UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
INSTITUTO DE MATEMÁTICA
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

RAFAEL S. SAMPAIO

Ambiente de Dados do SIHSUS com MongoDB

RIO DE JANEIRO
2019

RAFAEL S. SAMPAIO

Ambiente de Dados do SIHSUS com MongoDB

Trabalho de conclusão de curso de graduação apresentado ao Departamento de Ciência da Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Orientadora: Profa. Valeria Menezes Bastos
Co-orientadora: Profa. Myrian Christina de Aragão Costa

RIO DE JANEIRO
2019

CIP - Catalogação na Publicação

S192a

Sampaio, Rafael Soares
Ambiente de Dados do SIHSUS com MongoDB / Rafael Soares Sampaio. -- Rio de Janeiro, 2019.
73 f.

Orientadora: Valeria Menezes Bastos.
Coorientadora: Myrian Christina de Aragão Costa.
Trabalho de conclusão de curso (graduação) - Universidade Federal do Rio de Janeiro, Instituto de Matemática, Bacharel em Ciência da Computação, 2019.

1. Saúde Pública. 2. ETL. 3. DATASUS. 4. NoSQL. 5. MongoDB. I. Bastos, Valeria Menezes , orient. II. Costa, Myrian Christina de Aragão , coorient. III. Título.

Elaborado pelo Sistema de Geração Automática da UFRJ com os dados fornecidos pelo(a) autor(a), sob a responsabilidade de Miguel Romeu Amorim Neto - CRB-7/6283.

RAFAEL S. SAMPAIO

Ambiente de Dados do SIHSUS com MongoDB

Trabalho de conclusão de curso de graduação apresentado ao Departamento de Ciência da Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Aprovado em 28 de agosto de 2019.

BANCA EXAMINADORA:

Valeria Menezes Bastos
D.Sc. (Depto de Ciência da Computação - UFRJ)

Myrian Christina de Aragão Costa
D.Sc (COPPE Civil - UFRJ)

Adriana Santarosa Vivacqua
D.Sc. (Depto de Ciência da Computação - UFRJ)

Rejane Sobriño Pinheiro, D.Sc. (Instituto de Estudos de Saúde Coletiva - UFRJ)

Dedicatória: Dedico este trabalho a toda população brasileira, em especial aos que mais necessitam da saúde pública, afinal o objetivo final do trabalho é beneficiá-los.

AGRADECIMENTOS

Agradeço a minha orientadora e co-orientadora Valeria Menezes Bastos e Myrian Christina de Aragão Costa, respectivamente, pela valiosa orientação na execução deste trabalho, assim como a professora Rejane Sobriño Pinheiro com seus esclarecimentos importantes e necessários sobre a base de dados do DATASUS e questões relativas à saúde pública.

Também reservo minha gratidão à Universidade Federal do Rio de Janeiro e a todos os profissionais do Departamento de Ciência da Computação do Instituto de Matemática pela excelente formação recebida no curso de Bacharel em Ciência da Computação, no qual me formo.

"Raw data, now!"

Tim Berners-Lee

RESUMO

O sistema de saúde pública do Brasil SUS disponibiliza dados de eventos de saúde como internações e nascimentos de forma anonimizada para uso público. Tais dados são oferecidos em padrões e tipos de arquivos oriundos de sistemas legados, e que não são adequados para análise a partir de técnicas e sistemas modernos. O objetivo deste trabalho é realizar uma extração, transformação e carga (ETL) dos dados da base de internações (SIHSUS) para um banco de dados NoSQL, produzindo um ambiente analítico adequado para a aplicação de técnicas de estatística, aprendizado de máquina e mineração de dados a fim de se extrair conhecimentos úteis para o entendimento e melhoramento da saúde pública. Foi escolhido o banco de dados orientado a documento, MongoDB, e o processo envolveu etapas de conversão dos arquivos originais, importação para o banco e agregação dos dados de tabelas auxiliares à tabela principal. Após o processo de ETL, foram obtidas as bases de internações em um ambiente analítico no MongoDB, do estado do Rio de Janeiro, entre os anos de 2010 a 2015. A metodologia pode ser estendida para a obtenção das bases do Brasil inteiro, assim como para a realização do mesmo processo em outras bases, como a de nascimento (SINASC) e mortalidade (SIM).

Palavras-chave: Saúde Pública. ETL. DATASUS. NoSQL. MongoDB.

ABSTRACT

The public health system in Brazil SUS provides data from health events such as hospitalizations and births anonymously for public use. The purpose of this project is to perform an extraction, transformation and loading (ETL) of hospitalization base (SIH-SUS) to a NoSQL database, producing a suitable analytical environment for the application of statistical techniques, machine learning and data mining, in order to to extract useful knowledge for understanding and improving public health. The document-oriented database MongoDB was chosen and the process involved conversion of the original files, importation into the database and aggregation of the data from auxiliary tables to the main table. After the ETL process, whas obtained the hospitalization base in an analytical environment in MongoDB, with data from the state of Rio de Janeiro, between the years 2010 and 2015. The methodology can be extended to obtain bases from entire Brazil, as well as to carry out the same process on other bases like the birth (SINASC) and mortality base(SIM).

Keywords: Public Health. ETL. DATASUS. NoSQL. MongoDB.

LISTA DE ILUSTRAÇÕES

LISTA DE CÓDIGOS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

SGBD	Sistema Gerenciador de Banco de Dados
DML	Linguagem de Manipulaçao de Dados
SQL	Structured Query Language
CAP	Consistency, Availability, Partition-resilience
CA	Consistency, Availability
CP	Consistency, Partition-resilience
AP	Availability, Partition-resilience
web	World Wide Web
NoSQL	not only SQL
JSON	JavaScript Object Notation
BSON	Binary JSON
BASE	Basically Available, Soft state, Eventual consistency
ETL	Extract, Transform, Load
OLAP	Online analytical processing
BI	Business intelligence
SUS	Sitema Único de Saúde
SINASC	Sistema de informação de Nascidos Vivos
SIM	Sistema de informações de Mortalidade
SIH	Sistema de informações Hospitalares
CPF	Cadastro de pessoa física
DBC	Database Container
DBF	DataBase File
CSV	Comma-separated value
DEF	arquivo de definição

CID10	Código Internacional de Doenças
CNV	arquivo de conversão
AIH	autorização de internação hospitalar
IBGE	Instituto Brasileiro de Geografia e Estatística

SUMÁRIO

1 INTRODUÇÃO

Apesar do Sistema Gerenciador de Banco de Dados (SGBD) do tipo relacional ser o mais amplamente conhecido e comumente usado, existem bancos de dados que diferem em relação ao modelo dos dados, a forma de armazenamento e ao tratamento dado a cada informação durante a operação de gravação: objeto, objeto-relacional, hierárquico, redes e outros. Os bancos de dados dos tipos hierárquico e de redes são os mais antigos, podem ser encontrados em sistemas legados e estão em desuso nas aplicações atuais.

Em particular, os bancos de dados dos tipos objeto e objeto-relacional são mais recentes que o relacional puro, porém não alcançaram amplo uso em sistemas. A possível vantagem que pode ter ajudado ao SGBD relacional a se destacar em relação aos outros é a sua proposta de separar a representação dos dados nos modelos conceitual, lógico e físico, além de fornecer um fundamento matemático (álgebra relacional) para a consulta (RAMEZ; SHAMKANT, 2010).

A variedade de SGBDs não existe somente em relação ao modelos de dados, eles podem ser centralizados ou distribuídos, heterogêneos ou homogêneos ou ainda assumir outras classificações. Um fator muito associado aos SGBDs relacionais é o uso da Linguagem de Manipulaçao de Dados (DML) ou Structured Query Language (SQL), porém existem outros tipos de DML utilizadas em outros sistemas.

Um fato relevante na história recente de bancos de dados foi a publicação em 1999 do teorema CAP, que é a sigla de "Consistency, Availability, Partition-resilience"(FOX; BREWER, 1999).

- Consistency (consistência) - Uma vez que um estado no sistema é criado, qualquer consulta posterior a criação deste estado verá o banco de dados com o último estado.
- Availability (disponibilidade) - Um sistema apresenta disponibilidade se em caso de falha de um ou vários nós ainda assim sempre as requisições são respondidas, mesmo que de forma inconsistente.
- Partition-resilience (resistência a fragmentação) - Caso haja falha de comunicação entre dois ou mais nós, a resistência a fragmentação determina que o banco de dados deve continuar a responder corretamente.

O teorema determina que em um sistema contendo vários nós constituintes de um SGBD distribuído, existe uma troca entre as propriedades CAP, de modo que só é possível garantir duas entre três. Dessa forma, o sistema pode assumir uma das três características: CA (garante consistência e disponibilidade), CP (garante consistência e resistência a fragmentação) e AP (garante disponibilidade e resistência a fragmentação).

Além do teorema CAP, recentemente tem havido um reconhecimento das diferentes naturezas que os dados podem ter e também de diversos pré-requisitos que as aplicações possuem. Alguns sistemas, por exemplo, não exigem que os dados estejam no seu estado mais atual, ou seja, não requerem consistência, mas os dados precisam estar disponíveis. Os novos sistemas e padrões usados na World Wide Web (web) têm participação importante no reconhecimento da variabilidade de dados e requisitos de sistemas (CAO et al., 2011).

O surgimento da web e o aumento da quantidade de dados que vem sendo produzida originou também o fenômeno de grandes volumes de dados. De acodo com (DOUG, 2001) as características básicas existentes em grandes volumes de dados são conhecidas como os 3 Vs: variedade, velocidade e volume. Sistemas projetados para análise e manipulação desses dados possuem os 3 Vs como requisitos e portanto apresentam características específicas como o fato de serem distribuídos e possuírem escalabilidade horizontal, para armazenar um volume imenso e crescente de dados. Alguns também são não estruturados ou semi-estruturados para lidar com a característica variedade.

Impulsionado pelo teorema CAP, pelo fenômeno de grandes volume de dados e pela tendência de especialização dos SGBDs específicamente para uma aplicação/tipo de dado, tem surgido recentemente um movimento que criou uma categoria de sistemas chamados NoSQL. O nome é acrônimo de "not only SQL", em tradução "não somente SQL", e se deve a esses novos sistemas não usarem como linguagem de consulta o SQL, apesar de alguns oferecerem essa opção. O fato é que o nome não remete a todos os fatores impulsionadores e características que o movimento assume.

Dentre os SGBDs NoSQL, os principais são o orientado a coluna, a grafo, o do tipo chave-valor e o orientado a documento. O orientado a coluna possui armazenamento físico de dados de forma colunar em oposição ao armazenamento orientado a linha. O orientado a grafos é adequado para dados com muitos relacionamentos não regulares, e o orientado a chave-valor amazena grandes listas de pares chave-valor e emprega o paradigma MapReduce, sendo adequado para grandes volumes de dados. O orientado a documentos armazena os dados em uma estrutura JavaScript Object Notation (JSON) e possui armazenamento físico neste mesmo formato, possivelmente usando a forma binária (Binary JSON - BSON) (MongoDB Development Team, 2019b).

Especificamente neste trabalho foi usado o SGBD NoSQL MongoDB, que é classificado como orientado a documento. O MongoDB organiza os dados em estruturas JSON, sendo os arquivos JSON binários (BSON) armazenados em disco. Ao fazer uma analogia com o modelo relacional, pode-se comparar uma relação no relacional a uma coleção no orientado a documento, e uma tupla pode ser comparada a um documento, com a diferença notável que cada documento em uma coleção não possui estrutura definida, podendo ter variados campos, e isso se deve à sua característica de ser semi-estruturado.

O MongoDB ainda possui arquitetura distribuída, podendo ser instalado em uma única

máquina ou utilizado em um cluster de computadores (MongoDB Development Team, 2019d). Em relação ao torema CAP se encaixa no tipo CP, garantindo consistência e resistência à fragmentação, porém sem garantias de disponibilidade [1].

No caso de sistemas CP, abrir mão da disponibilidade torna o sistema não confiável. Para contornar esse problema foram criados SGBDs NoSQL com a propriedade BASE, que é a sigla de "Basically Available, Soft state, Eventual consistency". O MongoDB possui essa característica (BONNET et al., 2011).

- Basically Available (disponibilidade básica) - Garante que vai haver uma resposta para qualquer requisição ao banco, porém a resposta pode não encontrar os dados pesquisados, ser consistente ou incorreta.
- Soft state - O estado do sistema pode variar a qualquer momento, ainda que não haja atualizações em andamento.
- Eventual consistency (consistência tardia) - Existe garantia que haverá consistência em algum momento finito, se as atualizações dos dados cessarem.

Sempre que uma pessoa ou instituição possui uma grande quantidade de dados, é natural que se pense em processar e analisar estes dados a fim de extrair conhecimento deles, e essa é uma importante aplicação dos SGBDs. Em ambientes empresariais, uma abordagem comum é armazenar diversos dados de diversas fontes em um armazém de dados, em seguida estes sofrem um processo de extração, transformação e carga (Extract, transform, load - ETL) e são armazenados em um datamart em um esquema lógico chamado "estrela"(além deste, existe também, o esquema não tão eficiente chamado "floco de neve") que é adequado para consultas do tipo OLAP (Online analytical processing - OLAP). A essa técnica e estudos correlatos se dá o nome de inteligência de negócios (Business intelligence - BI), a partir da qual são gerados relatórios e insights úteis para a tomada de decisão estratégica por gestores (BARBIERI, 2011).

1.1 MOTIVAÇÃO

O Sitema Único de Saúde (SUS) disponibiliza através do seu portal DATASUS dados de saúde da população brasileira obtidos de atendimentos e procedimentos realizados pelo SUS e registrados em diversos sistemas. Algumas bases de dados estão disponíveis como a base do Sistema de informação sobre Nascidos Vivos (SINASC), a base do Sistema de informações sobre Mortalidade (SIM) e a base do Sistema de Informações Hospitalares do SUS (SIH). Apesar de existirem campos como o cadastro de pessoa física (CPF), estes são criptografados para anonimizar a base antes de serem disponibilizados ao público.

[1] <https://mongodbforabsolutebeginners.blogspot.com/2016/06/acid-and-cap-theroems.html>

Assim como qualquer instituição ou pessoa que de posse de uma grande quantidade de dados tende a querer analisá-los, não é diferente com os dados de saúde disponibilizados pelo SUS. A extração de conhecimento pode trazer vários benefícios, os gestores participantes em diversas esferas da saúde pública poderiam tomar decisões baseando-se nas análises para otimizar e melhorar os seviços prestados, possivelmente melhorando a saúde da população e economizando recursos. Pesquisas científicas em áreas da saúde também poderiam usar conclusões obtidas a partir dos dados para conhecer melhor as especificidades das populações ou propor tratamentos e métodos de prevenção que a beneficiem.

Atualmente as áreas de estudo inteligencia computacional, mineração de dados e a estatística oferecem diversos algoritmos e métodos úteis que poderiam ser usados para analisar os dados, obtendo insights, fazendo previsões e extraindo conhecimento. Infelizmente a forma atual em que os dados são disponibilizados não é adequada para a análise pois estes se encontram em arquivos contendo componentes de bases usando alguns padões legados e não estruturados para consultas analíticas. O software Tabwin disponibilizado pelo próprio SUS para exploração e análise dos dados possui apenas alguns tipos básicos de análise disponíveis e interface para o sistema R[2], porém sua estrutura de armazenamento não é otimizada para cargas analíticas.

O ambiente escolhido para proporcionar os dados de tal forma que pudessem ser analizados explorando todo o potencial de métodos computacionais foi o MongoDB, cujo escolha foi feita usando os critérios:

- Ambiente distribuído - Apesar do objetivo do presente trabalho ser lidar apenas com dados do Rio de Janeiro, é importante considerar dados do Brasil inteiro para se beneficiar do grande volume. Dado que os dados carregados futuramente serão proveniente de procedimentos de saúde do Brasil inteiro durante alguns anos e podem ter tamanho de armazenamento elevado, e ainda acrescenta-se a tendência de crescimento com o tempo da quantidade de dados, qualquer que seja a solução é necessário poder ser utilizada em clusters de computadores com recursos elevados e ter escalabilidade. O MongoDB possui escalabilidade horizontal podendo ser executado em cluster de arquitetura shared nothing.

- Velocidade de leitura - O MongoDB é adequado para ambiente analitico pois apresenta velocidade de leitura superior a bases relacionais (WU; HUANG; LEE, 2015).

