Universidade Federal do Paraná
Setor de Ciências Exatas
Departamento de Estatística
Programa de Especialização em *Data Science* e *Big Data*

Wesley Thiago dos Reis

Avaliação da descoberta de Denial Constraints em bases de dados com diferentes datas de referência

Curitiba
2020

Wesley Thiago dos Reis

Avaliação da descoberta de Denial Constraints em bases de dados com diferentes datas de referência

Monografia apresentada ao Programa de Especialização em *Data Science* e *Big Data* da Universidade Federal do Paraná como requisito parcial para a obtenção do grau de especialista.

Orientador: Prof. Eduardo Cunha De Almeida

Curitiba
2020

Avaliação da descoberta de Denial Constraints em bases de dados com diferentes datas de referência

Evaluation of the discovery of Denial Constraints in databases with different reference dates

Wesley T. Reis[1], Eduardo Cunha De Almeida[2]

[1]Departamento de Estatística, Universidade Federal do Paraná (UFPR) Campus III - Centro Politécnico, Rua Evaristo F. F. da Costa, 418 Jardim das Americas, Curitiba, PR, Brasil*
[2]Departamento de Estatística, Universidade Federal do Paraná (UFPR) Campus III - Centro Politécnico, Rua Evaristo F. F. da Costa, 418 Jardim das Americas, Curitiba, PR, Brasil†

Este estudo tem o objetivo de avaliar como pode ser utilizada a descoberta automática de Denial Constraints em processos estruturados de análise de dados, assim como entender a viabilidade desse uso, de forma a garantir a qualidade das bases de dados. Utilizamos como insumo bases históricas para então identificar possíveis problemas futuros que possam acontecer, de forma a minimizar impactos em processos que irão consumir estes dados.
Palavras-chave: Qualidade de dados, Restrições de Negação, Consistência da informação, Restrições de Integridade

This study aims to evaluate how the automatic discovery of Denial Constraints can be used in structured data analysis processes, as well as to understand the feasibility of this use, in order to guarantee the quality of the databases. We use historical bases as input to then identify possible future problems that may happen, in order to minimize impacts on processes that will consume this data.
Keywords: Data Quality, Denial Constraints, Consistency of information, Integrity Constraints

1. Introdução

As informações obtidas através da análise e manipulação dos dados de negócio são consideradas de grande valor para qualquer grande organização atualmente. Hoje em dia o processo de análise e manipulação de dados não é mais um diferencial, mas sim uma necessidade na hora de uma tomada de decisão, permitindo uma maior probabilidade de sucesso do objetivo almejado. Porém, da mesma forma que uma informação de qualidade pode ajudar uma organização, existe a possibilidade de uma informação equivocada provocar grandes prejuízos.

Considerando esse fato, é extremamente importante que sejam estabelecidos processos de governança e garantia de qualidade em todo o fluxo da construção dessa informação, como na coleta junto à fonte dos dados, transmissões e integrações entre sistemas distintos, modelagem para tratamento dos tipos dos dados, enriquecimento da informação, aplicação de regras de negócio e garantia da segurança da informação dentre outras possíveis etapas.

Para identificação de possíveis problemas, são importantes que sejam aplicados filtros de validação de qualidade da informação da forma mais automática possível, buscando identificar possíveis problemas que necessitem de correção antes do uso daqueles dados, mitigando os riscos envolvidos nesse processo.

É comum que em bases de dados, existam problemas relacionados à qualidade da informação, estes problemas podem ser de diversos tipos diferentes, como por exemplo, valores vazios que deveriam estar preenchidos, registros duplicados, campos que deveriam receber somente valores pré-determinados, mas que foram preenchidos com valores fora do domínio ou formato estabelecido, ou ainda não obedecem a regras específicas do negócio pelo qual é utilizado.

Estes problemas podem ser ocasionados, por fatores humanos como erros ao inserir uma determinada informação em um sistema, ou até mesmo erros sistê-

*wesley.tr@hotmail.com
†eduardo@inf.ufpr.br

micos, ocasionado, por exemplo, por erros de programação, integrações entre diferentes sistemas, manipulação da informação, dentre outros.

Restrições de Integridade buscam garantir a consistência dos dados através de regras com o objetivo de que os dados reflitam de forma assertiva a realidade das informações no modelo de um banco de dados. Os objetivos das restrições de integridade buscam prevenir a redundância, garantir a coerência e completude da informação baseando se, por exemplo, na característica dos atributos ou na relação entre os atributos da base.

