

Instituto Federal
Campus Goiânia

Bacharelado em Sistemas de Informação

Banco de Dados I



Prof. Dory Gonzaga Rodrigues





Agenda

- Introdução ao SQL
- Tipo de Dados
- Base de Dados
- Criação de Tabelas
- Integridade Referencial

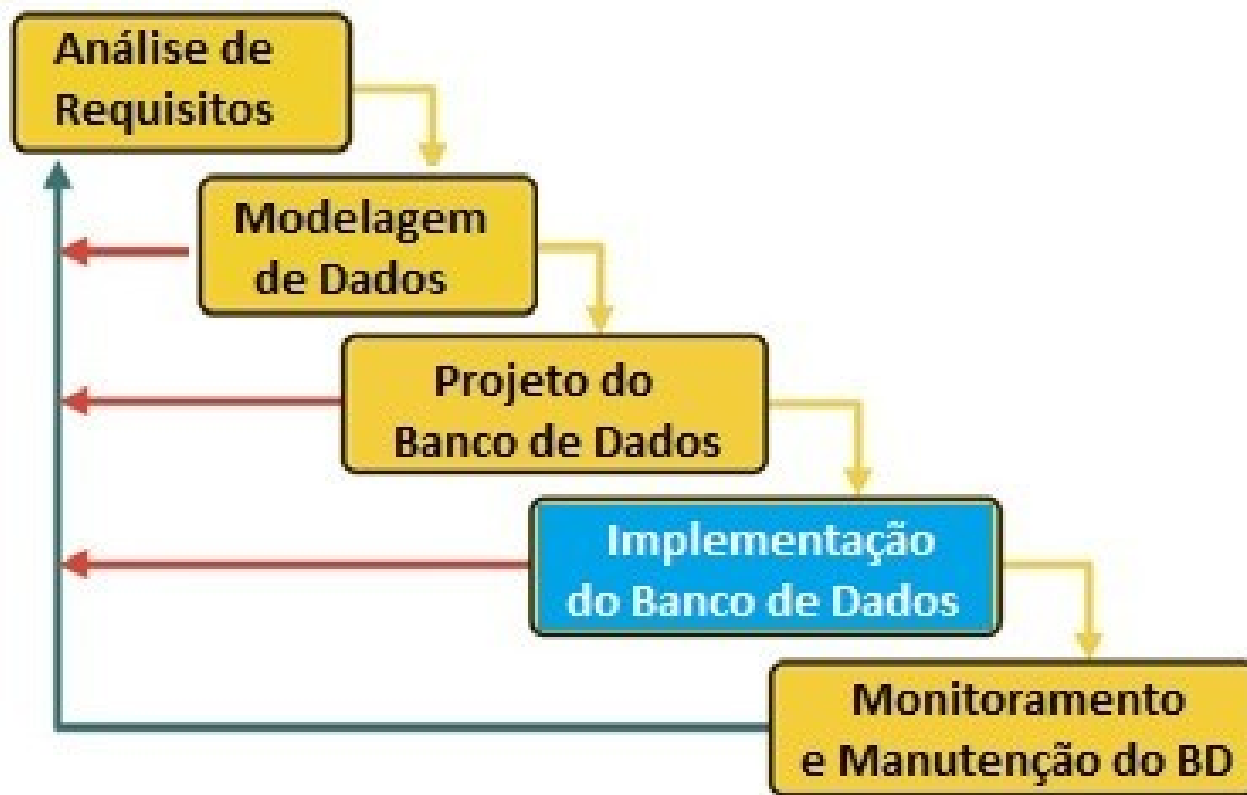




Introdução

INTRODUÇÃO – SQL

Onde estamos no ciclo de desenvolvimento do banco de dados ?





Introdução

INTRODUÇÃO – SQL

- A linguagem SQL (Structured Query Language) é uma linguagem declarativa, ao contrário das linguagens tradicionais, que são do tipo procedimental.
- Ou seja, permite ao usuário expressar aquilo que pretende sem ter que entrar em grandes detalhes sobre a localização física dos componentes, etc.
- A linguagem SQL é constituída por três sublinguagens:
 - DDL (Data Definition Language): Create, Alter, Drop, etc.
 - DML (Data Manipulation Language): Insert, Delete, Update, Select, etc.
 - DCL (Data Control Language): Grant, Revoke, etc.
- **Atenção:** Os comandos SQL utilizados durante o curso terão compatibilidade com o PostgreSQL.
- Para estudar utilize PostgreSQL Documentation disponível na internet ou no computador que possui o banco instalado.





