# Sistema de Eleitores 2018

Claudio Segala R. Silva Filho
15/0032552
claudiosegalafilho@gmail.com

Juliana Mayumi Hosoume
18/0048864
ju.hosoume@gmail.com

Khalil Carsten do Nascimento
15/0134495
khalilcarsten@gmail.com

Renato Avellar Nobre
15/0146698
rekanobre@gmail.com

Junho de 2018

**Abstract**

Projeto de Banco de Dados para um Sistema de Controle de Eleitores em uma Região

# 1 Introdução

Projetos de bancos de dados são essenciais para o desenvolvimento de uma aplicação que necessita consistência e organização de informações. Devido ao fato de estarmos em ano de eleições, é visível a necessidade de aplicações que estruturem dados e adquiriam informações sobre o cenário atual de eleitores. Este trabalho tem foco em desenvolver um sistema de registro de informações dos eleitores uma dada região (estado e município), além de treinar as habilidades adquiridas ao longo do curso e da matéria.

O foco principal do trabalho é estruturar um banco de dados do sistema eleitoral brasileiro de 2018. Para estruturar um banco de dados deste sistema realizou-se diversas etapas, descritas neste relatório. O primeiro passo foi realizar uma modelagem de Entidade-Relacionamento em formato de diagrama (DER). Com base no Diagrama Entidade-Relacionamento, foi construído o Modelo Relacional. Com o MR criado, realizou-se cinco consultas em álgebra relacional, cada consulta envolvendo pelo menos três tabelas. Com base também no modelo relacional verificou-se sua forma normal em cinco tabelas, escolhidas com base no foco principal do trabalho. Posteriormente o script de criação do SQL é descrito.

Após toda a estrutura ter sido criada passamos para a fase da implementação, na qual temos todo o banco de dados como fonte de leitura, mas apenas três entidades foram escolhidas para a realização do CRUD. apresentando como a interface gráfica do programa acessa a camada de persistência.

Para a implementação foi escolhido fazer um sistema web em Node, Express (back-end), Vue.js (front-end) e MySQL. A escolha dessas ferramentas se baseia na facilidade de criar uma interface gráfica como uma Aplicação de Página Única (SPA), onde a interação com o banco de dados seria intuitiva ao usuário.

# 2 Modelos

## 2.1 Diagrama de Entidade e Relacionamento

Como primeiro passo para o projeto do sistema de banco de dados, foram definidas as entidades que compõe os principais processos em um controle de eleitores por região(Figura 1). Nesse sentido, ainda que Estados não sejam comumente criados, escolheu-se por manter uma entidade Estado dada a importância e quantidade de consultas em que estará envolvido. Pela mesma razão, criou-se a tabela de Municípios. Como cada eleitor pode compartilhar endereços, além de informações como ruas e bairros serem relevantes para a definição do local e seção de votação, endereços foram mantidos como entidades. Ainda que os votos sejam secretos, para razões didáticas e de simulação, escolheu-se por guardar informações sobre quais foram os candidatos escolhidos por cada eleitor. Outras lógicas de negócio foram consideradas para a definição das tabelas e de seus relacionamentos, no entanto não são tão diretamente associadas ao controle de eleitores na região, estariam mais ligadas ao processo eleitoral como um todo.

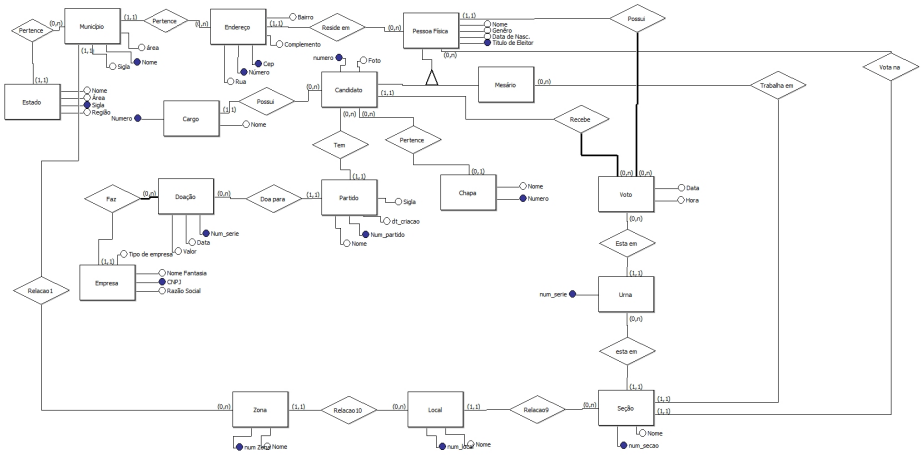


Figure 1: Diagrama Entidade Relacionamento (DER) para um sistema de controle de eleitores por região.