Dependências Funcionais [3] são dependências de dados que podem ser utilizadas nas definições de regras a serem respeitadas por um conjunto de dados e também chamadas de regras de negócio. As dependências funcionais são muito utilizadas atualmente nos bancos de dados para modelagem relacional. Porém existem regras comuns de negócio que não são cobertas utilizando Dependências Funcionais, como por exemplo, um cliente que não pode pagar um valor diferente de outro se o produto adquirido for exatamente o mesmo, ou ele não pode pagar uma taxa de juros diferente de outro se o período de atraso de um pagamento for exatamente o mesmo.

Escolhemos então para este estudo abordar as Denial Constraints (DC's), pois generalizam diversos tipos de restrições e são facilmente convertidas em código SQL que podem ser utilizadas facilmente na maior parte dos sistemas gerenciadores de banco de dados atuais.

Com base neste contexto, o objetivo deste estudo é entender a viabilidade da utilização de um algoritimo de descoberta automática de DC's para uma determinada base de dados com o intuito de garantir a qualidade dos dados. Utilizando o histórico dessa base como insumo para encontrar as DC's e mapear as de alta relevância que possam ajudar a identificar possíveis problemas de qualidade de dados em bases futuras. Considerando para isso safras de uma mesma base de dados que possuem informações de um período especifico (dia, mês, ano, ...).

O restante do artigo está dividido da seguinte forma: Seção 2 apresenta o que são as Denials Constraints, exemplifica o uso e mostra como uma DC pode ser transformada em código SQL. Seção 3 apresenta os algoritmos atuais mais avançados para descoberta de DC's. Seção 4 apresenta o caso de estudo que será abordado neste artigo. Seção 5 apresenta os resultados obtidas através do estudo realizado. Seção 6 apresenta a conclusão desse estudo.

2. O que são as Denial Constraints ?

Denial constraints (DC's) são expressões que buscam identificar problemas de integridade de dados relacionados a regras de negócio. As DC's encontram na base de dados analisada os registros que seriam os dados falhos ou dados potencialmente sujos, que não respeitam essas regras, que podem impactar de forma a causar erros em análises, construções de relatórios, utilização em modelos estatísticos ou qualquer outra finalidade para a qual aqueles dados serão utilizados.

Uma DC é uma expressão de negação constituída de um ou mais campos da base de dados realizando uma operação com estes mesmos campos em uma base espelho. Os campos possuem condições com eles mesmos através dos operadores {=, !=, <, <=, > ,>=}, onde para a realização da identificação dos registros com problemas é necessário realizar o cruzamento para cada um dos registros, gerando assim uma base com o produto cartesiano. A partir do resultado desse produto cartesiano verificamos as condições da DC para identificar quais são os registros que estão contidos naquela regra, caso existam, estes serão os registros considerados falhos e devem ser separados para posterior processo de limpeza de dados.

A Tabela 1 é utilizada para exemplificar o uso de uma DC para identificação de um dado sujo. Nessa tabela é possível identificar uma correlação entre o campo *{Produto}* e o *{Valor_boleto}*. Porém no registro onde o ID é igual a 6, essa correlação é quebrada, sendo este registro um possível caso de erro na base de dados. Nesse caso seria possível identifica-lo através da DC 1, ou seja no cruzamento de todos os registros dessa tabela com ela mesmo, onde o campo produto da primeira tabela for igual ao campo produto da segundo tabela, o Valor_boleto da primeira não pode ser menor que o Valor_boleto da segunda, caso isso aconteça, este registro, é identificado como problemático para esta DC:

DC 1: $(\neg[t0.produto = t1.produto \wedge t0.Valor\_boleto < t1.Valor\_boleto])$

As DC's são facilmente transformadas em códigos SQL, o que as torna praticas para utilização na maior parte dos bancos de dados utilizados atualmente. Para a construção da consulta através de código SQL, basta fazer a junção entre a base e ela mesma (ou auto-junção). Para todos os registros, sem a utilização de chaves para

Tabela 1: Exemplo de base de pagamentos de assinatura de TV e/ou internet.

