre sanitários, e a partir de 2019 alguns conjuntos são descontinuados, criando um conjunto de dados ausentes que irá aumentar ao longo dos anos.

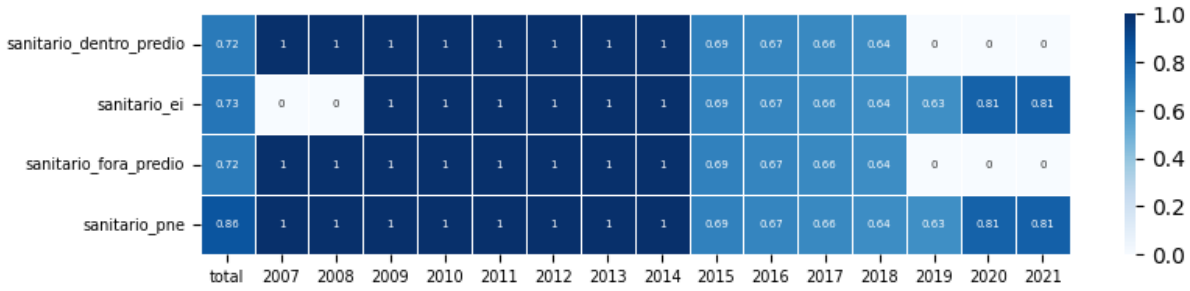


Figura 5.2: Proporção entre dados existentes e nulos ao longo dos anos para os atributos relacionados à sanitários.

O banco de dados contém tabelas muito longas, portanto será necessário avaliar o banco através de amostras. Para construir essas amostras foi utilizado um programa em Python utilizando a biblioteca PySpark para gerir a grande quantidade de dados na tabela.

## 5.1 AMOSTRAGEM DA TABELA

O Tane tem como limitação o tamanho de tuplas e, principalmente, de atributos na tabela onde o algoritmo é executado. Devido a isso, é necessário extrair amostras que representam a tabela original mas que conseguem ser analisadas pelo algoritmo.

Nos seguintes experimentos, cada amostra contém 300 tuplas de uma seleção de atributos da tabela *escola* relevantes para a análise. Todos os casos contém atributos base que estão presentes em todas as tuplas ao longo dos anos. Esses atributos servem de base para o Tane conseguir extrair as dependências funcionais da análise em conjunto com os atributos específicos da análise.

Os atributos base da tabela *escola* utilizados são os seguintes:

- *ano_censo*
- *id*
- *cod_orgao_regional_inep*
- *situacao_de_funcionamento*
- *estado_id*

- *municipio_id*
- *dependencia_adm_id*
- *localizacao_id*

# 6 ANÁLISE EM ATRIBUTOS RELACIONADOS À ÁGUA

Essa análise tem o intuito de provar a hipótese que o atributo redundante *agua_potavel* gera dependências funcionais fantasmas com a execução do algoritmo Tane.

Os atributos adicionais utilizados na criação das amostras para esta análise são os seguintes:

- *agua_filtrada*
- *agua_rede_publica*
- *agua_poco_artesiano*
- *agua_cacimba*
- *agua_fonte_rio*
- *agua_inexistente*
- *agua_potavel*

Desses itens adicionais, *agua_potavel* é um atributo redundante em relação ao atributo já existente *agua_filtrada*. A depreciação de *agua_filtrada* e a criação de *agua_potavel* ocorreu em 2019. *agua_filtrada* utiliza o tipo de dados *int*, onde 0 e 1 marcam se a escola possui água filtrada. Enquanto isso, *agua_potavel* utilizam tipo de dados booleano "true"e "false"para marcar se a escola possui água potável. Essa mudança no tipo de dados pode ter sido a causa da criação da coluna redundante.

Para realizar a análise, o algoritmo Tane será executado na amostra original e em 3 outras variações:

- **todos**: Amostra original, com todos os dados base e adicionais de água.
- Variação **mesclada**: Todos os dados do atributo *agua_potavel* foram convertidos para o tipo de dados *int* e inseridos no lugar dos dados ausentes em *agua_filtrada*
- Variação **sem filtrada**: O atributo *agua_filtrada* é removido da análise.
- Variação **sem potavel**: O atributo *agua_potavel* é removido da análise.

