SQL

STRUCTURED QUERY LANGUAGE
LINGUAGEM DE CONSULTA ESTRUTURA
DDL, tipos de dados

AGENDA

- Introdução
- SQL introdução
- SCHEMA
- TIPOS DE DADOS
- DDL
  - CREATE
  - CHECK

INTRODUÇÃO

- Uma das grandes responsáveis pelo sucesso dos SGBD
- Por ter um padrão, facilita a migração entre diferentes sistemas
- Cada SGBD pode ter especificações, mas sempre há o padrão SQL
- Foi criada pela IBM Research
- A criação da SQL foi esforço do American National Standards Institute (ANSI) e da International Standard Organization (ISO)
- Houveram vários padrões ao longo do tempo

DEFINIÇÕES E TIPOS DE DADOS EM SQL

SQL	MODELO RELACIONAL
TABELA	RELAÇÃO
LINHA	TUPLA
COLUNA	ATRIBUTO

SCHEMA

- Agrupamento de tabela e outras construções que pertencem a mesma aplicação de BD
- Geralmente os SGBD possuem um SCHEMA padrão.
- Podem ser criados outros geralmente por qualquer usuário
- Padrão:

```
CREATE SCHEMA EMPRESA AUTHORIZATION 'jsilva'
```

TIPOS DE LINGUAGEM SQL

- DDL - DATA DEFINITION LANGUAGE

Linguagem para criação/modificação das **estruturas** dos modelo de dados.

Exemplos: CREATE TABLE, ALTER TABLE, DROP...

- DML - DATA MANIPULATION LANGUAGE

Linguagem para **manipulação** dos dados armazenados (inserção, alteração, exclusão) em estruturas de dados.

Exemplos:INSERT, UPDATE, DELETE...

TIPOS DE DADOS

TIPOS DE DADOS - NUMÉRICOS

INTEIROS:

INTEGER ou INT, SMALLINT

PONTO FLUTUANTE:

FLOAT ou REAL, DOUBLE PRECISON

NÚMERO FORMATADOS:

DECIMAL(i,j), DEC(i,j), NUMERIC(i,j) -> i = número de dígitos / j = número de dígitos após o ponto decimal

TIPOS DE DADOS - CADEIAS DE TEXTO

TAMANHO FIXO:

CHAR(n) ou CHARACTER(n) -> n = número de caracteres

TAMANHO VARIÁVEL:

VARCHAR(n) ou CHAR VARYING(n) ou CHARACTER VARYING(n) -> n = número máximo de caracteres

Os valores devem estar sempre entre ‘ ’. Ex: ‘FERNANDO’ ou ‘F’.

Para concatenar valor, usa-se o ||.
Ex: ‘FER’ || ‘NANDO’ = ‘FERNANDO’

TIPOS DE DADOS - CADEIAS DE TEXTO

TAMANHO VARIÁVEL PARA GRANDES VOLUMES DE TEXTO:

CHARACTER LARGE OBJECT ou CLOB -> o tamanho máximo pode em:

killobytes (K) -> CLOB(10K)

megabytes (M) -> CLOB(10M)

gigabytes (G) -> CLOB(10G)

TIPOS DE DADOS - SEQUÊNCIA DE BITS

TAMANHO FIXO:

BIT(n) -> n = número de bits

TAMANHO VARIÁVEL:

BIT VARYING(n) -> n = número de bits

BINARY LARGE OBJECT ou BLOB -> para grandes valores de binários, como imagens. Segue o mesmo padrão do CLOB

Os valores devem estar sempre entre ‘ ’ precedidos de um B.
Ex: B‘101010001’.

TIPOS DE DADOS - BOOLEANO

REPRESENTAM VALORES TRUE ou FALSE.

PODE-SE TER UM VALOR "UNKNOWN"

assim tem-se uma lógica com três valores.