ID	Nome	Produto	Valor_boleto
1	João Silva	TV	150,00
2	Maria Souza	TV	150,00
3	Jose Reis	Internet	100,00
4	Ana Cruz	Internet	100,00
5	Marcos Rios	TV+Internet	200,00
6	Jorge Lima	TV+Internet	150,00
7	Jaqueline Castro	TV+Internet	200,00
8	Jorge Lima	TV+Internet	200,00

este cruzamento, e aplicando em seguida as clausulas condicionais presentes na DC através do comando where, segue exemplo para a DC 1.

```
1 select distinct t0.*
2 from boletos as t0, boletos as t1
3 where t0.produto = t1.produto and
4 t0.valor_boleto < t1.valor_boleto;
```

Com a execução desta consulta em código SQL, temos o retorno mostrado na Tabela 2, ou seja, somente o registro com o possível erro identificado pela DC 1:

Tabela 2: Exemplo de registros com problema da base de pagamentos de assinatura de TV e/ou internet com relação a DC 1.

ID	Nome	Produto	Valor_boleto
6	Jorge Lima	TV+Internet	150,00

Existem ainda situações em que para a identificação dos registros problemáticos será necessário desconsiderar o próprio registro com relação a ele mesmo na tabela espelho na execução da consulta. Por exemplo, na Tabela 1 caso considerarmos que os campos Nome e Produto não podem conter valores duplicado, esta base teria um caso de erro para os registros de ID 6 e 8. Então para encontrar este problema teriamos a DC 2, porém neste caso para transformar essa DC para um código SQL, será necessário incluir mais uma condição, que sera o ID da tabela 1 deverá ser diferente do ID de sua tabelha espelho, conforme exemplo:

DC 2: $(\neg[t0.Nome = t1.Nome \wedge t0.Produto = t1.Produto])$

```
1 select distinct t0.*
2 from boletos as t0, boletos as t1
3 where t0.Nome = t1.Nome and
4 t0.Produto = t1.Produto and
5 t0.ID != t1.ID;
```

Neste caso o retorno para esse consulta seria a Tabela 3, que contêm os dois registros problemáticos com relação a DC 2:

Tabela 3: Exemplo de registros com problema da base de pagamentos de assinatura de TV e/ou internet com relação a DC 2.

ID	Nome	Produto	Valor_boleto
6	Jorge Lima	TV+Internet	150,00
8	Jorge Lima	TV+Internet	200,00

3. Algoritmos para descoberta de DC's

A descoberta de DC's requer além de conhecimento técnico das tecnologias utilizadas, também o conhecimento sobre a base de dados analisada, sendo uma tarefa que exige um alto esforço se realizada de forma manual, assim como está sujeita a erros devido ao risco operacional envolvido na realização da tarefa.

Com o objetivo de otimizar esta tarefa, foram desenvolvidos algoritmos que são utilizados para encontrar DC's de forma automática, utilizando os próprios dados da base de dados como insumo para identificação das DC's nela contidos, como o DCFinder [1] que faz a busca de DC's aproximadas e exatas, e o Hydra que faz a busca por DC's exatas [2].

O Hydra se apróxima do conjunto de DC's através de uma amostragem inteligente, a apartir daí busca de forma eficiente verificar e realizar correções para então obter um resultado completo e preciso. O Hydra evita dessa forma a comparação de todos os pares de registros para todo a base de dados análisas. Com relação ao tempo de execução do Hydra, este cresce apenas de forma linear com relação ao número de registros da base.

O DCFinder realiza em um primeiro passo a organização dos dados em uma estrutura chamada de *lista de posições de indíces* (PLI's), que basicamente agrupa as informações iguais de cada campo da base, a partir dai particiona os pares de registros com base em seus intervalos. O DCFinder prepara então as evidências

de acordo com seletividade das regras, e realiza o preenchimento das evidências com base nas relações da estrutura PLI. O DCFinder utiliza a distribuição das evidências para buscar e calcular as medidas de número de violações das DC's e a importância estatística que ele possui com base na abrangência na base de dados análisada. De acordo com testes realizados com o DCFinder, ele é mais rápido que todos os algoritmos para descoberta de DC's de forma apróximada construídos anteriormente a ele, e ainda em alguns casos sendo mais rápido que algoritmos de descoberta exata de DC's.

Porém, é importante ressaltar que mesmo para os algoritmos mais eficientes atualmente, ainda é um problema o custo computacional e de tempo de execução, principalmente em grandes bases de dados, e considerando a necessidade das organizações buscando cada vez mais o uso da informação no menor tempo possível. Portanto o objetivo desse estudo é entender como seria possível utilizar as informações obtidas através de um desses algoritmos em um histórico de uma base de dados, para identificarmos então possíveis problemas em bases futuras.