## 2.2 Modelo Relacional

Com o projeto das entidades e seus relacionamentos (Seção 2.1), foram então estabelecidas as tabelas para estruturação do banco de dados. Para cada tabela, foi determinado como seria a sua identificação. Outrossim, foram definidas as relações por chave estrangeira, não esquecendo de marcar os contraints relevantes como auto incrementos e forma de deleção.

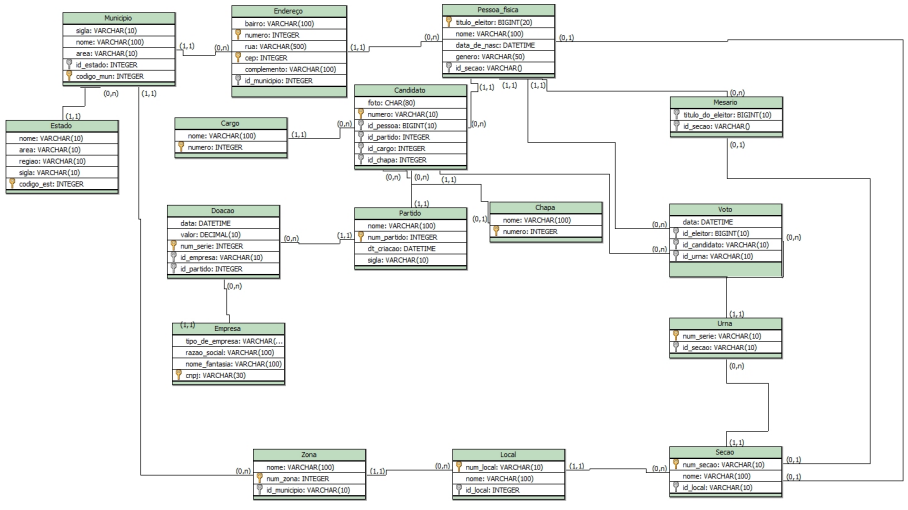


Figure 2: Modelo Relacional para um sistema de controle de eleitores por região.

# 3 Consultas em Álgebra Relacional

Uma das mais importantes utilidades de um banco de dados é a busca por informações registradas de maneira organizada. Para tanto, uma das ferramentas

de formalização de consultas é a álgebra relacional. Por conseguinte, aqui serão demonstradas quatro consultas relevantes para o sistema projetado.

## 3.1 Consulta 1

Para apresentar a média de idade dos eleitores de um candidato, precisa-se associar candidato com eleitores por meio do voto. Para isso, foi primeiramente testada a busca em SQL.

```
    mysql> SELECT candidato.id As candidato_id,
                AVG(TIMESTAMPDIFF(YEAR, data_de_nasc, CURDATE()))
                AS idade
           FROM eleitor, candidato, voto
           WHERE voto.id_candidato = candidato.id
                AND voto.id_eleitor = eleitor.id
           GROUP BY(candidato.id);
+--------------+----------+
| candidato_id | idade    |
+--------------+----------+
|            1 |  53.0000 |
|            2 |  58.0000 |
|            3 |  74.4286 |
|            4 |  79.6000 |
|            5 |  74.4286 |
|            6 |  40.0000 |
|            7 |  62.0000 |
|            8 |  45.5000 |
|            9 |  65.0000 |
|           10 | 113.0000 |
|           11 |  30.0000 |
|           12 |  51.3333 |
|           13 |  48.3333 |
|           14 |  71.7143 |
|           15 |  65.2500 |
|           16 |  58.4286 |
|           17 |  69.5000 |
|           18 |  80.2000 |
|           19 |  62.0000 |
|           20 |  44.0000 |
|           21 |  63.7143 |
|           22 |  87.3333 |
+--------------+----------+
22 rows in set (0.00 sec)
```

Nessa tabela, as médias de idade são grandes visto que as datas de nascimento são geradas randomicamente, sendo que possivelmente o mais novo possui 18 anos e o mais velho 120. Para a apresentação em álgebra relacional, considera-se que há um atributo virtual de idade para a tabela. Assim:
$eleitorVoto \leftarrow eleitor \bowtie_{(eleitor.id=voto.id\_eleitor)} voto$
$eleCandVot \leftarrow eleitorVoto \bowtie_{(voto.id\_candidato=candidato.id)} candidato$
$(candidato.id)\Im Media_{(eleitor.idade)}(eleCandVot)$