Introdução

Tipo de Dados

SQL – DDL

- Tipos de Dados (**Data Type**)

Tal como acontece com as variáveis em uma linguagem de programação tradicional, também cada coluna de uma tabela deve ser definida com um tipo de dados específico.

No PostgreSQL temos estes tipos classificados da seguinte forma:

- Numeric
- Monetary
- Character
- Date/Time
- Boolean
- Enumerated
- Arrays
- Composite
- Binary Data
- Geometric Types
- Network Address
- Bit String Types
- Text Search
- UUID
- XML
- JSON





Introdução

Tipo de Dados

SQL – DDL

- Tipos de Dados Numéricos (**Data Type: Numeric**)

Inteiros	Espaço	Descrição	Tamanho(Range)
<code>smallint</code>	2 bytes	inteiro de menor tamanho	-32768 até +32767
<code>integer</code> ou <code>int</code>	4 bytes	inteiro de tamanho padrão	-2147483648 até +2147483647
<code>bigint</code>	8 bytes	inteiro de maior tamanho	-9223372036854775808 até +9223372036854775807



Introdução

Tipo de Dados

SQL – DDL

- Tipos de Dados Numéricos (**Data Type: Numeric**)

Ponto Flutuante	Espaço	Descrição	Tamanho(Range)
decimal	variável	com precisão exata	acima de 131.072 dígitos antes e depois do ponto.
numeric	variável	com precisão exata	acima de 131.072 dígitos antes e depois do ponto.
real	4 bytes	precisão variável	precisão de 6 dígitos
double precision ou float	8 bytes	precisão variável	precisão de 15 dígitos





Introdução Tipo de Dados

SQL – DDL

- Tipos de Dados Numéricos (**Data Type: Numeric**)

Autoincrement	Espaço	Descrição	Tamanho(Range)
<code>smallserial</code>	2 bytes	inteiro de menor tamanho	1 to 32767
<code>serial</code>	4 bytes	inteiro de tamanho padrão	1 to 2147483647
<code>bigserial</code>	8 bytes	inteiro de maior tamanho	1 to 9223372036854775807

Atenção:

Estes tipos na verdade são **tipo inteiro** e uma **sequence como valor default**





Introdução

Tipo de Dados

SQL – DDL

- Tipos de Dados Numéricos (**Data Type: Monetary**)

Monetary	Espaço	Descrição	Tamanho(Range)
money	8 bytes	valor monetário	-92233720368547758. 08 to +92233720368547758. 07





Introdução > Tipo de Dados

SQL – DDL

- Tipos de Dados Numéricos (**Data Type: Character**)

Character	Descrição
character varying(n)	tamanho variável de acordo com o limite
varchar(n)	tamanho variável de acordo com o limite
char ou character	tamanho fixo e igual a 1
char(n) ou character(n)	tamanho variável de acordo com o limite (espaço em branco no final são ignorados)
text	variável sem limite de tamanho



Introdução

Tipo de Dados

SQL – DDL

- Tipos de Dados Numéricos (**Data Type: Data / Time**)

Data / Time	Espaço	Descrição	Tamanho
timestamp timestampz	8 bytes	data e hora <u>sem</u> fuso horário data e hora <u>com</u> fuso horário	4713 BC to 294276 AD
Date	4 bytes	date	4713 BC to 5874897 AD
Time	8 bytes	hora	00:00:00 to 24:00:00
	12 bytes	hora com fuso horário	00:00:00+1459 To 24:00:00-1459
Interval	16 bytes	intervalo de tempo	-178000000 years 178000000 years





Introdução Tipo de Dados

SQL – DDL

- Tipos de Dados Numéricos (**Data Type: Boolean**)

Boolean	Espaço	Descrição
boolean	1 byte	valor Verdadeiro ou Falso

Valores possíveis para TRUE

TRUE
't'
'true'
'y'
'yes'
'on'
'1'

Valores possíveis para FALSE

FALSE
'f'
'false'
'n'
'no'
'off'
'0'





Introdução

Tipo de Dados

SQL – DDL

- Tipos de Dados Numéricos (**Data Type: Enumerated**)

Enumerated

Descrição

enum

Enumerado são tipos de dados de composição estática, conjunto ordenado de valores. Eles são equivalentes para os tipos de enum suportadas em grande parte das linguagens de programação .