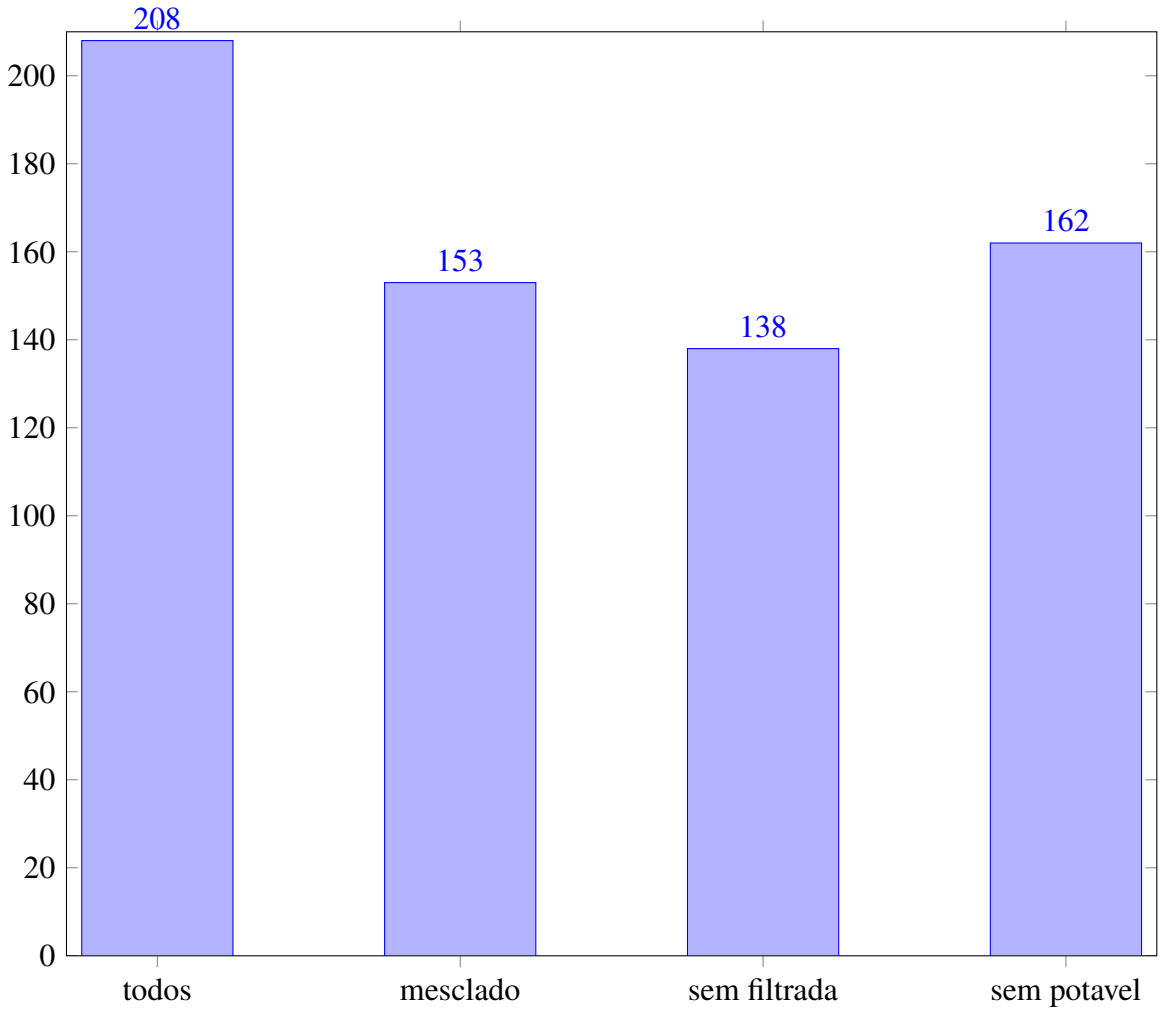


Figura 6.1: Número de dependência funcionais para cada caso apresentado nos atributos de água.

Após executar o algoritmo Tane sobre as amostras, é possível observar (figura 6.1) que todas as variações da amostra base apresentam uma redução no número de dependências funcionais. O caso da variação **mesclada** é especialmente forte: mantendo todos os mesmos dados da amostra base **todos** foi possível reduzir a quantidade de dependências funcionais de 208 para 153 DF's, uma redução de 55 DFs, ou seja, uma redução de 26% de DFs.

Portanto, podemos confirmar a hipótese que atributos redundantes aumentam o número de dependências funcionais fantasma.

De acordo com a figura 6.2, 9.9% das dependências funcionais encontradas na amostra **mesclada** apresentam cobertura total, enquanto este valor é de apenas 7.7% das DF's da amostra **todos**, uma redução de 23.4%.

