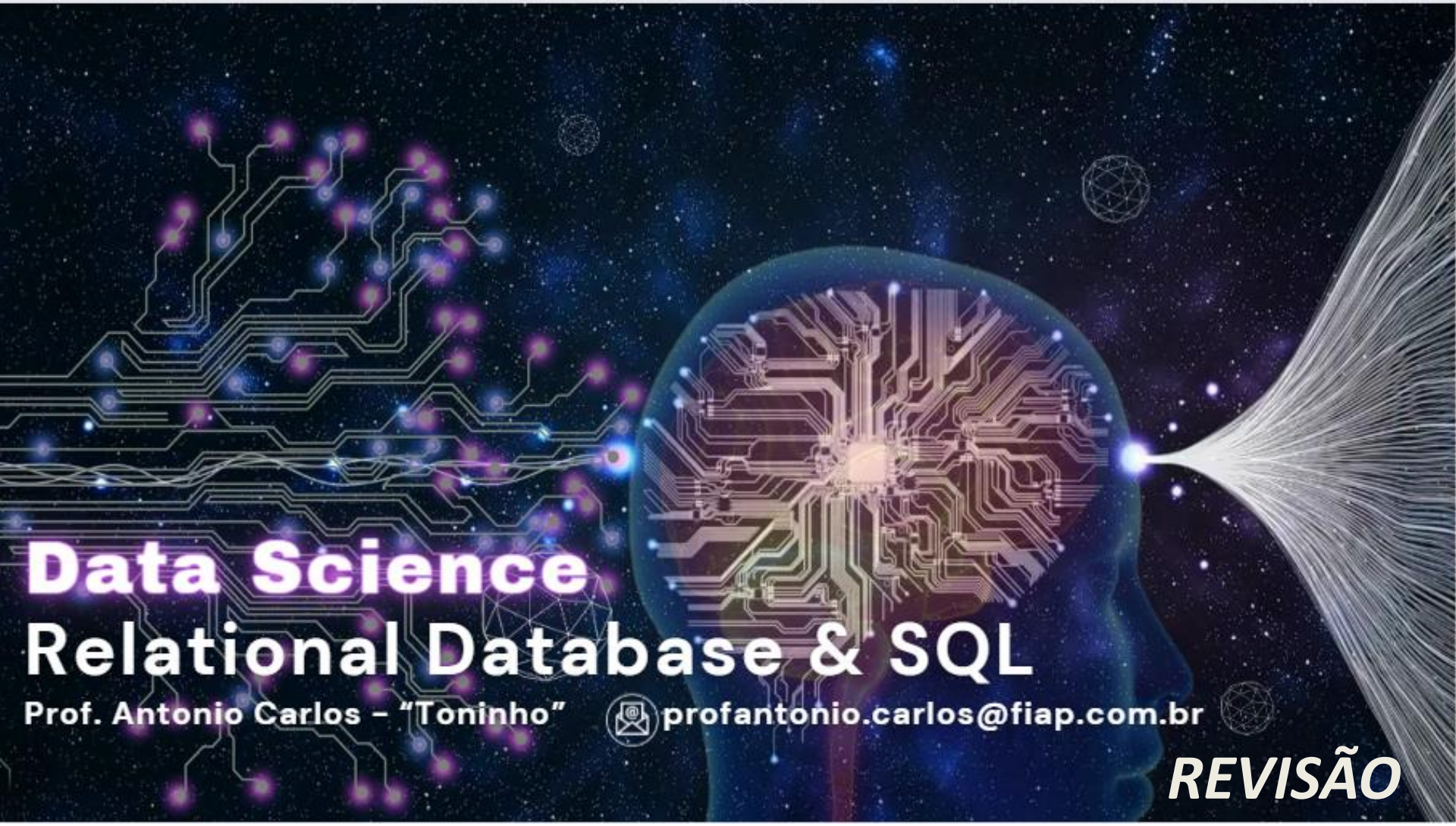
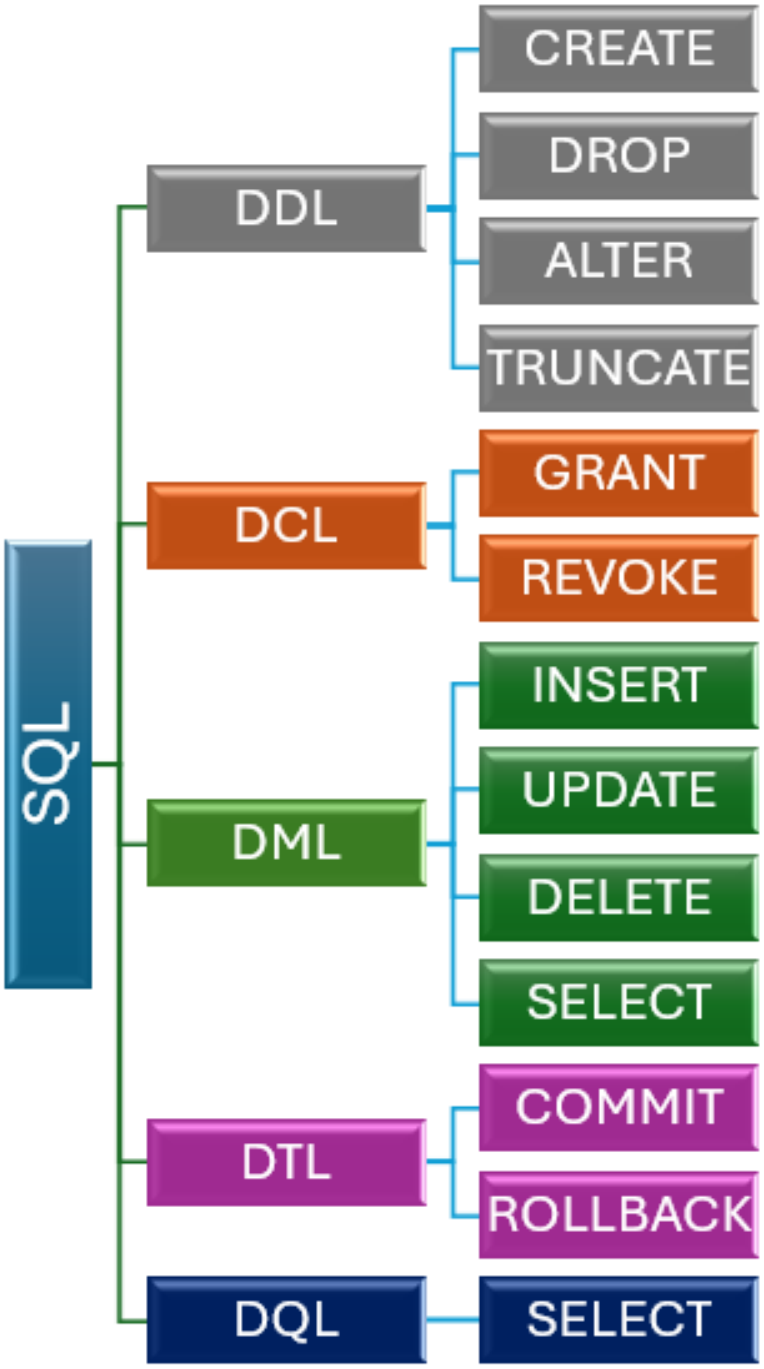
FIAP GRADUAÇÃO


