UMA FERRAMENTA PARA VALIDAÇÃO E VISUALIZAÇÃO DE RESTRIÇÕES DE INTEGRIDADE[1]

Fabiano Schaefer[2]
Alexandre Tagliari Lazzaretti[3]
Roberto Wiest[4]

RESUMO

Dentre as principais finalidades de um banco de dados, destaca-se a garantia de que os dados que estão armazenados estejam íntegros, ou seja, que eles representem a realidade modelada. Para garantir a consistência dos dados, os SGBDs utilizam o conceito de restrições de integridade, as quais são regras que são testadas toda vez que existe uma modificação nos dados, e indicam se os dados são válidos ou não. No entanto, existem restrições de integridade que são dinâmicas, pois dependem do conjunto de dados que se deseja avaliar. Por exemplo, deseja-se verificar se existem datas faltantes em um conjunto de dados diários. Neste sentido, este trabalho apresenta uma ferramenta *web* que auxilia na criação de restrições de integridade dinamicamente, permitindo sua visualização por meio de gráficos interativos. Afim de se testar as funcionalidades do sistema, foi realizado um estudo de caso com base nos dados climáticos observados no município de Itapiranga/SC.

Palavras-chave: Regras dinâmicas, Banco de dados, Consistência.

1 INTRODUÇÃO

Em banco de dados uma das principais finalidades é a busca pela manutenção da integridade dos seus dados. De acordo com Lazzaretti (2017), para garantir a consistência dos dados os SGBDs (Sistemas de Gerência de Banco de Dados) utilizam o conceito de restrições de integridade, as quais são regras que são testadas e indicam se os dados são válidos ou não. Bancos de dados com grandes

---

[1] Trabalho de Conclusão de Curso (TCC) apresentado ao Curso de Tecnologia em Sistemas para Internet do Instituto Federal Sul-rio-grandense, Câmpus Passo Fundo, como requisito parcial para a obtenção do título de Tecnólogo em Sistemas para Internet, na cidade de Passo Fundo, em 2018.

[2] Aluno do curso de Tecnologia em Sistemas para Internet no Instituto Federal de Educação, Ciência e Tecnologia Sul-Rio-Grandense de Passo Fundo (IFSul). E-mail: schaefer.fabiano@gmail.com.

[3] Orientador, professor no Instituto Federal de Educação, Ciência e Tecnologia Sul-Rio-Grandense de Passo Fundo (IFSul). E-mail: alexandre.lazzaretti@passofundo.ifsul.edu.br.

[4] Coorientador, professor no Instituto Federal de Educação, Ciência e Tecnologia Sul-Rio-Grandense de Passo Fundo (IFSul). E-mail: roberto.wiest@passofundo.ifsul.edu.br.

quantidades de registros, podem ter dificuldade em identificar se os dados estão íntegros ou sem a ocorrência de falhas.

O desenvolvimento de uma ferramenta que realize a validação dos dados dinamicamente por meio de restrições de integridade, possibilita realizar análises no conjunto de dados buscando identificar possíveis falhas e/ou inconsistências. Desta forma, torna-se possível fazer correções no conjunto de dados de maneira que estes possam ser utilizados com a certeza que estão consistentes e com maior segurança.

Com a finalidade de auxiliar agentes na tomada de decisão com base em dados armazenados em um banco de dados, este trabalho apresenta uma plataforma que possibilita a criação de regras de integridade de domínio de forma dinâmica, ou seja, o usuário pode montar a regra de integridade que deseja testar, e estas são verificadas e validadas em uma base de dados existente.

2 REFERENCIAL TEÓRICO

Neste capítulo, são apresentados os principais conceitos e tecnologias, baseados em pesquisas bibliográficas realizadas, que foram utilizados para o desenvolvimento deste trabalho.

2.1 RESTRIÇÕES DE INTEGRIDADE

A seção a seguir apresenta o estudo realizado acerca das restrições de integridade existentes em bancos de dados relacionais, seus conceitos, classificações e como são utilizadas.

2.1.1 Conceitos

O termo "integridade" refere-se à *precisão* ou *correção* de dados no banco de dados. As restrições de integridade resguardam o banco de dados contra danos acidentais, garantindo que mudanças feitas por *usuários autorizados* não resultem em perda da consistência de dados (DATE, 2003, SILBERSCHATZ *et al*., 2012).

Quando uma restrição de integridade de um banco de dados não é assegurada, têm-se valores indesejados, desconhecidos ou nulos e relacionamentos perdidos ou incorretos. No momento em que uma nova restrição de integridade é declarada, o banco de dados deve efetuar uma validação e aceitar ou não a restrição de integridade (LAZZARETTI, 2017). Se a restrição de integridade for aceita, passará a ser verificada pelo SGBD toda vez que uma modificação é feita nos dados aos quais ela remete.

Existe uma ampla variedade de restrições de integridade em bancos de dados relacionais. Lazzaretti (2017) classifica-as como: restrições de domínio, restrições de chaves, restrições de integridade referencial, restrições quanto ao momento de verificação e, por fim, restrições baseadas em eventos.