TIPOS DE DADOS - DATE

DATE = POSSUI 10 POSIÇÕES: DD-MM-YYYY

SENDO: DD = DAY(dia), MM = MONTH(mês), YYYY = YEAR(ano)

TIME = POSSUI OITO POSIÇÕES: HH:MM:SS

SENDO: HH = HOUR(hora), MM = MINUTE(minuto), SS = SECOND(segundo)

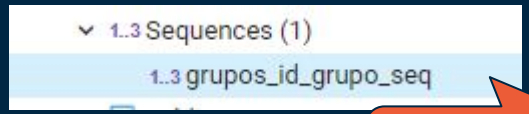
TIPOS DE DADOS - SERIAL -> PostgreSQL??

- Serial não é efetivamente um tipo de dado.
- Ele implementa o mesmo que **auto_incremento** em outras linguagens.
- O tipo do atributo na tabela será integer, porém ele cria uma estrutura (sequência, campo NOT NULL, Unique) para o atributo. Assim, o próprio banco gerencia as inserções e ordem.
- Ele é equivalente ao seguinte:

TIPOS DE DADOS - SERIAL -> PostgreSQL??

```
CREATE TABLE departamentos(
     id_departamento SERIAL,
     nome VARCHAR(60) NOT NULL
);
```

O resultado será esse:



Sequência criada.

Tabela/ Atributos criados

Que é equivalente a isso:

```
CREATE SEQUENCE seq_grupos_id AS integer;
CREATE TABLE departamentos(
     id_departamento NOT NULL DEFAULT NEXTVAL('seq_grupos_id'),
     ...
);
ALTER SEQUENCE seq_grupos_id OWNED BY departamentos.id_departamento
```

TIPOS DE DADOS - OUTROS

TIMESTAMP = Junção de DATE e TIME, mais um mínimo de seis posições para frações decimais de segundos e opcionalmente um qualificador de WITH TIME ZONE.

DDL:
DATA DEFINITION LANGUAGE

CREATE TABLE

- Especificar uma nova relação dando-lhe um nome e especificando seus atributos e restrições iniciais.
- Após a criação, usar o comando ALTER TABLE para modificações.

```
CREATE TABLE FUNCIONARIO
    (Primeiro_nome          VARCHAR(15)         NOT NULL,
    Nome_meio               CHAR,
    Ultimo_nome             VARCHAR(15)         NOT NULL,
    Cpf                     CHAR(11),           NOT NULL,
    Data_nascimento         DATE,
    Endereco                VARCHAR(30),
    Sexo                    CHAR,
    Salario                 DECIMAL(10,2),
    Cpf_supervisor          CHAR(11),
    Numero_departamento     INT                 NOT NULL,
    PRIMARY KEY (Cpf) );
CREATE TABLE DEPARTAMENTO
    (Nome_departamento      VARCHAR(15)         NOT NULL,
    Numero_departamento     INT                 NOT NULL,
    Cpf_gerente             CHAR(11),           NOT NULL,
    Data_inicio_gerente     DATE,
    PRIMARY KEY (Numero_departamento),
    UNIQUE (Nome_departamento),
```

CREATE TABLE

Exemplo criação modelo empresa de antes

```
    FOREIGN KEY (Cpf_gerente) REFERENCES FUNCIONARIO(Cpf) );
CREATE TABLE LOCALIZACOES_DEPARTAMENTO
    (Numero_departamento    INT             NOT NULL,
    Local                   VARCHAR(15)     NOT NULL,
    PRIMARY KEY (Numero_departamento, Local),
    FOREIGN KEY (Numero_departamento) REFERENCES
    DEPARTAMENTO(Numero_departamento) );
CREATE TABLE PROJETO
    (Nome_projeto           VARCHAR(15)     NOT NULL,
    Numero_projeto          INT             NOT NULL,
    Local_projeto           VARCHAR(15),
    Numero_departamento     INT             NOT NULL,
    PRIMARY KEY (Numero_projeto),
    UNIQUE (Nome_projeto),
    FOREIGN KEY (Numero_departamento) REFERENCES DEPARTAMENTO(Numero_
    departamento) );
```