Além disso, a quantidade de DF's com cobertura baixa na amostra **todos** é 51.8% maior que as DF's encotnradas na amostra **mesclado**. Dessa forma, é possível afirmar que a eliminação de atributos redundantes colabora na descoberta de DF's com uma métrica de cobertura melhor,

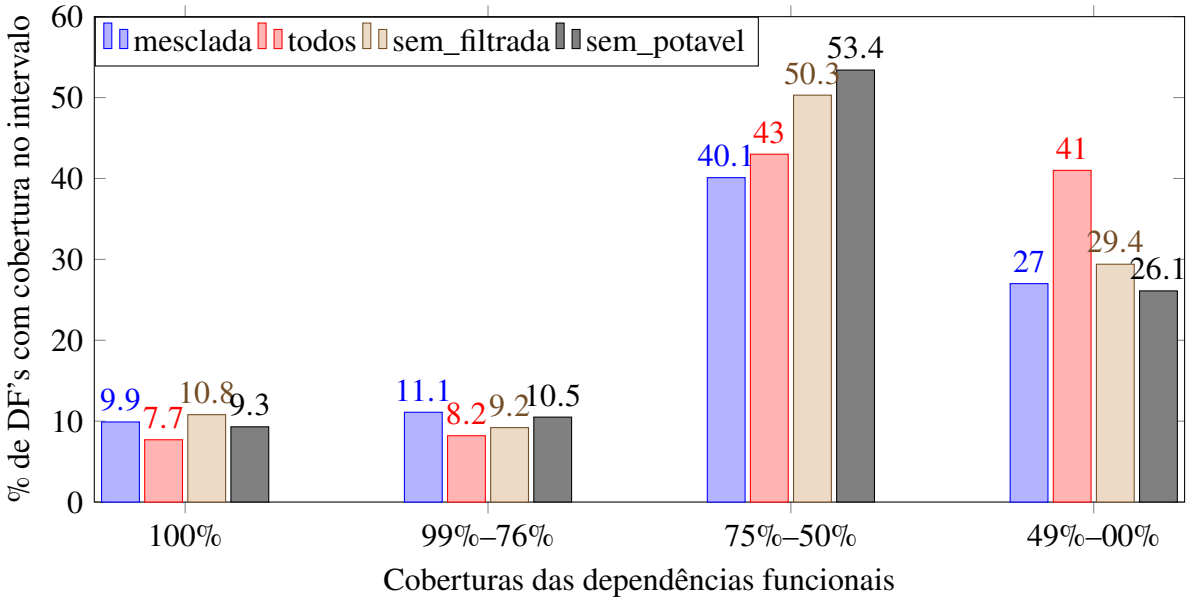


Figura 6.2: Coberturas das dependências funcionais relacionadas à água descobertas, dividido em intervalos.

aumentando a quantidade de DF's com alta cobertura e diminuindo a quantidade de DF's com baixa cobertura.

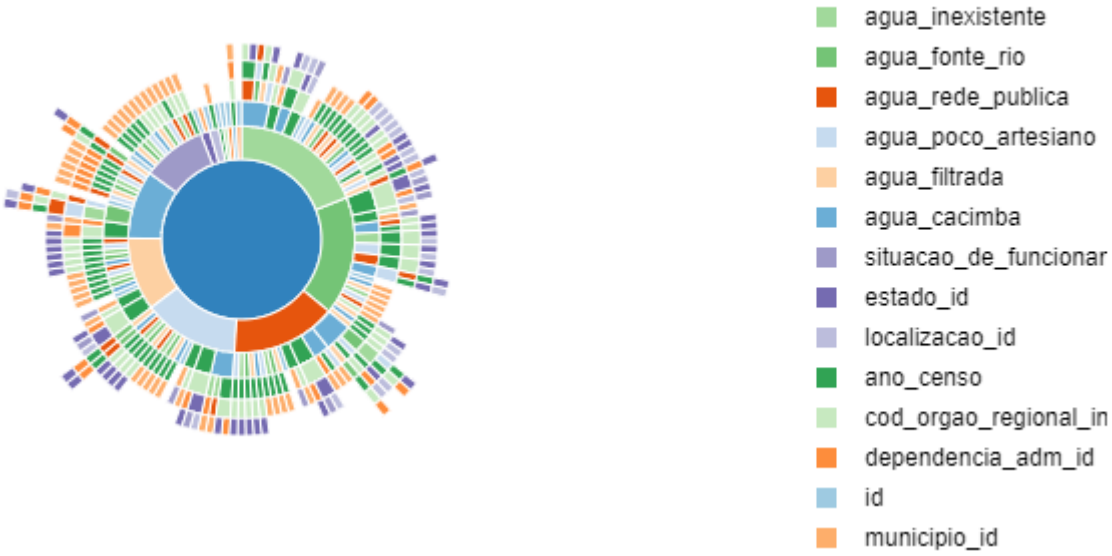


Figura 6.3: Visualização do Metanome dos DFs descobertos pela variação **mesclado**