- Semi-estruturado - Existem várias diferenças entre os campos existentes em diferentes anos. Alguns campos são criados ou destruídos, ainda podem ocorrer campos que mudam de nome, se unem em somente um ou se separam em dois. A natureza semi-esrtuturada do MongoDB facilita e permite de forma natural lidar com essa variabilidade de campos que os dados podem possuir.

[2] <https://www.r-project.org>

No trabalho correlato a este (SANTOS; CARVALHO, 2017) foi feito o ETL das bases SINASC e SIM, e foram realizadas consultas para comparação de tempos entre estas bases no MongoDB e no PostgreSQL[3]. Pôde-se observar que dependendo da consulta, o desempenho no MongoDB pode ser até cerca de 10 vezes mais rápido, sendo portanto adequado para consultas OLAP.

1.2 OBJETIVO

O objetivo deste trabalho é realizar o ETL dos dados da base SIH, obtidos no portal DATASUS, oferecendo como resultado uma base de dados analítica no MongoDB para ser usada em algoritmos diversos de aprendizado de máquina, mineração de dados e estatística, para a extração de conhecimento e obtenção de insights úteis para a saúde pública.

A base original obtida do DATASUS possui diversos códigos que fazem referência a dados existentes em tabelas auxiliares externas. O presente trabalho deve considerar que o MongoDB não foi projetado para a realizaçoes de junções e proporcionar uma organização dos dados tal que as informações provenientes das tabelas auxiliares estejam disponíveis junto à tabela principal, e com acesso facilitado para a execução eficiente de algoritmos. A solução é inserir os dados provenientes de tabelas auxiliares na coleção principal do MongoDB, na forma de subdocumentos.

No trabalho (SANTOS; CARVALHO, 2017) tais objetivos já foram alcançados especificamente para as tabelas do SUS SINASC e SIM. Portanto este trabalho é complementar e visa realizar o ETL para a tabela SIH, trazendo novos dados com informações de internação.

1.3 TRABALHOS CORRELATOS

No trabalho (MongoDB Development Team, 2019e) são listados diversos casos de uso do MongoDB, dentre os quais está o caso analítico. Algumas das capacidades que o MongoDB possui e que são pré-requisitos para essa aplicação são a escalabilidade de acordo com o crescimento do volume de dados e o esquema adaptável (semi-esrtuturado). No entanto o artigo elimina o processo de ETL, executando o tratamento dos dados em tempo de análise. No presente trabalho o ETL foi realizado agregando os dados de tabelas auxiliares à tabela principal, para simplificar e agilizar a análise dos dados.

O trabalho de (CHEVALIER et al., 2015) investiga o uso de Bases NoSQl, incluindo a orientação a documento, para a realização do processo inteiro de implementação de um sistema OLAP, do nível conceitual ao nível lógico. O principal diferencial em relação ao presente trabalho é que na representação do esquema estrela em um banco orientado a

[3] <https://www.postgresql.org/>

documento os fatos são modelados como subdocumentos cujos atributos são as medidas. As dimensões, assim como neste trabalho, são modeladas como subdocumentos.

No trabalho (CARNIEL et al., 2012) foram feitos testes comparando ambientes analíticos construídos em um banco de dados relacional e outros NoSQL. Os testes foram feitos em dados de uma aplicação de varejo, cuja tabela fato continha compras realizadas, e foram usadas consultas OLAP para avaliação de desempenho. Em suma, o trabalho conclui que o modelo orientado a coluna é melhor para cargas analíticas, seguido do banco de dados relacional otimizado com views e depois pelo MongoDB usando subdocumentos. Analizando o tamanho ocupado pelo armazenamento o MongoDB teve o pior desempenho, uma vez que possui muitos dados redundantes.

O trabalho de (SANTOS; CARVALHO, 2017) possui os mesmos objetivos deste trabalho, que é criar um ambiente analítico no MongoDB para bases do SUS, porém realizou o ETL para as bases SINASC e SIM. O presente trabalho busca realizar o ETl para a base SIH e possui uma metodologia diferente. No trabalho citado foi usado o Pentaho[4] e neste foi usado o pipeline de tratamenteo de dados do MongoDB, que possibilita a execução do ETL através de scripts, facilitando a automação e o encapsulamento de todo o processo, além de aproveitar a capacidade distribuída do MongoDB na execução do ETL.

1.4 ESTRUTURA DO DOCUMENTO

No capítulo 1 são introduzidos bancos NoSQL, é apresentada a motivação para uma base de dados de saúde no MongoDB e são apresentados trabalhos correlatos. No capítulo 2 são descritas todas as ferramentas, padrões de arquivos, linguagens de programação e bibliotecas utilizadas no trabalho. No capítulo 3 as bases do SIH usadas no trabalho, assim como bases auxiliares são identificadas e os campos usados são enumerados. No capítulo 4 todos os passos do processo de ETL para importação da base para o MongoDB são descritos. No capítulo 5 são apresentadas as bases e os resultados obtidos no ETL. No capítulo 6 conclui-se sobre a utilidade e importância da base obtida e são descritos trabalhos futuros complementares a este.

[4] <https://www.hitachivantara.com/go/pentaho.html>

2 ESTADO DA ARTE

Durante o processo de ETL dos dados disponibilizados pelo DATASUS para carga no MongoDB, foram necessários vários estágios de conversão de formatos de arquivos, junção com coleções auxiliares e importação para o ambiente do banco de dados. Nesta seção são descritos todos os padrões de arquivos e as ferramentas usadas nesse processo.

2.1 TABWIN

O Tabwin (Tabwin Development Team, 2008) é uma ferramenta de tabulação desenvolvida pelo DATASUS para necessidades de análise e vizualização de dados de gestores nas diversas esferas do governo, e atualmente se encontra na versão 4.15. Dentre as suas principais funções estão a geração de estatísticas e gráficos sobre os dados, com o objetivo de auxiliar o planejamento e a tomada de decisão para usuários do setor de saúde pública.

Apesar do Tabwin ter trazido grande avanço e configurabilidade para as bases disponibilizadas no site do SUS, suas capacidade de análise e visualização possuem limitações, ele possui apenas 6 tipos de gráficos que podem ser feitos, 4 estatísticas (descritivas de colunas, coeficiente de correlação, regressão linear e histograma), 14 operações sobre os dados e a possibilidade de importar mapas de programas de geoprocessamento de uso mais comum como Atlas-GIS[1] e MapInfo[2], permitindo a visualização de dados sobre os mapas. Já estão disponíveis mapas de todo o Brasil, até o nível municipal. Nenhum tipo de processamento personalizado sobre os dados é permitido e o sistema não oferece suporte a diversas técnicas de análise que poderiam ser feitas, como por exemplo a aplicação de algoritmos de mineração de dados e aprendizado de máquina. Uma alternativa mais poderosa é usar a interface que o Tabwin possui para o sistema R, porém as bases ainda não estão em um ambiente otimizado para a axecução rápida de consultas.

Os dados disponibilizados pelo DATASUS se encontram no formato de arquivo compactado Database Container (DBC), que não é manipulado pela maioria dos SGBDs. Entretanto, a função básica do Tabwin, de abrir e manipular este tipo de arquivo, foi explorada totalmente neste trabalho. A ferramenta foi usada para abrir os arquivos e fazer a conversão de DBC para o formato DataBase File (DBF), para posteriormente serem transformados no formato separado por vírgula (Comma-separated values - CSV).

O Tabwin pode ser obtido de <http://www2.datasus.gov.br/DATASUS/index.php?area=060805&item=3> clicando-se no link "Download Progama".

[1] <http://www.atlasgis.net/>

[2] <https://www.pitneybowes.com/us/location-intelligence/geographic-information-systems/mapinfo-pro.html>

2.2 FORMATOS DE ARQUIVOS

O formato primário no qual os dados de saúde são disponibilizados pelo DATASUS (SUS, 2008) é o DBC. De acordo com (GEATER, 2011), este é um formato associado ao Visual Foxpro Database Container (Microsoft Corporation) e é possível observar na sua vizualização no Tabwin que ele armazena dados no formato tabular.

O próprio Tabwin consegue converter os arquivos no formato DBC para um formato DBF, que, de acordo com (WHATIS.COM, 1999) é um arquivo usado originalmente no sistema dBase II. Enquanto o formato DBC armazena os dados de forma compactada, o formato DBF expande o arquivo, ficando com tamanho maior e podendo ser lido por alguns softwares de planilha eletrônica.

Os outros dois tipos de arquivos manipulados pelo Tabwin e que foram usados neste trabalho são as bases de configuração do tabulador encontradas em arquivos de definição (DEF) e de conversão (CNV). O primeiro se trata de um arquivo que possui diversas informações sobre a base de dados, em especial a que foi útil para o trabalho foi a associação entre as colunas e suas respectivas tabelas auxiliares. Os arquivos CNV possuem dados no formato chave-valor e são usados como tabelas auxiliares a alguns campos nas tabelas principais.

2.3 ESPECIFICAÇÕES DO FORMATO CNV

Nenhuma documentação ou especificação foi encontrada sobre o formato CNV, e pesquisas realizadfas sempre retornam como resultado links associados ao DATASUS, o que permite concluir que provavelmente o formato foi criado pelo DATASUS para suas próprias aplicações.

Devido à falta de informação e documentação disponível, todo o conhecimento desse tipo de arquivo foi obtido através de engenharia reversa e da observação dos arquivos usados. Os arquivos CNV apresentam uma estrutura com chaves e valores organizados de forma particular, contendo também uma linha de cabeçalho e um campo inicial que informa o número da linha contando a partir de 1. Foi possível concluir sobre a formatação:

- O cabeçalho possui, após uma tabulação, duas ou três strings separadas por espaços, a primeira string informa a quantidade de linhas ou registros do arquivo, e a segunda informa quantos algarismos o valor possui e a terceira pode existir ou não. Só foi possível concluir que se houver a terceira string (normalmente a sring é "L") indica que a chave não é constituída apenas por números.

- Cada linha com um par chave-valor é iniciada por uma tabulação e zero ou mais espaços em branco, após isso verifica-se o número da linha contando a partir de 1.

- Após o número da linha ocorre um ou mais espaços seguidos pelo valor.

- A chave sempre se inicia no sexagésimo primeiro caracter da linha, seguindo o formato especificado no cabeçalho.
- A chave pode conter "-", indicando uma sequência de valores (e.g. 0000-9999).
- A chave pode conter ",", indicando uma lista de valores (e.g. 1000,1005,1009).
- A chave pode conter qualquer quantidade de "," e "-", sendo composta por uma ou mais listas e uma ou mais sequências de valores.
- Qualquer string após a contagem da linha e após espaços posteriores a chave constitui o valor.

Abaixo são mostradas as primeiras linhas do arquivo de municípios br_municip.cnv, que possui 5597 linhas e a chave (código do IBGE do Município) contém 6 caracteres numéricos:

```
5597  6
    1  110001 Alta Floresta D'Oeste                    110001
    2  110037 Alto Alegre dos Parecis                   110037
    3  110040 Alto Paraíso                              110040
    4  110034 Alvorada D'Oeste                          110034
    5  110002 Ariquemes                                 110002
    6  110045 Buritis                                   110045
    7  110003 Cabixi                                    110003
```

Abaixo são mostradas as primeiras linhas do arquivo CID10_2.CNV, com o código internacional de doenças (Código Internacional de Doenças - CID10), que possui 739 linhas e a chave contém até 4 caracteres incluindo letras:

```
739 4 L
    1  C00.0 Labio super externo                       C000,
    2  C00.1 Labio infer externo                       C001,
    3  C00.2 Labio externo NE                          C002,
    4  C00.3 Labio super face interna                  C003,
    5  C00.4 Labio infer face interna                  C004,
    6  C00.5 Labio s/especificacao face interna        C005,
    7  C00.6 Comissura labial                          C006,
    8  C00.8 Lesao invasiva do labio                   C008,
    9  C00.9 Labio NE                                  C009,
   10  C01   Neopl malig da base da lingua             C01 ,C010,
```

A principal dificuldade em construir o analisador deste tipo de arquivo foi identificar o fim da lista de códigos, pois haviam casos em que a descrição ocupava a linha inteira, com posições antes e após os códigos.

2.4 PYTHON

Em diversas fases do trabalho houve a necessidade de se criar scripts para automatizar tarefas repetitivas, tais como a conversão em lote de arquivos DBF para CSV, e para a importação de vários arquivos CSV para databases no MongoDB. Além disso, o próprio processo de ETL sobre os dados brutos foi feito através de programação.

A linguagem escolhida para tais tarefas foi o Python (Python Development Team, 2001) na versão 3.5. Ocorreram muitas mudanças entre as versões 2.X e 3.X, tornando a mais recente incompatível com a mais antiga. Porém, a escolha da versão 3 se deve aos melhoramentos introduzidos e a maturidade que a linguagem atingiu, além do fim de vida da versão 2 ter sido determinada pelos desenvolvedores para 2020 (Python Development Team, 1990).

2.5 DBF2CSV

Os dados originais obtidos do DATASUS possuem formatação própria, o que exigiu um prodcedimento manual de conversões (SANTOS; CARVALHO, 2017), do formato DBC para DBF, de DBF para CSV, e finalmente a importação para o MongoDB. O processo de conversão de DBF para CSV foi feito através de um script em Python, chamado DBF2CSV (SPAAN, 2019).

Para a execução do script foi necessário instalar, como dependência, a biblioteca dbfpy (SMYSHLIAEV; SAMCHUCK, 2005), que possui rotinas úteis para a manipulação de arquivos DBF. Além disso, o script foi originalmente feito para o Python 2.X, foi então necessário fazer a migração que consiste apenas em modificar as chamadas para a função "print()", que no Python 3.X leva os parênteses na sintaxe.

2.6 MONGODB

O MongoDB (MongoDB Development Team, 2019a) é uma base de dados NoSQL baseada em documentos JSON-like, armazenando informações de forma semi-estruturada, através de documentos com diferentes campos. Seu código é aberto e o sistema possui uma versão comunitária disponibilizada de forma gratuita.

O objetivo de analisar os dados pode ser cumprido usando o MongoDB pois este permite consultas ad hoc, porém, a característica mais importante para o presente trabalho é que o MongoDB é distribuído e possui escalabilidade horizontal, com isso pode acompa-

nhar a tendência de crescimento elevado dos dados e uma infra-estrutura distribuída pode ser aproveitada para a execução de algoritmos com alta complexidade computacional.

A versão do MongoDB utilizada neste trabalho foi a 3.6.

2.7 PYMONGO

O MongoDB possui um front-end com interface gráfica chamado Atlas[3], além de um terminal e de bibliotecas que permitem a manipulação das bases nas seguintes linguagens: Python, Javascript, C++, C#, Java e Ruby. Devido à facilidade de uso e à familiaridade com o uso da linguagem, foi escolhida Python como linguagem no presente trabalho.

A biblioteca através da qual é possível acessar e manipular as bases de dados no MongoDB usando Python é a PyMongo (PyMongo Development Team,). Essa biblioteca possui funções usadas para a execução de consultas, manipulações de dados e tarefas administrativas como criação e remoção de coleções, criação de índices, etc.

A versão usada no trabalho foi a PyMongo 3.7.2, que é a mais recente compatível com o Python 3.X. A instalação do PyMongo pode ser feita através do instalador de pacotes do Pyhon pip, usando o comando:

```
$sudo pip install pymongo
```

2.8 PIPELINE DE TRATAMENTOS DE DADOS NO MONGODB

Apesar do MongoDB possuir funções para a execução de consultas ad hoc, a sua natureza de armazenar na forma de documentos isolados não favorece operações de junção. Para esta e outras operações típicas de tratamento de dados e ETL, o MongoDB oferece um framework de agregação (MongoDB Development Team, 2019c).

O framework funciona montando um pipeline de tarefas que são executadas, se possível em paralelo, seguindo um fluxo dos dados. Algumas tarefas utilizadas neste trabalho foram:

- Project - operação de projeção, produzindo um resultado com apenas campos selecionados.
- Unwind - desconstrói um array, adicionando determinado elemento do array como valor de um campo no documento.
- Lookup - análogo à junção, acrescentando campos ao resultado provenientes de uma coleção diferente, dado um predicado.

[3] <https://www.mongodb.com/cloud/atlas>

3 IDENTIFICAÇÃO DAS BASES DE DADOS DO DATASUS

O DATASUS (SUS, 2008) é um portal do SUS que disponibiliza dados de saúde da população brasileira, obtidos de diferentes sistemas e especificamente sobre a ocorrência de um fato de saúde como internação ou nascimento. Além de dados de saúde, dados auxiliares, documentação e o software Tabwin, desenvolvido para tabular as bases, também podem ser obtidos nesse portal.

