escola britânica de artes criativas & tecnologia

Desenvolvedor Full Stack Python

Definindo a Estrutura de um Banco de Dados (DDL)

escola britânica de artes criativa & tecnologia

AGENDA

- **Base de Dados e Schemas**
- **Tabelas e Colunas**
- **Tipos de Dados e Protegendo a Integridade de dados com constraint**
- **ID`s automáticos**

Base de Dados e Esquemas

Na Base de Dados definimos toda a nossa estrutura de dados do mais alto nível até o mais baixo:

Base de Dados > Schema > Tabelas

Para criar uma base de dados basta apenas executar o comando:

CREATE DATABASE <nome da tabela>;

Base de Dados e Esquemas

Schemas são espaços onde adicionamos e organizamos tabelas, funções e entre outros objetos de banco de dados.

A sintaxe para criação de um schema é:

CREATE SCHEMA <schema_name>; ou
CREATE SCHEMA IF NOT EXISTS <schema_name>;

Criar Tabelas

Para criar **Tabelas** e **Colunas** no SQL precisamos utilizar o comando **CREATE TABLE** seguido das informações de Colunas:

```
CREATE TABLE table_name(
        column1 datatype,
        column2 datatype,
        column3 datatype,
        .....
        columnN datatype,
        PRIMARY KEY (  one or more columns  )
```

Remover Tabelas

Para remover **Tabelas:**

DROP TABLE <Nome da Tabela>

DROP TABLE Customer

Adicionar Colunas

Para **adicionar Colunas** em tabelas existentes no SQL precisamos utilizar o comando **ALTER TABLE** seguido das informações da coluna que queremos adicionar

ALTER TABLE <Nome da Tabela> **ADD** <nome da coluna> <tipo do campo>
ALTER TABLE Customer **ADD** email varchar;

Remover Colunas

Para **remover Colunas** em tabelas existentes no SQL precisamos utilizar o comando **ALTER TABLE** seguido das informações da coluna que queremos **remover**

ALTER TABLE <Nome da Tabela> **DROP** <nome da coluna> <tipo do campo>;
ALTER TABLE Customer **DROP** email varchar;

Tipos de dados em Colunas

Oracle	PostgreSQL
VARCHAR2(n)	VARCHAR(n)
CHAR(n)	CHAR(n)
NUMBER (n,m)	NUMERIC (n,m)
NUMBER(4)	SMALLINT
NUMBER(9)	INT
NUMBER(18)	BIGINT
NUMBER(n)	NUMERIC (n)
DATE	TIMESTAMP (0)
TIMESTAMP WITH LOCAL TIME ZONE	TIMESTAMPTZ
CLOB	TEXT
BLOB RAW(n)	BYTEA(1 GB limit) Large object

Tipos de dados foram criados para proteger as **informações** dos dados da nossa tabela.

Tipos de dados em Colunas

Para alterar o tipo de dados de um campo específico em tabelas existentes:

ALTER TABLE <nome_da_tabela> ALTER COLUMN <nome_da_coluna> TYPE varchar(30);

Caso o tipo do campo seja diferente do tipo que queremos criar (exemplo texto para número) temos que adicionar um **asset_no** para o postgres fazer a conversão do tipo:

ALTER TABLE nome_da_tabela> ALTER COLUMN <nome_da_coluna> TYPE INT USING <nome_da_coluna> ::integer;

Constraints...

Constraints foram criados para **proteger** e prevenir dados incompletos entrarem na nossa base além de dados **repetidos** mantendo dados únicos dentro da nossa base.

Constraints Not Null

Para não aceitar valores Nulos na nossa base podemos utilizar o **NOT NULL:**

```
        ID INT PRIMARY    KEY          NOT NULL,
        NAME                    TEXT          NOT
NULL,
        AGE               INT          NOT NULL,
        ADDRESS           CHAR (50),
```

Para adicionar constraints **Not Null** em tabelas existentes:

```
ALTER TABLE <nome_da_tabela> ALTER COLUMN <nome_da_coluna> SET NOT NULL;
```

Constraints Unique

Para não aceitar valores Repetidos na nossa base podemos utilizar o **UNIQUE:**

```
CREATE TABLE products (
        product_no integer,
        name text,
        price numeric,
        UNIQUE (product_no)
);
```

Para adicionar constraints **Unique** em tabelas existentes:

ALTER TABLE <nome_da_tabela> ALTER COLUMN <nome_da_coluna> SET NOT NULL;

Valores Automáticos

Podemos usar o **serial** do Postgres para criar tabelas com números de id **automáticos**, geralmente usamos a coluna **ID** para receber esses valores automáticos:

```
testdb=# CREATE TABLE COMPANY (
        ID      SERIAL PRIMARY KEY,
        NAME                TEXT
NOT NULL,
        AGE             INT             NOT
NULL,
        ADDRESS     CHAR (50),
```

Valores Automáticos

Agora para inserir valores na nossa tabela, não precisamos mais referenciar o campo de ID:

```
INSERT INTO COMPANY (NAME,AGE,ADRESS,SA-
LARY)
VALUES (‘Paul’, 32, ‘California’, 20000.00 );

INSERT INTO COMPANY (NAME,AGE,ADRESS,SA-
LARY)
VALUES (‘Allen’, 25, ‘Texas’, 15000.00 );

INSERT INTO COMPANY (NAME,AGE,ADRESS,SA-
LARY)
```

Resumo

Aprendemos um pouco como a estrutura de um banco de dados funciona e quais são as principais funções do **DDL** como:

- Tabelas
- Colunas
- Constraints
- Números de ID automáticos

Exercício

escola britânica de artes criativa & tecnologia

Nesse exercício, vamos criar as **tabelas de cliente, estoque e produto,** utilize os desenhos que fizemos no lucidchart como referência.