

# Comandos DDL (CREATE, ALTER, DROP)

## Estudo Dirigido

R.A.: 1221116101

Nome: Victor Vaz de Oliveira

Os comandos DDL (*Data Definition Language*) são um conjunto de instruções que permitem definir, modificar ou excluir estruturas de dados em um banco de dados. Esses comandos são usados para criar, alterar ou excluir tabelas, índices, exibições, procedimentos armazenados, funções e outras estruturas de banco de dados. Alguns exemplos de comandos DDL incluem:

- **CREATE TABLE:** cria uma nova tabela no banco de dados
- **ALTER TABLE:** modifica a estrutura de uma tabela existente
- **DROP TABLE:** exclui uma tabela do banco de dados
- **CREATE INDEX:** cria um novo índice em uma tabela
- **ALTER INDEX:** modifica um índice existente
- **DROP INDEX:** exclui um índice de uma tabela
- **CREATE VIEW:** cria uma nova visão a partir de uma ou mais tabelas
- **ALTER VIEW:** modifica uma visão existente
- **DROP VIEW:** exclui uma visão do banco de dados

Perceba que nos exemplos acima 3 palavras-chave se repetem, são elas: **CREATE**, **ALTER**, **DROP**. Esses comandos fazem parte da Linguagem de Definição de Dados e são usados para:

ALTER	DROP	CREATE
O comando <b>ALTER</b> é usado para modificar a estrutura de um objeto existente em um banco de dados. Por exemplo, é possível adicionar uma	O comando <b>DROP</b> é usado para excluir um objeto existente em um banco de dados. Por exemplo, é possível excluir uma tabela, um índice,	O comando <b>CREATE</b> é usado para criar um novo objeto em um banco de dados. Por exemplo, é possível criar uma nova tabela, um novo índice, uma

nova coluna em uma tabela, alterar o tipo de dados de uma coluna, renomear um objeto, entre outras modificações.	uma exibição, uma função, entre outros objetos.	nova exibição, uma nova função, entre outros objetos. Ao criar um novo objeto, é preciso especificar suas características, como nome, tipo de dados, tamanho, entre outras informações relevantes para a sua criação.
--	---	---

## Exemplos

```
-- Cria um novo banco de dados chamado rh
CREATE DATABASE rh;

-- Exclui a tabela Cliente do banco de dados
DROP TABLE Cliente;

-- Altera o tipo de dados da coluna cpf de INT para CHAR(11) na tabela Cliente
ALTER TABLE Cliente
ALTER COLUMN cpf CHAR(11);

-- Acrescenta a coluna email na tabela Cliente
ALTER TABLE Cliente
ADD email VARCHAR(255);
```

**Importante:** É sempre recomendável fazer backup do banco de dados antes de executar qualquer comando DDL, para evitar perda de dados acidental.

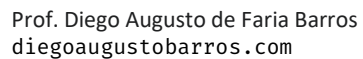
Os comandos DDL (Data Definition Language) exercem um importante papel no ciclo de vida e um banco de dados, pois é através deles que nós podemos criar os objetos em nossa base de dados. Após a elaboração do Modelo Físico o próximo passo é implementarmos nosso Diagrama Entidade Relacionamento (DER) usando o comando CREATE para criar nosso esquema e todas as tabelas, e, caso seja necessário, usaremos os comandos ALTER e DROP para corrigirmos algum erro cometido durante a codificação. Neste exercício nós iremos implementar o modelo físico para um Estacionamento Universitário.

## 1. Estacionamento Universitário

Uma Universidade deseja construir um sistema para facilitar e agilizar o controle de acesso de veículos aos seus cinco estacionamentos para prover mais segurança e comodidade para os seus usuários. O sistema deverá permitir que se cadastre todos os tipos de usuários (alunos, professores e funcionários), que receberão um cartão com um código de barras para sua identificação.

Cada usuário poderá solicitar o cadastramento de vários veículos com os quais utiliza os estacionamentos da universidade. Ao chegar a qualquer portão de acesso à universidade, o vigilante irá informar a placa do veículo e o usuário deverá passar o cartão magnético em um leitor de código de barras, e com isso, o sistema irá identificar se o veículo está relacionado com a identificação do usuário. Ao sair, o usuário simplesmente passa o seu cartão em outra leitora de código de barras.

O visitante (usuário não cadastrado) deverá pegar um cartão especial com os vigilantes. Através desses procedimentos, o sistema poderá fornecer dados de ocupação de cada estacionamento, além de permitir a consulta de quais os veículos estão, ou estiveram, dentro da universidade em um determinado dia e horário.