2.1.1.1 Restrições de domínio

Segundo Lazzaretti (2017), restrições de domínio são as mais elementares formas de restrições de integridade, sendo verificadas toda vez que um item é incorporado ou modificado no banco de dados. Estas restrições verificam os valores inseridos ou modificados no banco de dados, além de realizar testes de consulta a fim de garantir que comparações feitas tenham sentido. As restrições de domínio podem ser classificadas da seguinte forma:

- → Restrições de atributo: definem os valores válidos que um atributo pode vir a assumir;
- → Restrições de tipo: estabelecem o tipo de um atributo;
- → Restrições de tupla: é uma restrição sobre uma única tupla, podendo agregar restrições sobre vários atributos da mesma. Normalmente executadas após uma instrução que possa fazer com que sejam violadas;
- → Restrições de banco de dados: envolve duas ou mais tuplas distintas. Neste caso, todas as restrições de tupla relacionadas deverão ser atendidas ou então a operação não será efetuada;
- → Restrições de transição de estado: consideram os estados corretos dos valores dos atributos de um banco de dados, assegurando que a mudança de um estado correto para outro seja válido.

2.1.1.2 Restrições de chaves

Para DATE (2003), o modelo relacional sempre enfatizou o conceito de chaves, embora seja na realidade apenas um caso especial de restrições. Chave é o conceito básico para identificar linhas e estabelecer as relações em bancos de dados relacionais (HEUSER, 2009). As restrições de chaves são classificadas de três formas, são elas:

- → Chaves candidatas: seja X um conjunto de atributos de uma tupla Y. X é uma chave candidata se obedecer às propriedades de unicidade e irredutibilidade (LAZZARETTI, 2005);
- → Chaves primárias ou alternativas: no caso de existir mais de uma chave candidata, o modelo relacional exige que, uma dessas chaves seja definida como chave primária. As outras são chamadas, automaticamente, de chaves "alternativas";
- → Chaves estrangeiras: é um conjunto de atributos de uma tupla cujos valores devem, obrigatoriamente, corresponder a valores de alguma chave primária de uma outra ou de uma mesma tupla.

2.1.1.3 Restrições de integridade referencial

As regras de integridade referencial garantem que o banco de dados não inclua valores inválidos ou inexistentes de chave estrangeira (LAZZARETTI, 2017). Sendo assim, uma chave estrangeira de um relacionamento deve coincidir com a chave primária a qual faz referência, em outras palavras, se B faz relação a A, então A deve existir.

2.1.1.4 Restrições quanto ao momento de verificação

Restrições quanto ao momento de verificação podem ser subdivididas em duas categorias de verificação:

- → Imediata: a verificação é feita no momento da ocorrência de uma operação, ou seja, imediatamente;
- → Postergada: as restrições são verificadas no momento que uma instrução COMMIT (indica o término de uma operação bem-sucedida, onde todas as

operações feitas são gravadas) é executada ou em algum momento posterior a execução da operação, sendo esse momento definido pelo usuário.

2.1.1.5 Restrições baseadas em eventos

Segundo Lazzaretti (2017), restrições de integridade baseadas em eventos podem ser programadas pelo utilizador, e sua verificação é independente da execução de operações de atualização no banco de dados. Neste caso, as restrições são verificadas através de uma chamada da aplicação.

2.2 TECNOLOGIAS

A seguir são apresentadas as principais tecnologias utilizadas para o desenvolvimento da plataforma.

2.2.1 PostgreSQL

De acordo com ELMASRI & NAVATHE (2011, p. 3), um sistema gerenciador de banco de dados (SGBD) é uma coleção de programas que permite aos usuários criar e manter um banco de dados. O SGBD é um sistema de *software* de uso geral que visa facilitar o processo de definição, construção, manipulação e compartilhamento de bancos de dados entre diversos usuários e aplicações.

O PostgreSQL é um sistema de gerenciamento de banco de dados objeto-relacional de código fonte aberto. Atualmente, de acordo com a página oficial, o PostgreSQL está em conformidade com a norma ANSI-SQL:2008, abrangendo a maior parte dos tipos de dados SQL-2008, além de suportar o armazenamento de objetos binários grades, incluindo imagens, sons ou vídeos.

O modelo objeto-relacional surgiu como uma forma de estender os SGBDs relacionais com características presentes em SGBDs orientados a objeto (ELMASRI & NAVATHE, 2011). Uma dessas características é o mecanismo de tipos definidos

pelo usuário (UDTs – *User-Defined Types*) que permitem a criação de objetos estruturados complexos

2.2.2 PHP

De acordo com o site oficial, o PHP: *Hypertext Preprocessor*, é uma linguagem de script de código aberto, interpretada, amplamente utilizada para aplicações no lado do servidor e desenvolvimento de páginas *Web*, podendo ser mesclada dentro do código HTML. Apesar de permitir a mesclagem no HTML, o código PHP é executado no lado do servidor, gerando o HTML que é enviado ao navegador (PHP, 2017). É uma linguagem extremamente simples para um iniciante, porém fornece recursos avançados para programadores profissionais.

Uma das características mais fortes e significativas do PHP é seu suporte a uma ampla variedade de banco de dados. Escrever uma página *Web* consultando um banco de dados é relativamente simples usando uma das extensões específicas de um banco de dados, ou usando uma camada de abstração (PHP, 2017). Além disso, é suportado pela maioria dos sistemas operacionais e servidores *Web* atualmente disponíveis.

2.2.3 Linguagem R e pacote Shiny

R é um sistema para computação estatística e gráficos. Ele fornece, por exemplo, uma linguagem de programação, gráficos de alto nível, interfaces para outras linguagens e facilidades de depuração (R Core Team, 2016). A linguagem R também permite conexão com bancos de dados e leitura de arquivos, como por exemplo a leitura de planilhas (.csv) e de arquivos texto (.txt).