Um exemplo de um tipo de enumeração podem ser os dias da semana, ou um conjunto de valores de status para uma peça de dados

Exemplo:

```
CREATE TYPE sexo AS ENUM ('Masculino', 'Feminino');
```





Introdução Tipo de Dados

SQL – DDL

Data type	Access	SQLServer	Oracle	MySQL	PostgreSQL
<i>boolean</i>	Yes/No	Bit	Byte	N/A	Boolean
<i>integer</i>	Number (integer)	Int	Number	Int Integer	Int Integer
<i>float</i>	Number (single)	Float Real	Number	Float	Numeric
<i>currency</i>	Currency	Money	N/A	N/A	Money
<i>string (fixed)</i>	N/A	Char	Char	Char	Char
<i>string (variable)</i>	Text (<256) Memo (65k+)	Varchar	Varchar Varchar2	Varchar	Varchar
<i>binary object</i>	OLE Object Memo	Binary (fixed up to 8K) Varbinary (<8K) Image (<2GB)	Long Raw	Blob Text	Binary Varbinary





Introdução

Tipo de Dados

SQL - DDL

CREATE

SQL – DDL

- Comando CREATE DATABASE:

O comando **CREATE DATABASE** é responsável pela criação do banco de dados. A partir deste momento poderemos criar as tabelas, armazenar e manipular os dados no sistema SGBD.

A sintaxe do comando é

CREATE DATABASE <nome do banco de dados>;

Exemplos:

CREATE DATABASE ifgoias;





Introdução

Tipo de Dados

SQL - DDL

DROP

SQL – DDL

- Comando DROP DATABASE:

O comando **DROP DATABASE** permite remover um determinado Banco de Dados, apagando todas as tabelas e estruturas associadas e, conseqüentemente, todos os dados existentes nas tabelas.

A sintaxe do comando é

DROP DATABASE [IF EXISTS] <nome do banco de dados>;

Exemplos:

DROP DATABASE ifgoias;

DROP DATABASE IF EXISTS ifgoias;





Introdução

Tipo de Dados

SQL - DDL

CREATE

SQL – DDL

- Comando CREATE TABLE:

O comando **CREATE TABLE** é responsável pela criação das Tabelas permanentes, sendo considerado o principal comando da linguagem DDL.

A sintaxe básica do comando é

```
CREATE TABLE [ IF NOT EXISTS ] <nome da tabela> (  
    <nome_da_coluna>    <tipo_de_dado> <constraint>,  
    <nome_da_coluna>    <tipo_de_dado> <constraint>,  
    ...                  ...  
    <nome_da_coluna>    <tipo_de_dado> <constraint>,  
    <constraint>  
);
```





Introdução

Tipo de Dados

SQL - DDL

CREATE

SQL – DDL

- Comando CREATE TABLE:

```
CREATE TYPE genero AS ENUM ('Masculino', 'Feminino');
```

```
CREATE TABLE IF NOT EXISTS Alunos (
```

id	serial,
cpf	integer,
nome	varchar,
primeiro_nome	varchar(20),
sexo	genero,
altura	numeric,
peso	float,
valor_matricula	money
data_matricula	date,
hora_matricula	time,
matricula_cancelada	boolean

```
);
```





Introdução

Tipo de Dados

SQL - DDL

DROP

SQL – DDL

- Comando DROP TABLE:

O comando **DROP TABLE** é responsável pela destruição das Tabelas.

A sintaxe básica do comando é:

```
DROP TABLE [ IF EXISTS ] [ CASCADE ou RESTRICT ] <nome da tabela> ;
```

IF EXISTS

O comando é executado apenas quando a tabela existir

CASCADE

Automaticamente apaga os objetos que dependentes desta tabela (i.e views).

RESTRICT

Não executa o comando se existir objetos dependentes da tabela.

Opção padrão (default).





Introdução

Tipo de Dados

SQL - DDL

DROP

SQL – DDL

- Comando DROP TABLE:

O comando **DROP TABLE** é responsável pela destruição da Tabela.

Exemplo:

```
DROP TABLE IF EXISTS Alunos CASCADE;
```



Atenção: Caso existam objetos Views e Constraints associadas à tabela, estes também serão apagados.





Introdução

Tipo de Dados

SQL - DDL

DELETE

SQL – DDL

- Comando DELETE:

DELETE é responsável por apagar linhas que satisfazem a cláusula **[WHERE]** da tabela especificada. Se a cláusula **[WHERE]** estiver ausente, o efeito é excluir todas as linhas na tabela. O resultado é uma tabela válida, porém vazia.

TRUNCATE é uma extensão do PostgreSQL que fornece um mecanismo mais rápido para remover todas as linhas de uma tabela .