## 1. Criar Esquema

Um esquema de banco de dados é uma descrição formal e organizada da estrutura de um banco de dados. Ele define a forma como os dados serão armazenados, organizados e relacionados entre si, assim como as restrições e regras de integridade que devem ser seguidas. O esquema inclui as tabelas ou coleções, seus campos ou atributos, as chaves primárias e estrangeiras, e as relações entre elas. Ele é essencial para a criação e manutenção de um banco de dados, pois permite que os usuários compreendam a estrutura e as regras de negócio que governam o uso dos dados. O esquema também pode ser utilizado para garantir a consistência e a integridade dos dados armazenados, pois especifica as restrições que devem ser aplicadas aos dados. Além disso, ele é um importante recurso para a documentação de um banco de dados, permitindo que os desenvolvedores e usuários compreendam o significado dos dados e como eles devem ser utilizados. Um esquema é criado por meio da instrução `CREATE SCHEMA` ou em alguns sistemas através do sinônimo `CREATE DATABASE`.

### 1.1. Esquema Estacionamento Universitário

```
CREATE DATABASE estacionamento_universitario;
```

## 2. Criar Tabelas e Índices

Uma tabela em um banco de dados relacional é uma estrutura que contém dados organizados em linhas e colunas. As tabelas são a base para armazenar e manipular dados em um banco de dados. Cada tabela tem um nome único e é composta por uma ou mais colunas, que definem os diferentes tipos de dados que podem ser armazenados em cada linha. Para criar tabelas usando o comando `CREATE TABLE` NomeTabela, por exemplo:

```
CREATE TABLE Vendedor (  
    id_vendedor CHAR(10),  
    nome_vendedor VARCHAR(256),  
    nascimento DATE,  
    salario DECIMAL(10, 2),  
    PRIMARY KEY (id_vendedor)  
);
```

Existem várias convenções para nomes de tabelas em um banco de dados relacional, e a escolha pode variar de acordo com as preferências pessoais ou as normas adotadas por uma organização. No entanto, algumas boas práticas e convenções amplamente utilizadas incluem:

- Não use caracteres especiais, tais como: símbolos, espaços ou caracteres acentuados;
- Use nomes significativos e descritivos que representem claramente o conteúdo da tabela.
- Evite abreviações desnecessárias ou siglas que possam não ser facilmente compreendidas por outros usuários.
- Separe as palavras com sublinhados ( \_ ), para melhor legibilidade de nomes de colunas e tabelas;
- Use nomes de colunas no singular;
- Evite usar apenas **id** como nome de identificador primário da tabela;
- Não adicione uma coluna com o mesmo nome da tabela e vice versa;
- Para nomes de colunas, sempre utilize caixa baixa com nomes compostos separados por sublinhado (*underscore*), por exemplo: **data\_hora\_inicio**
- Evite sempre que possível concatenar dois nomes de tabelas para criar o nome de uma tabela associativa. Por exemplo, no lugar de **Obra\_Autor**, prefira **Autoria**;
- **Evite Plurais!** Use nomes de tabelas no singular ou no coletivo. Por exemplo, em ordem de preferência use Empregado ou Pessoal ao invés de Empregados (menos indicado)
- **Não utilize prefixos**, por exemplo: **tbl\_** antes do nome da tabela; ou qualquer outro prefixo descritivo ou notação húngara.

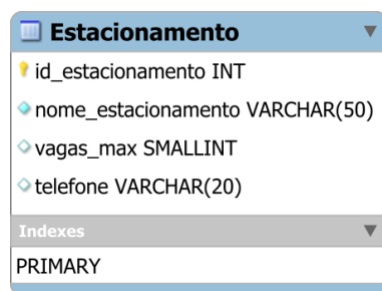
Seguir essas convenções pode tornar o banco de dados mais fácil de entender, manter e atualizar, além de ajudar na integração com outras aplicações e sistemas. A seguir, crie as tabelas do banco de dados usando o comando **CREATE TABLE**.

Em bancos de dados relacionais, um índice é uma estrutura de dados que ajuda a melhorar o desempenho das consultas em uma tabela. Ele funciona como uma tabela de pesquisa que armazena os valores de uma ou mais colunas da tabela em ordem específica, permitindo que o banco de dados localize mais rapidamente as linhas que atendem a uma consulta específica.