CREATE TABLE

Exemplo criação modelo empresa de antes

```
CREATE TABLE TRABALHA_EM
    (Cpf_funcionario        CHAR(11)        NOT NULL,
    Numero_projeto          INT             NOT NULL,
    Horas                   DECIMAL(3,1)    NOT NULL,
    PRIMARY KEY (Cpf_funcionario, Numero_projeto),
    FOREIGN KEY (Cpf_funcionario) REFERENCES FUNCIONARIO(Cpf),
    FOREIGN KEY (Numero_projeto) REFERENCES PROJETO(Numero_projeto) );
CREATE TABLE DEPENDENTE
    (Cpf_funcionario        CHAR(11),       NOT NULL,
    Nome_dependente         VARCHAR(15)     NOT NULL,
    Sexo                    CHAR,
    Data_nascimento         DATE,
    Parentesco                              VARCHAR(8),
    PRIMARY KEY (Cpf_funcionario, Nome_dependente),
    FOREIGN KEY (Cpf_funcionario) REFERENCES FUNCIONARIO(Cpf) );
```

CONSTRAINTS - RESTRIÇÕES

Restrições são limitações impostas ao modelo, no momento da definição dos atributos e tabelas ou após sua criação, mas antes do uso.

Podem ser:

- Restrição de valores Nulos = **NOT NULL**
- Restrições Relacionais = **PRIMARY KEY e FOREIGN KEY**
- Restrição de valores na tuplas = **CHECK**
- Valor Padrão = **DEFAULT**

RESTRIÇÕES - NOT NULL

Como SQL aceita valores do tipo NULL (vazio), é possível limitar o atributo a não permitir valores assim.

Pode ser feito na criação da tabela ou depois, na alteração.

```
CREATE TABLE DEPARTAMENTO(
    id_departamento INT NOT NULL,
    nome VARCHAR(60)
);
```

```
ALTER TABLE departamentos ALTER COLUMN nome SET NOT NULL;

ALTER TABLE departamentos ALTER COLUMN nome DROP NOT NULL;
```

RESTRIÇÕES - PRIMARY KEY

Define qual será a chave primária da tabela, ou seja, atributo com valor único. Precisa ser NOT NULL. Pode ser indicada de três formas:

```
CREATE TABLE DEPARTAMENTO(
     id_departamento INT UNIQUE NOT NULL,
     nome VARCHAR(60)
);
```

```
CREATE TABLE DEPARTAMENTO(
     id_departamento INT PRIMARY KEY,
     nome VARCHAR(60)
);
```

```
CREATE TABLE DEPARTAMENTO(
     id_departamento INT NOT NULL,
     nome VARCHAR(60),
     PRIMARY KEY (id_departamento)
);
```

RESTRIÇÕES - FOREIGN KEY

Define qual será a chave estrangeira, ou seja, atributo que depende do valor estar presente em outra tabela. Pode ser indicada de duas formas:

```
CREATE TABLE grupo(
    id_grupo INT PRIMARY KEY,
    nome VARCHAR(60),
    id_departamento INT REFERENCES departamento (id_departamento)
);
```

```
CREATE TABLE grupo(
    id_grupo INT PRIMARY KEY,
    nome VARCHAR(60),
    id_departamento INT REFERENCES departamento
);
```

RESTRIÇÕES - FOREIGN KEY

dando um nome para a CONSTRAINT:

```
CREATE TABLE grupo(
    id_grupo  INT PRIMARY KEY,
    nome VARCHAR(60),
    id_departamento INT NOT NUILL,
    FOREING KEY grupos_departamento_fk REFERENCES departamento
(id_departamento)
);
```