### 3.2 Consulta 2

De maneira semelhante a consulta da média de idade, pode-se querer verificar a idade da pessoa mais velha a votar por município. Portanto a álgebra relacional é:

$eleitorVoto \leftarrow eleitor \bowtie_{(eleitor.id=voto.id\_eleitor)} voto$
$enderecoMun \leftarrow endereco \bowtie_{(endereco.id\_municipio=municipio.id)} municipio$
$eleitoresMun \leftarrow eleitorVoto \bowtie_{(eleitor.id\_endereco=endereco.id)} enderecoMun$
$(municipio.nome)\Im MAX_{(eleitor.idade)}(eleitoresMun)$

### 3.3 Consulta 3

Para controle de eleitores por região, é necessário saber a quantidade de eleitores nos estados. A álgebra relacional para se obter a informação é:
$estadoMun \leftarrow estado \bowtie_{(estado.id=municipio.id\_estado)} municipio$
$enderecoEstado \leftarrow endereco \bowtie_{(endereco.id\_municipio=municipio.id)} estadoMun$
$eleitoresEstado \leftarrow eleitor \bowtie_{(eleitor.id\_endereco=endereco.id)} enderecoEstado$
$(estado.nome)\Im COUNT_{(eleitor.id)}(eleitoresEstado)$

### 3.4 Consulta 4

Por região também é importante saber quais os são mesários por Município.
$mesarioSecao \leftarrow mesario \bowtie_{(mesario.id\_secao=secao.id)} secao$
$mesarioLocal \leftarrow local \bowtie_{(secao.id\_local=local.id)} mesarioSecao$
$mesarioZona \leftarrow zona \bowtie_{local.id\_zona=zona.id)} mesarioLocal$
$mesarioMunicipio \leftarrow municipio \bowtie_{zona.id\_municipio=municipio.id)} mesarioZona$
$(municipio.id)\Im COUNT_{(mesario.id)}(mesarioMunicipio)$

### 3.5 Consulta 5

Para identificar se há problemas e algum eleitor menor de idade votou em determinado estado, pode-se buscar o minimo de um atributo virtual.
$eleitorVoto \leftarrow eleitor \bowtie_{(eleitor.id=voto.id\_eleitor)} voto$
$enderecoMun \leftarrow endereco \bowtie_{(endereco.id\_municipio=municipio.id)} municipio$
$eleitoresMun \leftarrow eleitorVoto \bowtie_{(eleitor.id\_endereco=endereco.id)} enderecoMun$
$eleitoresEstado \leftarrow eleitoresMun \bowtie_{(municipio.id\_estado=estado.id)} estado$
$(estado.nome)\Im MIN_{(eleitor.idade)}(eleitoresEstado)$