O índice é criado usando a instrução `CREATE INDEX`, que especifica o nome do índice, o nome da tabela e as colunas que serão indexadas. Por exemplo, para criar um índice para a coluna "nome" da tabela "clientes", pode-se usar a seguinte instrução: `CREATE INDEX idx_nome ON Cliente (nome);`

É importante lembrar que a criação de índices pode afetar o desempenho da inserção e atualização de dados na tabela, por isso é recomendado que sejam criados com cuidado e avaliando seu impacto no desempenho geral do banco de dados. Geralmente criamos índices para dados não voláteis, ou seja, colunas que não são atualizadas com frequência, por exemplo: chaves primárias ou estrangeiras; logo, não crie índices em colunas que são voláteis, por exemplo: saldo.

## 2.1. Tabela: Estacionamento



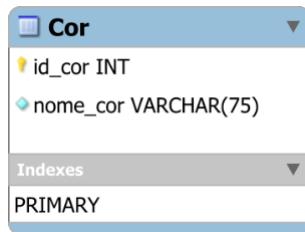
### 2.1.1. Descrição da Tabela

Field	Type	Null	Key	Default	Extra
id_estacionamento	int	NO	PRI	NULL	auto_increment
nome_estacionamento	varchar(50)	NO		NULL	
vagas_max	smallint unsigned	YES		0	
telefone	varchar(20)	YES		NULL	

### 2.1.2. Código SQL para Criar Tabela

```
CREATE TABLE Estacionamento (
    id_estacionamento INT NOT NULL AUTO_INCREMENT,
    nome_estacionamento VARCHAR(50) NOT NULL,
    vagas_max SMALLINT UNSIGNED DEFAULT 0,
    telefone VARCHAR(20) NULL DEFAULT NULL,
    PRIMARY KEY (id_estacionamento)
);
```

## 2.2. Tabela: Cor



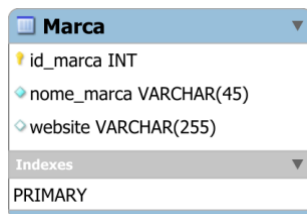
### 2.2.1. Descrição da Tabela

Field	Type	Null	Key	Default	Extra
id_cor	int	NO	PRI	NULL	auto_increment
nome_cor	varchar(75)	NO		NULL	

### 2.2.2. Código SQL para Criar Tabela

```
CREATE TABLE IF NOT EXISTS Cor (
    id_cor INT NOT NULL AUTO_INCREMENT,
    nome_cor VARCHAR(75) NOT NULL,
    PRIMARY KEY (id_cor)
);
```

## 2.3. Tabela: Marca



### 2.3.1. Descrição da Tabela

Field	Type	Null	Key	Default	Extra
id_marca	int	NO	PRI	NULL	auto_increment
nome_marca	varchar(45)	NO		NULL	
website	varchar(255)	YES		NULL	



```
+-----+-----+-----+-----+-----+-----+
```

### 2.3.2. Código SQL para Criar Tabela

```
CREATE TABLE IF NOT EXISTS Marca (
    id_marca INT NOT NULL AUTO_INCREMENT,
    nome_marca VARCHAR(45) NOT NULL,
    website VARCHAR(255) NULL,
    PRIMARY KEY (id_marca)
);
```

## 2.4. Tabela: Tipo Veículo

Tipo_Veiculo	
id_tipo_veiculo	CHAR(1)
tipo_veiculo	VARCHAR(45)
Indexes	
PRIMARY	

### 2.4.1. Descrição da Tabela

Field	Type	Null	Key	Default	Extra
id_tipo_veiculo	char(1)	NO	PRI	NULL	
tipo_veiculo	varchar(45)	NO		NULL	

### 2.4.2. Código SQL para Criar Tabela

```
CREATE TABLE IF NOT EXISTS Tipo_Veiculo (
    id_tipo_veiculo CHAR(1) NOT NULL,
    tipo_veiculo VARCHAR(45) NOT NULL,
    PRIMARY KEY (id_tipo_veiculo)
);
```

## 2.5. Tabela: Tipo Usuário

Tipo_Usuario	
id_tipo_usuario	CHAR(1)
tipo_usuario	VARCHAR(45)
Indexes	
PRIMARY	

### 2.5.1. Descrição da Tabela

Field	Type	Null	Key	Default	Extra
id_tipo_usuario	char(1)	NO	PRI	NULL	
tipo_usuario	varchar(45)	NO		NULL	

### 2.5.2. Código SQL para Criar Tabela

```
CREATE TABLE IF NOT EXISTS Tipo_Usuario (
  id_tipo_usuario CHAR(1) NOT NULL,
  tipo_usuario VARCHAR(45) NOT NULL,
  PRIMARY KEY (id_tipo_usuario)
);
```