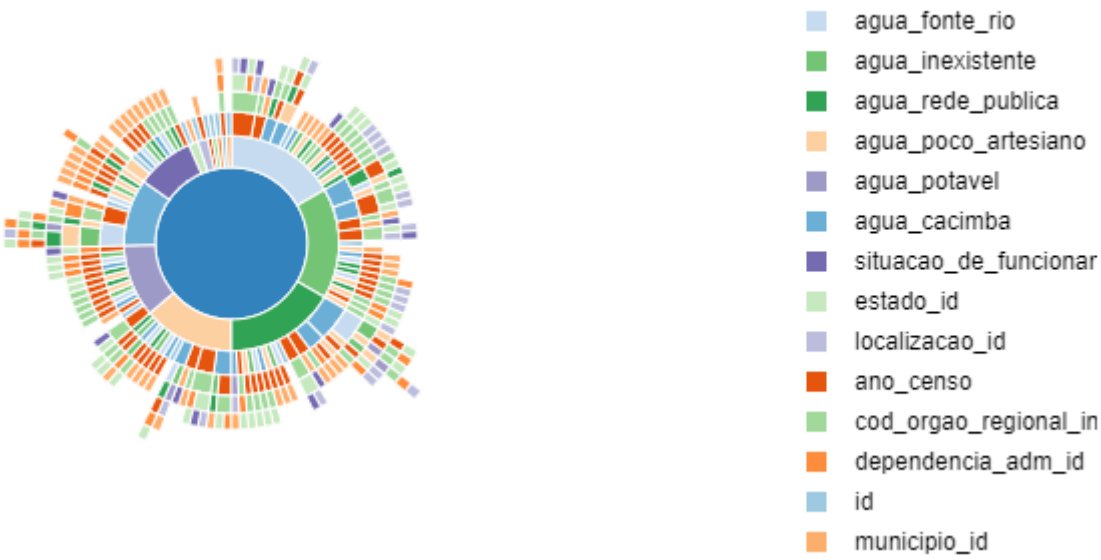


Figura 6.4: Visualização do Metanome dos DFs descobertos pela variação **sem_filtrado**


Figura 6.5: Visualização do Metanome dos DFs descobertos pela variação **sem_potavel**

# 7 ANÁLISE EM ATRIBUTOS RELACIONADOS À SANITÁRIOS

Essa análise tem o intuito de provar a hipótese que a mudança dos critérios utilizados para inserir dados na tabela ao longo dos anos gera dependências funcionais fantasmas e com métrica de cobertura reduzida com a execução do algoritmo Tane.

Os atributos adicionais utilizados na criação das amostras para esta análise são os seguintes:

- *sanitario_fora_predio*
- *sanitario_dentro_predio*
- *sanitario_ei*
- *sanitario_pne*

Nestes itens adicionais nota-se uma anomalia ao olhar para a figura 5.2: a partir de 2015 os atributos que costumavam sempre ter seus dados completamente preenchidos passam a ter dados ausentes nos atributos adicionais. Notavelmente esses atributos têm a mesma razão de dados ausentes entre todos, algo que se repete ao longo dos anos e continua até o final dos dados da tabela em 2021.

Esse comportamento padrão entre os números de dados ausentes entre os diferentes atributos indicam que esses dados ausentes foram inseridos pelo mecanismo de inserção "MAR"("missing at random").

Para realizar a análise, o algoritmo Tane será executado na amostra original e em 2 outras variações:

- **todos**: Amostra original, com todos os dados base e adicionais dos sanitários.
- Variação **pre 2015**: Todos as tuplas da amostra tem como dado de seu atributo *ano_censo* um número menor que 2015.
- Variação **pos 2015**: Todos as tuplas da amostra tem como dado de seu atributo *ano_censo* um número igual ou maior que 2015.

Após executar o algoritmo Tane sobre as amostras, é possível observar (figura 7.1) o aumento da quantidade de dados ausentes gera um aumento nas dependências funcionais fantasmas descobertos pelo algoritmo. A análise feita com apenas com tuplas criadas posteriormente ao ano de 2015 apresenta 36 dependências funcionais adicionais comparado a análise dos anos