## 4 Script SQL

O SGBD escolhido foi o MySQL. Utilizando a linguagem SQL foram então construídas 16 tabelas distintas, de forma que tentou-se incluir pelo menos 5 atributos em cada uma das tabelas.

```
CREATE TABLE local (
```

```
    id INTEGER PRIMARY KEY AUTO_INCREMENT,
    nome VARCHAR(100),
    id_zona INTEGER
);

CREATE TABLE municipio (
    sigla VARCHAR(10),
    nome VARCHAR(100),
    area VARCHAR(10),
    id_estado INTEGER,
    id INTEGER PRIMARY KEY AUTO_INCREMENT
);

CREATE TABLE partido (
    nome VARCHAR(100),
    id INTEGER PRIMARY KEY,
    dt_criacao DATE,
    sigla VARCHAR(10)
);

CREATE TABLE chapa (
    nome VARCHAR(100),
    id INTEGER PRIMARY KEY AUTO_INCREMENT
);

CREATE TABLE candidato (
    foto BLOB,
    id INTEGER PRIMARY KEY AUTO_INCREMENT,
    id_pessoa INTEGER,
    id_partido INTEGER,
    id_cargo INTEGER,
    id_chapa INTEGER,
    UNIQUE(id_pessoa),
    FOREIGN KEY(id_partido) REFERENCES partido (id) ON DELETE SET NULL,
    FOREIGN KEY(id_chapa) REFERENCES chapa (id) ON DELETE SET NULL
);

CREATE TABLE voto (
    data DATETIME,
    id_eleitor INTEGER,
    id_candidato INTEGER,
    id_urna INTEGER,
    PRIMARY KEY(id_eleitor, id_candidato),
    FOREIGN KEY(id_candidato) REFERENCES candidato (id) ON DELETE CASCADE
);

CREATE TABLE urna (
    id INTEGER PRIMARY KEY AUTO_INCREMENT,
    id_secao INTEGER
);
```

```
CREATE TABLE estado (
    nome VARCHAR(100),
    area VARCHAR(20),
    regiao VARCHAR(100),
    sigla VARCHAR(10),
    id INTEGER PRIMARY KEY
);

CREATE TABLE empresa (
    id INTEGER PRIMARY KEY AUTO_INCREMENT,
    tipo_de_empresa VARCHAR(100),
    razao_social VARCHAR(100),
    nome_fantasia VARCHAR(100),
    cnpj VARCHAR(30)
);

CREATE TABLE endereco (
    bairro VARCHAR(100),
    id INTEGER,
    rua VARCHAR(500),
    cep INTEGER,
    complemento VARCHAR(100),
    id_municipio INTEGER,
    PRIMARY KEY(id,cep),
    FOREIGN KEY(id_municipio) REFERENCES municipio (id) ON DELETE CASCADE
);

CREATE TABLE zona (
    nome VARCHAR(100),
    id INTEGER PRIMARY KEY AUTO_INCREMENT,
    id_municipio INTEGER,
    FOREIGN KEY(id_municipio) REFERENCES municipio (id) ON DELETE CASCADE
);

CREATE TABLE cargo (
    nome VARCHAR(100),
    id INTEGER PRIMARY KEY AUTO_INCREMENT
);

CREATE TABLE secao (
    id INTEGER PRIMARY KEY AUTO_INCREMENT,
    nome VARCHAR(100),
    id_local INTEGER,
    FOREIGN KEY(id_local) REFERENCES local (id)
);

CREATE TABLE doacao (
    data DATETIME,
    valor DECIMAL(12,2),
```

```
    id INTEGER PRIMARY KEY,
    id_empresa INTEGER,
    id_partido INTEGER,
    FOREIGN KEY(id_empresa) REFERENCES empresa (id),
    FOREIGN KEY(id_partido) REFERENCES partido (id)
);

CREATE TABLE mesario (
    id INTEGER PRIMARY KEY AUTO_INCREMENT,
    id_eleitor INTEGER,
    id_secao INTEGER,
    FOREIGN KEY(id_secao) REFERENCES secao (id),
    UNIQUE(id_eleitor)
);

CREATE TABLE eleitor (
    id INTEGER PRIMARY KEY AUTO_INCREMENT,
    titulo_eleitor BIGINT(20),
    nome VARCHAR(100),
    data_de_nasc DATE,
    genero VARCHAR(50),
    id_secao INTEGER,
    cep_endereco INTEGER,
    id_endereco INTEGER,
    FOREIGN KEY(id_secao) REFERENCES secao (id),
    FOREIGN KEY(id_endereco, cep_endereco)
        REFERENCES endereco (id, cep)
);

ALTER TABLE local ADD FOREIGN KEY(id_zona)
    REFERENCES zona (id);
ALTER TABLE municipio ADD FOREIGN KEY(id_estado)
    REFERENCES estado (id);
ALTER TABLE candidato ADD FOREIGN KEY(id_pessoa)
    REFERENCES eleitor (id);
ALTER TABLE candidato ADD FOREIGN KEY(id_cargo)
    REFERENCES cargo (id);
ALTER TABLE voto ADD FOREIGN KEY(id_eleitor)
    REFERENCES eleitor (id);
ALTER TABLE voto ADD FOREIGN KEY(id_urna)
    REFERENCES urna (id);
ALTER TABLE urna ADD FOREIGN KEY(id_secao)
    REFERENCES secao (id);
ALTER TABLE mesario ADD FOREIGN KEY(id_eleitor)
    REFERENCES eleitor (id);
```

# 5 Normalização

## 5.1 Primeira Forma Normal

Uma relação está em 1FN se o valor de uma coluna de uma tabela é indivisível.

- **Município** - (idMunicipio, idEstado, area, nome, sigla)
- **Estado** - (idEstado, nome, area, regiao, sigla)
- **Endereço** - (numeroEnd, cepEnd, rua, complemento, idMunicipio, bairro)
- **Candidato** - (numero, idPessoa, idPartido, idCargo, idChapa, foto)
- **Pessoa Física** - (idPessoa, nome, dtNasc, genero, idSecao)

Note que todos o valores das colunas das nossas tabelas só podem ser atômicos, e portanto, já está em sua primeira forma normal.