## 2.6. Tabela: Sexo

Sexo	
idsexo	CHAR(1)
sexo	VARCHAR(50)
Indexes	
PRIMARY	

### 2.6.1. Descrição da Tabela

Field	Type	Null	Key	Default	Extra
idsexo	char(1)	NO	PRI	NULL	
sexo	varchar(50)	NO		NULL	

### 2.6.2. Código SQL para Criar Tabela

```

3. CREATE TABLE IF NOT EXISTS Sexo (
4.     id_sexo CHAR(1) NOT NULL,
5.     sexo VARCHAR(50) NOT NULL,
6.     PRIMARY KEY (id_sexo)
7. );

```

## 7.1. Tabela Categoria Modelo

Categoria_Modelo	
id_categoria	INT
tipo_modelo	VARCHAR(45)
Indexes	
PRIMARY	

### 7.1.1. Descrição da Tabela

Field	Type	Null	Key	Default	Extra
id_categoria_modelo	int	NO	PRI	NULL	auto_increment
tipo_modelo	varchar(45)	NO		NULL	

### 7.1.2. Código SQL para Criar Tabela

```

CREATE TABLE IF NOT EXISTS Categoria_Modelo (
    id_categoria_modelo INT NOT NULL AUTO_INCREMENT,
    tipo_modelo VARCHAR(45),
    PRIMARY KEY (id_categoria_modelo)
);

```

## 7.2. Tabela: Combustível

Combustivel	
id_combustivel	TINYINT
tipo_combustivel	VARCHAR(45)
Indexes	
PRIMARY	

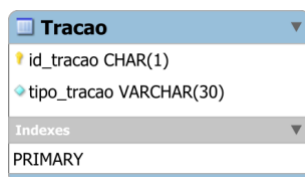
### 7.2.1. Descrição da Tabela

Field	Type	Null	Key	Default	Extra
id_combustivel	tinyint	NO	PRI	NULL	auto_increment
tipo_combustivel	varchar(45)	NO		NULL	

### 7.2.2. Código SQL para Criar Tabela

```
CREATE TABLE IF NOT EXISTS Combustivel (
  id_combustivel TINYINT NOT NULL AUTO_INCREMENT,
  tipo_combustivel VARCHAR(45) NOT NULL,
  PRIMARY KEY (id_combustivel)
);
```

## 7.3. Tabela: Tração



Field	Type
id_tracao	CHAR(1)
tipo_tracao	VARCHAR(30)

Indexes

PRIMARY
---------

### 7.3.1. Descrição da Tabela

Field	Type	Null	Key	Default	Extra
id_tracao	char(1)	NO	PRI	NULL	
tipo_tracao	varchar(30)	NO		NULL	

### 7.3.2. Código SQL para Criar Tabela

```
CREATE TABLE IF NOT EXISTS Tracao (
  id_tracao CHAR(1) NOT NULL,
  tipo_tracao VARCHAR(30) NOT NULL,
  PRIMARY KEY (id_tracao)
);
```

## 7.4. Tabela: Transmissão

Transmissao	
id_transmissao	CHAR(1)
tipo_transmissao	VARCHAR(10)
Indexes	
PRIMARY	

### 7.4.1. Descrição da Tabela

Field	Type	Null	Key	Default	Extra
id_transmissao	char(1)	NO	PRI	NULL	
tipo_transmissao	varchar(10)	NO		NULL	

### 7.4.2. Código SQL para Criar Tabela

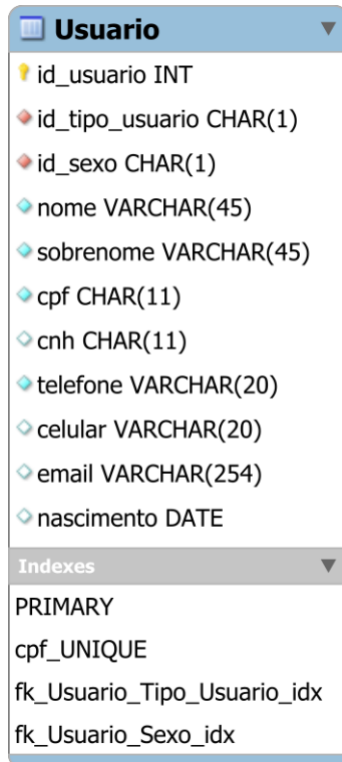
```

8. CREATE TABLE IF NOT EXISTS Transmissao (
9.     id_transmissao CHAR(1) NOT NULL,
10.    tipo_transmissao VARCHAR(10) NOT NULL,
11.    PRIMARY KEY (id_transmissao)
12. );