RESTRIÇÕES - CHECK

Limitar valores do atributo do domínio, após sua declaração.

```
CREATE TABLE DEPARTAMENTO(
    numero_departamento INT NOT NULL CHECK
        (numero_departamento > 0 AND numero_departamento < 21)
)
```

```
CREATE DOMAIN D_NUM AS INTEGER
CHECK (D_NUM >0 AND D_NUM<21)
```

RESTRIÇÕES - DEFAULT

Atribuir um valor por padrão, caso não seja definido no momento da inserção de um registro.

```
CREATE TABLE grupo(
     id_grupo INT PRIMARY KEY,
     nome VARCHAR(60) DEFAULT 'SEM NOME',
     id_departamento INT REFERENCES departamento (id_departamento)
);
```

```
ALTER TABLE grupos ALTER COLUMN nome SET DEFAULT 'SEM NOME'

ALTER TABLE grupos ALTER COLUMN nome DROP DEFAULT 'SEM NOME'
```

ALTER TABLE

Após a construção de uma tabela, é possível modificá-la. ALTER TABLE tem esse papel.

É possível modificar:

- Adicionar ou Remover columns
- Adicionar ou Remover constraints
- Mudar valores default
- Mudar tipos de dados da coluna
- Renomear Colunas
- Renomear Tabelas

EXEMPLOS ALTER TABLE

```
ALTER TABLE departamentos ALTER COLUMN nome SET NOT NULL;

ALTER TABLE departamentos ALTER COLUMN nome DROP NOT
NULL;

ALTER TABLE grupos ALTER COLUMN nome SET DEFAULT 'SEM
NOME'

ALTER TABLE grupos ALTER COLUMN nome DROP DEFAULT 'SEM
NOME'
```

EXEMPLOS ALTER TABLE

```
ALTER TABLE grupos ADD COLUMN responsavel VARCHAR(30) NOT NULL

ALTER TABLE grupos DROP COLUMN responsavel

>Quando ela é FOREIGN KEY:
ALTER TABLE grupos DROP COLUMN responsavel CASCADE

ALTER TABLE grupos ADD CHECK (nome <> '');

ALTER TABLE grupos ADD CONSTRAINT nome_restrição UNIQUE
(id_departamento);

ALTER TABLE grupos ADD FOREIGN KEY (departamentos_gropos_id)
REFERENCES departamentos;
```

EXEMPLOS ALTER TABLE

```
ALTER TABLE grupos ALTER COLUMN responsavel TYPE VARCHAR(60)

ALTER TABLE grupos RENAME COLUMN responsavel TO nome_responsavel

ALTER TABLE grupo RENAME TO grupos
```

OUTROS COMANDOS

DROP TABLE grupos; **-> Excluir a tabela**

CREATE TABLE IF NOT EXISTS grupos; **-> Se a tabela já existir, apenas retornará uma mensagem avisando e não um erro.**

CONSIDERAÇÕES FINAIS

A linguagem DDL é menos utilizada no dia a dia do que a DML.
No entanto, em algumas tarefas pode ser muito útil.

Enter o que pode ser feito é o principal:
CREATE,
ALTER TABLE,
ADD,
DROP,
ALTER TABLE …. COLUMN
CONSTRAINTS

Ter muito cuidado quando já houver valores em produção na tabela.

***SEMPRE TESTE ANTES PARA CONFIRMAR O COMNADO.**

REFERÊNCIAS

Elmasi E.; navathe, S. B. Sistemas de Banco de dados. São paulo: Pearson. 2018. Disponível em: https://plataforma.bvirtual.com.br/Acervo/Publicacao/168492 (Biblioteca Pearson Unipar - acesso pelo aluno on-line, menu a direita).

POSTGRESQL. Documentação oficial. SQL. https://www.postgresql.org/docs/13/sql.html

POSTGRESQL. Documentação oficial. DDL. https://www.postgresql.org/docs/13/ddl.html

Postgres Tutorial. https://www.postgresqltutorial.com