Através do endereço <http://www2.datasus.gov.br/DATASUS/index.php?area=0901> é possível encontrar os dados de diversos sistemas disponíveis para download, entre eles os dados do SINASC (Sistema sobre Nascidos Vivos) e do SIM (Sistema de Mortalidade), cujo ETL e inserção em uma base de dados do MongoDB foi explorada no trabalho (SANTOS; CARVALHO, 2017). No presente trabalho, os dados do SIH (Sistema de Internações Hospitalares) foram incorporados aos dados das coleções pré-existentes (SINASC e SIM).

3.1 BASE SIH

A base do Sistema de Informações Hospitalares (SIH) possui dados de todas as autorizações de internação hospitalar (AIH) que foram financiadas pelo SUS e o seu propósito é gerar relatórios para que os gestores possam fazer pagamentos aos estabelecimentos de saúde (SUS, 2019). Dentre as suas funcionalidades estão a programação do orçamento dos estabelecimentos e a avaliação de condições sanitárias através de taxas de óbito e infecção hospitalar nesses estabelecimentos.

A base do SIH está integrada ao DATASUS, a partir do qual os dados são disponibilizados em 4 tabelas: "AIH Rejeitadas", contém as autorizações de internação que foram rejeitadas, "AIH Rejeitadas com código de erro", contém, além das autorizações de internação rejeitadas, o código que informa o motivo da rejeição, "Serviços profissionais", contém procedimentos hospitalares não classificados como internação, porém autorizados pelo SUS e "AIH reduzida", contém todas as internações autorizadas de forma reduzida, contendo apenas o procedimento principal realizado na intervenção. Uma vez que serviços profissionais e internações rejeitadas não carregam informações relevantes sobre saúde, a tabela utilizada nesse trabalho foi a AIH reduzida.

O download da base SIH e da sua documentação pode ser feito acessando <http://www2.datasus.gov.br> e navegando para:
Início > Serviços > Transferência/Download de Arquivos > Arquivos de Dados > SIHSUS.
As bases são disponibilizadas no portal mensalmente e em formato dbc. Para fazer o download:

- Escolha a modalidade de arquivos: "Dados".

- Escolha a tipo de arquivo: "RD - AIH Reduzida".
- Selecione o ano desejado.
- Selecione uma ou mais UFs desejadas.
- Selecione um ou mais meses desejados.
- Clique em "enviar"e faça o download no lado direito da tela.

A documentação de cada sistema é composta por um dicionário de dados, que enumera os campos e informa seu tipo, sendo sempre uma string com determinada quantidade de caracteres, e uma breve descrição do campo. Também é explicitada a data de processamento, uma vez que bases de anos diferentes podem possuir diferentes campos. Para fazer download da documentação:

- Escolha a modalidade de arquivos: "Documentação".
- Clique em "enviar"e faça o download no lado direito da tela.

3.2 AUXILIARES DA BASE SIH

A base SIH não é autocontida e necessita de tabelas auxiliares. O campo "sexo"por exemplo pode assumir os valores de 0 a 9. Tal código é mapeado para uma descrição, que pode ser: "Masculino", "Feminino"ou "Ignorado", e que se encontram no arquivo auxiliar "SEXO.CNV".

O download das tabelas auxiliares da base SIH pode ser feito acessando <http://www2.datasus.gov.br> e navegando para:
Início > Serviços > Transferência/Download de Arquivos > Arquivos de Dados > SIHSUS

Para fazer download:

- Escolha a modalidade de arquivos: "Arquivos Auxiliares de Tabulação".
- Clique em "enviar"e faça o download no lado direito da tela.

O arquivo obtido é compactado com diversos arquivos, entre eles as tabelas auxiliares nos formatos DBF e CNV e arquivos com a extensão ".DEF", cujas informações serão detalhadas no capítulo 3.2.1. Por exemplo, o arquivo "RD2008.DEF"possui informações sobre a base de internações "AIH Reduzida", e, portanto, foi utilizado neste trabalho.

3.2.1 Arquivos de Tabulação do TABWIN (.DEF)

Os arquivos com a extensão .DEF são arquivos de tabulação com metadados usados pelo Tabwin para suas funções. É um formato específicamente feito pelo DATASUS para a utilização do tabulador e, como nenhuma documentação sobre a organização deste arquivo foi encontrada, todas as conclusões aqui presentes foram obtidas através de observação e engenharia reversa:

- O arquivo possui seções representadas por três linhas iniciadas com ";", e a linha central contém o título da seção que descreve as "Variáveis referentes ao hospital"ou as "Variáveis referentes ao paciente".
- Dentro de cada seção, cada linha possui um conjunto de 4 valores separados por vírgula.
- O primeiro valor possui uma descrição semântica do campo a ser usada nos menus do tabulador.
- O segundo valor se refere ao nome do campo na base SIH.
- O terceiro valor pode assumir um numeral ou uma string em alguns casos, e não foi possível identificar sua função.
- O quarto valor possui o nome do arquivo onde se encontram os dados auxiliares ao campo em questão da base SIH, com os códigos específicos.

Abaixo, são apresentados dois exemplos do conteúdo do arquivo de tabulação para a base "AIH Reduzida", contendo as seções "Variáveis referentes ao hospital"e "Variáveis referentes ao paciente"e suas primeiras linhas de dados:

```
;
; Variáveis referentes ao hospital
;
LRegião e UF int    ,MUNIC_MOV ,1          ,REGUF.CNV
LUF int             ,MUNIC_MOV ,1          ,UF.CNV
LRegião int         ,MUNIC_MOV ,1          ,REGIAO.CNV
CRegião int         ,MUNIC_MOV ,1          ,REGIAOC.CNV

;
;   Variáveis referentes ao paciente
;
XSexo               ,SEXO      ,1          ,SEXO.CNV
XFaixa etária (5)   ,COD_IDADE ,1          ,IDADEBAS.CNV
XFaixa etária (9)   ,COD_IDADE ,1          ,IDADEPUB.CNV
```

```
XFaixa etária (18) ,COD_IDADE ,1          ,IDADE18.CNV
XIdade detalhada   ,COD_IDADE ,1          ,IDADEDET.CNV
XCor/raça          ,RACA_COR  ,1          ,RACACOR.CNV
XEtnia             ,ETNIA     ,1          ,ETNIA.CNV
```

3.3 COMPLETUDE DA BASE SIH

Foi feita uma análise de completude de todos os campos da base SIH para o ano de 2010. A completude é expressa em percentagem unitária de registros preenchidos. O resultado se encontra na tabela 4 em anexo.

3.4 DOMÍNIO DA BASE SIH

Foi feita uma análise do domínio de todos os campos relevantes da base SIH para o ano de 2010. O domínio é expresso pelos valores mínimo e máximo que cada campo assume. No caso de campos não numéricos, estes foram interpretados como string. O resultado se encontra na tabela 5 em anexo.

3.5 ESCOLHA DOS CAMPOS DA BASE SIH

O domínio e a completude de cada campo foram discutidos em reunião junto às especialistas da área de saúde que selecionaram os campos mais relevantes e que podem servir para posteriores avaliações de comportamento das internações efetuadas ao longo dos anos, sendo consideradas as bases do SIH de 2010 até 2015.

Devido à natureza semi-estruturada do MongoDB e do seu modo de armazenamento orientado a documentos, os campos com baixa completude não provocam um aumento na quantidade de dados armazenados, uma vez que valores faltantes não são armazenados (em oposição ao modelo relacional que armazena "NULL"). Isso foi levado em consideração na análise de campos usados, fazendo com que campos com baixa completude mas de alta importância fossem incorporados.

Diante da conclusão que campos com baixa completude não possuem impacto negativo na base, a abordagem junto as especialistas da área de saúde passou a ser a avaliação individual de todos os campos do dicionário de dados. Observa-se que o dicionário possui todos os campos existentes a partir de 2010, ainda que em alguns anos alguns campos possam não existir, como por exemplo os campos TPDISEC1,TPDISEC2,..., TPDISEC9 que só estão presentes a partir de 2014. Depois da apreciação de cada campo foram escolhidos aqueles que seriam aproveitados na base analítica construída no trabalho, sendo o critério usado pelas especialistas a importância analítica para fenômenos de saúde pública.

Houve dúvida por parte da especialista se alguns campos teriam utilidade analítica, onde optou-se por incorporá-los, já que a adição de um ou mais campos não causaria um

aumento significativo de processamento e memória para armazená-los. Um exemplo onde isso ocorreu foi com os campos DIAGSEC1,DIAGSEC2,DIAGSEC3,... DIAGSEC9 que possuíam completude bem baixa, mas ao percorrer a base e verificar que poucos registros estavem preenchidos a especialista demonstrou interesse, revelando a importância que estes campos podem possuir.

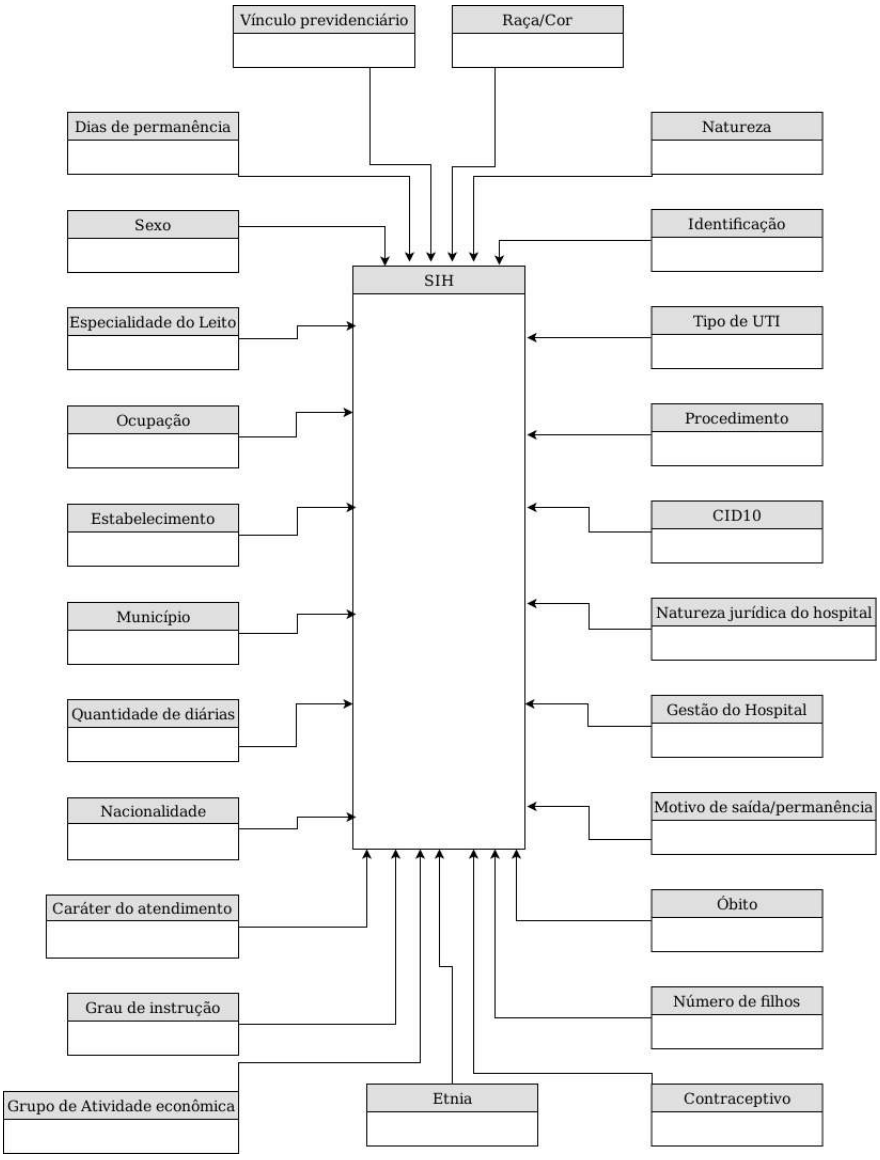
Para todos os campos da SIH que são chaves de tabelas auxiliares, optou-se por fazer a junção com a tabela auxiliar correspondente no processo de ETL, incorporando na base analítica os dados dessas tabelas auxiliares. Tal decisão foi tomada pois permite efetuar análises ad-hoc também através desses campos.

Na tabela 6 em anexo estão todos os campos que foram escolhidos para compor a base SIH no MongoDB. Também foram colocadas as tabelas auxiliares quando existentes.

3.6 MODELO CONCEITUAL DA BASE SIH

Na figura 1 é exibido o modelo conceitual da base SIH e todas as tabelas auxiliares que são usadas para complementar as informações. Algumas tabelas auxiliares apenas possuem descrições dos códigos usados na tabela principal, como é o caso do campo IDADE. Porém, outras acrescentam informações externas, provenientes de outras bases como a tabela CID10 e a de estabelecimentos CNES. Nota-se porém, que algumas tabelas foram usadas mais de uma vez, como a CID10 por exemplo, para prover a descrição de campos como DIAG_PRINC e DIAGSEC1, dentre outros.

Figura 1 – Modelo Conceitual da Base SIH.



4 EXTRAÇÃO, TRANSFORMAÇÃO E CARGA (ETL)

Diversos scripts foram elaborados para lidar com formatos de arquivos específicos ou para automatizar tarefas repetitivas e longas. Apenas os scripts centrais para a execução do trabalho foram inseridos nesse documento nos anexos, porém todos os códigos utilizados estão disponíveis para consulta no repositório online <https://github.com/velhoti/SIHSUS>.

4.1 CONVERSÕES

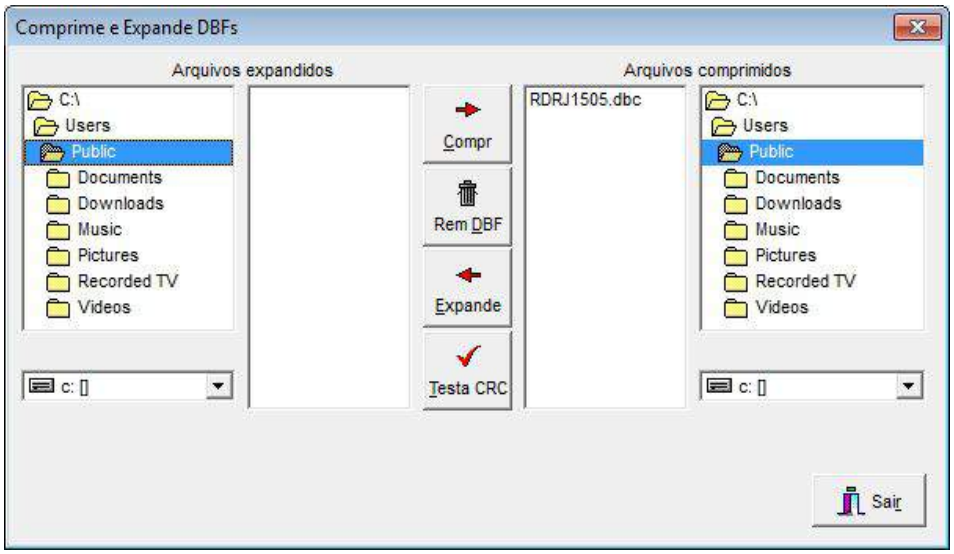
4.1.1 DBC para DBF

As bases do SIH são obtidas do DATASUS no formato DBC. O primeiro passo para a importação destas tabelas para o MongoDB é a conversão do formato DBC para DBF. Uma vez que o formato DBC é uma versão comprimida do formato DBF pode-se usar o termo descompressão ao invés de conversão. Esta descompressão foi feita através do próprio Tabwin fornecido no portal do DATASUS:

Clicando-se em Arquivo > Comprime/Expande .DBF

Uma janela como a da figura 2 é mostrada. No lado direito são listados os arquivos DBC na pasta, e no lado esquerdo estão os arquivos DBF. Esta janela possui tanto a funcionalidade de compressão DBF para DBC quanto descompressão.

Figura 2 – Caixa de descompressão de DBC para DBF no Tabwin



Para descomprimir, o arquivo desejado é selecionado e clica-se em "Expandir". Após o processo, o arquivo aparece na pasta no lado esquerdo da janela (figuras 3 e 4).