4. Caso de Estudo

Dentro das organizações, é comum haver processos estruturados que recebem informações dentro de uma determinada frequência, podendo variar como processos anuais, mensais, semanais, diários, esporádico, a cada hora, minuto, tempo real, dentre outros. Ou seja, essas bases possuem o mesmo layout, mudando apenas a data de referência da informação, mas mesmo para processos bem estruturados e automatizados, existe a possibilidade de serem disponibilizadas informações problemáticas.

Por exemplo para um processo automatizado que gera um relatório diáriamente que tem como um dos insumos uma informação que também é recebida diáriamente. Considere que em um determinando momento foi realizada uma manutenção no sistema origem e um dos campos dessa base passou a ser preenchido com valores vázios, neste caso, existe uma grande chance, ou é quase certo que o relatório final que utiliza esses dados como insumo poderá conter informações incorretas. Na Figura 1 é demonstrado um cenário de forma simplificada de um problema na base de dados ocasionado por uma manutenção do sistema origem da informação. Este Problema impactou no processo que consome estes dados e passou a gerar uma informação inválida ou incorreta.

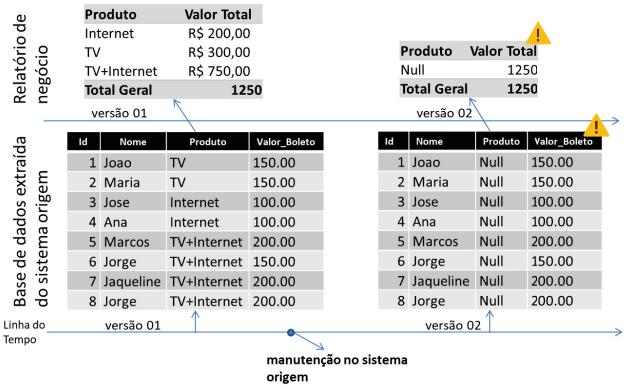

Figura 1: Fluxo simplificado que demonstra possíveis problemas nos dados e o impacto causado pela utilização sem a devida correção.

Por esse motivo existe a necessidade de aferir a qualidade desses dados. Para este estudo buscamos identificar e entender o comportamento do algoritmo de busca de DC's chamado DCFinder, e como identificar DC's de alta relevância para processos com esta característica. O objetivo é entender como isso pode ajudar na descoberta de possíveis problemas nos dados de forma eficiente e a se de fato seria possível aplicar em um processo real de forma a agregar valor ao mesmo mitigando possíveis erros.

5. Resultados

Para as execuções do DCFinder, as bases foram extraídas do repositorio e executadas em uma máquina com o sistema Operacional Windows 10, com processador Intel(R) Xeon(R) CPU E5-2650L v2 @ 1.70GHz 1.70GHz (2 processadores); 8 GB RAM, 100GB SSD, rodando com Java 1.8 with the JVM heap space limited to 4 GB.

Após as execuções, os resultados obtidos através do DCFinder foram transmitidos para o sistema SAS para realização dos tratamentos dos dados e processamento das informações para geração dos relatórios quantitativos. Este ambiente possui sistema operacional Linux Hat Rat, ~50 processadores Intel(R) Xeon(R) Gold 5120 CPU 2.20GHz, ~700 GB RAM, ~2Tb SSD.

Neste estudo foram então avaliadas duas bases de dados que são disponibilizadas diariamente dentro de uma grande corporação bancária para estudos analíticos relacionados a riscos. Cada base de dados possui uma amostra de 30 versões diárias em sequência ordenadas da mais antiga para a mais recente, e exe-

cutamos então o DCFinder para cada uma delas, e análisamos a relação entre os resultados de uma delas.

Como uma DC's pode conter inúmeros campos, e o DCFinder nem sempre gera uma DC' com os campos na mesma ordem, com o resultado do DCFinder foi necessário realizar o tratamento dos dados para a ordenação de cada uma das DC's geradas para então conseguir relacioná-las.