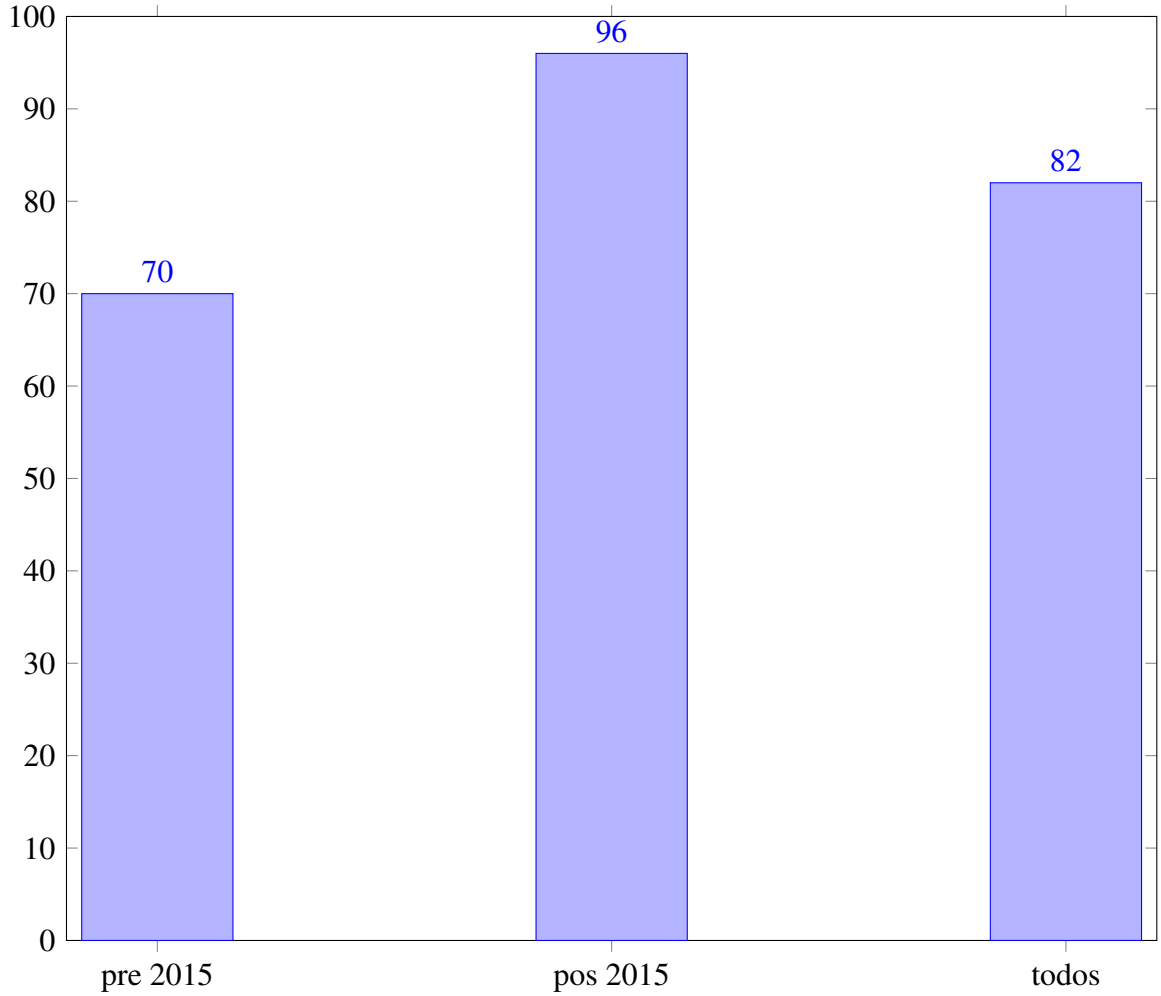


Figura 7.1: Número de dependência funcionais para cado caso apresentado nos atributos sanitários.

anteriores a 2015, um aumento 51% na quantidade de DFs. A análise da amostra base apresenta um número de dependências funcionais entre o resultado das duas amostras com variações.

É possível observar na figura 7.2 que o número de dependências funcionais que apresentam cobertura total é drasticamente maior na amostra **sanitario_pre_2015** do que nas outras. A amostra **sanitario_pos_2015** apresenta a menor proporção de DF's com cobertura total entre as amostras e a maior proporção de DF's com cobertura abaixo de 49%.

Com esse resultado, é possível confirmar a hipótese que o aumento de dados ausentes ao longo dos anos aumenta as dependências fantasmas e reduz a qualidade das dependências funcionais encontradas.