```

## 12.1. Tabela: Usuário

Uma chave estrangeira é um atributo ou conjunto de atributos em uma tabela de um banco de dados relacional que faz referência a uma chave primária em outra tabela. Essa chave é usada para estabelecer uma relação entre duas tabelas, permitindo que os dados sejam correlacionados entre elas. As chaves estrangeiras são amplamente utilizadas em bancos de dados relacionais para estabelecer relações entre as tabelas e garantir a integridade e consistência dos dados. A sua utilização é uma boa prática de modelagem de banco de dados, pois ajuda a evitar problemas de inconsistência e redundância de dados. Para criar uma chave estrangeira usamos a cláusula **FOREIGN KEY** que define a coluna como uma chave estrangeira e a cláusula **REFERENCES** indica a tabela e a coluna da chave primária que a chave estrangeira fará referência. Observe no exemplo abaixo:



### 12.1.1. Descrição da Tabela

Field	Type	Null	Key	Default	Extra
id_usuario	int	NO	PRI	NULL	auto_increment
id_tipo_usuario	char(1)	NO	MUL	NULL	
id_sexo	char(1)	NO	MUL	NULL	
nome	varchar(45)	NO		NULL	
sobrenome	varchar(45)	NO		NULL	
cpf	char(11)	NO	UNI	NULL	
cnh	char(11)	YES		NULL	
telefone	varchar(20)	NO		NULL	
celular	varchar(20)	YES		NULL	
email	varchar(254)	YES		NULL	
nascimento	date	YES		NULL	

### 12.1.2. Código SQL para Criar Tabela

```
CREATE TABLE Usuario (
    id_usuario INT NOT NULL AUTO_INCREMENT,
    id_tipo_usuario CHAR(1) NOT NULL,
    id-sexo CHAR(1) NOT NULL,
    nome VARCHAR(45) NOT NULL,
    sobrenome VARCHAR(45) NOT NULL,
    cpf CHAR(11) NOT NULL,
```

```

cnh CHAR(11) NULL DEFAULT NULL,
telefone VARCHAR(20) NOT NULL,
celular VARCHAR(20) NULL DEFAULT NULL,
email VARCHAR(254) NULL DEFAULT NULL,
nascimento DATE NULL DEFAULT NULL,
PRIMARY KEY (id_usuario),
CONSTRAINT cpf_unique UNIQUE(cpf),
CONSTRAINT fk_usuario_tipo_usuario FOREIGN KEY (id_tipo_usuario)
REFERENCES Tipo_usuario (id_tipo_usuario),
CONSTRAINT fk_usuario_sexo FOREIGN KEY (id_sexo)
REFERENCES Sexo(id_sexo)
);

```

### 12.1.3. Código SQL para Criar Índices

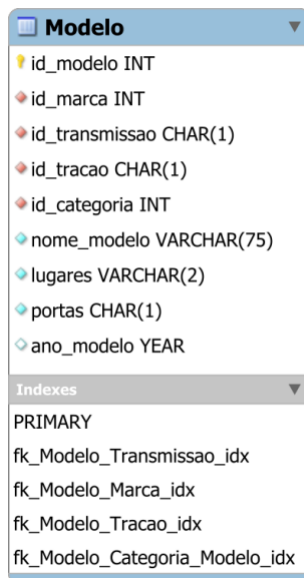
```

CREATE INDEX fk_usuario_tipo_usuario_idx ON Usuario(id_tipo_usuario);

CREATE INDEX fk_usuario_sexo_idx ON Usuario(id_sexo);

```

## 12.2. Tabela: Modelo



### 12.2.1. Descrição da Tabela

Field	Type	Null	Key	Default	Extra
id_modelo	int	NO	PRI	NULL	auto_increment
id_marca	int	NO	MUL	NULL	
id_transmissao	char(1)	NO	MUL	NULL	
id_tracao	char(1)	NO	MUL	NULL	

id_categoria_modelo	int	NO	MUL	NULL		
nome_modelo	varchar(75)	NO		NULL		
lugares	varchar(2)	NO		NULL		
portas	char(1)	NO		NULL		
ano_modelo	year	YES		NULL		
+-----+-----+-----+-----+-----+-----+						