Exemplos:

DELETE FROM Alunos;

DELETE FROM Alunos **WHERE** nome = 'Dory';

TRUNCATE FROM Alunos;

TRUNCATE FROM Alunos, Professores;

TRUNCATE FROM Alunos **CASCADE**;

TRUNCATE FROM Alunos **RESTART IDENTITY**;





Introdução

Tipo de Dados

SQL - DDL

CONSTRAINTS

SQL – DDL

- Integridade Referencial:

A integridade referencial é conhecida como **CONSTRAINTS**

- São regras agregadas a colunas ou tabelas. Estas regras definem questões como: quais valores uma ou mais colunas devem obedecer.

- Variam muito de um banco de dados para outro. Pode ser que algumas delas não esteja presente no banco de dados utilizado.

Exemplos:

Sexo	só poderá conter os valores 'F' e 'M'
Idade	não poderá conter valores negativos





Introdução

Tipo de Dados

SQL - DDL

CONSTRAINTS

SQL – DDL

- Integridade Referencial:

A utilização de **CONSTRAINTS** é a única garantia que temos de que os dados existentes nas colunas estão de acordo com as regras especificadas no projeto do Banco de Dados.

Existem alguns tipos de restrições que se aplica à colunas:

- NOT NULL ou NULL
- UNIQUE
- PRIMARY KEY
- DEFAULT
- AUTO_INCREMENT
- CHECK ()
- REFERENCES





Introdução

Tipo de Dados

SQL - DDL

CONSTRAINTS


NOT NULL

SQL – DDL

- Integridade Referencial (CONSTRAINTS): **NOT NULL**

A Constraint **NOT NULL** é restrição de uso mais comum, pois impede a inserção de um registro com valor nulo na coluna em que foi especificada.

```
CREATE TABLE Alunos (  
    noMatr          INTEGER NOT NULL ,  
    nomeAluno       VARCHAR(100) NOT NULL ,  
    idadeAluno      INTEGER      ,  
    cpfAluno        VARCHAR(20)  
);
```



As colunas que possuem a **restrição NULL**, permitem a inserção com valores nulos. O padrão é NULL;





Introdução

Tipo de Dados

SQL - DDL

CONSTRAINTS

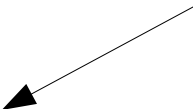
UNIQUE

SQL – DDL

- Integridade Referencial (CONSTRAINTS): **UNIQUE**

A Constraint **UNIQUE** permite identificar que os valores dessa coluna não podem se repetir, podendo ser NULL.

```
CREATE TABLE Alunos (  
    noMatr          INTEGER NOT NULL UNIQUE ,  
    nomeAluno       VARCHAR(100) NOT NULL ,  
    idadeAluno      INTEGER      ,  
    cpfAluno        VARCHAR(20)  
);
```



É permitida várias ocorrências da cláusula **UNIQUE**.





Introdução

Tipo de Dados

SQL - DDL

CONSTRAINTS

PRIMARY KEY

SQL – DDL

- Integridade Referencial (CONSTRAINTS): **PRIMARY KEY**

A Constraint **PRIMARY KEY** incorpora às cláusulas NOT NULL + UNIQUE , isto é, o comando define que o conteúdo da coluna não pode ser nulo e não admite repetições. Além disso, a cláusula define a chave primária da tabela.

```
CREATE TABLE Alunos (  
    noMatr          INTEGER PRIMARY KEY ,  
    nomeAluno       VARCHAR(100) ,  
    idadeAluno      INTEGER ,  
    cpfAluno        VARCHAR(20)  
);
```



É permitida apenas uma cláusula **PRIMARY KEY** na tabela.





Introdução

Tipo de Dados

SQL - DDL

CONSTRAINTS

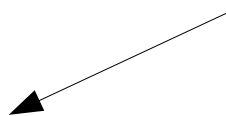
PRIMARY KEY

SQL – DDL

- Integridade Referencial (CONSTRAINTS): PRIMARY KEY

Outra forma de incluir a **PRIMARY KEY** é depois da declaração das colunas. Neste caso, podemos criar chaves primárias compostas por mais de uma coluna.

```
CREATE TABLE Alunos (  
    noMatr          INTEGER      ,  
    nomeAluno       VARCHAR(100) ,  
    idadeAluno      INTEGER      ,  
    cpfAluno        VARCHAR(20) ,  
    PRIMARY KEY (noMatr, cpfAluno)  
);
```