A primeira base de dados testada, chamada neste estudo de de BASE_EXEC1, foi testada para suas safras disponibilizadas no período de 21/05/2020 até 02/07/2020. Essa base de dados possuí 9 campos, e a média de 2767 registros na amostra testada. Para esta base de dados a quantidade média de DC's geradas pelo DCFinder foi de 24.5, e no total foram geradas 29 DC's únicas, onde dessas, 20 estavam presentes em todas as 30 bases da amostra. Neste caso houve uma estabilização das DC's encontradas em comum após a 11ª base, onde a partir dai as DC's mantidas permaneceram constantes. Mesmo identificando novas DC's, as 20 DC's encontradas nesse ponto que estavam presentes para em todas as bases anteriores se mantiveram no restantes da amostra, através do gráfico na Figura 2 pode ser observado esta evolução:

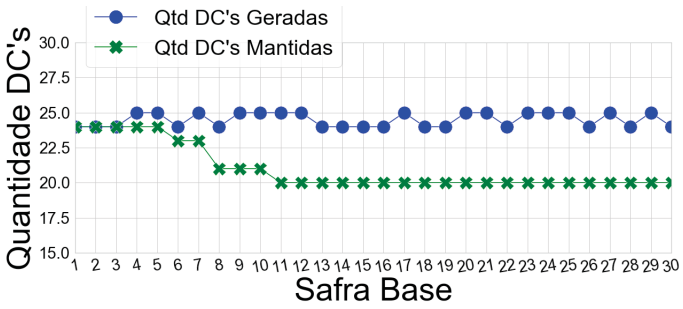

Figura 2: Gráfico que mostra a quantidade de DC's encontradas e quantidade de DC's que se mantiveram na linha do tempo para a base BASE_EXEC1.

A segunda base de dados testada, chamada neste estudo de BASE_EXEC2, foi testada para suas safras disponibilizadas no período de 26/05/2020 até 06/07/2020. Essa base de dados possuí 16 campos e a média de 9485 registros na amostra testada. Para esta base de dados a quantidade média de DC's geradas pelo DCFinder foi de 4568.8, e no total foram geradas 12496 DC's únicas, onde dessas, 1241 estavam presentes em todas as 30 bases da amostra. Neste caso não houve uma estabilização das DC's encontradas, porém a curva da quantidade de DC's mantidas foi mais acentuada no inicio entre a 1ª safra e a 9ª safra, se mantendo em queda a partir desse ponto porém de forma mais suave, através do gráfico na Figura 3 pode ser observado esta evolução:

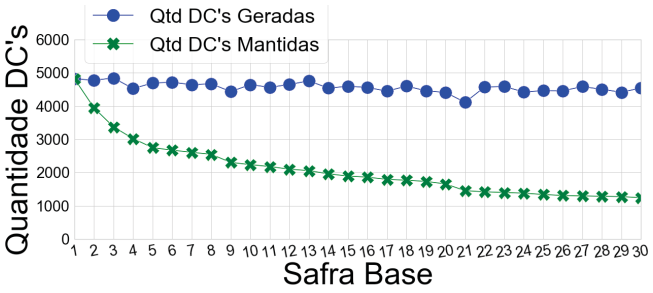

Figura 3: Gráfico que mostra a quantidade de DC's encontradas e quantidade de DC's que se mantiveram na linha do tempo para a base BASE_EXEC2.

A partir desse experimento é possível entender, que a quantidade de DC's em comum para as safras estudadas tende a diminuir e chegar a uma constante. Podendo então indicar que essas DC's que se mantem para todo o histórico até a estabilização, são relevantes para o negócio. A quantidade de bases necessárias para essa estabilização aparentemente esta diretamente relacionada à quantidade de campos e registros que a base possui, ou ainda a complexidade das regras de negócio presentes na base.

Por exemplo, na base BASE_EXEC1 que possuía uma quantidade menor de campos e registros, a estabilização ocorreu com um histórico pequeno de bases. Na BASE_EXEC2 que possuia um numero maior de campos e também de registros, não foi possível chegar a essa estabilização com o histórico avaliado. Neste caso seria necessário um histórico maior para que pudéssemos identificar de fato as DC's mais importantes.

Para entender a relevância dessas DC's para a base BASE_EXEC1, foram então criadas sintéticamente 5 bases de dados sujas ou com problemas para então executar o algoritmos DCFinder nelas e entender se essas DC's se mantinham. As caractericas dessas bases sujas foram as seguintes, na 1ª base de dados suja, foi removido um tipo especifico de produto da base, onde um determinado campo tem sempre o mesmo valor, removendo cerca de 20% de registros da base. Na 2ª base de dados suja, foi alterado um campo númerico para valores errados, cópiados de outro campo número rico da mesma base. Na 3ª base de dados suja, foram alterados 2 campos de formato string para somente registros vazios/missing. Na 4ª base dados suja, foi alterado um campo númerico e um campo string para conter somente registros vazios/missing, E por fim na 5ª base de dados suja, a base foi alterada deixando so-

mente 50% dos registros da base, removendo a metade final da base, tentando simular um possível corte no arquivo por erro de transmissão.