### 12.2.2. Código SQL para Criar Tabela

```
CREATE TABLE IF NOT EXISTS Modelo (
    id_modelo INT NOT NULL AUTO_INCREMENT,
    id_marca INT NOT NULL,
    id_transmissao CHAR(1) NOT NULL,
    id_tracao CHAR(1) NOT NULL,
    id_categoria_modelo INT NOT NULL,
    nome_modelo VARCHAR(75) NOT NULL,
    lugares VARCHAR(2) NOT NULL,
    portas CHAR(1) NOT NULL,
    ano_modelo YEAR NULL DEFAULT NULL,
    PRIMARY KEY (id_modelo),
    CONSTRAINT fk_Modelo_Transmissao FOREIGN KEY (id_transmissao)
        REFERENCES Transmissao (id_transmissao),
    CONSTRAINT fk_Modelo_Marca FOREIGN KEY (id_marca)
        REFERENCES Marca (id_marca),
    CONSTRAINT fk_Modelo_Tracao FOREIGN KEY (id_tracao)
        REFERENCES Tracao (id_tracao),
    CONSTRAINT fk_Modelo_Categoria_Modelo FOREIGN KEY
(id_categoria_modelo)
        REFERENCES Categoria_Modelo (id_categoria_modelo)
);
```

### 12.2.3. Código SQL para Criar Índices

```
CREATE INDEX fk_Modelo_Transmissao_idx ON Modelo(id_transmissao);
CREATE INDEX fk_Modelo_Marca_idx ON Modelo(id_marca);
CREATE INDEX fk_Modelo_Tracao_idx ON Modelo(id_tracao);
CREATE INDEX fk_Modelo_Categoria_Modelo ON Modelo(id_categoria_modelo);
```



## 12.3. Tabela: Veículo

Veiculo	
renavam	CHAR(11)
id_marca	INT
id_modelo	INT
id_tipo_veiculo	CHAR(1)
id_cor	INT
placa	VARCHAR(8)
chassi	VARCHAR(20)
ano_fabricacao	YEAR
Indexes	
PRIMARY	
placa_UNIQUE	
chassi_UNIQUE	
fk_Veiculo_Marca_idx	
fk_Veiculo_Modelo_idx	
fk_Veiculo_Tipo_Veiculo_idx	
fk_Veiculo_Cor_idx	

### 12.3.1. Descrição da Tabela

Field	Type	Null	Key	Default	Extra
renavam	char(11)	NO	PRI	NULL	
id_marca	int	NO	MUL	NULL	
id_modelo	int	NO	MUL	NULL	
id_tipo_veiculo	char(1)	NO	MUL	NULL	
id_cor	int	NO	MUL	NULL	
placa	varchar(8)	NO	UNI	NULL	
chassi	varchar(20)	YES	UNI	NULL	
ano_fabricacao	year	NO		NULL	

### 12.3.2. Código SQL para Criar Tabela

```
CREATE TABLE IF NOT EXISTS Veiculo (
    renavam CHAR(11) NOT NULL,
    id_marca INT NOT NULL,
    id_modelo INT NOT NULL,
    id_tipo_veiculo CHAR(1) NOT NULL,
    id_cor INT NOT NULL,
    placa VARCHAR(8) NOT NULL,
    chassi VARCHAR(20) NULL DEFAULT NULL,
    ano_fabricacao YEAR NOT NULL,
    PRIMARY KEY(renavam),
```

```
CONSTRAINT placa_unique UNIQUE (placa),
CONSTRAINT chassi_unique UNIQUE (chassi),
CONSTRAINT fk_Veiculo_Marca FOREIGN KEY (id_marca)
    REFERENCES Marca (id_marca),
CONSTRAINT fk_Veiculo_Modelo FOREIGN KEY (id_modelo)
    REFERENCES Modelo (id_modelo),
CONSTRAINT fk_Veiculo_Tipo_Veiculo FOREIGN KEY (id_tipo_veiculo)
    REFERENCES Tipo_Veiculo (id_tipo_veiculo),
CONSTRAINT fk_Veiculo_Cor FOREIGN KEY (id_cor)
    REFERENCES Cor (id_cor)
);
```

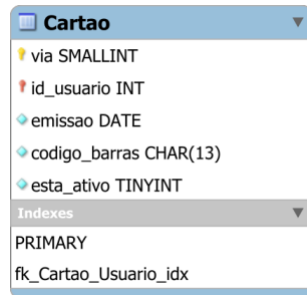
### 12.3.3. Código SQL para Criar Índices

```
CREATE INDEX fk_Veiculo_Marca_idx ON Veiculo(id_marca);
CREATE INDEX fk_Veiculo_Modelo_idx ON Veiculo(id_modelo);
CREATE INDEX fk_Veiculo_Tipo_Veiculo_idx ON Veiculo(id_tipo_veiculo);
CREATE INDEX fk_Veiculo_Cor_idx ON Veiculo(id_cor);
```