A linguagem R foi criada em 1996 pelo neozelandês Ross Ihaka e pelo canadense Robert Gentleman, inicialmente como um projeto de pesquisa. Foi baseada em duas outras linguagens, sendo elas a linguagem S e Scheme (MELLO et al., 2013). A linguagem e o ambiente S foram desenvolvidas na Bell Laboratories (anteriormente AT&T, agora Lucent Technologies) por John Chambers e colegas. A linguagem R pode ser considerada como uma implementação diferente da S.

Existem algumas diferenças importantes, mas muito código escrito para S é executado inalterado no R (R, 2017). A linguagem R encontra-se disponível sob licença GNU (*General Public License*).

O Shiny é um *framework* de aplicação *Web* para o R. Transforma suas análises em aplicações *Web* interativas, não havendo a necessidade de um conhecimento prévio nas tecnologias HTML (*HyperText Markup Language*), CSS (*Cascading Style Sheets*) ou JavaScript. Permite à programadores R transformarem sua análise em aplicativos *Web* interativos e acessíveis por todos nos navegadores (RADU *et al.*, 2014).

2.2.4 Tecnologias de Interface

Segundo Lamin (2014) em desenvolvimento *Web*, *front-end* refere-se a etapa inicial de coleta de dados do usuário, realizando o processamento dos mesmos a fim de adequá-los às especificações de utilização do *back-end* (etapa final). Para o desenvolvimento do *front-end* da aplicação *Web*, foram utilizadas as tecnologias descritas a seguir.

HTML é uma linguagem de marcação com uma sintaxe específica que fornece instruções ao navegador sobre como exibir uma página (Tableless, 2017). Distingue e separa o conteúdo (palavras, imagens, vídeo, etc.) da forma como o mesmo é apresentado (MDN, 2017). Atualmente, encontra-se em sua versão 5.

CSS é uma linguagem para definir a apresentação (aparência) de documentos que adotam para o seu desenvolvimento linguagens de marcação (como o HTML). O CSS irá definir como serão expostos os elementos contidos no código de um documento, destacando-se a separação entre o formato e o conteúdo de um documento (Tableless, 2017).

O JavaScript é uma linguagem de programação dinâmica, normalmente utilizada como parte de páginas *Web*, cujas implementações permitem que o script do lado do cliente interaja com o usuário e crie páginas dinâmicas. É uma linguagem de programação *interpretada*, com recursos orientados a objetos (Tutorials Point, 2017). Dentre inúmeros efeitos possíveis, pode-se citar como exemplo um *Slider* de imagens. Toda a movimentação, ações de cliques nas setas dentre outras, fica ao encargo do JavaScript.

Criada e lançada por John Resig em 2006, o JQuery é uma biblioteca JavaScript de código aberto criada para simplificar a criação de scripts em páginas HTML (JQuery, 2017). Dentre as suas funcionalidades, destaca-se a sua utilização na redução de código, possibilidade de utilizar plugins desenvolvidos pela comunidade, além de descomplicar chamadas AJAX e manipulações DOM (W3C School, 2017, JQuery - Tutorial, 2017).

Desenvolvido em meados de 2010 e lançado em agosto de 2011, o Bootstrap é um *framework* de código aberto utilizado no desenvolvimento de aplicações responsivas através de HTML, CSS e JavaScript (Bootstrap, 2017). Dentre suas inúmeras características, destaca-se a agilidade e facilidade com que uma página responsiva pode ser concebida, além da baixa curva de aprendizagem necessária para utilização do *framework*.

3 RESULTADOS

A seguir são apresentados o estudo de caso e os resultados obtidos neste trabalho.

3.1 Estudo de caso

Para o estudo de caso foram utilizados os dados climáticos observados no município de Itapiranga/SC. O registro climático de Itapiranga passou a ser efetuado a partir de 1935 por Bruno Lengert. Após seu falecimento, as anotações diárias passaram a ser realizadas pelo seu filho Wolfgang J. Lengert, perdurando até os dias atuais (Força d'Oeste, 2014).

Como resultado, tem-se mais de 80 anos de dados climáticos diários, armazenados num livro de registros. Recentemente, estes dados foram transferidos para planilhas eletrônicas. A Tabela 1 apresenta a forma como estes dados estão estruturados:

Tabela 1 – Anotações diárias mensais – janeiro/2010.

DAT	MIN	MAX	MED	VENTO	TEMPO	HYGRO	BAROM	CHUVA
1	21,1	35,2	28,2	W-o no no no n no	Ns S S w S Ws w S	96-45	1006-999	
2	22,3	36,9	29,3	Nw- so o o nw os n	n n S SSS w S w S w SS	95-40	1004-1004	
3	23,9	38,7	31,4	n n n n n n n n n n	n S S S Sw Sws s W Ws	94-49	999-1006	
4	24,4	36,4	30,4	nw nw nw nwnw w	N S S S w Ss w SWwrr	95-55	1004-1002	3 mm
5	24,3	37,2	30,8	N nw nw nw nw n	n S S Sws sw SWrr	94-51	1002-995	

Fonte: Wolfgang Lengert[5].

