



**PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS  
NÚCLEO DE EDUCAÇÃO À DISTÂNCIA**

*Analytics & Business Intelligence - Pós-graduação Lato Sensu*

**RELATÓRIO TÉCNICO**  
**ANÁLISE DE ACIDENTES EM RODOVIAS FEDERAIS BRASILEIRAS**

Kaio Oliveira Peixoto

Belo Horizonte  
2022

# Sumário

<b>1. Introdução</b>	
1.1 Contexto	3
1.2 Objetivos	3
1.3 Público Alvo	3
<b>2. Modelos de Dados</b>	
2.1 Modelo Dimensional	4
<b>3. Integração, Tratamento e Carga</b>	
3.1 Fonte de Dados	5
3.2 Processo de ETL	5
<b>3. Integração, Tratamento e Carga</b>	
3.1 Fonte de Dados	5
3.2 Processo de ETL	5
3.2 Extração	6
3.2 Transformação	6
3.2 Carga	8
<b>4. Camada de Apresentação</b>	
4.1 Dashboard	9
4.2 Análises Avançadas	9
<b>5. Registros de Homologação</b>	9
<b>6. Conclusão</b>	9
<b>7. Links</b>	10
<b>Referências</b>	11

# **1. Introdução**

## **1.1. Contexto**

A malha rodoviária brasileira é uma das maiores do mundo. Isso não é novidade quando trata-se de interligar um país de dimensões continentais, o quinto maior do planeta. No entanto, diferentemente de outros países com proporções parecidas como Estados Unidos, Canadá, Rússia e China, o Brasil depende quase que exclusivamente de sua malha rodoviária para transportar pessoas, bens e serviços.

Sendo assim, o fluxo diário de milhões de pessoas e enorme pressão num único modal, aumenta a preocupação da sociedade sobre a ocorrência de acidentes e a segurança como um todo. Neste cenário, surgem questões como quais fatores contribuem para mais acidentes? Quais rodovias e quais trechos são os mais perigosos? Quais cidades e estados tem os maiores números de vítimas fatais?

Para responder a estas e outras perguntas, bem como fazer um diagnóstico da segurança do transporte na malha rodoviária nacional, o presente trabalho analisou a base de dados de acidentes ocorridos nas estradas brasileiras entre os anos de 2007 a 2020, disponibilizada pela Polícia Rodoviária Federal.

## **1.2. Objetivos**

O objetivo principal da análise é trazer compreensão para que medidas eficientes de prevenção possam ser tomadas pelos agentes públicos. No setor logístico, é de grande valia para o planejamento de rotas e elucidação para tomada de decisões.

## **1.3. Público alvo**

Agentes públicos, estudantes e profissionais da área de dados do setor logístico. É também de interesse de qualquer cidadão brasileiro como principal usuário direta ou indiretamente da malha rodoviária federal nacional.