## 12.4. Tabela: Cartão de Estacionamento

Uma chave primária composta é uma chave primária que consiste em duas ou mais colunas em uma tabela de um banco de dados relacional. Em vez de uma única coluna que atua como chave primária, as colunas são combinadas para criar uma chave primária única para identificar cada registro da tabela. A principal vantagem de usar uma chave primária composta é que ela permite uma maior granularidade e precisão na identificação de registros. Em algumas situações, uma única coluna pode não ser suficiente para identificar exclusivamente cada registro em uma tabela, então a combinação de colunas é necessária. Esse tipo de chave é muito comum em tabelas associativas onde a chave primária é constituída pela concatenação de chaves de outras tabelas, ou em relacionamento com entidades fracas, por exemplo: **PRIMARY KEY (id\_estacionamento, renavam, data\_hora\_entrada)**.

O relacionamento entre uma entidade fraca e forte é chamado de relacionamento identificador (*identifying relationship*) nele uma tabela (tabela dependente) possui uma chave primária que inclui a chave primária de outra tabela (tabela principal). Nesse caso, a chave primária da tabela dependente é uma chave primária composta, observe **PRIMARY KEY (via, id\_usuario)** no exemplo abaixo:



#### 12.4.1. Descrição da Tabela

Field	Type	Null	Key	Default	Extra
via	smallint	NO	PRI	NULL	
id_usuario	int	NO	PRI	NULL	
emissao	date	NO		NULL	
codigo_barras	char(13)	NO		NULL	
esta_ativo	tinyint	NO		0	

#### 12.4.2. Código SQL para Criar Tabela

```
CREATE TABLE Cartao (
  via SMALLINT NOT NULL,
  id_usuario INT NOT NULL,
  emissao DATE NOT NULL,
  codigo_barras CHAR(13) NOT NULL,
  esta_ativo TINYINT NOT NULL DEFAULT 0,
  PRIMARY KEY (via, id_usuario),
  CONSTRAINT fk_Cartao_Usuario FOREIGN KEY (id_usuario)
    REFERENCES Usuario(id_usuario)
);
```

#### 12.4.3. Código SQL para Criar Índices

```
CREATE INDEX fk_Cartao_Usuario_idx ON Cartao(id_usuario);
```

## 12.5. Tabela: Histórico de Estacionamento

Column	Type	Null	Key	Default	Extra
id_estacionamento	INT	NO	PRI	NULL	
renavam	CHAR(11)	NO	PRI	NULL	
data_hora_entrada	TIMESTAMP	NO	PRI	CURRENT_TIMESTAMP	DEFAULT_GENERATED
data_hora_saida	DATETIME	YES		NULL	
permanencia	TIME	YES		NULL	STORED GENERATED

### 12.5.1. Descrição da Tabela

Field	Type	Null	Key	Default	Extra
id_estacionamento	int	NO	PRI	NULL	
renavam	char(11)	NO	PRI	NULL	
data_hora_entrada	timestamp	NO	PRI	CURRENT_TIMESTAMP	DEFAULT_GENERATED
data_hora_saida	datetime	YES		NULL	
permanencia	time	YES		NULL	STORED GENERATED

### 12.5.2. Código SQL para Criar Tabela

```
CREATE TABLE Historico_Estacionamento (
    id_estacionamento INT NOT NULL,
    renavam CHAR(11) NOT NULL,
    data_hora_entrada TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    data_hora_saida DATETIME NULL DEFAULT NULL,
    permanencia TIME GENERATED ALWAYS AS (TIMEDIFF(data_hora_saida, data_hora_entrada)) STORED,
    PRIMARY KEY (id_estacionamento, renavam, data_hora_entrada),
    CONSTRAINT fk_Historico_Estacionamento_Estacionamento FOREIGN KEY (id_estacionamento)
        REFERENCES Estacionamento(id_estacionamento),
    CONSTRAINT fk_Historico_Estacionamento_Veiculo FOREIGN KEY (renavam)
        REFERENCES Veiculo(renavam)
);
```

Observe que a coluna *permanencia*, é um atributo derivado (*GENERATED COLUMN*) que será calculado como a diferença da data e hora de saída e entrada. Além disso, o resultado gerado será guardado no banco porque usamos a palavra *STORED*.