Para obtenção das medições de temperatura mínima (MIN) e máxima (MAX), é utilizado um termômetro digital que armazena a menor e a maior temperatura do dia, após é realizado o cálculo da média (MED) da temperatura diária. As variáveis VENTO e TEMPO são observadas durante o dia, sem auxílio de aparelhos, onde a direção do vento é estabelecida de acordo com a direção das nuvens e o tempo é determinado de acordo com a quantidade de nebulosidade.

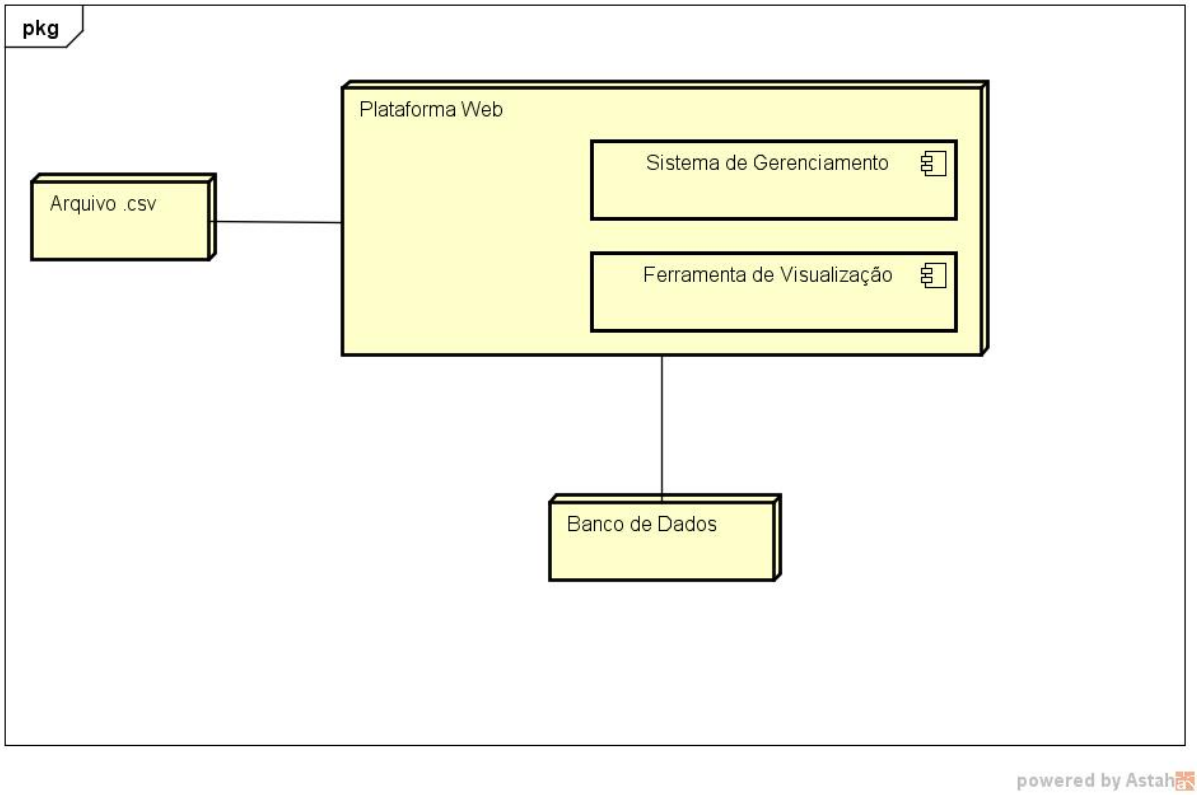
A indicação da umidade do ar (HYGRO) e da pressão barométrica (BAROM) é realizada pelo mesmo equipamento que faz a medição da temperatura. Por fim, em caso de precipitação (CHUVA), a medição é feita por um pluviômetro.

3.2 Proposta

[5] Dados disponibilizados por Wolfgang Lengert para a construção do estudo de caso. Não possui fontes online.

Com base no estudo de caso, foi construída uma ferramenta que possibilita a importação dos dados através de um arquivo .csv, e após os insere no banco de dados genérico como demonstrado na Figura 1:

Figura 1 – Estrutura da Plataforma



Fonte: do Autor.

Depois da inserção, é possível selecionar os dados para visualização e, após, determinar quais as restrições de domínio deseja-se avaliar. Por exemplo, ao definir uma restrição para o campo TEMP_MAX (temperatura máxima registrada no dia), onde é definido o valor mínimo como sendo -15º Celsius e o valor máximo +45º Celsius, o usuário estará limitando os valores da coluna TEMP_MAX a regra de integridade definida.

Após criadas todas as restrições de integridade de acordo com as demandas do usuário, é possível verificar as mesmas na ferramenta de visualização. Para tanto, deve-se escolher a variável que se deseja avaliar e o período de tempo (se houver).