## 5.2 Segunda Forma Normal

Uma relação está na 2FN se, e somente se, estiver na 1FN e cada atributo não-chave for dependente da chave primária inteira, isto é, cada atributo não-chave não poderá ser dependente de apenas parte da chave.

Para analisarmos se as tabelas estão na segunda forma normal precisamos analisar as dependências funcionais de cada tabela.

**Tabela de Município:**
{idMunicipio} → idEstado, area, nome, sigla

**Tabela de Estado:**
{idEstado} → nome, area, regiao, sigla

**Tabela de Cantidato:**
{numero} → idPessoa, idPartido, idCargo, idChapa, foto

**Tabela de Pessoa Física:**
{idPessoa} → nome, dtNasc, genero, idSecao

**Tabela de Endereço:**
{cepEnd} → rua, bairro, idMunicipio
{cepEnd, numeroEnd} → complemento

Nota-se que as relações mostradas na 1FN se mantém sem nenhuma alteração. E portanto, já está na segunda forma normal.

## 5.3 Terceira Forma Normal

Uma relação R está na 3FN se ela estiver na 2FN e cada atributo não-chave de R não possuir dependência transitiva, para cada chave candidata de R. Todos os atributos dessa tabela devem ser independentes uns dos outros, ao mesmo tempo que devem ser dependentes exclusivamente da chave primária da tabela

Ao observar as relações da nossa segunda forma normal, percebemos que não há nenhuma dependência transitiva e logo está na terceira forma normal.

# 6 Criação do Sistema

Com o esquema do banco de dados concretizado, ele foi então populado com dados randômicos. Apenas para as algumas tabelas, como Estados e Partidos é que as informações foram fornecidas por referências. O banco foi então conectado ao Express e ao Node JS, para que assim fosse criada uma API. Dessa maneira, para todas as tabelas foi criada uma função de leitura de dados que serve as informações armazenadas em formato JSON. Para três tabelas, sendo elas endereço, eleitor e voto, foi completado o CRUD. Para isso, foram desenvolvidas funções em javascript que fazem requisições em SQL para o banco de dados, a informação recebida é então encaminhada para o usuário. Para a facilidade do uso dessa API, foi então desenvolvida uma camada de visualização em navegador.

Um dos tópicos do projeto se refere a criação de procedures e views. Assim foram feitos os códigos SQL a seguir.

```
CDROP PROCEDURE IF EXISTS checkSecao;

delimiter //
CREATE PROCEDURE checkSecao(tit_eleitor BIGINT(20))
BEGIN
    DECLARE MUN_SEC INT;
    DECLARE MUN_END INT;

    SELECT zona.id_municipio
    INTO MUN_SEC
    FROM eleitor, zona, local, secao
    WHERE eleitor.titulo_eleitor = tit_eleitor AND eleitor.id_secao = secao.id

    SELECT endereco.id_municipio
    INTO MUN_END
    FROM eleitor, endereco
    WHERE eleitor.titulo_eleitor = tit_eleitor AND eleitor.id_endereco = endere

    IF MUN_SEC = MUN_END
    THEN
        SELECT 'TRUE' AS mesmo_municipio;
    ELSE
        SELECT 'FALSE' AS mesmo_municipio;
    END IF;
END//
delimiter ;
```

# 7 Camada de Mapeamento

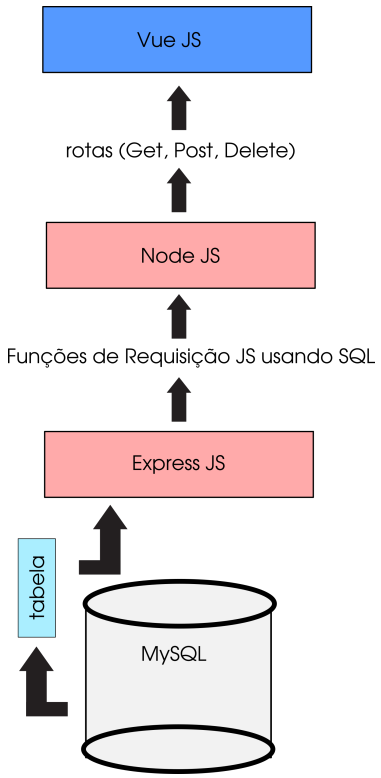


Figure 3: Diagrama com o mapeamento das informações da tabela até a visualização pelo usuário.

# References