Senac - Desenvolver Banco de Dados - UC10 - 2024_A6-1/2

SQL - Data Definition Language (DDL)

Modificação e Criação de Estruturas de Dados (DDL - Data Definition Language) com SQLite e Python

A **DDL (Data Definition Language)** no SQL é responsável por criar, modificar e remover a estrutura do banco de dados. No **SQLite**, a DDL inclui a criação e alteração de **tabelas**, além da definição de **restrições (constraints)** que garantem a integridade dos dados.

---

1. Declarações DDL

1.1. CREATE TABLE

- **Descrição**:
  O comando CREATE TABLE no SQL é usado para **criar uma nova tabela** em um banco de dados. Ele define o nome da tabela e as colunas que ela terá, especificando o tipo de dados de cada coluna e, opcionalmente, restrições como chaves primárias, valores únicos ou valores não nulos.

  Exemplo em Python:

```python
import sqlite3

# Conectar ao banco de dados (ou criar se não existir)
conn = sqlite3.connect('exemplo.db')
cursor = conn.cursor()

# Criar uma tabela 'alunos'
cursor.execute('''
CREATE TABLE IF NOT EXISTS alunos (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    nome TEXT NOT NULL,
    idade INTEGER CHECK (idade >= 18),
    email TEXT UNIQUE
)
''')

print("Tabela 'alunos' criada com sucesso!")

# Fechar a conexão
conn.close()
```

- **Explicação**:
  - `id` é a chave primária com **AUTO INCREMENT**.
  - `nome` é obrigatório (**NOT NULL**).
  - `idade` deve ser maior ou igual a 18 (**CHECK**).
  - `email` precisa ser único (**UNIQUE**).

1.2. DROP TABLE

- **Descrição**:
    - O comando DROP TABLE é utilizado no SQL para **remover uma tabela existente** no banco de dados, junto com todos os seus dados e estrutura. Depois que a tabela é excluída, não é possível recuperar seus dados, a menos que haja um backup.
    - O comando `DROP TABLE` remove uma tabela existente e todos os seus dados.

Exemplo em Python:

```
cursor.execute('DROP TABLE IF EXISTS alunos')
```

1.3. ALTER TABLE

- **Descrição**:
    - O comando ALTER TABLE no SQL é usado para **modificar a estrutura de uma tabela** existente. Com ele, você pode adicionar, remover ou alterar colunas, além de modificar restrições, como chaves primárias ou índices.
    - O comando `ALTER TABLE` modifica a estrutura de uma tabela existente, como adicionar ou remover colunas.

Exemplo em Python:

```
cursor.execute('ALTER TABLE alunos ADD COLUMN curso TEXT DEFAULT "Indefinido"')
```

---

2. Restrições (Constraints)

As restrições (constraints) no SQL são regras aplicadas às colunas para garantir a integridade e consistência dos dados no banco de dados. Elas controlam quais tipos de dados são permitidos e como os dados podem ser manipulados.

Principais Restrições:

- NOT NULL: Impede que a coluna receba valores nulos.
- UNIQUE: Assegura que os valores em uma coluna sejam únicos.
- CHECK: Define uma condição que os valores devem satisfazer.
- DEFAULT: Define um valor padrão para uma coluna.
- PRIMARY KEY: Garante que a coluna tenha valores únicos e não nulos.
- FOREIGN KEY: Mantém a integridade referencial entre duas tabelas.
- AUTOINCREMENT: Faz com que valor de uma coluna seja automaticamente incrementado a cada novo registro

2.1. NOT NULL

- **Descrição**:
    - A restrição NOT NULL no SQL garante que uma coluna não pode conter valores nulos. Isso significa que, ao inserir ou atualizar um registro, é obrigatório fornecer um valor válido para essa coluna.
    - Garante que uma coluna **não possa conter valores nulos**.

Exemplo em Python:

```
cursor.execute('''
CREATE TABLE IF NOT EXISTS cursos (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    nome TEXT NOT NULL
)
''')
```

Aqui, a coluna `nome` é obrigatória e **não pode ser nula**.

2.2. UNIQUE

- **Descrição**:
    - A restrição UNIQUE no SQL garante que todos os valores em uma coluna (ou conjunto de colunas) sejam únicos, ou seja, não podem se repetir. Isso é útil para evitar duplicação de dados em campos importantes, como emails ou CPF.
    - Garante que todos os valores em uma coluna sejam **únicos** (sem duplicação).

Exemplo em Python:

```
cursor.execute('''
CREATE TABLE IF NOT EXISTS emails (
    id INTEGER PRIMARY KEY,
    email TEXT UNIQUE
)
''')
```

- Se dois registros tentarem inserir o mesmo email, a operação será rejeitada.

2.3. CHECK

- **Descrição**:
    - A restrição CHECK no SQL é usada para definir uma condição lógica que os valores inseridos em uma coluna devem obedecer. Se a condição não for satisfeita, o banco de dados rejeita a inserção ou atualização.
    - Define uma **condição que os valores da coluna devem obedecer**.