Introdução

Tipo de Dados

SQL - DDL

CONSTRAINTS

DEFAULT

SQL – DDL

- Integridade Referencial (CONSTRAINTS): **DEFAULT**

A Constraint **DEFAULT** serve para atribuir um valor-padrão a uma coluna da tabela, sempre que for incluída uma nova linha na tabela e o valor da coluna não for especificado no comando.

```
CREATE TABLE Alunos (  
    noMatr          INTEGER    PRIMARY KEY,  
    nomeAluno       VARCHAR(100) NOT NULL,  
    idadeAluno      INTEGER    DEFAULT 0,  
    cpfAluno        VARCHAR(20) UNIQUE  
);
```





Introdução

Tipo de Dados

SQL - DDL

CONSTRAINTS

AUTO_INCREMENT

SQL – DDL

- Integridade Referencial (CONSTRAINTS): **AUTO_INCREMENT**

A Constraint **AUTO_INCREMENT** serve para atribuir um valor automaticamente a uma coluna da tabela, sempre que for incluída uma nova linha na tabela o valor será incrementado (+1) em relação ao última linha inserida na tabela.

Existe apenas no MySQL.

No PostgreSQL e no ORACLE ?

Utilizamos o tipo **SERIAL** (cria uma **SEQUENCE** e a inclui na constraint **DEFAULT**).

```
CREATE TABLE Alunos (  
    noMatr          SERIAL ,  
    nomeAluno       VARCHAR(100) ,  
    idadeAluno      INTEGER ,  
    cpfAluno        VARCHAR(20)  
);
```





Introdução

Tipo de Dados

SQL - DDL

CONSTRAINTS

CHECK

SQL – DDL

- Integridade Referencial (CONSTRAINTS): **CHECK ()**

A Constraint **CHECK ()** permite realizar a validação dos dados inseridos na coluna, através da especificação de uma condição. São admitidos apenas os dados cujo resultado da avaliação da condição seja verdadeiro.

```
CREATE TABLE Professor (  
    id            INTEGER ,  
    nome          VARCHAR(100) ,  
    idade         INTEGER CHECK ( idade >= 0 ) ,  
    cpf           VARCHAR(20) ,  
    salario       FLOAT CHECK ( salario >= 780 AND salario <=30000 )  
);
```





Introdução

Tipo de Dados

SQL - DDL

CONSTRAINTS

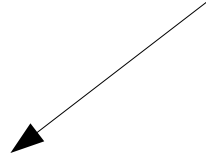
REFERENCE

SQL – DDL

- Integridade Referencial (CONSTRAINTS): REFERENCE / FOREIGN KEY

A Constraint **REFERENCES** permite fazer a atribuição e validação das chaves estrangeiras na própria coluna.

```
CREATE TABLE Alunos (  
    noMatr          INTEGER PRIMARY KEY ,  
    nomeAluno       VARCHAR(100) UNIQUE ,  
    idadeAluno      INTEGER ,  
    cpfAluno        VARCHAR(20) ,  
    noCidade        INTEGER REFERENCES Cidade(noCidade)  
);
```



Isto é, os valores inseridos na coluna **noCidade** será obrigatoriamente um valor existente na coluna **noCidade** da tabela **Cidade**.





Introdução

Tipo de Dados

SQL - DDL

CONSTRAINTS

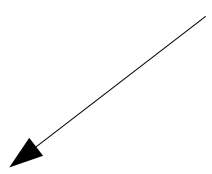
FOREIGN KEY

SQL – DDL

- Integridade Referencial (CONSTRAINTS): REFERENCE / FOREIGN KEY

Outra forma de incluir a **REFERENCES** é depois da declaração das colunas. Neste caso, podemos criar chaves estrangeiras compostas por mais de uma coluna.

```
CREATE TABLE Alunos (  
    noMatr          INTEGER PRIMARY KEY ,  
    nomeAluno       VARCHAR(100) UNIQUE ,  
    idadeAluno      INTEGER ,  
    cpfAluno        VARCHAR(20) ,  
    noCidade        INTEGER ,  
    FOREIGN KEY (noCidade) REFERENCES Cidade (noCidade)  
);
```





Introdução

Tipo de Dados

SQL - DDL

CONSTRAINTS

FOREIGN KEY

SQL – DDL

- Integridade Referencial (CONSTRAINTS): **REFERENCE / FOREIGN KEY**

Opções: [**NO ACTION** ou **RESTRICT** ou **CASCADE**]

RESTRICT não permite a exclusão da linha referenciada (contém a chave estrangeira).

CASCADE quando a linha referencia (contém a chave estrangeira) for excluída, a linha da tabela a que detém a chave primária será apagada automaticamente.

NO ACTION permite que seja excluída a linha referenciada (contém a chave estrangeira) e a linha da tabela que detém a chave primária não sofrerá qualquer alteração.