Figura 3 – Processo de descompressão no Tabwin

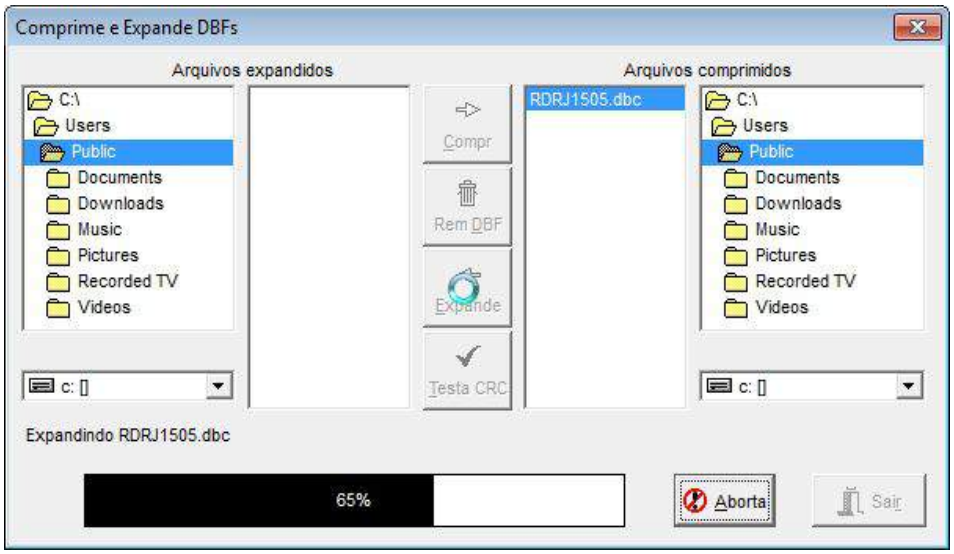
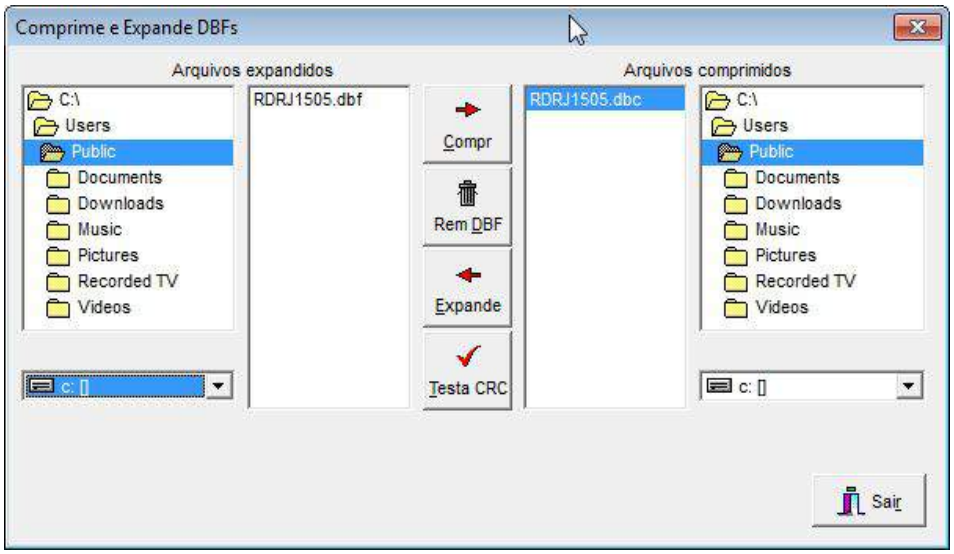


Figura 4 – Resultado da descompressão no Tabwin



4.1.2 DBF para CSV

Além dos arquivos originais da base SIH serem convertidos para DBF, algumas tabelas auxiliares também são disponibilizadsa pelo DATASUS neste formato. Para a importação de dados para o MongoDB é necessário ainda converter DBF para CSV, que é um formato aceito.

A conversão de DBF para CSV é feita com o script dbf2csv mostrado no capítulo 2.5. Após a instalação das dependências o script foi colocado na pasta */usr/local/bin* que faz parte do PATH do sistema, e desta forma não é necessário o ponto e barra (./) para a sua execução. O modo de uso é simples, basta digitar o comando seguido do nome de um arquivo DBF válido que é feita a conversão e um arquivo final CSV com o mesmo nome do original é criado na pasta:

```
$dbf2csv RDRJ1505.dbf
```

4.1.3 CNV para CSV

Arquivos do tipo CNV provavelmente foram criados pelo próprio DATASUS para suas aplicações, e sua codificação obtida através de engenharia reversa foi descrita no capítulo 2.3. Apesar da representação dos dados ainda serem arquivos textos, o que facilita sua leitura e entendimento, para a conversão de CNV para CSV foi desenvolvida uma biblioteca em Python, que considera toda a estrutura interna de um CNV e produz um arquivo CSV com dois campos: código e valor.

A biblioteca confeccionada cujo código 7 está em anexo, tem como função principal a "CNV2CSV", que recebe como parâmetro de entrada o nome de um arquivo do tipo CNV que deve estar presente no diretório atual. A função cria um arquivo de saída com o mesmo nome de entrada, porém com a extensão CSV.

4.2 PROCESSO DE ETL

O processo de ETL é realizado nos dados originais obtidos do DATASUS até que se chegue aos dados finais no MongoDB pronto para serem analisados.

4.2.1 1º Passo - Decisão Sobre os Campos Aproveitados e Tabelas Auxiliares

Antes do processo de ETL, um passo preliminar a ser feito é a determinação de quais campos das tabelas originais serão aproveitados. Os campos foram escolhidos em reunião com especialistas da área de saúde que possuíam conhecimento das bases e do significado e contexto de cada campo, levando em consideração ainda as análises de completude e domínio feitas (3.5).

Uma vez que os campos aproveitados foram selecionados, a inspeção do arquivo de metadados do tipo DEF *SP2008.DEF* foi feita, e as tabelas auxiliares de algum dos campos selecionados foram incorporadas. A lista com os campos selecionados e as respectivas tabelas auxiliares, quando o campo possuía alguma, pode ser vista na tabela 6 em anexo.

4.2.2 2º Passo - Conversão da Tabela SIH de DBC para DBF

As tabelas SIH são fornecidas pelo DATASUS no formato DBC e a conversão inicial, que pode ser chamada de expansão, é para formato expandido DBF. Tal conversão foi feita usando o próprio software Tabwin disponibilizado pelo DATASUS. As instruções para o uso da funcionalidade de Compressão/Expansão do Tabwin estão descritas em 4.1.1.

As tabelas so SIH são disponibilizadas em períodos de um mês, mas o Tabwin permite a conversão de vários arquivos de uma só vez, bastando colocá-los todos em uma

pasta, selecionar todos os meses desejados, através da sua interface (figura 3), e clicar em "Expandir", para que o processo seja feito em lote.

Tanto o download das bases mensais entre os anos de 2010 até 2015 quanto a sua conversão de DBC para DBF foram feitos manualmente, sem qualquer script de automatização da tarefa.

4.2.3 3º Passo - Conversão da Tabela SIH de DBF para CSV

Após a obtenção das tabelas SIH mensais em DBF, estas foram convertidas para o formato final CSV, que pode ser importado diretamente para o MongoDB.

A conversão para CSV é feita através do script DBF2CSV (seção 2.5). Como a quantidade de arquivos DBF (um por mês, de 2010 a 2015, sendo 60 no total) era elevada, foi feito um script em Python para automatizar a tarefa, que pode ser visto no repositório de códigos. O script lista todos os arquivos da pasta atual em que foi executado, e caso o arquivo seja do tipo DBF (extensão "DBF"ou "dbf") será executado o comando para conversão:

```
$dbf2csv nome_do_arquivo.DBF
```

A conversão foi feita em lote, colocando-se todos os arquivos em uma pasta e executando-se o script.

4.2.4 4º Passo - Importação da Tabela SIH para o MongoDB

A importação dos arquivos é feita através do comando fornecido junto com o mongo *mongoimport*, que aceita o formato CSV. Um exemplo de importação de arquivo CSV:

```
$mongoimport -d projeto -c sih --headerline --type=csv
arquivo.csv
```

onde os parâmetros possuem a função:

- -d - Especifica a base de dados do MongoDB em que a importação será feita, neste exemplo na base "projeto".
- -c - Especifica a coleção do MongoDB em que a importação será feita, neste exemplo na coleção "sih".
- –drop - Caso a coleção com o mesmo nome já exista, a exclui antes da importação. O não uso deste parâmetro em coleções já existentes causa a adição dos documentos importados na coleção.
- –headerline - Informa que a primeira linha do arquivo CSV é um cabeçalho, e os nomes dos campos especificados pelo cabeçalho serão usados como nomes de campos na coleção.

- –type - Especifica o tipo de arquivo do qual os dados serão importados, neste exemplo é "csv".

Devido ao fato de serem muitos arquivos, um script em Python foi usado para automatizar essa tarefa, e pode ser visto no repositório de códigos <https://github.com/velhoti/SIHSUS/tree/master/conversaoImportacaoSIH>. O script lista todos os arquivos da pasta atual em que foi executado, e caso possuam a extensão "CSV"ou "csv"o comando é invocado importando os dados para o MongoDB.

Todos os arquivos CSV referentes aos meses de um mesmo ano foram reunidos em uma pasta e o script foi executado usando para o nome da coleção o prefixo "sih", sendo que o ano é usado como sufixo do nome do arquivo (por ex. sih2011_csv). É importante ressaltar que no script o comando de importação foi usado sem o parâmetro --drop", dessa forma todos os dados mensais foram acumulados em uma única coleção, sem que esta fosse deletada a cada importação.

4.2.5 5º Passo - Conversão das Tabelas Auxiliares de DBF para CSV

As tabelas auxiliares fornecidas pelo DATASUS estavam em dois formatos, CNV e DBF.

A conversão das tabelas auxiliares de DBF para CSV é semelhante a conversão da tabela SIH, o procedimento se resume a colocar todas as tabelas no formato DBF em uma pasta comum e em seguida executar o script que converte todos os arquivos DBF da pasta para o formato CSV de forma automática.

4.2.6 6º Passo - Conversão das Tabelas Auxiliares de CNV para CSV

A conversão das tabelas auxiliares de CNV para CSV foi feita em lote, e consiste em colocar todos os arquivos do tipo CNV em uma pasta e em seguida executar o script que lista todos os arquivos com extensão "CNV"ou "cnv"da pasta e os converte. Como resultado são criados arquivos com o mesmo nome porém com extensão CSV. Internamente, o que o script usa é a biblioteca desenvolvida para essa tarefa e descrita em 4.1.3, executando a função "CNV2CSV"para cada arquivo listado.

4.2.7 7º Passo - Importação das Tabelas Auxiliares para o MongoDB

Existem dois tipos de tabelas auxiliares quanto a sua forma de importação para o MongoDB. O primeiro tipo são tabelas que estão contidas em um único arquivo, e o segundo são tabelas que estão espalhadas em vários arquivos, cada um contendo uma porção da tabela.

Após a conversão para o formato CSV, a importação para o MongoDB das tabelas auxiliares espalhadas em vários arquivos é idêntica à importação da tabelas SIH, e para tabelas em um único arquivo o processo é bastante parecido.

Para as tabelas contidas em arquivo único, o script executa o comando de importação com o parâmetro --drop", e dessa forma uma nova coleção no MongoDB é criada para cada tabela importada. O scripts usados se encontram em <https://github.com/velhoti/SIHSUS/tree/master/1-conversaoImportacaoCNV> e <https://github.com/velhoti/SIHSUS/tree/master/1-conversaoImportacaoDBF>.

Para tabelas separadas em vários arquivos, os arquivos precisam ser reunidos em uma pasta única e em seguida, um script que executa o comando do MongoDB de importação sem o parâmetro --drop"é invocado. A ausência deste parâmetro faz com que o conteúdo de cada arquivo seja acrescentado a uma única coleção do MongoDB sem deletá-la. Os scripts usados encontram-se em <https://github.com/velhoti/SIHSUS/tree/master/2-conversaoImportacaoCNV> e <https://github.com/velhoti/SIHSUS/tree/master/2-conversaoImportacaoDBF>.

4.2.8 8º Passo - Execução do Script com Pipeline de Tratamento de Dados do MongoDB

Uma vez que os dados das tabelas SIH e auxiliares já foram importados para as suas respectivas coleções no MongoDB, o próximo passo é executar o pipeline de tratamento de dados que produz uma coleção final onde se encontram todos os dados da SIH e os dados das tabelas auxiliares, armazenados sob a forma de subdocumentos nos seus respectivos campos. O script que contém esse processo inteiro pode ser visto no código 7 em anexo.

Abaixo (Código 4.1) é mostrado um script simplificado e ilustrativo, que executa o lookup apenas em um campo, usado para explicar os passos da transformação que é feita com os dados.

Na linha 7 é criado um índice na coleção SIH no campo "DIAG_PRINC"que melhora o desempenho da operação de lookup, porém, apenas possui efeito na primeira vez que é feito um lookup. Após isso, os resultados são armazenados em tabelas e estruturas intermediáris do MongoDB, onde não existem mais índices. Para aproveitar o ganho de desempenho da indexação, a operação de lookup mais dispendiosa computacionalmente é realizada primeiro.

Código 4.1 – Código simplificado e ilustrativo do pipeline de tratamento de dados no MongoDB.

```
1  from pymongo import MongoClient
2
3  #pega a database
4  projeto=MongoClient().projeto
5
6  #indexa o campo que sofrera a primeira juncao, que deve ser a juncao
       mais cara computacionalmente
7  projeto.sih_pura.create_index("DIAG_PRINC")
8
9  #array com as juncoes a serem feitas
10 pipeline=[]
11
12 #projecao que seleciona campos usados
13 camposUsados={'UF_ZI':1,'ANO_CMPT':1,'MES_CMPT':1,'ESPEC':1,'CGC_HOSP'
       :1,'N_AIH':1,'IDENT':1,'CEP':1,'MUNIC_RES':1,...}
14 project={ "$project" : camposUsados}
15 pipeline.append(project)
16
17 #juncao de DIAG_PRINC - CID10.CNV
18 projeto['cid10_cnv'].create_index("codigo")
19 lookup={"$lookup": { "from": 'cid10_cnv', "localField": "DIAG_PRINC", "
       foreignField": "codigo", "as": "DIAG_PRINC"}}
20 unwind={ "$addFields": {"DIAG_PRINC": { "$arrayElemAt": [ "$DIAG_PRINC",
        0 ] }}}
21 pipeline.append(lookup)
22 project={ "$project" : {"DIAG_PRINC._id" : 0}}
23
24 pipeline.append(unwind)
25 pipeline.append(project)
26
27 out={ "$out" : "sih15" }
28 pipeline.append(out)
29
30 #executa a agregacao, onde ocorrem as juncoes
31 projeto.sih1pura.aggregate(pipeline)
```

Na linha 10 é criado um array com o pipeline de tratamento, nele serão inseridas as operações do processo.

Nas linhas 13, 14 e 15 os campos, selecionados pela equipe especialista em saúde pública, são explicitados e uma operação de projeção produz como saída uma nova tabela apenas com os campos escolhidos.

Nas linhas 18 e 19 é feita a indexação do campo de junção da coleção auxiliar CID10, e em seguida a operação de lookup, que incorpora o elemento desta coleção na SIH, sob

a forma de um subdocumento.

A operação de lookup é semelhante a uma junção e produz no campo a criação e um subdocumento com os dados da tabela auxiliar. Porém no MongoDB, ao invés de um simples subdocumento é criado um array com todos que obedecem ao campo do lookup. Na linha 20, a operação de unwind é feita para se extrair o primeiro elemento do array.

Na linha 22, uma nova operação de projeção é realizada, mas para retirar o campo "_id"present no subdocumento. Este campo é um metadado usado pelo MongoDB em situações como indexação, mas não é utilizado no presente trabalho.

A linha 27 estabelece que o resultado final da transformação deve ser armazenado em uma nova coleção com o nome "sih15".

As linhas 15, 21, 24, 25 e 28 são usadas para inserir no array do pipeline as operações a serem realizadas, sendo que a ordem em que são executadas afeta o resultado.

Por fim, a linha 31 executa o pipeline com todas as operações. O script usado é semelhante a esse mas repete 54 vezes as operações de lookup, unwind e project, cada repetição para um dos campos da SIH que possuem tabela auxiliar.