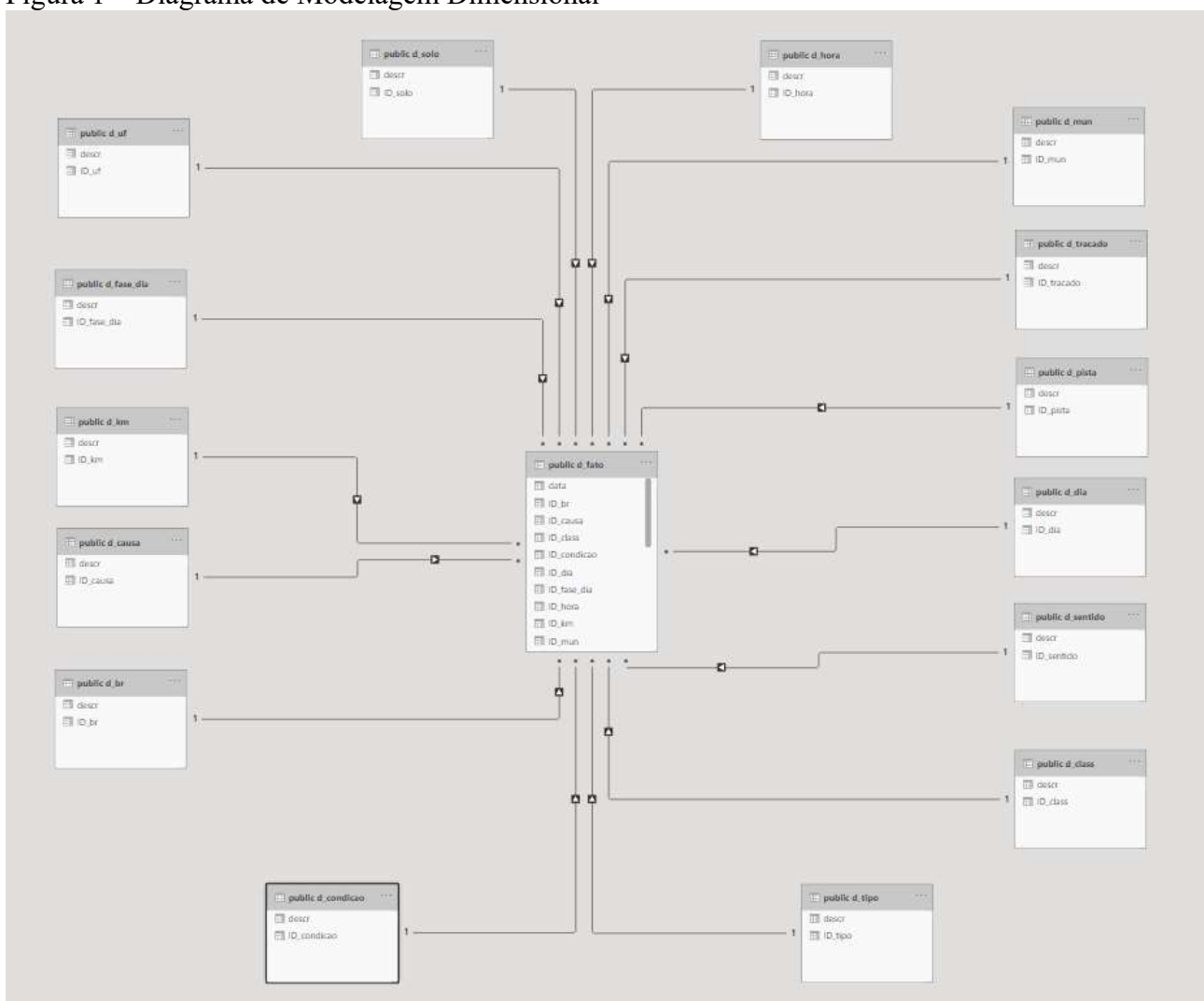
## 2. Modelo de Dados

### 2.1. Modelo Dimensional

Para relacionarmos as informações de maneira que seja combinada como um cubo a modelagem desse projeto foi feita com base no modelo *Star Schema* e nas quatro etapas de desenvolvimento: “Selecione o processo de negócio, Declare a granularidade, Identifique as dimensões e Identifique os fatos.” (KIMBALL, 2021).

A análise do arquivo resultou em uma tabela fato e 15 tabelas dimensão. Algumas colunas com ocorrência de muitos valores nulo ou que não eram importantes para a análise foram excluídas do modelo. Veja na figura abaixo como ficou o modelo final em star schema.

Figura 1 – Diagrama de Modelagem Dimensional



### 3. Integração, Tratamento e Carga de Dados

#### 3.1. Fonte de Dados


A fonte de dados deste estudo é um arquivo csv onde está unificada toda a base coletada pela Polícia Rodoviária Federal entre os anos de 2007 e 2020. O arquivo está no repositório deste projeto no GitHub e o link para pode ser encontrado na seção 6 deste documento. Para facilitar a manipulação e distribuição do arquivo, o mesmo foi reduzido para o formato tar.xz.

#### 3.2. Processo de ETL

O banco de dados escolhido foi o PostgreSQL pelas suas conhecidas vantagens, como economia e alto desempenho. Este SGBD suporta um intenso fluxo de dados com garantia de estabilidade e segurança.

Para o fluxo de ETL foi utilizada a linguagem de programação Python além das bibliotecas Pandas, Numpy e SQLAlchemy para manipulação e análise de dados. O SQLAlchemy permite transformar o banco de dados num objeto manipulável em Python, facilitando assim nossa integração e trabalho de ETL. O repositório encontra-se público, disponível no Link 2. Neste repositório do GitHub encontra-se o notebook onde consta o código Python criado para esta etapa do projeto.

Figura 2 – Jupyter Notebook



```
Importando bibliotecas a serem utilizadas

In [4]: import os
import pandas as pd
import numpy as np
from sqlalchemy import create_engine
import psycopg2
from time import sleep

Leitura do arquivo

In [6]: df = pd.read_csv('acidentes2007-2020.tar.xz', compression='xz', usecols=[
'id', 'id_unico', 'dia_semana', 'horario', 'uf', 'br', 'km', 'municipio',
'causa_acidente', 'tipo_acidente', 'classificacao_acidente', 'fase_dia',
'sentido_via', 'condicao_meteorologica', 'tipo_pista', 'tracado_via',
'uso_solo', 'pessoas', 'mortos', 'feridos_leves', 'feridos_graves', 'ilesos',
'ignorados', 'feridos', 'veiculos'], dtype={'br': 'str', 'km': 'str'})

In [7]: df.columns

Out[7]: Index(['id', 'id_unico', 'dia_semana', 'horario', 'uf', 'br', 'km',
'municipio', 'causa_acidente', 'tipo_acidente',
'classificacao_acidente', 'fase_dia', 'sentido_via',
'condicao_meteorologica', 'tipo_pista', 'tracado_via', 'uso_solo',
'pessoas', 'mortos', 'feridos_leves', 'feridos_graves', 'ilesos',
'ignorados', 'feridos', 'veiculos'],
dtype='object')

In [8]: df.head()

Out[8]:
```