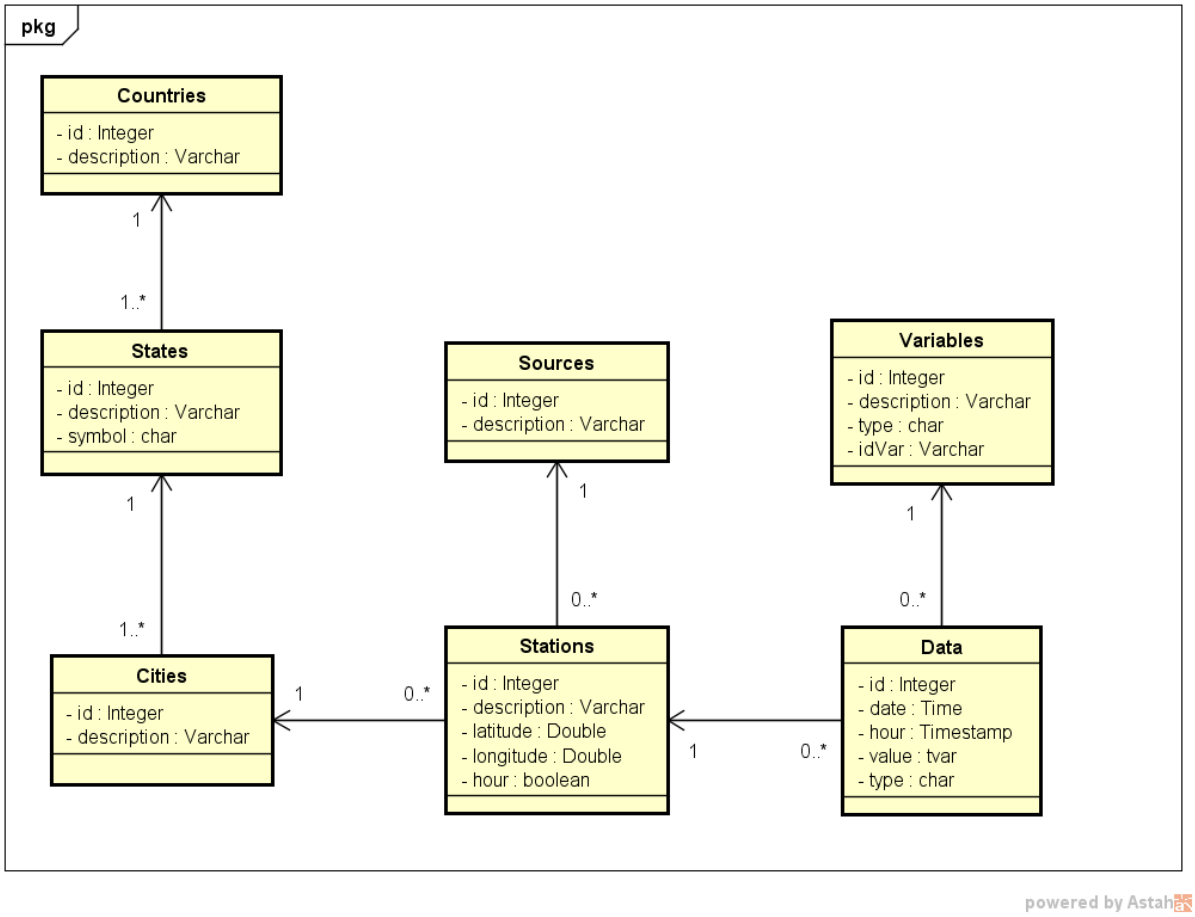
Todos os registros que não respeitarem a regra de integridade de domínio que foi imposta pelo usuário serão mostrados na ferramenta. Como exemplo, se para a variável PRECIPITACAO, a restrição de integridade imposta indica que a

variável não deve permitir valores negativos ou maiores que 500, todo o registro que não respeitar a regra de integridade e estiver no intervalo definido pelo usuário será mostrado na ferramenta.

3.3 Banco de Dados

Para armazenar os dados submetidos a plataforma para posterior visualização, foi construído um banco de dados conforme a modelagem apresentada no diagrama de classes descrito na Figura 2:

Figura 2 – Diagrama de classes da ferramenta



Fonte: do Autor.

As classes *Countries*, *States* e *Cities* são responsáveis pelo armazenamento dos países, estados e cidades respectivamente. A classe *Sources* é responsável por armazenar a fonte/organização que realizou a coleta dos dados submetidos a plataforma. Uma organização pode possuir mais de uma estação, as quais são

representadas na classe *Stations* (para o estudo de caso a fonte foi definida como “Wolfgang Lengert” e a estação “Linha Becker”). A tabela *Variables* é uma abstração das colunas do arquivo submetido, e a tabela *Data* armazena os dados diários da coluna em si.

Para a modelagem do banco de dados, utilizou-se o paradigma objeto-relacional. Na implementação do banco de dados, foi definido um UDT chamado “tvar” como demonstrado na Figura 3.

Figura 3 – Definição de tipo tvar

```
65  create type tvar as (
66      tvar_integer integer,
67      tvar_double double precision,
68      tvar_char char,
69      tvar_time time,
70      tvar_text text
71  );
```

Fonte: do Autor.

Este UDT foi criado para armazenar dados do tipo inteiro, ponto flutuante, caractere, texto ou datas. O campo *value* da tabela *Data* utiliza este UDT para armazenar o dado recebido do arquivo submetido a plataforma. Por exemplo, se o valor da coluna no arquivo for 23.7, o tipo do dado assumido pela coluna *value* será tvar_double (ponto flutuante).

3.4 Importação dos dados

Para realizar a análise dos dados através das regras de integridade dinâmicas, faz-se necessário, que estes estejam armazenados no banco de dados desenvolvido. Desta forma, foi desenvolvida uma funcionalidade para importação de dados. Atualmente, a ferramenta possibilita a importação de arquivos com a extensão .csv (*Comma-Separated Values*). Para tanto, foi definida uma estrutura padronizada de importação (Figura 4).