5 RESULTADOS

Todos os passos e processos descritos no capítulo 4 foram executados para seis conjuntos de dados, que correspondem a tabela SIH nos anos de 2010 a 2015. O procedimento só foi realizado em uma amostra que corresponde aos dados do estado do Rio de Janeiro, porém a metodologia estabelecida pode ser empregada para a construção de bases com dados de todos os estados brasileiros e outros períodos de tempo.

Para cada ano foi construída uma coleção no MongoDB sendo cada documento um registro de uma internação pelo SUS, e códigos que faziam referência a tabelas auxiliares se transformaram em subdocumentos. Os campos existentes podem ser vistos na tabela 6 em anexo, porém alguns documentos podem não ter alguns campos pois os registros de internações não foram totalmente preenchidos. Além disso existem campos que não existem em alguns anos. O código do anexo 7 contém o exemplo de um registro (documento no MongoDB) de internação de 2015.

Na tabela 1 estão os tempos aproximados gasto na execução do ETL para cada ano, assim como o tamanho final da coleção no MongoDB e a quantidade de documentos. Os tempos são apenas relativos ao script final de junção, não sendo consideradas as conversões e as importações para o MongoDB.

Tabela 1 – Tempos, tamanho e quantidade de documentos para os ETLs da tabela SIH entre os anos 2010-2015.

Ano	Tempo (min)	Documentos	Tamanho (GB)
2010	77	735294	1,66
2011	80	724421	1,66
2012	76	688578	1,60
2013	74	672708	1,60
2014	76	684037	1,63
2015	79	720322	1,73

Para ilustrar o uso e a utilidade da base obtida, foram realizadas algumas consultas que podem ocorrer em situações práticas. Para este procedimento foi usado o terminal do MongoDB. Também foram medidos os tempos que levaram as consultas, e observou-se que houve grande variação. A partir das medidas cogita-se que a primeira consulta é mais demorada, porém uma vez que partes ou a totalidade da coleção já está na memória as próximas consultas podem ser até 10 vezes mais rápidas. Os tempos medidos expostos aqui são sempre os maiores obtidos, ou seja, refletem o pior caso medido.

Primeira consulta: quantidade de internações cujo diagnóstico é dengue clássica (código CID10 A90). A consulta é mostrada logo abaixo. Os resultados e tempos podem ser vistos na tabela 2.

```
db.sih.aggregate([ {$match: {'DIAG_PRINC.codigo':"A90"}},
{$group: { _id: null, total: { $sum: 1 } } } ])
```

Tabela 2 – Quantidade de internações com diagnóstico dengue clássica no estado do Rio de Janeiro entre os anos 2010-2015.

Ano	Internações (dengue clássica)	Tempo (ms)
2010	8198	11098
2011	8943	6684
2012	3558	8854
2013	4035	7931
2014	494	8858
2015	2505	9793

Segunda consulta: quantidade de internações em 2015 agregada por cada diagnóstico, ordenada em forma decrescente e mostrando apenas os três primeiros resultados, ou diagnósticos com maior número de internações. Esta consulta levou 9897 milisegundos para ser executada. A consulta é mostrada logo abaixo.

```
db.sih.aggregate([{$group: { _id: '$DIAG_PRINC.valor',
total: { $sum: 1 } } }, {$sort:{"total":-1}},{ $limit: 3 }])
```

O resultado pode ser visto na figura 5. Conclui-se que os diagnósticos de internações mais numerosos, em ordem decrescente, são: Parto espontâneo cefálico, Parto único espontâneo e Pneumonia NE (5).

Figura 5 – Quantidade de internações ordenadas de forma decrescente, por diagnóstico em 2015.

```
{ "_id" : "O80.0 Parto espontaneo cefalico", "total" : 66499 }
{ "_id" : "O80.9 Parto unico espontaneo NE", "total" : 14746 }
{ "_id" : "J18.9 Pneumonia NE", "total" : 14393 }
```

Terceira consulta: constituída de duas subconsultas, a idade média dos pacientes internados com diagnóstico Pneumonia NE (código CID10 J18.9) e a a idade média dos pacientes internados com diagnóstico Pneumonia NE que vieram a óbito. As duas consultas podem ser vistas logo abaixo.

```
db.sih.aggregate([ {$match: {'DIAG_PRINC.codigo':"J189"}},
{$group: { _id: null, idade_media: { $avg : '$IDADE' } } } ])
```

```
db.sih.aggregate([ {$match:{ $and: [{'DIAG_PRINC.codigo':"J189"
    },
{'MORTE.codigo':1}]}}, {$group: { _id: null, idade_media:
{ $avg : '$IDADE' } } } ])
```

As consultas foram executadas para cada ano entre 2010 e 1015, resultados anuais e tempos estão expostos na tabela 3.

Tabela 3 – Idade média de pacientes em internações com diagnóstico Pneumonia NE.

Ano	Idade média	Tempo (ms)	Idade média (com óbito)	Tempo (ms)
2010	35,26	5958	69,91	12523
2011	38,06	7781	69,07	7724
2012	36,75	9172	69,00	9063
2013	39,61	8065	70,57	8483
2014	41,61	8914	70,37	9339
2015	40,52	10147	70,58	9946

6 CONCLUSÃO

Os dados originais na forma que são disponibilizados pelo portal DATASUS são provenientes de sistemas legados e por vezes possuem formato ou codificação próprios, estas características dificultam ou impedem que sejam aproveitados e integrados em diversos outros sistemas ou ferramentas analíticas. A transformação da base para um formato (coleções no MongoDB) mais atual e com padrão popular é um passo no sentido de permitir o uso mais amplo e integrado dos dados em novas aplicações.

Um possível uso da base construída no presente trabalho é em visualização de dados. Existem alguns sistemas de relatório como o Tableau[1] que podem ser integrados no MongoDB para a geração de painéis. Também é possível o uso de bibliotecas como a D3.js[2] para gerar as visualizações.

Um grande número de linguagens de programação possuem bibliotecas para integração com o MongoDB, possibilitando a construção de sistemas diversos. Em especial, sistemas web tendem a se beneficiar com isso, uma vez que o MongoDB proporciona um armazenamento JSON-like, que é adequado a forma como esses sistemas transmitem os dados.

Uma aplicação particularmente interessante é o uso dos dados como fonte para algoritmos de inteligência computacional, mineração de dados e estatística, a fim de se extrair conhecimento. Era mais trabalhoso e menos eficiente lidar com os dados no formato original, enquanto a base no MongoDB oferece uma forma mais fácil e um ambiente projetado para ter agilidade, eficiência e escalabilidade em consultas analíticas.

Um outro aspecto muito dificultado pela forma original dos dados é a integração entre diferentes bases. Não era possível juntar dados das tabelas de nascimento (SINASC), mortalidade (SIM) e internações (SIH) para tentar traçar um histórico médico de um indivíduo. Com a migração para o MongoDB essa tarefa se tornou possível.

Além de dados provenientes do DATASUS como a tabela de nascimentos SINASC, a forma atual dos dados no MongoDB também tende a ajudar em integrações com outras bases de informações socioeconômicas, tais como o censo demográfico do Instituto Brasileiro de Geografia e Estatística (IBGE)[3], bolsa família, renda per capta, avaliações de dados demográficos, etc.

[1] <https://www.tableau.com/pt-br>
[2] <https://d3js.org/>
[3] <https://ww2.ibge.gov.br/>

7 TRABALHOS FUTUROS

As tarefas de fazer download dos arquivos com a base SIH do DATASUS e a conversão de DBC para DBF, através do software Tabwin, foram feitas manualmente. O trabalho manual é aceitável dado que apenas foi feito o ETL dos dados do estado do Rio de Janeiro, entre os anos de 2010 a 2015. Para fazer o ETL de todos os estados do Brasil, assim como das tabelas dos próximos anos, será necessário automatizar essas tarefas e encapsulá-las em um script único que executa todos os passos do processo.

Uma vez que a base está disponível em ambiente adequado para consultas analíticas, seria proveitoso gerar visualizações de dados agregados disponibilizando-os ao público. Também podem ser projetados sistemas voltados para gestores da saúde pública com dados menos agregados e carregando informações úteis para tarefas administrativas e de acompanhamento da saúde pública. Existem diferentes bibliotecas de visualização de dados e ferramentas que podem ser usadas para gerar tais sistemas. Destaca-se ainda o uso de um banco de dados orientado a documento (MongoDB), que facilita a integração com sistemas web pois estes usam majoritariamente a linguagem JavaScript, logo a base é propícia para a construção de sistemas web com visualização dos dados.

A metodologia aqui proposta também pode ser usada para outras bases com conteúdo sócio econômico como bolsa família, renda per capta, demografia, etc., sendo necessárias algumas adaptações. O uso da metodologia em outras bases pode ser entendido como o uso das mesmas ferramentas e da mesma abordagem, a fim de se obter como resultado uma coleção no MongoDB com organização semelhante a usada no SIH. Tais bases extras, em conjunto com as bases do SUS, podem ajudar a determinar a influência de fatores socioeconômicos na saúde pública.

Uma vez que estejam disponíveis bases de conteúdo sócio econômico, sejam elas obtidas com a metodologia do presente trabalho ou não, seria útil fazer o cruzmento de dados destas tabelas com a SIH obtida para enriquecer os registros de internações da população em questão. Isso poderia aumentar o poder de análise e gerar novos insight sobre a saúde de populações baseados em critérios regionais como renda, clima, densidade demográfica e outros.

As tabelas SIH de anos diferentes podem possuir alguns campos extras ou faltando. Pode ainda ocorrer de campos serem fundidos em um único ou se separarem em vários. Até o momento é necessário que o usuário da base tenha conhecimento da semântica de cada campo e leve isso em consideração ao fazer a consulta em anos diferentes. Mas seria útil se no ETL isso fosse considerado e campos fossem separados ou unificados, oferecendo uma base para o usuário com semântica uniforme entre anos diferentes.

Para traçar um histórico de saúde de um indivíduo desde o seu nascimento, passando pelas internações e possivelmente o seu falecimento, é necessário fazer junções entre as três

tabelas do DATASUS: SIM, SINASC e SIH. A junção entre as tabelas SIM e SINASC é feita através do campo em comum Número da Declaração de Nascimento (NUMERODN) (SANTOS; CARVALHO, 2017). Porém, a tabela SIH não possui tal campo de junção, e para obter as internações de um indivíduo identificado na SIM e SINASC é necessário usar técnicas de linkage de dados (CHRISTEN, 2012), que consideram outros campos como nome, data de nascimento e nome da mãe para estabelecer de forma probabilística junções entre registros de tabelas diferentes. Já é notável a utilidade destas técnicas para dados de saúde pública ((COELI, 2015)). Apesar das limitaçoes e restrições que a base anonimizada possui, o trabalho futuro é aproveitar o ambiente otimizado para consultas para aplicar técnicas de linkage de dados.

Ainda que as bases usadas neste trabalho sejam anonimizadas e públicas, sempre se deve ter em mente a privacidade e o sigilo dos dados, especialmente em dados altamente sensíveis como registros médicos. Supõe-se que não seja possível retirar o anonimato mas considerando a otimização de consultas alcançada e o poder de algoritmos de mineração de dados e inteligência computacional, é necessário o uso cauteloso e responsável para evitar que quaisquer cruzamentos possam violar a privacidade revelando condições clínicas de pacientes a terceiros, ou pior, a empresas ou entidades que façam uso dos dados em benefício próprio, prejudicando-os. Por isso a base não deve ser aberta para consultas indiscriminadas pelo público, recomenda-se apenas o uso e acesso por pesquisadores, profissionais de saúde ou gestores com interesses únicos na saúde pública. Pode-se ainda disponibilizar sistemas que forneçam dados agregados de forma criteriosa mas é necessário cuidado para não permitir a livre consulta.

REFERÊNCIAS

BARBIERI, C. **BI2: Business Intelligence: modelagem e qualidade**. Rio de Janeiro: Elsevier Editora LTDA, 2011. 392 p.

BONNET, L. et al. Reduce, you say: What nosql can do for data aggregation and bi in large repositories. **22nd International Workshop on Database and Expert Systems Applications. IEEE**, p. 483–488, 2011.

CAO, Y. et al. Es 2: A cloud data storage system for supporting both oltp and olap. **IEEE 27th International Conference on Data Engineering**, p. 291–302, 2011.

CARNIEL, A. C. et al. Query processing over data warehouse using relational databases and nosql. **XXXVIII Conferencia Latinoamericana En Informatica (CLEI). IEEE**, p. 1–9, 2012.

CHEVALIER, M. et al. Implementing multidimensional data warehouses into nosql. **17th International Conference on Enterprise Information Systems (ICEIS 2015) held in conjunction with ENASE 2015 and GISTAM**, p. 108–130, 2015.

CHRISTEN, P. **Data matching: concepts and techniques for record linkage, entity resolution, and duplicate detection**. [S.l.: s.n.], 2012. 1349-1350 p.

COELI, C. M. **A qualidade do linkage de dados precisa de mais atenção**. [S.l.]: SciELO Public Health, 2015.

DOUG, L. 3d data management: Controlling data volume, velocity and variety. **META group research note**, v. 6, n. 70, p. 1, 2001.

FOX, A.; BREWER, E. A. Harvest, yield, and scalable tolerant systems. **Proceedings of the Seventh Workshop on Hot Topics in Operating Systems. IEEE**, p. 174–178, 1999.

GEATER, J. **O que é a extensão dearquivo DBC?** 2011. Disponível em:<https://www.solvusoft.com/pt-br/file-extensions/file-extension-dbc/>. Acessado em: 17 nov. 2018.

MongoDB Development Team. **MongoDB**. 2019. MongoDB. Disponível em:<https://www.mongodb.com/>. Acessado em: 02 mai. 2019.

MongoDB Development Team. **MongoDB JSON-BSON**. 2019.

MongoDB Development Team. **MongoDB Pipeline**. 2019.

MongoDB Development Team. **MongoDB Sharded Cluster**. 2019. <https://docs.mongodb.com/manual/tutorial/deploy-shard-cluster/>. Acessado em: 04 ago. 2019.

MongoDB Development Team. **Use Case Guidance: Selecting the Best Apps for MongoDB, and when to evaluate other options**. [S.l.], 2019.

PyMongo Development Team. **PyMongo**. PyMongo. Documentação disponível em:<https://api.mongodb.com/python/current/>. Acessado em: 02 mai. 2019.

Python Development Team. **Python EOL**. 1990. Anúncio do fim de vida do Python2.7. Disponível em:<https://legacy.python.org/dev/peps/pep-0373/>. Acessado em: 17 nov. 2018.

Python Development Team. **Python**. 2001. Disponível em:<https://www.python.org/>. Acessado em: 17 nov. 2018.

RAMEZ, E.; SHAMKANT, N. B. **Fundamentals of Database Systems**. 6. ed. Massachusetts: Addison-Wesley, 2010. 1172 p.

SANTOS, I. A. dos; CARVALHO, R. O. de. **Ambiente de exploração de dados de saúde usando um Banco de Dados NoSQL**. 66 f. Monografia (Trabalho de Conclusão de Curso) — Departamento de Ciência da Computação-Instituto de Matemática, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2017.

SMYSHLIAEV, A.; SAMCHUCK, Y. **dbfpy**. 2005. Biblioteca dbfpy. Disponível em:<http://sourceforge.net/projects/dbfpy/files/latest/download?source=files>. Acessado em: 17 nov. 2018.

SPAAN, B. **DBF2CSV**. 2019. Script DBF2CSV. Disponível em:<https://gist.github.com/bertspaan/8220892>. Acessado em: 4 ago. 2019.

SUS. **DATASUS**. 2008. Portal de saúde SUS. Disponível em:<http://www2.datasus.gov.br/DATASUS/index.php>. Acessado em: 17 nov. 2018.

SUS. **Sistema de Informações Hospitalares (SIH)**. 2019. Sistema de Informações Hospitalares (SIH). Disponível em:<http://datasus.saude.gov.br/sistemas-e-aplicativos/hospitalares/sihsus>. Acessado em: 04 ago. 2019.

Tabwin Development Team. **Tabwin**. 2008. Sistema de tabulação Tabwin. Disponível em:<http://www2.datasus.gov.br/DATASUS/index.php?area=060805>. Acessado em: 17 nov. 2018.

WHATIS.COM. **DBF File Format**. 1999. Disponível em:<https://whatis.techtarget.com/fileformat/DBF-dBASE-file>. Acessado em: 17 nov. 2018.