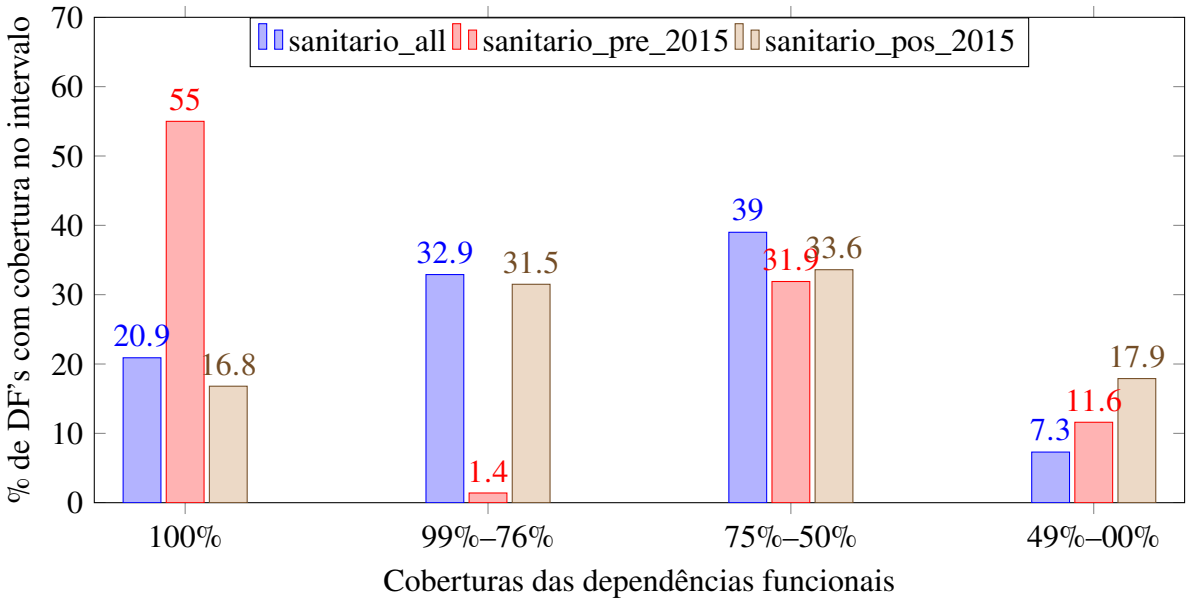


Figura 7.2: Coberturas das dependências funcionais relacionadas aos sanitários descobertas, dividido em intervalos.


Figura 7.3: Visualização do Metanome dos DFs descobertos pela variação **todos**


Figura 7.4: Visualização do Metanome dos DFs descobertos pela variação **pre-2015**


Figura 7.5: Visualização do Metanome dos DFs descobertos pela variação **pre-2015**

# 8 RELAÇÃO ENTRE SITUAÇÃO DA ESCOLA, ANO DO CENSO E DADOS AUSENTES DOS ATRIBUTOS SANITÁRIOS

Ao analisar a amostra variante **pos-2015** é possível reparar dois detalhes:

- Todos as tuplas com dados ausentes entre os atributos sanitários tem dados ausentes em todos os mesmos atributos. Não existe nenhuma tupla onde um dado ausente existe em apenas um atributo sanitário.
- Todas as tuplas com dados presentes contém o dado **1** no atributo *situacao_de_funcionamento*, enquanto tuplas onde esse dado apresentam **2**, **3** ou **4** contém dados ausentes em todos os atributos sanitários.

Essa análise comprova que o padrão entre a quantidade de dados ausentes de diferentes atributos é causado por um mecanismo de inserção de dado ausente MAR ("missing at random"). Neste caso, a inserção do dado ausente é diretamente relacionado com os atributos conhecidos *ano_censo* e *situacao_de_funcionamento*: caso *ano_censo* for **2015** ou maior e *situacao_de_funcionamento* for diferente de **1**, todos os atributos sanitários são dados ausentes.

Uma análise na amostra variante **pre-2015** demonstra que todos as tuplas que apresentam *situacao_de_funcionamento* diferente de "1"tem como dado de seus atributos sanitários o valor "false".

Com essas conclusões, novas análises foram realizadas para determinar o impacto da exclusão de tuplas com o valor de *situacao_de_funcionamento* diferente de "1"nas variantes anteriores. Outra análise foi realizada na amostra base dos sanitários, preenchendo todos os valores ausentes em tuplas onde *situacao_de_funcionamento* é diferente de "1".

- Variação **todos**: Todos as tuplas da amostra **todos**.
- Variação **todos-substituido**: Todos as tuplas da amostra **todos**, onde tuplas com dado de *situacao_de_funcionamento* diferente de 1 tem seus atributos sanitários ausentes inseridos por "false".
- Variação **pre 2015-situação**: Todos as tuplas da amostra **pre 2015** que tem como dado de seu atributo *ano_censo* um valor menor que 2015 e *situacao_de_funcionamento* igual a **1**.
- Variação **pos 2015-situação**: Todos as tuplas da amostra **pos 2015** quetem como dado de seu atributo *ano_censo* um valor igual ou maior que 2015 e *situacao_de_funcionamento* igual a **1**.