Figura 4 – Estrutura de arquivo .csv

Janeiro.csv

```
1  DAT;MIN;MAX;MED;VENTO;TEMPO;HYGRMT;BRMTR;PRECPT
2  01-01-2010;21,1;35,2;28,2;W-o no n o no n no  ;ns S S w S w S Ws w S;96-45;1006-999;Pp,0,5m
3  02-01-2010;22,3;36,9;29,3;Nw- so o o  nw os n;n  n S SSS w S w S w SS;95-40;1004-1004;
4  03-01-2010;23,9;38,7;31,4;n  n  n  n  n n n n n  ;n S S S SwSWs s W Ws      ;94-49      ; 999-1006;
5  04-01-2010;24,4;36,4;30,4;nw nw nw nwnw w; n S S S w Ss w SWwrr;95-55;1004-1002;3 mm
```

Fonte: do Autor.

A submissão do arquivo é feita por meio de um formulário e os dados são armazenados em uma tabela temporária. Após, é disparada uma função no banco de dados que realiza a inserção dos registros. Para cada coluna do arquivo csv, a função realiza uma inserção na tabela *Data*, armazenando o valor e a data em que o registro foi observado. Por exemplo, no mês de janeiro de 2010, a função irá ler o cabeçalho e quando a coluna estiver na posição 2, fará a inserção dos registros de temperatura mínima na tabela. Na Figura 5 é descrito o trecho da função responsável por esta parte.

Figura 5 – Trecho da função insert_csv_values()

```
121     -- Seleciona a coluna MIN do cabeçalho
122     v_column = split_part(v_header, ';', 2);
123
124     -- Compara a coluna MIN do cabeçalho com a coluna idVar da tabela variables
125     select type, id
126     from variables
127     where idVar = v_column
128     into v_type, v_variable;
129
130     -- Insere os dados de data e temperatura MÍNIMA na tabela DATA
131     insert into data (date, value.tvar_double, type, station, variable)
132     select cast(split_part(description, ';', 1) as date),
133            cast(replace(split_part(description, ';', 2), ',', '.') as double precision),
134            v_type,
135            station,
136            v_variable
137     from temp_table
138     where split_part(description, ';', 1) <> 'DAT';
```

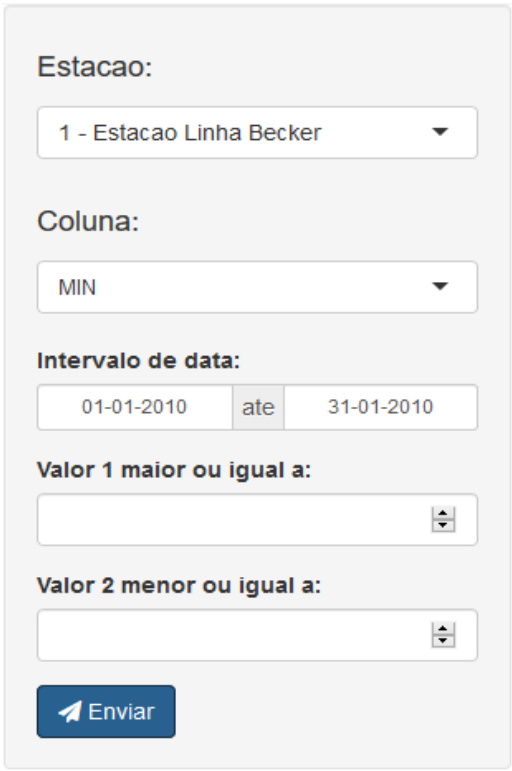
Fonte: do Autor.

A variável v_column (linha 121) recebe o valor presente na segunda coluna do cabeçalho do arquivo, que como pode se observar na Figura 4 corresponde ao valor MIN. Entre as linhas 125 e 128 são selecionados *type* e *id* da tabela *variables* que são armazenados nas variáveis v_type e v_variable respectivamente, respeitando a comparação que é feita entre *idVar*(campo da tabela *variables*) e v_column. Por último é feita a inserção dos dados na tabela *data* (entre as linhas 131 e 138), onde é informado a data do registro, valor, tipo, a qual estação pertence e variável. Pelo fato das informações estarem em texto no arquivo .csv, é necessário realizar a conversão de alguns dados para seus respectivos tipos no banco de dados como mostrado nas linhas 132 e 133.

3.5 Visualização dos Dados

Após a importação dos dados no banco de dados, é possível fazer a visualização e validação dos dados na plataforma. Na figura 6 estão as opções de filtro disponíveis para gerar o gráfico que foram implementadas:

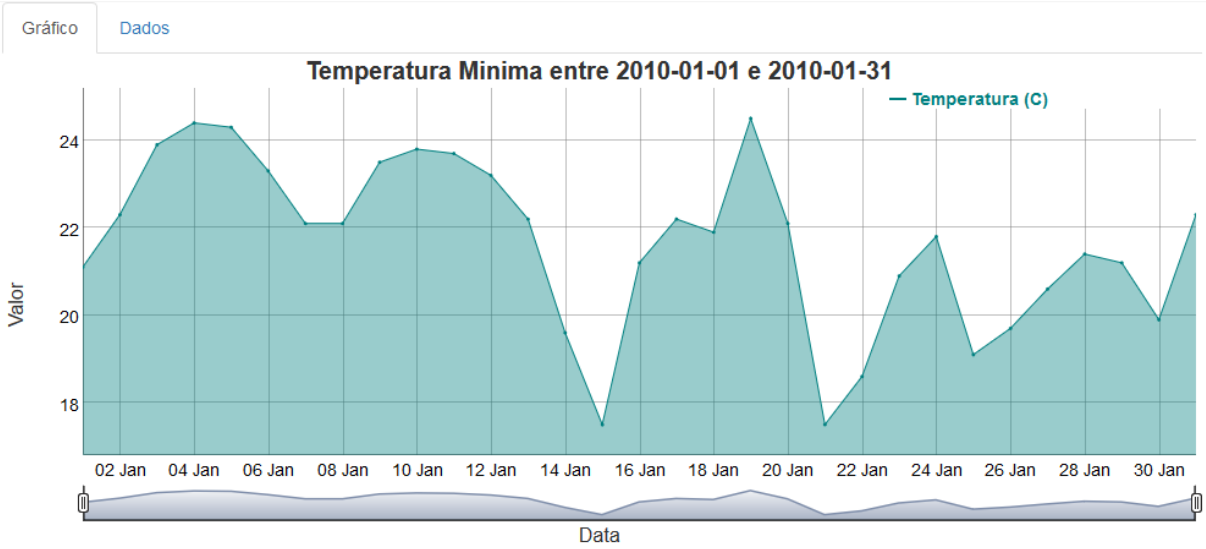
Figura 6 – Formulário



Fonte: do Autor.

Para montar a visualização das restrições, é necessário selecionar a estação de onde os dados estão armazenados, a variável que se deseja mostrar, o intervalo de datas e, por fim, se existe algum valor específico ou um intervalo de valores que se deseja informar para restringir a busca (opcional). A Figura 7 mostra um gráfico gerado buscando dados de temperatura mínima da Estação Linha Becker no mês de janeiro/2010 sem filtro de valores.

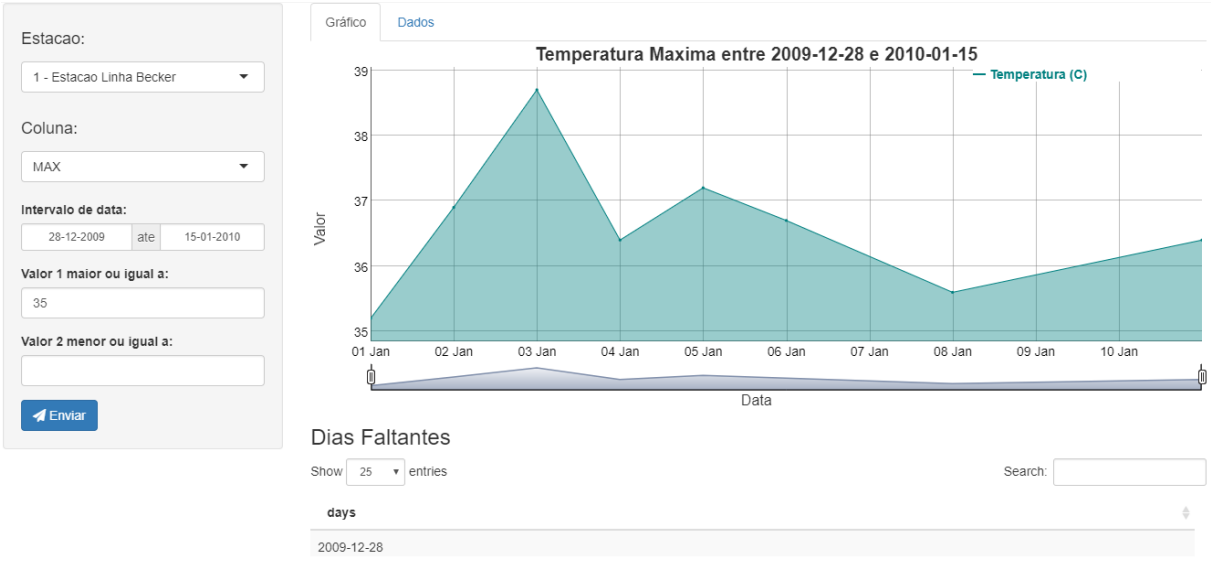
Figura 7 – Gráfico de temperatura mínima



Fonte: do Autor.

Na Figura 8, é montada uma restrição que busca dados de temperatura máxima no período de 28 de dezembro/2009 até 15 de janeiro/2010. Além disso, somente os dados de temperatura maior ou igual a 35 ºC serão exibidos. Entretanto, não foram inseridos registros anteriores a janeiro/2010, desta forma além de exibir o gráfico são mostradas as datas faltantes.

Figura 8 – Gráfico de temperatura máxima



Fonte: do Autor.

Na Figura 9, é mostrada uma tabela onde são indicadas as datas que não possuem registros que contemplem a restrição montada pelo usuário.

Figura 9 – Datas faltantes

Fonte: do Autor.

4 CONSIDERAÇÕES FINAIS

O presente trabalho apresentou uma plataforma para criação e visualização de restrições de integridade dinâmicas a fim de se verificar as informações armazenadas em um banco de dados.

O banco de dados, por meio de uma modelagem objeto-relacional foi projetado para ser genérico, ou seja, capaz de receber diferentes tipos de dados. A capacidade de receber dados de diferentes tipos é definida pelo UDT. Contudo, para a inserção de dados no sistema foi necessário definir uma estrutura de arquivo para possibilitar a inserção correta dos mesmos no banco.