id	id_unico	dia_semana	horario	uf	br	km	municipio	causa_acidente	tipo_acidente	...	tracado_via	uso_solo	pessoas	mortos	feri
----	----------	------------	---------	----	----	----	-----------	----------------	---------------	-----	-------------	----------	---------	--------	------

### 3.2.1. Extração

Através da biblioteca pandas descompactamos o arquivo e já fazemos a leitura do mesmo. Veja que também já é possível indicar os tipos de dados de algumas colunas, como 'br' e 'km' que, apesar de dados numéricos, trataremos como strings.

Figura 2 – Leitura do arquivo no Jupyter notebook

```
Leitura do arquivo

In [6]: df = pd.read_csv('acidentes2007-2020.tar.xz', compression='xz', usecols=[
    'id', 'id_unico', 'dia_semana', 'horario', 'uf', 'br', 'km', 'municipio',
    'causa_acidente', 'tipo_acidente', 'classificacao_acidente', 'fase_dia',
    'sentido_via', 'condicao_meteorologica', 'tipo_pista', 'tracado_via',
    'uso_solo', 'pessoas', 'mortos', 'feridos_leves', 'feridos_graves', 'ileso',
    'ignorado', 'feridos', 'veiculos'], dtype={'br': 'str', 'km': 'str'})

In [7]: df.columns

Out[7]: Index(['id', 'id_unico', 'dia_semana', 'horario', 'uf', 'br', 'km',
    'municipio', 'causa_acidente', 'tipo_acidente',
    'classificacao_acidente', 'fase_dia', 'sentido_via',
    'condicao_meteorologica', 'tipo_pista', 'tracado_via', 'uso_solo',
    'pessoas', 'mortos', 'feridos_leves', 'feridos_graves', 'ileso',
    'ignorado', 'feridos', 'veiculos'],
    dtype='object')
```

### 3.2.2. Transformação

Essa é a parte que ocupa a maior parte do tempo e de processamento. Veja abaixo como iniciamos esta fase conhecendo quais são os tipos de dados de cada coluna.

Figura 3 – Avaliando tipos de dados de cada coluna

```
Conhecendo tipos de dados de cada coluna da nossa base

In [10]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1851866 entries, 0 to 1851865
Data columns (total 25 columns):
#   Column              Dtype
---  ---
0   id                  float64
1   id_unico            object
2   dia_semana          object
3   horario             object
4   uf                  object
5   br                  object
6   km                  object
7   municipio           object
8   causa_acidente      object
9   tipo_acidente       object
10  classificacao_acidente object
11  fase_dia            object
12  sentido_via         object
13  condicao_meteorologica object
14  tipo_pista          object
15  tracado_via         object
16  uso_solo            object
17  pessoas             float64
18  mortos              float64
19  feridos_leves       float64
20  feridos_graves      float64
21  ileso               float64
22  ignorado            float64
23  feridos             float64
24  veiculos            float64
dtypes: float64(9), object(16)
memory usage: 353.2+ MB
```

Depois fazemos uma checagem da qualidade dos dados de todas as colunas. O método `isna()` de pandas nos permite visualizar quais colunas possuem valores nulos e qual a quantidade. Dessa forma podemos ver rapidamente quais colunas precisam de uma revisão.

Figura 4 – Checando qualidade das colunas

Vamos checar quantos e onde estão os valores nulos

```
In [11]: df.isna().sum()
Out[11]: id                1
id_unico                1
dia_semana              1
horario                1
uf                    1
br                   514
km                   514
municipio              1
causa_acidente         1
tipo_acidente          13
classificacao_acidente 25
fase_dia               2
sentido_via            1
condicao_metereologica  4
tipo_pista             11
tracado_via            1
uso_solo               1
pessoas               1
mortos                1
feridos_leves          1
feridos_graves         1
ilesos                1
ignorados              1
feridos                1
veiculos              1
dtype: int64
```

A partir daí iniciamos a limpeza propriamente dita. Abaixo tem um exemplo do primeiro passo dado, que foi excluir colunas com quantidades significativa de valores nulos.