WU, C. M.; HUANG, Y. F.; LEE, J. Comparisons between mongodb and ms-sql databases on the twc website. **American Journal of Software Engineering and Applications.**, v. 4, n. 3, p. 35–41, 2015.

ANEXOS

ANEXO A – COMPLETUDE DOS DADOS PARA A BASE SIH DE 2010

Tabela 4 – Completude de dados para a base SIH de 2010.

Campo	Completude	Campo	Completude
UF_ZI	1	NATUREZA	1
ANO_CMPT	1	GESTAO	1
MES_CMPT	1	RUBRICA	1
ESPEC	1	IND_VDRL	1
CGC_HOSP	0.88	MUNIC_MOV	1
N_AIH	1	COD_IDADE	1
IDENT	1	IDADE	1
CEP	1	DIAS_PERM	1
MUNIC_RES	1	MORTE	1
NASC	1	NACIONAL	1
SEXO	1	NUM_PROC	0
UTI_MES_IN	1	CAR_INT	1
UTI_MES_AN	1	TOT_PT_SP	1
UTI_MES_AL	1	CPF_AUT	0
UTI_MES_TO	1	HOMONIMO	1
MARCA_UTI	1	NUM_FILHOS	1
UTI_INT_IN	1	INSTRU	1
UTI_INT_AN	1	CID_NOTIF	0
UTI_INT_AL	1	CONTRACEP1	1
UTI_INT_TO	1	CONTRACEP2	1
DIAR_ACOM	1	GESTRISCO	1
QT_DIARIAS	1	INSC_PN	1
PROC_SOLIC	1	SEQ_AIH5	1
PROC_REA	1	CBOR	1
VAL_SH	1	CNAER	1
VAL_SP	1	VINCPREV	1
VAL_SADT	1	GESTOR_COD	0
VAL_RN	1	GESTOR_TP	1
VAL_ACOMP	1	GESTOR_CPF	1
VAL_ORTP	1	GESTOR_DT	0
VAL_SANGUE	1	CNES	1

VAL_SADTSR	1	CNPJ_MANT	0.45
VAL_TRANSP	1	INFEHOSP	0
VAL_OBSANG	1	CID_ASSO	0.01
VAL_PED1AC	1	CID_MORTE	0.05
VAL_TOT	1	COMPLEX	1
VAL_UTI	1	FINANC	1
US_TOT	1	FAEC_TP	0.02
DT_INTER	1	REGCT	1
DT_SAIDA	1	RACA_COR	1
DIAG_PRINC	1	ETNIA	0.24
DIAG_SECUN	0.13	SEQUENCIA	1
COBRANCA	1	REMESSA	1

ANEXO B – DOMÍNIO DOS DADOS PARA A BASE SIH DE 2010

Tabela 5 – Domínio dos dados para a base SIH de 2010.

Campo	Mínimo	Máximo
UF_ZI	330000	330630
ANO_CMPT	2010	2010
MES_CMPT	1	12
ESPEC	1	14
CGC_HOSP	254512000184	72386212000160
N_AIH	3307105070533	9910300011832
IDENT	1	5
CEP	1001000	99990000
MUNIC_RES	110002	530180
NASC	18990624	20101230
SEXO	1	3
UTI_MES_IN	0	0
UTI_MES_AN	0	0
UTI_MES_AL	0	0
UTI_MES_TO	0	209
MARCA_UTI	0	99
UTI_INT_IN	0	0
UTI_INT_AN	0	0

UTI_INT_AL	0	0
UTI_INT_TO	0	170
DIAR_ACOM	0	345
QT_DIARIAS	0	345
PROC_SOLIC	201010038	506020045
PROC_REA	201010038	506020045
VAL_SH	15.4	98499.49
VAL_SP	1.39	15288.76
VAL_SADT	0	0
VAL_RN	0	0
VAL_ACOMP	0	0
VAL_ORTP	0	0
VAL_SANGUE	0	0
VAL_SADTSR	0	0
VAL_TRANSP	0	0
VAL_OBSANG	0	0
VAL_PED1AC	0	0
VAL_TOT	20.74	112041.41
VAL_UTI	0	100708.7
US_TOT	11.45	65140.35
DT_INTER	20000801	20101231
DT_SAIDA	20090801	20101231
DIAG_PRINC	A000	Z988
DIAG_SECUN	A000	Z948
COBRANCA	11	67
NATUREZA	20	61
GESTAO	1	2
RUBRICA	0	0
IND_VDRL	0	1
MUNIC_MOV	330010	330630
COD_IDADE	2	5
IDADE	0	99
DIAS_PERM	0	345
MORTE	0	1
NACIONAL	10	348
NUM_PROC	None	None
CAR_INT	1	6
TOT_PT_SP	0	0
CPF_AUT	None	None

HOMONIMO	0	2
NUM_FILHOS	0	37
INSTRU	0	4
CID_NOTIF	O100	Z302
CONTRACEP1	0	12
CONTRACEP2	0	12
GESTRISCO	0	1
INSC_PN	0	9010206600
SEQ_AIH5	0	7
CBOR	0	841745
CNAER	0	950
VINCPREV	0	5
GESTOR_COD	None	None
GESTOR_TP	0	1
GESTOR_CPF	0	98617729791
GESTOR_DT	None	None
CNES	12505	6586767
CNPJ_MANT	394544017150	60922168000186
INFEHOSP	None	None
CID_ASSO	A049	Z992
CID_MORTE	A000	Z988
COMPLEX	2	3
FINANC	4	6
FAEC_TP	40018	40049
REGCT	0	7110
RACA_COR	1	99
ETNIA	0	223
SEQUENCIA	1	33191

ANEXO C – CAMPOS APROVEITADOS COM AS RESPECTIVAS TABELAS AUXILIARES DA BASE SIH DE 2010

Tabela 6 – Campos da SIH escolhidos para carga no MongoDB.

Campo	Tabela Auxiliar

UF_ZI	br_municgestor.cnv
ANO_CMPT	
MES_CMPT	
ESPEC	LEITOS.CNV
CGC_HOSP	TCHBR.DBF
N_AIH	
IDENT	IDENT.CNV
CEP	
MUNIC_RES	br_municip.cnv
NASC	
SEXO	SEXO.CNV
UTI_MES_TO	DIARIASUTI.CNV
MARCA_UTI	MARCAUTI.CNV
UTI_INT_TO	
DIAR_ACOM	
QT_DIARIAS	
PROC_REA	TB_SIGTAP.DBF
VAL_SH	
VAL_SP	
VAL_TOT	
VAL_UTI	
US_TOT	
DT_INTER	
DT_SAIDA	
DIAG_PRINC	CID10_01.CNV até CID10_22.CNV
DIAG_SECUN	CID10_01.CNV até CID10_22.CNV
COBRANCA	SAIDAPERM.CNV
NATUREZA	NATUREZA.CNV
NAT_JUR	NATJUR.CNV
GESTAO	GESTAO.CNV
IND_VDRL	SIMNAO.CNV
MUNIC_MOV	br_municip.cnv
COD_IDADE	IDADEBAS.CNV
IDADE	
DIAS_PERM	PERM.CNV
MORTE	MORTES.CNV
NACIONAL	NACION3D.CNV
CAR_INT	CARATEND.CNV

HOMONIMO	
NUM_FILHOS	NUMFILH.CNV
INSTRU	INSTRU.CNV
CID_NOTIF	VCID10_01.CNV até CID10_22.CNV
CONTRACEP1	CONTRAC.CNV
CONTRACEP2	CONTRAC.CNV
GESTRISCO	SIMNAO.CNV
SEQ_AIH5	SEQAIH.CNV
CBOR	CBO2002.CNV
CNAER	CNAEG.CNV, CNAE.CNV
VINCPREV	VINCPREV.CNV
CNES	TCNESBR.DBF
CID_ASSO	CID10_01.CNV até CID10_22.CNV
CID_MORTE	CID10_01.CNV até CID10_22.CNV
COMPLEX	COMPLEX2.CNV
RACA_COR	RACACOR.CNV
ETNI	ETNIA.CNV
AVAL_UCI	
MARCA_UCI	
DIAGSEC1	cid10.dbf
DIAGSEC2	cid10.dbf
DIAGSEC3	cid10.dbf
DIAGSEC4	cid10.dbf
DIAGSEC5	cid10.dbf
DIAGSEC6	cid10.dbf
DIAGSEC7	cid10.dbf
DIAGSEC8	cid10.dbf
DIAGSEC9	cid10.dbf
TPDISEC1	TP_DIAGSEC.CNV
TPDISEC2	TP_DIAGSEC.CNV
TPDISEC3	TP_DIAGSEC.CNV
TPDISEC4	TP_DIAGSEC.CNV
TPDISEC5	TP_DIAGSEC.CNV
TPDISEC6	TP_DIAGSEC.CNV
TPDISEC7	TP_DIAGSEC.CNV
TPDISEC8	TP_DIAGSEC.CNV
TPDISEC9	TP_DIAGSEC.CNV

ANEXO D – EXEMPLO DE REGISTRO NO MONGODB DA BASE SIH.


```
{
'NATUREZA': {
        'codigo': 40,
        'valor': 'Ignorado'
},
'GESTAO': {
        'codigo': 2,
        'valor': 'Estadual plena'
},
'TPDISEC1': {
        'codigo': 2,
        'valor': 'Adquirido'
},
'VAL_UCI': 0.0,
'ANO_CMPT': 2015,
'VINCPREV': {
        'codigo': 0,
        'valor': 'Não classificado'
},
'SEQ_AIH5': {
        'codigo': 0,
        'valor': 'Sequencial zerado'
},
'UTI_MES_TO': {
        'codigo': 0,
        'valor': 'Não'
},
'COMPLEX': {
        'codigo': 2,
        'valor': 'Não se aplica'
},
'NAT_JUR': {
        'codigo': 1023,
        'valor': '1. Administração Pública'
},
'DIAG_PRINC': {
        'codigo': 'T850',
        'valor': 'T85.0 Complic mecanica shunt ventric intracraniano'
},
'NACIONAL': {
        'codigo': 10,
        'valor': 'Brasil'
```

```
},
'MARCA_UTI': {
        'codigo': 0,
        'valor': 'Não utilizou UTI'
},
'ESPEC': {
        'codigo': 1,
        'valor': '01-Cirúrgico'
},
'NUM_FILHOS': {
        'codigo': 0,
        'valor': 'Sem filhos/Não inform'
},
'IDADE': 8,
'MUNIC_RES': {
        'codigo': 330250,
        'valor': '330250 Magé'
},
'CID_MORTE': 0,
'CNES': {
        'NOMEFANT': 'SES RJ HOSPITAL ESTADUAL ADAO PEREIRA NUNES',
        'CMPT': 201803,
        'UF_ZI': 33,
        'CNES': 2290227
},
'PROC_REA': {
        'COMPET': 201803,
        'IP_DSCR': 'TRATAMENTO C/ CIRURGIAS MULTIPLAS',
        'IP_COD': 415010012
},
'DT_INTER': 20150401,
'RACA_COR': {
        'codigo': 2,
        'valor': 'Sem informação'
},
'COD_IDADE': {
        'codigo': 4,
        'valor': 'Ign'
},
'N_AIH': 3315103625361L,
'CEP': 25916488,
'ETNIA': {
        'codigo': 0,
        'valor': 'NÃO INFORMADO'
},
'CNAER': {
        'codigo': 0,
```

```
        'valor': 'Não classificado'
},
'DIAGSEC1': {
        'CD_COD': 'X599',
        'CD_DESCR': 'X59.9 Local NE'
},
'DIAR_ACOM': 31,
'IDENT': {
        'codigo': 1,
        'valor': 'Normal'
},
'DT_SAIDA': 20150502,
'CID_ASSO': 0,
'MES_CMPT': 6,
'MUNIC_MOV': {
        'codigo': 330170,
        'valor': '330170 Duque de Caxias'
},
'SEXO': {
        'codigo': 1,
        'valor': 'Ignorado'
},
'MARCA_UCI': 0,
'HOMONIMO': 0,
'UF_ZI': {
        'codigo': 330000,
        'valor': '330000 Rio de Janeiro - Gestão estadual'
},
'COBRANCA': {
        'codigo': 26,
        'valor': 'Permanência por mudança de procedimento'
},
'QT_DIARIAS': 32,
'IND_VDRL': {
        'codigo': 0,
        'valor': 'Não'
},
'CAR_INT': {
        'codigo': 2,
        'valor': '02 Urgência'
},
'CONTRACEP1': {
        'codigo': 0,
        'valor': 'Ignorado/não se aplica'
},
'GESTRISCO':
        {'codigo': 1,
```

```
        'valor': 'Sim'
},
'VAL_SH': 2412.64,
'CID_NOTIF': {
        'codigo': '', 'valor': 'Não preenchido'
},
'CONTRACEP2': {
        'codigo': 0,
        'valor': 'Ignorado/não se aplica'
},
'US_TOT': 934.52,
'VAL_SP': 755.41,
'UTI_INT_TO': 0,
'NASC': 20060503,
'INSTR': {
        'codigo': 0,
        'valor': 'Ignorado/não se aplica'
},
'CGC_HOSP': {
        'UF_ZI': 12,
        'CMPT': 201502,
        'RAZAO': 'SEM CNPJ PROPRIO',
        'CGC_HOSP': ''
},
'CBOR': {
        'codigo': 0,
        'valor': 'Não informado'
},
'VAL_TOT': 3168.05,
'DIAG_SECUN': {
        'codigo': 0,
        'valor': 'Não preenchido'
},
'MORTE': {
        'codigo': 0,
        'valor': 'Sem óbito'
},
'VAL_UTI': 0.0
}
```


ANEXO E – SCRIPT DE CONVERSÃO CNV2CSV

```
import codecs
```

```
def CNV2CSV(fileName):
        countLinha=1
        nomeSaida=fileName.split('.')[0]+'.csv'
        saida=open(nomeSaida,'w')
        saida.write('valor,codigo\n')
        with codecs.open(fileName,'r',encoding='iso8859-1') as f:
                for line in f:
                        if(countLinha>1):
                                #line=line.strip(' ')
                                line=line.strip('\n')
                                line=line.strip('\r')
                                line=line.strip(',')
                                campos=divide_duas_colunas(line)
                                campos[0]=remove_contagem(campos[0])
                                campos[1]=campos[1].strip('␣')
                                if(campos!=[]):
                                        camposFinal=pegaSubCampos(campos
                                            )
                                        for linhaSaida in camposFinal:
                                                arraySaida=juntaLinha(
                                                    linhaSaida)
                                                saida.write(arraySaida)
                        countLinha=countLinha+1


def remove_contagem(campo):
        primeiro=0
        campo=campo.strip('␣')
        for i in range(len(campo)):
                if(campo[i]=='␣'):
                        primeiro=i
                        break
        for i in range(primeiro,len(campo)):
                if(campo[i]!='␣'):
                        primeiro=i
                        break
        return campo[primeiro:len(campo)]


def divide_duas_colunas(line):
        return [line[0:59],line[59:len(line)]]

def juntaLinha(array):
        arraySaida=''
        cont=0
        for i in array:
```

```
                if(cont!=0):
                        arraySaida=arraySaida+','+i
                else:
                        arraySaida=arraySaida+i
                cont=cont+1
        arraySaida=arraySaida+'\n'
        return arraySaida

def pegaSubCampos(array):
        retorno=[]
        codigos=array[len(array)-1].split(',')
        for codigo in codigos:
                limites=codigo.split('-')
                if(len(limites)==2 ):
                        for i in range(int(limites[0]),int(limites[1])
                           +1):
                                retorno.append([array[0],str(i)])
                elif(len(limites)==1):
                        retorno.append([array[0],limites[0]])
        return retorno
```