Com a ferramenta de visualização, foi possível verificar a integridade dos dados inseridos no banco de dados através da criação de restrições de integridade dinâmicas. A ferramenta também informa ao usuário se existe algum dado faltante no período de tempo selecionado pelo usuário ao criar a restrição. Essas funcionalidades podem auxiliar na tomada de decisão pois oferecem uma visão integral da real situação em que se encontram os dados em um banco de dados.

Como trabalhos futuros, existem melhorias na parte de controle de acesso a plataforma, restringindo o acesso aos dados apenas àqueles que realizaram a submissão dos mesmos. Desta forma, a plataforma poderia receber dados de fontes variadas, mas os usuários estariam autorizados a manipular somente os dados aos quais possuem permissão. Outra melhoria seria no sentido de permitir a submissão

de diferentes formatos de arquivo além do .csv como por exemplo arquivos Excel (.xlsx).

ABSTRACT

Among the main purposes of a database is the guarantee that the data that is stored are intact, that is, they represent the modeled reality. To ensure data consistency, DBMSs use the concept of integrity constraints, which are rules that are tested every time there is a change in the data and indicate whether the data is valid or not. However, there are integrity constraints that are dynamic because they depend on the data set to be evaluated. For example, you want to check for missing dates in a daily dataset. In this sense, this work presents a web platform that assists in the creation of integrity constraints dynamically, allowing their visualization through interactive graphics. In order to test the functionalities of the system, a case study was carried out based on the climatic data observed in the municipality of Itapiranga / SC.

Keywords: Dynamic rules, Database, Consistency.

REFERÊNCIAS

BOOTSTRAP. *History*. Disponível em: <https://goo.gl/Tt4jA0>. Acesso em: maio de 2017.

DATE, C. J. *Introdução a sistemas de banco de dados*. Ed. Elsevier, 2003. 8ª Edição.

ELMASRI, Ramez; NAVATHE, Shamkant B. *Sistemas de Bancos de Dados*. Ed. Pearson Addison Wesley, 2011. 6ª Edição.

Força d'Oeste. *Wolfgang Lengert é homenageado no legislativo*. Disponível em: <https://goo.gl/ZJVMw5>. Acesso em março de 2017.

HEUSER, Carlos Alberto. *Projeto de banco de dados*. Ed. Bookman, 2009. 6ª Edição.

JQuery-tutorial. *What is JQuery?* Disponível em: <https://goo.gl/YtuZed>. Acesso em: maio de 2017.

JQuery. *What is JQuery?* Disponível em: <https://goo.gl/kP0kQD>. Acesso em: maio de 2017.

LAMIN, Jonathan. *Afinal, o que é Frontend e o que é Backend?* Disponível em: < https://goo.gl/gQyEos>. Acesso em junho de 2017.

LAZZARETTI, Alexandre Tagliari. *Controle de Restrições de Integridade de Domínio em Documentos XML*. Disponível em: <https://goo.gl/mgaISW>. Acesso em: março de 2017.

LAZZARETTI, Alexandre Tagliari. *XDC: Uma Proposta de Controle de Restrições de Integridade de Domínio em Documentos XML*. Disponível em: <https://goo.gl/6B0fLt>. Acesso em: março de 2017.

MELLO, Marcio; PETERNELLI, Luiz. *Conhecendo o R: Uma visão estatística*. Ed. UFV, 2013. 1ª Edição.

MDN. *Introdução ao HTML*. Disponível em: <https://goo.gl/KQipsx>. Acesso em: abril de 2017.

PostgreSQL. *About*. Disponível em: <https://goo.gl/FMFqNu>. Acesso em: maio de 2017.

PHP. *O que é PHP?*. Disponível em: <https://goo.gl/R7mGau>. Acesso em: maio de 2017.

PHP. *Prefácio*. Disponível em: <https://goo.gl/9Qle5i>. Acesso em: maio de 2017.

RADU, Marius; MUREŞAN, Ioan; NISTOR, Răzva. Using R To Get Value Out Of Public Data. *Revista Română de Statistică nr. 2 / 2014*. Disponível em: <https://goo.gl/f9ErKb>. Acesso em: maio de 2017.

R Core Team. *R Language Definition*. Disponível em: <https://goo.gl/e0DwRP>. Acesso em: maio de 2017.

R. *What is R?*. Disponível em: <https://goo.gl/omqJ90>. Acesso em: maio de 2017.

Shiny. Disponível em: <https://goo.gl/dbnALR>. Acesso em: maio de 2017.

SILBERSCHATZ, Abraham; KORTH, Henry F.; SUDARSHAN; S. *Sistemas de Banco de Dados*. Ed. Elsevier, 2012. 6ª Edição.

TABLELESS. *O que é CSS?* Disponível em: <https://goo.gl/rOfyd3>. Acesso em: maio de 2017.

TABLELESS. *O que é HTML?* Disponível em: <https://goo.gl/f2wDyt>. Acesso em: abril de 2017.

Tutorials Point. *JavaScript Language*. Disponível em: <https://goo.gl/f1W7AK>. Acesso em: maio de 2017.

W3Schools.com. *JQuery Introduction*. Disponível em: <https://goo.gl/1IhLgo>. Acesso em: maio de 2017.