🚲 Sistema de Gestão de Ocorrências de Furto e Recuperação de Bicicletas

Este projeto é um sistema de banco de dados relacional simples, construído em **SQLite**, para gerenciar o cadastro de vítimas, bicicletas e ocorrências de furto e recuperação. Foi desenvolvido para demonstrar e praticar a modelagem de dados, a implementação de restrições de Chave Estrangeira (`FOREIGN KEY`) e a execução de consultas SQL.

🏗️ Estrutura do Projeto

O projeto é baseado em um único arquivo de banco de dados SQLite e scripts SQL para criação de tabelas e inserção de dados.

Esquema do Banco de Dados

O sistema é composto pelas seguintes tabelas principais, que se relacionam de forma hierárquica (Vítima ➡️ Bicicleta ➡️ Ocorrência):

Tabela	Função	Chaves Estrangeiras (FKs)
`vitima`	Armazena dados pessoais dos proprietários.	Nenhuma (É a tabela Pai)
`bicicleta`	Armazena os detalhes da bicicleta furtada/recuperada.	`id_vitima` (referencia `vitima`)
`ocorrencia`	Armazena os detalhes do furto, localização e data/hora.	`id_bicicleta` (referencia `bicicleta`)
`evidencia`	(Se existir)	
`investigacao`	(Se existir)	

🚀 Como Utilizar (SQLiteStudio)

Para explorar e executar as consultas SQL deste projeto, você precisará de um cliente SQLite:

1. **Instalação do SQLiteStudio:** Baixe e instale o SQLiteStudio.
2. **Abrir o Banco de Dados:**
    - Abra o SQLiteStudio.
    - Clique em `Database` > `Add a database...`.
    - Selecione o arquivo `.sqlite` do projeto (por exemplo, `Sistema de Gestao de Ocorrencias.sqlite`).
3. **Executar Consultas:**
    - Clique no banco de dados na barra lateral esquerda.
    - Clique no ícone do **Editor SQL** (SQL Editor) para abrir uma nova aba de consulta.
    - Cole e execute os scripts de criação de tabelas e inserção de dados.

🔑 Restrições de Integridade (Pontos Cruciais)

O principal desafio deste projeto é garantir a **Integridade Referencial**. Ao inserir dados, siga rigorosamente esta ordem para evitar o erro `FOREIGN KEY constraint failed`:

1. **Vítima:** Deve ser inserida primeiro.
2. **Bicicleta:** O `id_vitima` usado aqui **deve** existir na tabela `vitima`.
3. **Ocorrência:** O `id_bicicleta` usado aqui **deve** existir na tabela `bicicleta`.

Exemplo de Inserção Correta (Ordem)

SQL

```
-- 1. Inserir a Vítima (Pai)
INSERT INTO vitima (id_vitima, nome, CPF, telefone, email)
VALUES (1, 'Exemplo Vítima', '11122233300', '999999999',
'contato@exemplo.com');

-- 2. Inserir a Bicicleta (Filha de Vítima)
INSERT INTO bicicleta (id_bicicleta, n_de_serie, marca, modelo, cor,
aro, id_vitima)
VALUES (101, 'SERIAL12345', 'Caloi', 'Sport', 'Preta', 26, 1);

-- 3. Inserir a Ocorrência (Filha de Bicicleta)
INSERT INTO ocorrencia (id_ocorrencia, data_furto, hora_do_furto,
rua_bairro_do_furto, id_bicicleta)
VALUES (1, '2025-12-01', '10:00', 'Rua Principal, Centro', 101);
```

📄 Scripts SQL Essenciais

Aqui estão os comandos básicos para iniciar o banco de dados:

Criação da Tabela `ocorrencia` (Formato Correto)

SQL
```
CREATE TABLE ocorrencia (
    id_ocorrencia         INTEGER    PRIMARY KEY,
    data_furto            TEXT (15)  NOT NULL,
    hora_do_furto         TEXT (6),
    rua_bairro_do_furto   TEXT (100) NOT NULL,
    id_bicicleta          INTEGER    REFERENCES bicicleta
(id_bicicleta)
);
```

Comandos de Consulta (Exemplo)

SQL
```
-- Consultar todas as bicicletas furtadas
SELECT
    b.marca,
    b.n_de_serie,
    v.nome AS Proprietário,
    o.data_furto
FROM bicicleta b
JOIN vitima v ON b.id_vitima = v.id_vitima
JOIN ocorrencia o ON b.id_bicicleta = o.id_bicicleta
ORDER BY o.data_furto DESC;
```

🤝 Contribuição

Sinta-se à vontade para sugerir melhorias na modelagem, adicionar mais tabelas ou propor consultas mais complexas!