ANEXO F – SCRIPT PRINCIPAL DO ETL: JUNCOESSIH

```
from pymongo import MongoClient

#pega a database
projeto=MongoClient().projeto2

#indexa o campo que sofrera a primeira juncao, que deve ser a juncao
   mais cara computacionalmente
projeto.sih15_pura.create_index("DIAG_PRINC")

#array com as juncoes a serem feitas
pipeline=[]

#projecao que seleciona campos usados
camposUsados={'UF_ZI':1,'ANO_CMPT':1,'MES_CMPT':1,'ESPEC':1,'CGC_HOSP'
   :1,'N_AIH':1,'IDENT':1,'CEP':1,'MUNIC_RES':1,'NASC':1,'SEXO':1,'
   UTI_MES_TO':1,'MARCA_UTI':1,'UTI_INT_TO':1,'DIAR_ACOM':1,'QT_DIARIAS'
   :1,'PROC_REA':1,'VAL_SH':1,'VAL_SP':1,'VAL_TOT':1,'VAL_UTI':1,'US_TOT
   ':1,'DT_INTER':1,'DT_SAIDA':1,'DIAG_PRINC':1,'DIAG_SECUN':1,'COBRANCA
   ':1,'NATUREZA':1,'NAT_JUR':1,'GESTAO':1,'IND_VDRL':1,'MUNIC_MOV':1,'
   COD_IDADE':1,'IDADE':1,'DIAS_PERM':1,'MORTE':1,'NACIONAL':1,'CAR_INT'
```

```
    :1,'HOMONIMO':1,'NUM_FILHOS':1,'INSTRU':1,'CID_NOTIF':1,'CONTRACEP1'
    :1,'CONTRACEP2':1,'GESTRISCO':1,'SEQ_AIH5':1,'CBOR':1,'CNAER':1,'
    VINCPREV':1,'CNES':1,'CID_ASSO':1,'CID_MORTE':1,'COMPLEX':1,'RACA_COR
    ':1,'ETNIA':1,'VAL_UCI':1,'MARCA_UCI':1,'DIAGSEC1':1,'DIAGSEC2':1,'
    DIAGSEC3':1,'DIAGSEC4':1,'DIAGSEC5':1,'DIAGSEC6':1,'DIAGSEC7':1,'
    DIAGSEC8':1,'DIAGSEC9':1,'TPDISEC1':1,'TPDISEC2':1,'TPDISEC3':1,'
    TPDISEC4':1,'TPDISEC5':1,'TPDISEC6':1,'TPDISEC7':1,'TPDISEC8':1,'
    TPDISEC9':1}
project={ "$project" : camposUsados}
pipeline.append(project)

#juncao de DIAG_PRINC - CID10.CNV
projeto['cid10_cnv'].create_index("codigo")
lookup={"$lookup": { "from": 'cid10_cnv', "localField": "DIAG_PRINC", "
    foreignField": "codigo", "as": "DIAG_PRINC"}}
unwind11={ "$addFields": {"DIAG_PRINC": { "$arrayElemAt": [ "$DIAG_PRINC
    ", 0 ] }}}
pipeline.append(lookup)
project11={ "$project" : {"DIAG_PRINC._id" : 0}}

#juncao de PROC_REA - TB_SIGTAP.DBF
projeto["tb_sigtap_dbf"].create_index("IP_COD")
lookup={"$lookup": { "from": "tb_sigtap_dbf", "localField": "PROC_REA",
    "foreignField": "IP_COD", "as": "PROC_REA"}}
unwind1={ "$addFields": {"PROC_REA": { "$arrayElemAt": [ "$PROC_REA", 0
    ] }}}
pipeline.append(lookup)
project1={ "$project" : {"PROC_REA._id" : 0}}

#juncao de CNES - TCNESBR.DBF
projeto['tcnesbr_dbf'].create_index("CNES")
lookup={"$lookup": { "from": 'tcnesbr_dbf', "localField": "CNES", "
    foreignField": "CNES", "as": "CNES"}}
unwind2={ "$addFields": {"CNES": { "$arrayElemAt": [ "$CNES", 0 ] }}}
pipeline.append(lookup)
project2={ "$project" : {"CNES._id" : 0}}

#juncao de UF_ZI - br_municgestor.cnv
projeto['br_municgestor_cnv'].create_index("codigo")
lookup={"$lookup": { "from": 'br_municgestor_cnv', "localField": "UF_ZI"
    , "foreignField": "codigo", "as": "UF_ZI"}}
unwind3={ "$addFields": {"UF_ZI": { "$arrayElemAt": [ "$UF_ZI", 0 ] }}}
pipeline.append(lookup)
project3={ "$project" : {"UF_ZI._id" : 0}}


#juncao de ESPEC - LEITOS.CNV
```

```
projeto['leitos_cnv'].create_index("codigo")
lookup={"$lookup": { "from": 'leitos_cnv', "localField": "ESPEC", "
    foreignField": "codigo", "as": "ESPEC"}}
unwind4={ "$addFields": {"ESPEC": { "$arrayElemAt": [ "$ESPEC", 0 ] }}}
pipeline.append(lookup)
project4={ "$project" : {"ESPEC._id" : 0}}

#juncao de CGC_HOSP - TCHBR.DBF
projeto['tchbr_dbf'].create_index("CGC_HOSP")
lookup={"$lookup": { "from": 'tchbr_dbf', "localField": "CGC_HOSP", "
    foreignField": "CGC_HOSP", "as": "CGC_HOSP"}}
unwind5={ "$addFields": {"CGC_HOSP": { "$arrayElemAt": [ "$CGC_HOSP", 0
    ] }}}
pipeline.append(lookup)
project5={ "$project" : {"CGC_HOSP._id" : 0}}

#juncao de IDENT - IDENT.CNV
projeto['ident_cnv'].create_index("codigo")
lookup={"$lookup": { "from": 'ident_cnv', "localField": "IDENT", "
    foreignField": "codigo", "as": "IDENT"}}
unwind6={ "$addFields": {"IDENT": { "$arrayElemAt": [ "$IDENT", 0 ] }}}
pipeline.append(lookup)
project6={ "$project" : {"IDENT._id" : 0}}

#juncao de MUNIC_RES - br_municip.cnv
projeto['br_municip_cnv'].create_index("codigo")
lookup={"$lookup": { "from": 'br_municip_cnv', "localField": "MUNIC_RES"
    , "foreignField": "codigo", "as": "MUNIC_RES"}}
unwind7={ "$addFields": {"MUNIC_RES": { "$arrayElemAt": [ "$MUNIC_RES",
    0 ] }}}
pipeline.append(lookup)
project7={ "$project" : {"MUNIC_RES._id" : 0}}

#juncao de UTI_MES_TO - DIARIASUTI.CNV
projeto['diariasuti_cnv'].create_index("codigo")
lookup={"$lookup": { "from": 'diariasuti_cnv', "localField": "UTI_MES_TO
    ", "foreignField": "codigo", "as": "UTI_MES_TO"}}
unwind8={ "$addFields": {"UTI_MES_TO": { "$arrayElemAt": [ "$UTI_MES_TO"
    , 0 ] }}}
pipeline.append(lookup)
project8={ "$project" : {"UTI_MES_TO._id" : 0}}

#juncao de MARCA_UTI - MARCAUTI.CNV
projeto['marcauti_cnv'].create_index("codigo")
lookup={"$lookup": { "from": 'marcauti_cnv', "localField": "MARCA_UTI",
    "foreignField": "codigo", "as": "MARCA_UTI"}}
unwind9={ "$addFields": {"MARCA_UTI": { "$arrayElemAt": [ "$MARCA_UTI",
```

```
    0 ] }}}
pipeline.append(lookup)
project9={ "$project" : {"MARCA_UTI._id" : 0}}

#juncao de SEXO - SEXO.CNV
projeto['sexo_cnv'].create_index("codigo")
lookup={"$lookup": { "from": 'sexo_cnv', "localField": "SEXO", "
    foreignField": "codigo", "as": "SEXO"}}
unwind10={ "$addFields": {"SEXO": { "$arrayElemAt": [ "$SEXO", 0 ] }}}
pipeline.append(lookup)
project10={ "$project" : {"SEXO._id" : 0}}


#juncao de DIAG_SECUN - CID10.CNV
projeto['cid10_cnv'].create_index("codigo")
lookup={"$lookup": { "from": 'cid10_cnv', "localField": "DIAG_SECUN", "
    foreignField": "codigo", "as": "DIAG_SECUN"}}
unwind12={ "$addFields": {"DIAG_SECUN": { "$arrayElemAt": [ "$DIAG_SECUN
    ", 0 ] }}}
pipeline.append(lookup)
project12={ "$project" : {"DIAG_SECUN._id" : 0}}

#juncao de COBRANCA - SAIDAPERM.CNV
projeto['saidaperm_cnv'].create_index("codigo")
lookup={"$lookup": { "from": 'saidaperm_cnv', "localField": "COBRANCA",
    "foreignField": "codigo", "as": "COBRANCA"}}
unwind13={ "$addFields": {"COBRANCA": { "$arrayElemAt": [ "$COBRANCA", 0
     ] }}}
pipeline.append(lookup)
project13={ "$project" : {"COBRANCA._id" : 0}}

#juncao de NATUREZA - NATUREZA.CNV
projeto['natureza_cnv'].create_index("codigo")
lookup={"$lookup": { "from": 'natureza_cnv', "localField": "NATUREZA", "
    foreignField": "codigo", "as": "NATUREZA"}}
unwind14={ "$addFields": {"NATUREZA": { "$arrayElemAt": [ "$NATUREZA", 0
     ] }}}
pipeline.append(lookup)
project14={ "$project" : {"NATUREZA._id" : 0}}

#juncao de NAT_JUR - NATJUR.CNV
projeto['natjur_cnv'].create_index("codigo")
lookup={"$lookup": { "from": 'natjur_cnv', "localField": "NAT_JUR", "
    foreignField": "codigo", "as": "NAT_JUR"}}
unwind15={ "$addFields": {"NAT_JUR": { "$arrayElemAt": [ "$NAT_JUR", 0 ]
     }}}
pipeline.append(lookup)
```

```
project15={ "$project" : {"NAT_JUR._id" : 0}}

#juncao de GESTAO - GESTAO.CNV
projeto['gestao_cnv'].create_index("codigo")
lookup={"$lookup": { "from": 'gestao_cnv', "localField": "GESTAO", "
    foreignField": "codigo", "as": "GESTAO"}}
unwind16={ "$addFields": {"GESTAO": { "$arrayElemAt": [ "$GESTAO", 0 ]
    }}}
pipeline.append(lookup)
project16={ "$project" : {"GESTAO._id" : 0}}

#juncao de IND_VDRL - SIMNAO.CNV
projeto['simnao_cnv'].create_index("codigo")
lookup={"$lookup": { "from": 'simnao_cnv', "localField": "IND_VDRL", "
    foreignField": "codigo", "as": "IND_VDRL"}}
unwind17={ "$addFields": {"IND_VDRL": { "$arrayElemAt": [ "$IND_VDRL", 0
     ] }}}
pipeline.append(lookup)
project17={ "$project" : {"IND_VDRL._id" : 0}}

#juncao de MUNIC_MOV - br_municip.cnv
projeto['br_municip_cnv'].create_index("codigo")
lookup={"$lookup": { "from": 'br_municip_cnv', "localField": "MUNIC_MOV"
    , "foreignField": "codigo", "as": "MUNIC_MOV"}}
unwind18={ "$addFields": {"MUNIC_MOV": { "$arrayElemAt": [ "$MUNIC_MOV",
     0 ] }}}
pipeline.append(lookup)
project18={ "$project" : {"MUNIC_MOV._id" : 0}}

#juncao de COD_IDADE - IDADEBAS.CNV
projeto['idadebas_cnv'].create_index("codigo")
lookup={"$lookup": { "from": 'idadebas_cnv', "localField": "COD_IDADE",
    "foreignField": "codigo", "as": "COD_IDADE"}}
unwind19={ "$addFields": {"COD_IDADE": { "$arrayElemAt": [ "$COD_IDADE",
     0 ] }}}
pipeline.append(lookup)
project19={ "$project" : {"COD_IDADE._id" : 0}}

#juncao de DIAS_PERM - PERM.CNV
projeto['perm_cnv'].create_index("codigo")
lookup={"$lookup": { "from": 'perm_cnv', "localField": "DIAS_PERM", "
    foreignField": "codigo", "as": "DIAS_PERM"}}
unwind20={ "$addFields": {"DIAS_PERM": { "$arrayElemAt": [ "$DIAS_PERM",
     0 ] }}}
pipeline.append(lookup)
project20={ "$project" : {"DIAS_PERM._id" : 0}}
```

```
#juncao de MORTE - MORTES.CNV
projeto['mortes_cnv'].create_index("codigo")
lookup={"$lookup": { "from": 'mortes_cnv', "localField": "MORTE", "
    foreignField": "codigo", "as": "MORTE"}}
unwind21={ "$addFields": {"MORTE": { "$arrayElemAt": [ "$MORTE", 0 ] }}}
pipeline.append(lookup)
project21={ "$project" : {"MORTE._id" : 0}}

#juncao de NACIONAL - NACION3D.CNV
projeto['nacion3d_cnv'].create_index("codigo")
lookup={"$lookup": { "from": 'nacion3d_cnv', "localField": "NACIONAL", "
    foreignField": "codigo", "as": "NACIONAL"}}
unwind22={ "$addFields": {"NACIONAL": { "$arrayElemAt": [ "$NACIONAL", 0
    ] }}}
pipeline.append(lookup)
project22={ "$project" : {"NACIONAL._id" : 0}}

#juncao de CAR_INT - CARATEND.CNV
projeto['caratend_cnv'].create_index("codigo")
lookup={"$lookup": { "from": 'caratend_cnv', "localField": "CAR_INT", "
    foreignField": "codigo", "as": "CAR_INT"}}
unwind23={ "$addFields": {"CAR_INT": { "$arrayElemAt": [ "$CAR_INT", 0 ]
    }}}
pipeline.append(lookup)
project23={ "$project" : {"CAR_INT._id" : 0}}

#juncao de NUM_FILHOS - NUMFILH.CNV
projeto['numfilh_cnv'].create_index("codigo")
lookup={"$lookup": { "from": 'numfilh_cnv', "localField": "NUM_FILHOS",
    "foreignField": "codigo", "as": "NUM_FILHOS"}}
unwind24={ "$addFields": {"NUM_FILHOS": { "$arrayElemAt": [ "$NUM_FILHOS
    ", 0 ] }}}
pipeline.append(lookup)
project24={ "$project" : {"NUM_FILHOS._id" : 0}}

#juncao de INSTRU - INSTRU.CNV
projeto['instru_cnv'].create_index("codigo")
lookup={"$lookup": { "from": 'instru_cnv', "localField": "INSTRU", "
    foreignField": "codigo", "as": "INSTRU"}}
unwind25={ "$addFields": {"INSTRU": { "$arrayElemAt": [ "$INSTRU", 0 ]
    }}}
pipeline.append(lookup)
project25={ "$project" : {"INSTRU._id" : 0}}

#juncao de CID_NOTIF - CID10.CNV
projeto['cid10_cnv'].create_index("codigo")
lookup={"$lookup": { "from": 'cid10_cnv', "localField": "CID_NOTIF", "
```

```
    foreignField": "codigo", "as": "CID_NOTIF"}}
unwind26={ "$addFields": {"CID_NOTIF": { "$arrayElemAt": [ "$CID_NOTIF",
     0 ] }}}
pipeline.append(lookup)
project26={ "$project" : {"CID_NOTIF._id" : 0}}

#juncao de CONTRACEP1 - CONTRAC.CNV
projeto['contrac_cnv'].create_index("codigo")
lookup={"$lookup": { "from": 'contrac_cnv', "localField": "CONTRACEP1",
    "foreignField": "codigo", "as": "CONTRACEP1"}}
unwind27={ "$addFields": {"CONTRACEP1": { "$arrayElemAt": [ "$CONTRACEP1
    ", 0 ] }}}
pipeline.append(lookup)
project27={ "$project" : {"CONTRACEP1._id" : 0}}

#juncao de CONTRACEP2 - CONTRAC.CNV
projeto['contrac_cnv'].create_index("codigo")
lookup={"$lookup": { "from": 'contrac_cnv', "localField": "CONTRACEP2",
    "foreignField": "codigo", "as": "CONTRACEP2"}}
unwind28={ "$addFields": {"CONTRACEP2": { "$arrayElemAt": [ "$CONTRACEP2
    ", 0 ] }}}
pipeline.append(lookup)
project28={ "$project" : {"CONTRACEP2._id" : 0}}

#juncao de GESTRISCO - SIMNAO.CNV
projeto['simnao_cnv'].create_index("codigo")
lookup={"$lookup": { "from": 'simnao_cnv', "localField": "GESTRISCO", "
    foreignField": "codigo", "as": "GESTRISCO"}}
unwind29={ "$addFields": {"GESTRISCO": { "$arrayElemAt": [ "$GESTRISCO",
     0 ] }}}
pipeline.append(lookup)
project29={ "$project" : {"GESTRISCO._id" : 0}}

#juncao de SEQ_AIH5 - SEQAIH.CNV
projeto['seqaih_cnv'].create_index("codigo")
lookup={"$lookup": { "from": 'seqaih_cnv', "localField": "SEQ_AIH5", "
    foreignField": "codigo", "as": "SEQ_AIH5"}}