### 12.5.3. Código SQL para Criar Índices

```
CREATE INDEX fk_Historico_Estacionamento_Veiculo_idx ON
Historico_Estacionamento(renavam);
```

```
CREATE INDEX fk_Historico_Estacionamento_Estacionamento_idx ON
Historico_Estacionamento (id_estacionamento);
```

## 12.6. Tabela: Propriedade

Column	Type
renavam	CHAR(11)
id_usuario	INT
data_inicio	TIMESTAMP
data_fim	DATETIME

Index	Type
PRIMARY	PRIMARY
fk_Propriedade_Proprietario_idx	FOREIGN KEY
fk_Propriedade_Veiculo_idx	FOREIGN KEY

### 12.6.1. Descrição da Tabela

Field	Type	Null	Key	Default	Extra
renavam	char(11)	NO	PRI	NULL	
id_usuario	int	NO	PRI	NULL	
data_inicio	timestamp	NO	PRI	CURRENT_TIMESTAMP	DEFAULT_GENERATED
data_fim	datetime	YES		NULL	

### 12.6.2. Código SQL para Criar Tabela

```
CREATE TABLE IF NOT EXISTS Propriedade (
    renavam CHAR(11) NOT NULL,
    id_usuario INT NOT NULL,
    data_inicio TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    data_fim DATETIME NULL DEFAULT NULL,
    PRIMARY KEY (renavam, id_usuario, data_inicio),
    CONSTRAINT fk_Propriedade_Proprietario FOREIGN KEY (id_usuario)
        REFERENCES Usuario (id_usuario),
    CONSTRAINT fk_Propriedade_Veiculo FOREIGN KEY (renavam)
        REFERENCES Veiculo (renavam)
);
```

### 12.6.3. Código SQL para Criar Índices

```
CREATE INDEX fk_Propriedade_Proprietario_idx ON Propriedade (id_usuario);
CREATE INDEX fk_Propriedade_Veiculo_idx ON Propriedade (renavam);
```

## 12.7. Tabela: Tipo Combustível

Tipo_Combustivel	
id_modelo	INT
id_combustivel	TINYINT
Indexes	
PRIMARY	
fk_Tipo_Combustivel_Combustivel_idx	
fk_Tipo_Combustivel_Modelo_idx	

### 12.7.1. Descrição da Tabela

Field	Type	Null	Key	Default	Extra
id_modelo	int	NO	PRI	NULL	
id_combustivel	tinyint	NO	PRI	NULL	

### 12.7.2. Código SQL para Criar Tabela

```
CREATE TABLE IF NOT EXISTS Tipo_Combustivel (
    id_modelo INT NOT NULL,
    id_combustivel TINYINT,
    PRIMARY KEY (id_modelo, id_combustivel),
    CONSTRAINT fk_Tipo_Combustivel_Combustivel FOREIGN KEY (id_combustivel)
        REFERENCES Combustivel (id_combustivel),
    CONSTRAINT fk_Tipo_Combustivel_Modelo FOREIGN KEY (id_modelo)
        REFERENCES Modelo (id_modelo)
);
```

### 12.7.3. Código SQL para Criar Índices

```
CREATE INDEX fk_Tipo_Combustivel_Combustivel_idx ON Tipo_Combustivel
(id_combustivel);
CREATE INDEX fk_Tipo_Combustivel_Modelo_idx ON Tipo_Combustivel (id_modelo);
```

## 13. Criar Arquivo SQL

Para criar um arquivo SQL, você pode usar qualquer editor de texto simples, como o Bloco de Notas (no Windows) ou o TextEdit (no Mac), desde que o arquivo seja salvo com a extensão .sql. Por exemplo, para criar um arquivo SQL usando o Bloco de Notas no Windows, faça:

1. Abra o Bloco de Notas;

2. Digite ou copie os comandos SQL que deseja executar no arquivo. No nosso caso, você deverá copiar todos os comandos SQL deste arquivo para o bloco de notas;
3. Salve o arquivo com a extensão .sql. Para fazer isso, faça:
  - 3.1. Clique em **Arquivo** no menu superior do Bloco de Notas;
  - 3.2. Selecione **Salvar Como**;
  - 3.3. Escolha um nome para o arquivo e adicione a extensão .sql no final do nome por exemplo:  
`meu_arquivo.sql`
  - 3.4. Certifique-se de escolher: **Todos os arquivos**; como o tipo de arquivo antes de salvar o arquivo.

### 13.1. Execute seu Arquivo .SQL

Pronto! Agora você tem um arquivo SQL que pode ser executado em seu banco de dados. Para executar os comandos contidos neste arquivo, você pode usar um cliente SQL, como o MySQL Workbench, ou executá-lo usando a linha de comando do seu banco de dados.