Figura 5 – Iniciando o tratamento

No código abaixo vamos excluir as linhas de colunas que tiveram alguns valores nulos

```
In [12]: df.dropna(subset=['br', 'km', 'tipo_acidente', 'classificacao_acidente',
'fase_dia', 'condicao_metereologica', 'tipo_pista'], axis=0, inplace=True)
```

Como ficou nossa base

```
In [13]: df.head()
```

```
Out[13]:
```

	id	id_unico	dia_semana	horario	uf	br	km	municipio	causa_acidente	tipo_acidente	...	tracado_via	uso_solo	pessoas	mortos	feri
1	1032898.0	10328982007-08-13	segunda	14:25:00	MG	40	585.5	ITABIRITO	outras	Saída de Pista	...	Reta	Rural	3.0	0.0	
2	1051130.0	10511302007-02-12	segunda	02:10:00	MA	135	11.0	SAO LUIS	animais na pista	Atropelamento de Animal	...	Reta	Urbano	5.0	2.0	
3	1066824.0	10668242007-11-20	terça	05:30:00	CE	222	30.8	CAUCAIA	defeito mecânico no veículo	Capotamento	...	Reta	Rural	1.0	0.0	
4	1069918.0	10699182007-12-16	domingo	17:40:00	MA	230	14.0	BARAO DE GRAJAU	outras	Capotamento	...	Curva	Rural	1.0	0.0	
5	1070971.0	10709712007-03-05	segunda	08:10:00	PR	277	584.4	CASCADEL	outras	Colisão Lateral	...	Curva	Urbano	2.0	0.0	

5 rows × 17 columns

Para demais detalhes e informações desta etapa, o link 3 da seção 6 deste projeto oferece acesso ao notebook completo com todas as etapas e descrições, não só da fase de tratamento, mas de todo o processo de ETL.

### 3.2.2. Carga

Concomitante ao processo de transformação e limpeza foi construída a estrutura física de cada tabela dimensão que compõe este projeto. Abaixo, por exemplo, é mostrada a criação da tabela dimensão `d_solo`.

Logo após criamos uma função python que modela cada tabela dimensão e já salva na estrutura de tabela correspondente criada no banco de dados PostgreSQL. Daí basta colocar esta função num loop para percorrer por todas as tabelas dimensão da lista campos e concluir o processo de carga.

Figura 6 – Linguagem SQL para criação das tabelas no banco de dados



```
postgres/postgres@PostgreSQL 13 v
Query Editor  Query History
1 CREATE TABLE solo (
2   id_solo PRIMATY KEY,
3   descr VARCHAR(50)
4 );
```

Figura 7 – Código criado em Python para carga das tabelas direto para o banco de dados



```
Código para transformar colunas dimensão em tabelas e já salvar para o banco

In [48]: def criar_dim(coluna, salva=False):
          unicos = df_dim[coluna].unique().copy()
          tabela_dim = pd.DataFrame({'ID_{coluna[3:]}':range(1, len(unicos)+1), 'descr':unicos})
          if salva:
              tabela_dim.to_sql(f'd_{coluna[3:]}', con=con, index=False, if_exists='replace')
          return tabela_dim

In [49]: campos

Out[49]: ['ID_dia',
          'ID_hora',
          'ID_uf',
          'ID_br',
          'ID_km',
          'ID_mun',
          'ID_causa',
          'ID_tipo',
          'ID_class',
          'ID_fase_dia',
          'ID_sentido',
          'ID_condicao',
          'ID_pista',
          'ID_tracado',
          'ID_solo']

Aqui vamos criar um laço que vai percorrer por toda a lista acima e criar as tabelas no banco uma a uma

In [50]: for dimensao in campos:
          criar_dim(dimensao, True)
          sleep(5)
```



## **4. Camada de Apresentação**

### 4.1. Dashboard

*Módulo B*

### 4.2. Análises Avançadas

*Módulo C*

## **5. Registros de Homologação**

*Módulo B*

## **6. Conclusão**

*Módulo C*

## 7. Links

Link 1 [Fonte de Dados Arquivo CSV] Disponível em:

[https://github.com/KPxto/pucminas\\_tcc\\_mba/blob/master/data/acidentes2007-2020.tar.xz](https://github.com/KPxto/pucminas_tcc_mba/blob/master/data/acidentes2007-2020.tar.xz)

Link 2 [Repositório Github] Disponível em:

[https://github.com/KPxto/tcc\\_mba](https://github.com/KPxto/tcc_mba)

Link 3 [Jupyter Notebook] Disponível em:

[https://github.com/KPxto/pucminas\\_tcc\\_mba/blob/master/tratamento\\_tcc\\_completo.ipynb](https://github.com/KPxto/pucminas_tcc_mba/blob/master/tratamento_tcc_completo.ipynb)

## REFERÊNCIAS:

G1. Disponível em:

<https://g1.globo.com/economia/noticia/por-que-o-brasil-depender-tanto-do-transporte-rodoviario.ghtml>

PANDAS

<https://pandas.pydata.org/>