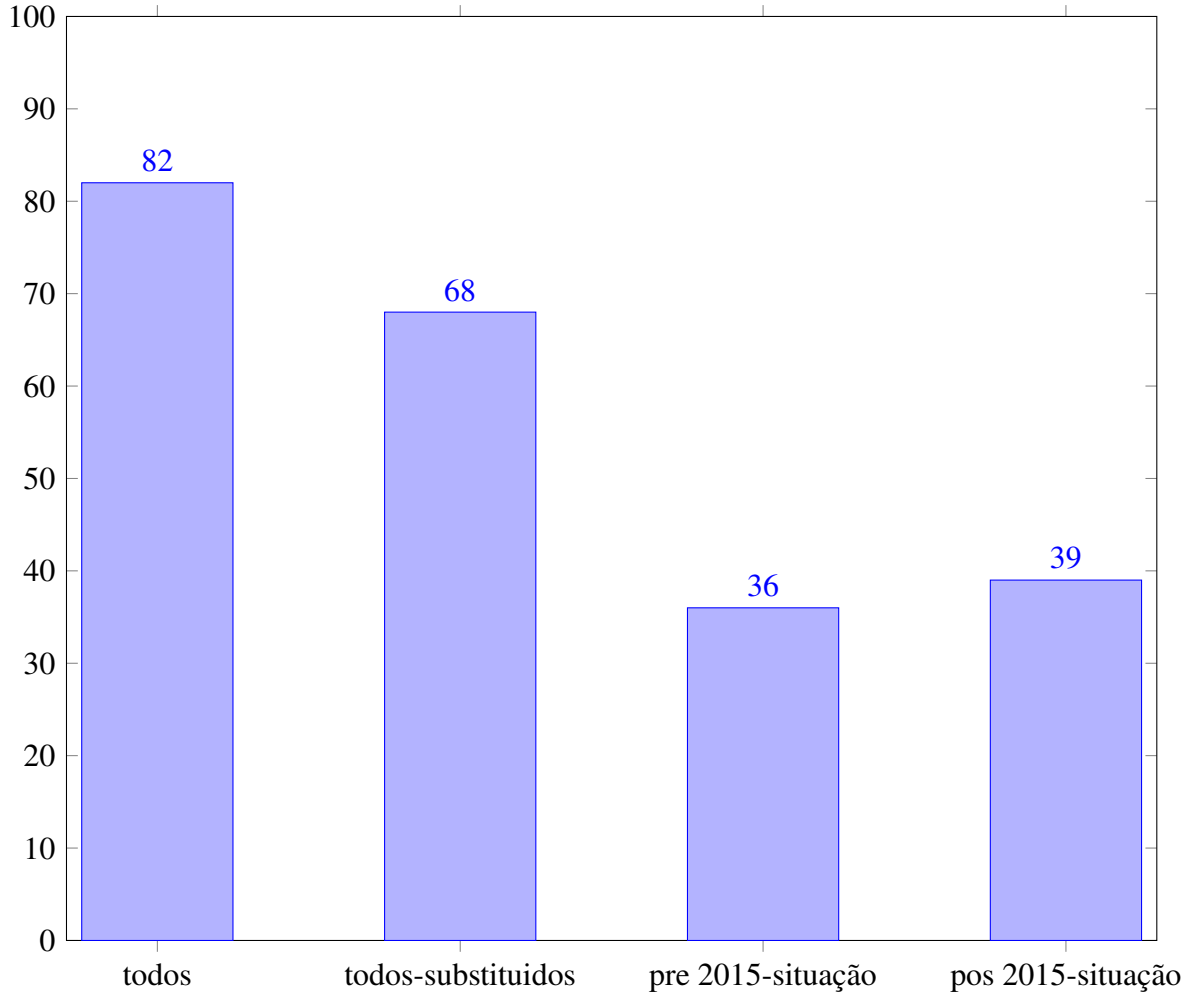


Figura 8.1: Número de dependência funcionais para cada caso apresentado nos atributos sanitários, onde o atributo *situacao_de_funcionamento* é 1.

Na figura 8.1, é possível observar a diferença entre o número de dependências funcionais encontrado entre as variantes **pre 2015-situação** e **pos 2015-situação** é muito menor em comparação a diferença entre as variantes **pre-2015** e **pos-2015** encontrada na análise do capítulo 7. O crescimento na quantidade de DF's considerando a *situacao_de_funcionamento* da tupla entre os períodos pre-2015 e pós-2015 é de 8%, uma quantiade significativamente menor que os 51% quando o atributo *situacao_de_funcionamento* não é considerado.

Essa análise comprova que a divergência entre a quantidade de dependências funcionais observadas na primeira análise dos atributos sanitários, causada pela mudança da forma de preencher dados nos anos 2015, se torna irrelevante quando as dependências são construídas apenas com tuplas com a mesma *situacao_de_funcionamento*.

A análise comprova que a substituição dos dados ausentes que não foram inseridos após 2015 colabora na diminuição de dependência funcionais fantasmas encontradas quando executada em cima da amostra de atributos sanitários **todos**.