Foi então executado o DCFinder para estas bases sujas, e como resultado desta execução, nenhuma delas registrou todas as 20 DC's que se mantiveram em comum na amostra análisado para esta base. No melhor dos casos, 14 DC's estavam presentes e no pior dos casos apenas 6 estiveram presentes, o que corrobora com a tese de que as DC's em comum podem ser importante na identificação das bases que possuem problemas, ou seja a falta delas em bases que deveriam contê-las é um indicativo de possível problema, na Tabela 4 pode ser observado estes resultados.

Tabela 4: Resultados obtidos no experimento das bases sujas.

ID	Qtd DC's encontradas	Qtd DC's em comum
1	16	7
2	19	14
3	8	6
4	12	9
5	46	10

Para a base BASE_EXEC1 foi testado também, com duas outras bases adicionais, uma base apendada com todos as bases da amostra e outra base com o append das 12 primeiras bases da amostra, e para ambas estas execuções, em cada uma delas foram identificadas 25 DC's, e todas as 20 DC's que se mantiveram em comum nos testes da amostra inicial de 30 bases estiveram presentes também nessas novas bases testadas.

Para a base BASE_EXEC2 não foi possível executar o teste para a base apendada, devido erro de memória: OutOfMemory.

Com o objetivo de entender se as DC's em comum poderiam ser utilizadas como regras de validação para identificação de registros sujos, antes da execução de um determinado processo que consume estas informações, foram criados os códigos SQL e executados em algumas safras da base_exec1. Porém, foram marcados como sujos uma grande quantidade registros corretos, para algumas regras chegou a quase 100% de registros da base idenificados como sujo, o que dificulta muito essa utilização para as bases testadas, devido ao esforço para avaliar estes registros posteriormente.

Também foram testados no DCFinder a execução de bases com uma quantidade maior de registros e de quantidade de campos. Por exemplo, bases com 90 campos e 50.000 registros, base com 41 campo e 21.000 registros, base com 70 campos e 4.000 registros, e base com 28 campos e 15.000, retornando sempre com erro na execução 'OutOfMemory', entendendo assim que seria necessário um ambiente mais robusto, ou até mesmo a implementação em outros tipos de tecnologias para viabilizar estes tipos de execução.

6. Conclusão

A partir desse estudo pode ser entendido que é possível utilizar algoritmos de descoberta de Denial Constraints utilizando as bases históricas para mapear DC's de alta relevância para um processo. Observamos também que é possível utilizar a descoberta das DC's que permanecem em todo o histórico para identificar possíveis problemas nos dados, ou seja quando essas DC's não aparecem, é grande a possíbilidade de haver erros nos dados. Podendo então utilizar essas DC's para identificar problemas de qualidade de dados e assim gerar ganhos para o negócio. Por exemplo evitar um processamento daquele dado, o que geraria custo computacional e de tempo, para somente após o processamento identificar que houve um problema, ou pior ainda, tomar alguma decisão com base em dados incorretos.

A execução das DC's encontradas não se mostrou eficiente para identificação de registros sujos de forma a garantir a integridade dos dados, ou seja para criação de restrições de validação a partir das próprias DC's, pois em todos os casos testados foram identificados uma quantidade elevada de registros corretos, sendo marcados como sujos pelas DC's identificadas como relevantes, o que ocasiona um alto indice de falsos positivos.

A execução do DCFinder em bases com grande quantidade de registros ou grande quantidade de campos pode ser dificil de ser viabilizado devido ao alto uso de memória e tempo de execução.

Agradecimentos

W.T.R. gostaria de agradecer ao departamento de Estatística da Universidade Federal do Paraná - UFPR.

Referências

[1] Eduardo H M Pena, Eduardo Cunha De Almeida, Felix Naumann, *Discovery of approximate (and exact) denial constraints*, (2019)

[2] Chu, X., Ilyas, I. F., Papotti, P., *Discovering denial constraints*, (2013)
[3] Beeri, C., Dowd, M., Fagin, R., and Statman, R., *On the structure of Armstrong relations for functional dependencies*, (1984)