Exemplo em Python:

```
cursor.execute('''
CREATE TABLE IF NOT EXISTS funcionarios (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    nome TEXT NOT NULL,
    salario REAL CHECK (salario > 0)
)
''')
```

- O valor do `salario` deve ser maior que 0.

2.4. DEFAULT

- **Descrição**:
    - A restrição DEFAULT no SQL define um **valor padrão para uma coluna**, que será usado automaticamente se nenhum valor for inserido durante a inserção de um novo registro.
    - Define um **valor padrão** para uma coluna quando nenhum valor é fornecido.

Exemplo em Python:

```
cursor.execute('''
CREATE TABLE IF NOT EXISTS estudantes (
    id INTEGER PRIMARY KEY,
    nome TEXT NOT NULL,
    cidade TEXT DEFAULT 'São Paulo'
)
''')
```

- Se nenhum valor for informado para `cidade`, o valor será `'São Paulo'`.

2.5. PRIMARY KEY

- **Descrição**:
    - A PRIMARY KEY no SQL é uma restrição que identifica de forma única cada registro em uma tabela. Uma coluna ou um conjunto de colunas definidas como chave primária não pode conter valores duplicados ou nulos.

      Características:

        - Única por tabela: Cada tabela pode ter apenas uma PRIMARY KEY.
        - Combinação de colunas: Pode ser composta por mais de uma coluna (chave composta).
        - Índice automático: Ao definir uma PRIMARY KEY, o banco cria um índice para otimizar a busca
    - A **chave primária (PRIMARY KEY)** identifica unicamente cada registro na tabela.

Exemplo em Python:

```
cursor.execute('''
CREATE TABLE IF NOT EXISTS professores (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    nome TEXT NOT NULL
)
''')
```

- O campo `id` é uma **chave primária** única e incrementada automaticamente.

2.6. FOREIGN KEY

- **Descrição**:
  - A FOREIGN KEY é uma restrição usada para garantir a integridade referencial entre duas tabelas no banco de dados. Ela cria um vínculo entre uma coluna em uma tabela e a PRIMARY KEY de outra tabela. Isso garante que os valores inseridos na coluna referenciada correspondam a valores existentes na tabela pai, evitando inconsistências.
  - A **chave estrangeira (FOREIGN KEY)** cria uma relação entre duas tabelas.

Exemplo em Python:

```
cursor.execute('''
CREATE TABLE IF NOT EXISTS turmas (
    id INTEGER PRIMARY KEY,
    aluno_id INTEGER,
    FOREIGN KEY (aluno_id) REFERENCES alunos (id)
)
''')
```

- A coluna `aluno_id` deve referenciar um `id` existente na tabela `alunos`.

2.7. AUTO INCREMENT

- **Descrição**:
  - A restrição AUTO INCREMENT no SQL é usada para gerar automaticamente valores únicos e sequenciais para uma coluna, geralmente uma chave primária. A cada novo registro inserido, o banco de dados incrementa automaticamente o valor sem a necessidade de especificação manual.
  - **AUTO INCREMENT** faz com que o valor de uma coluna seja automaticamente incrementado a cada novo registro.

Exemplo em Python:

```
cursor.execute('''
CREATE TABLE IF NOT EXISTS matriculas (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    aluno TEXT NOT NULL
)
''')
```

Exemplo Completo: Estrutura Completa com Várias Restrições

```
# Criar tabela com várias restrições
cursor.execute('''
CREATE TABLE IF NOT EXISTS alunos (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    nome TEXT NOT NULL,
    idade INTEGER CHECK (idade >= 18),
    email TEXT UNIQUE,
    curso TEXT DEFAULT 'Indefinido'
)
''')

# Inserir dados na tabela
cursor.execute('''
INSERT INTO alunos (nome, idade, email)
VALUES ('Gustavo', 20, 'gustavo@example.com')
''')

cursor.execute('''
INSERT INTO alunos (nome, idade, email)
VALUES ('Nycollas', 22, 'nycollas@example.com')
''')

# Consultar dados
cursor.execute('SELECT * FROM alunos')
resultados = cursor.fetchall()
for linha in resultados:
    print(linha)
```

Resumo

Declaração SQL	Descrição
CREATE TABLE	Cria uma nova tabela.
DROP TABLE	Remove uma tabela existente.
ALTER TABLE	Modifica uma tabela existente.

Restrição	Descrição
NOT NULL	Garante que a coluna não aceite valores nulos.
UNIQUE	Garante que os valores sejam únicos.
CHECK	Impõe uma condição aos valores.
DEFAULT	Define um valor padrão.
PRIMARY KEY	Identifica unicamente cada registro.
FOREIGN KEY	Estabelece uma relação entre duas tabelas.
AUTO INCREMENT	Incrementa automaticamente a chave primária.