Figura 8.2: Visualização do Metanome dos DFs descobertos pela variação **todos-situação**


Figura 8.3: Visualização do Metanome dos DFs descobertos pela variação **pre 2015-situação**


Figura 8.4: Visualização do Metanome dos DFs descobertos pela variação **pos 2015-situação**

# 9 CONCLUSÃO

## 9.1 HIPÓTESES

Através das análises neste trabalho foi possível confirmar as seguintes hipóteses:

- Atributos redundantes nas tabelas do banco do INEP levam ao aumento de dependências funcionais fantasmas encontradas pelo algoritmo Tane. Reunir os atributos reduz as dependências funcionais fantasmas encontradas.
- A separação das amostras ajuda a diminuir o quantidade de dependências funcionais fantasmas encontradas num algoritmo de descoberta de DF's quando há uma mudança na inserção de dados ao longo dos anos.
- A presença de dados ausentes numa tabela, causada por atributos redundantes e mudança na inserção de dados, piora a métrica de cobertura nas dependências funcionais encontradas pelo algoritmo Tane.
- Existe um mecanismo de inserção de dados nulos MAR ("missing at random") no banco de dados do INEP causado pela variação da representação de certos atributos de determinados tipos de escola o longo dos anos. Este mecanismo MAR leva a uma relação entre os dados dos atributos *situacao_de_funcionamento* e *ano_de_censo* a probabilidade de um determinado valor ser estar ausente. A aplicação do tratamento de substituição de dados ausentes consegue mitigar as dependências funcionais fantasmas criados por esses dados.

## 9.2 SUGESTÕES AO GERIR BANCO DE DADOS

Sugiro as seguintes intervenções para os administradores do banco de dados:

- A depreciação de certos atributos das tabelas do banco prejudicam imensamente a descoberta de dependências funcionais. Este problema tende a se agravar, pois a cada novo ano de censo novos dados nulos são inseridos nas tabelas. A existência desses atributos prejudica o trabalho de pesquisadores que queiram trabalhar em cima dos dados do banco, necessitando que precisem dividir os dados do censo pelos diferentes anos em que ele aconteceu.
- É necessário cuidado ao inserir um novo atributo para ver se ele não será redundante. Caso exista uma mudança na tipo do dado, é preferível que todos os dados de um mesmo sejam padronizados do que a criação de um novo atributo para o novo tipo de dado.

- Não alterar a lógica de inserção de dados ausentes ao longo dos anos. Caso seja necessário, retroativamente mude dados antigos para a nova lógica.
- Ser mais transparente no significado de cada um dos atributos nas tabelas. Como por exemplo, a *situacao_de_funcionamento* é especialmente vaga sendo que os valores são apenas números.

## 9.3 SUGESTÕES A PESQUISADORES

Sugiro as seguintes dicas de tratamento de dados para pesquisadores que queiram trabalhar com a descoberta de dependências funcionais neste banco de dados.

- Identifique atributos redundantes e unifique-os em apenas um só atributo.
- Caso queire utilizar as DF's para validar a inserção de dados futuros, desconsidere atributos depreciados.
- Caso queire utilizar as DF's para realizar limpeza de dados, encontre as DF's por selecionando apenas os atributos de um determinado tema e um conjunto de atributos base.
- Separe as escolas para a descobertas de DF's por suas diferentes *situacao_de_funcionamento*. Cada *situacao_de_funcionamento* apresenta dados ausentes em diferentes contextos, frequentemente mudando ao longo dos anos. Esses dados ausentes criam DF's fantasmas durante a execução do algoritmo de descoberta de DF.
- Caso um atributo comece a apresentar dados ausentes em uma tupla com uma certa *situacao_de_funcionamento* que não apresentava dados ausentes em anos anteriores, confira se a lógica da *situacao_de_funcionamento* implique que os atributos ausentes podem ser substituídos pelo valor "false".

# REFERÊNCIAS

Berti-Équille, L., Harmouch, H., Naumann, F., Novelli, N. e Thirumuruganathan, S. (2018). Discovery of genuine functional dependencies from relational data with missing values. *Proc. VLDB Endow.*, 11(8):880–892.

Chu, X., Ilyas, I. F. e Papotti, P. (2013). Discovering denial constraints. *Proc. VLDB Endow.*, 6(13):1498–1509.

Elmasri, R. e Shamkant, N. (1989). *Fundamentals of Database Systems*.

Huhtala, Y., Kärkkäinen, J., Porkka, P. e Toivonen, H. (1999). Tane: An efficient algorithm for discovering functional and approximate dependencies. *The Computer Journal*, 42(2):100–111.

Pena, E. H. M., de Almeida, E. C. e Naumann, F. (2019). Discovery of approximate (and exact) denial constraints. *Proc. VLDB Endow.*, 13(3):266–278.