unwind30={ "$addFields": {"SEQ_AIH5": { "$arrayElemAt": [ "$SEQ_AIH5", 0
     ] }}}
pipeline.append(lookup)
project30={ "$project" : {"SEQ_AIH5._id" : 0}}

#juncao de CBOR - CBO2002.CNV
projeto['cbo2002_cnv'].create_index("codigo")
lookup={"$lookup": { "from": 'cbo2002_cnv', "localField": "CBOR", "
    foreignField": "codigo", "as": "CBOR"}}
unwind31={ "$addFields": {"CBOR": { "$arrayElemAt": [ "$CBOR", 0 ] }}}
```

```
pipeline.append(lookup)
project31={ "$project" : {"CBOR._id" : 0}}

#juncao de CNAER - CNAE.CNV
projeto['cnaeg_cnv'].create_index("codigo")
lookup={"$lookup": { "from": 'cnaeg_cnv', "localField": "CNAER", "
    foreignField": "codigo", "as": "CNAER"}}
unwind32={ "$addFields": {"CNAER": { "$arrayElemAt": [ "$CNAER", 0 ] }}}
pipeline.append(lookup)
project32={ "$project" : {"CNAER._id" : 0}}

#juncao de VINCPREV - VINCPREV.CNV
projeto['vinvprev_cnv'].create_index("codigo")
lookup={"$lookup": { "from": 'vincprev_cnv', "localField": "VINCPREV", "
    foreignField": "codigo", "as": "VINCPREV"}}
unwind33={ "$addFields": {"VINCPREV": { "$arrayElemAt": [ "$VINCPREV", 0
    ] }}}
pipeline.append(lookup)
project33={ "$project" : {"VINCPREV._id" : 0}}

#juncao de COMPLEX - COMPLEX2.CNV
projeto['complex2_cnv'].create_index("codigo")
lookup={"$lookup": { "from": 'complex2_cnv', "localField": "COMPLEX", "
    foreignField": "codigo", "as": "COMPLEX"}}
unwind34={ "$addFields": {"COMPLEX": { "$arrayElemAt": [ "$COMPLEX", 0 ]
    }}}
pipeline.append(lookup)
project34={ "$project" : {"COMPLEX._id" : 0}}

#juncao de RACA_COR - RACACOR.CNV
projeto['racacor_cnv'].create_index("codigo")
lookup={"$lookup": { "from": 'racacor_cnv', "localField": "RACA_COR", "
    foreignField": "codigo", "as": "RACA_COR"}}
unwind35={ "$addFields": {"RACA_COR": { "$arrayElemAt": [ "$RACA_COR", 0
    ] }}}
pipeline.append(lookup)
project35={ "$project" : {"RACA_COR._id" : 0}}

#juncao de ETNIA - ETNIA.CNV
projeto['etnia_cnv'].create_index("codigo")
lookup={"$lookup": { "from": 'etnia_cnv', "localField": "ETNIA", "
    foreignField": "codigo", "as": "ETNIA"}}
unwind36={ "$addFields": {"ETNIA": { "$arrayElemAt": [ "$ETNIA", 0 ] }}}
pipeline.append(lookup)
project36={ "$project" : {"ETNIA._id" : 0}}

#juncao de DIAGSEC1 - cid10.dbf
```

```
projeto['cid10_dbf'].create_index("CD_COD")
lookup={"$lookup": { "from": 'cid10_dbf', "localField": "DIAGSEC1", "
    foreignField": "CD_COD", "as": "DIAGSEC1"}}
unwind37={ "$addFields": {"DIAGSEC1": { "$arrayElemAt": [ "$DIAGSEC1", 0
    ] }}}
pipeline.append(lookup)
project37={ "$project" : {"DIAGSEC1._id" : 0}}


#juncao de DIAGSEC2 - cid10.dbf
projeto['cid10_dbf'].create_index("CD_COD")
lookup={"$lookup": { "from": 'cid10_dbf', "localField": "DIAGSEC2", "
    foreignField": "CD_COD", "as": "DIAGSEC2"}}
unwind38={ "$addFields": {"DIAGSEC2": { "$arrayElemAt": [ "$DIAGSEC2", 0
    ] }}}
pipeline.append(lookup)
project38={ "$project" : {"DIAGSEC2._id" : 0}}

#juncao de DIAGSEC3 - cid10.dbf
projeto['cid10_dbf'].create_index("CD_COD")
lookup={"$lookup": { "from": 'cid10_dbf', "localField": "DIAGSEC3", "
    foreignField": "CD_COD", "as": "DIAGSEC3"}}
unwind39={ "$addFields": {"DIAGSEC3": { "$arrayElemAt": [ "$DIAGSEC3", 0
    ] }}}
pipeline.append(lookup)
project39={ "$project" : {"DIAGSEC3._id" : 0}}

#juncao de DIAGSEC4 - cid10.dbf
projeto['cid10_dbf'].create_index("CD_COD")
lookup={"$lookup": { "from": 'cid10_dbf', "localField": "DIAGSEC4", "
    foreignField": "CD_COD", "as": "DIAGSEC4"}}
unwind40={ "$addFields": {"DIAGSEC4": { "$arrayElemAt": [ "$DIAGSEC4", 0
    ] }}}
pipeline.append(lookup)
project40={ "$project" : {"DIAGSEC4._id" : 0}}

#juncao de DIAGSEC5 - cid10.dbf
projeto['cid10_dbf'].create_index("CD_COD")
lookup={"$lookup": { "from": 'cid10_dbf', "localField": "DIAGSEC5", "
    foreignField": "CD_COD", "as": "DIAGSEC5"}}
unwind41={ "$addFields": {"DIAGSEC5": { "$arrayElemAt": [ "$DIAGSEC5", 0
    ] }}}
pipeline.append(lookup)
project41={ "$project" : {"DIAGSEC5._id" : 0}}

#juncao de DIAGSEC6 - cid10.dbf
projeto['cid10_dbf'].create_index("CD_COD")
```

```
lookup={"$lookup": { "from": 'cid10_dbf', "localField": "DIAGSEC6", "
    foreignField": "CD_COD", "as": "DIAGSEC6"}}
unwind42={ "$addFields": {"DIAGSEC6": { "$arrayElemAt": [ "$DIAGSEC6", 0
    ] }}}
pipeline.append(lookup)
project42={ "$project" : {"DIAGSEC6._id" : 0}}

#juncao de DIAGSEC7 - cid10.dbf
projeto['cid10_dbf'].create_index("CD_COD")
lookup={"$lookup": { "from": 'cid10_dbf', "localField": "DIAGSEC7", "
    foreignField": "CD_COD", "as": "DIAGSEC7"}}
unwind43={ "$addFields": {"DIAGSEC7": { "$arrayElemAt": [ "$DIAGSEC7", 0
    ] }}}
pipeline.append(lookup)
project43={ "$project" : {"DIAGSEC7._id" : 0}}

#juncao de DIAGSEC8 - cid10.dbf
projeto['cid10_dbf'].create_index("CD_COD")
lookup={"$lookup": { "from": 'cid10_dbf', "localField": "DIAGSEC8", "
    foreignField": "CD_COD", "as": "DIAGSEC8"}}
unwind44={ "$addFields": {"DIAGSEC8": { "$arrayElemAt": [ "$DIAGSEC8", 0
    ] }}}
pipeline.append(lookup)
project44={ "$project" : {"DIAGSEC8._id" : 0}}

#juncao de DIAGSEC9 - cid10.dbf
projeto['cid10_dbf'].create_index("CD_COD")
lookup={"$lookup": { "from": 'cid10_dbf', "localField": "DIAGSEC9", "
    foreignField": "CD_COD", "as": "DIAGSEC9"}}
unwind45={ "$addFields": {"DIAGSEC9": { "$arrayElemAt": [ "$DIAGSEC9", 0
    ] }}}
pipeline.append(lookup)
project45={ "$project" : {"DIAGSEC9._id" : 0}}

#juncao de TPDISEC1 - TP_DIAGSEC.CNV
projeto['tp_diagsec_cnv'].create_index("codigo")
lookup={"$lookup": { "from": 'tp_diagsec_cnv', "localField": "TPDISEC1",
    "foreignField": "codigo", "as": "TPDISEC1"}}
unwind46={ "$addFields": {"TPDISEC1": { "$arrayElemAt": [ "$TPDISEC1", 0
    ] }}}
pipeline.append(lookup)
project46={ "$project" : {"TPDISEC1._id" : 0}}

#juncao de TPDISEC2 - TP_DIAGSEC.CNV
projeto['tp_diagsec_cnv'].create_index("codigo")
lookup={"$lookup": { "from": 'tp_diagsec_cnv', "localField": "TPDISEC2",
    "foreignField": "codigo", "as": "TPDISEC2"}}
```

```
unwind47={ "$addFields": {"TPDISEC2": { "$arrayElemAt": [ "$TPDISEC2", 0
    ] }}}
pipeline.append(lookup)
project47={ "$project" : {"TPDISEC2._id" : 0}}

#juncao de TPDISEC3 - TP_DIAGSEC.CNV
projeto['tp_diagsec_cnv'].create_index("codigo")
lookup={"$lookup": { "from": 'tp_diagsec_cnv', "localField": "TPDISEC3",
    "foreignField": "codigo", "as": "TPDISEC3"}}
unwind48={ "$addFields": {"TPDISEC3": { "$arrayElemAt": [ "$TPDISEC3", 0
    ] }}}
pipeline.append(lookup)
project48={ "$project" : {"TPDISEC3._id" : 0}}

#juncao de TPDISEC4 - TP_DIAGSEC.CNV
projeto['tp_diagsec_cnv'].create_index("codigo")
lookup={"$lookup": { "from": 'tp_diagsec_cnv', "localField": "TPDISEC4",
    "foreignField": "codigo", "as": "TPDISEC4"}}
unwind49={ "$addFields": {"TPDISEC4": { "$arrayElemAt": [ "$TPDISEC4", 0
    ] }}}
pipeline.append(lookup)
project49={ "$project" : {"TPDISEC4._id" : 0}}

#juncao de TPDISEC5 - TP_DIAGSEC.CNV
projeto['tp_diagsec_cnv'].create_index("codigo")
lookup={"$lookup": { "from": 'tp_diagsec_cnv', "localField": "TPDISEC5",
    "foreignField": "codigo", "as": "TPDISEC5"}}
unwind50={ "$addFields": {"TPDISEC5": { "$arrayElemAt": [ "$TPDISEC5", 0
    ] }}}
pipeline.append(lookup)
project50={ "$project" : {"TPDISEC5._id" : 0}}

#juncao de TPDISEC6 - TP_DIAGSEC.CNV
projeto['tp_diagsec_cnv'].create_index("codigo")
lookup={"$lookup": { "from": 'tp_diagsec_cnv', "localField": "TPDISEC6",
    "foreignField": "codigo", "as": "TPDISEC6"}}
unwind51={ "$addFields": {"TPDISEC6": { "$arrayElemAt": [ "$TPDISEC6", 0
    ] }}}
pipeline.append(lookup)
project51={ "$project" : {"TPDISEC6._id" : 0}}

#juncao de TPDISEC7 - TP_DIAGSEC.CNV
projeto['tp_diagsec_cnv'].create_index("codigo")
lookup={"$lookup": { "from": 'tp_diagsec_cnv', "localField": "TPDISEC7",
    "foreignField": "codigo", "as": "TPDISEC7"}}
unwind52={ "$addFields": {"TPDISEC7": { "$arrayElemAt": [ "$TPDISEC7", 0
    ] }}}
```

```
pipeline.append(lookup)
project52={ "$project" : {"TPDISEC7._id" : 0}}

#juncao de TPDISEC8 - TP_DIAGSEC.CNV
projeto['tp_diagsec_cnv'].create_index("codigo")
lookup={"$lookup": { "from": 'tp_diagsec_cnv', "localField": "TPDISEC8",
    "foreignField": "codigo", "as": "TPDISEC8"}}
unwind53={ "$addFields": {"TPDISEC8": { "$arrayElemAt": [ "$TPDISEC8", 0
    ] }}}
pipeline.append(lookup)
project53={ "$project" : {"TPDISEC8._id" : 0}}

#juncao de TPDISEC9 - TP_DIAGSEC.CNV
projeto['tp_diagsec_cnv'].create_index("codigo")
lookup={"$lookup": { "from": 'tp_diagsec_cnv', "localField": "TPDISEC9",
    "foreignField": "codigo", "as": "TPDISEC9"}}
unwind54={ "$addFields": {"TPDISEC9": { "$arrayElemAt": [ "$TPDISEC9", 0
    ] }}}
pipeline.append(lookup)
project54={ "$project" : {"TPDISEC9._id" : 0}}


#executa os unwinds para retirar os dados dos arrays
pipeline.append(unwind1)
pipeline.append(unwind2)
pipeline.append(unwind3)
pipeline.append(unwind4)
pipeline.append(unwind5)
pipeline.append(unwind6)
pipeline.append(unwind7)
pipeline.append(unwind8)
pipeline.append(unwind9)
pipeline.append(unwind10)
pipeline.append(unwind11)
pipeline.append(unwind12)
pipeline.append(unwind13)
pipeline.append(unwind14)
pipeline.append(unwind15)
pipeline.append(unwind16)
pipeline.append(unwind17)
pipeline.append(unwind18)
pipeline.append(unwind19)
pipeline.append(unwind20)
pipeline.append(unwind21)
pipeline.append(unwind22)
pipeline.append(unwind23)
pipeline.append(unwind24)
```

```
pipeline.append(unwind25)
pipeline.append(unwind26)
pipeline.append(unwind27)
pipeline.append(unwind28)
pipeline.append(unwind29)
pipeline.append(unwind30)
pipeline.append(unwind31)
pipeline.append(unwind32)
pipeline.append(unwind33)
pipeline.append(unwind34)
pipeline.append(unwind35)
pipeline.append(unwind36)
pipeline.append(unwind37)
pipeline.append(unwind38)
pipeline.append(unwind39)
pipeline.append(unwind40)
pipeline.append(unwind41)
pipeline.append(unwind42)
pipeline.append(unwind43)
pipeline.append(unwind44)
pipeline.append(unwind45)
pipeline.append(unwind46)
pipeline.append(unwind47)
pipeline.append(unwind48)
pipeline.append(unwind49)
pipeline.append(unwind50)
pipeline.append(unwind51)
pipeline.append(unwind52)
pipeline.append(unwind53)
pipeline.append(unwind54)



pipeline.append(project1)
pipeline.append(project2)
pipeline.append(project3)
pipeline.append(project4)
pipeline.append(project5)
pipeline.append(project6)
pipeline.append(project7)
pipeline.append(project8)
pipeline.append(project9)
pipeline.append(project10)
pipeline.append(project11)
pipeline.append(project12)
pipeline.append(project13)
pipeline.append(project14)
```

```
pipeline.append(project15)
pipeline.append(project16)
pipeline.append(project17)
pipeline.append(project18)
pipeline.append(project19)
pipeline.append(project20)
pipeline.append(project21)
pipeline.append(project22)
pipeline.append(project23)
pipeline.append(project24)
pipeline.append(project25)
pipeline.append(project26)
pipeline.append(project27)
pipeline.append(project28)
pipeline.append(project29)
pipeline.append(project30)
pipeline.append(project31)
pipeline.append(project32)
pipeline.append(project33)
pipeline.append(project34)
pipeline.append(project35)
pipeline.append(project36)
pipeline.append(project37)
pipeline.append(project38)
pipeline.append(project39)
pipeline.append(project40)
pipeline.append(project41)
pipeline.append(project42)
pipeline.append(project43)
pipeline.append(project44)
pipeline.append(project45)
pipeline.append(project46)
pipeline.append(project47)
pipeline.append(project48)
pipeline.append(project49)
pipeline.append(project50)
pipeline.append(project51)
pipeline.append(project52)
pipeline.append(project53)
pipeline.append(project54)


out={ "$out" : "sih15" }
pipeline.append(out)

#agregacao onde ocorrem as juncoes
projeto.sih15_pura.aggregate(pipeline)
```

```
cursor=projeto.sih15.find()
if __name__ == '__main__':
        i=0
        saida=open("juncoes.out","w")
        for ele in cursor:
                #print(ele)
                if(i==60):break
                saida.write(str(ele)+"\n")
                i=i+1
        saida.close()
```