Data Science
Relational Database & SQL
Prof. Antonio Carlos – "Toninho"
profantonio.carlos@fiap.com.br
REVISÃO

Principais Comandos SQL



--> CONEXÃO AO BANCO DE DADOS ORACLE FIAP

Nome BD: BD_XXXXXX

User Name: RMxxxxxxx

Password: Data Aniversário

Hostname: oracle.fiap.com.br

Port: 1521

SID: orcl

DDL – Data Definition Language

- Podemos criar uma tabela qualquer dentro de um banco de dados. A sintaxe básica para criarmos é:

```
CREATE TABLE nome_tabela (
nome_campo_1  tipo_1,
nome_campo_2  tipo_2,
...
nome_campo_n tipo_n,
);
```

DDL – Data Definition Language

- **CREATE TABLE** é o comando usado para criação da tabela e deve ser seguida pelo nome que daremos à tabela. Dentro do comando, devemos definir os nomes dos campos de acordo com a conveniência do banco de dados, e determinar o ***tipo de dado*** que poderá ser incluído neste campo.

- **PRIMARY KEY** define a chave primária da tabela, isto é, o campo que serve como chave da tabela e que não pode ser repetido.

- **NOT NULL** é usado quando desejamos que um campo seja de preenchimento obrigatório.

Ex:

```
CREATE TABLE nome_tabela (
nome_campo_1 tipo_1 NOT NULL,
nome_campo_2 tipo_2,
...
nome_campo_n tipo_n,
PRIMARY KEY(campo_x,...)
);
```

Tipos de Dados

***Tipos de dados* definem os tipos de informação que podem ser inseridos em um campo. Abaixo os tipos suportados por um banco de dados Oracle:**

NVARCHAR(N) : Permite a inclusão de dados alfanuméricos com tamanho pré-definido. O número de caracteres é definido entre os parênteses.Limite de 4000 bytes.

NUMBER(E,P) ou (N,D) : Permite a inclusão de números positivos ou negativos de ponto flutuante. "**E**" ou "**N**" corresponde ao número máximo de Dígitos, sem considerar o ponto decimal e "**P**" ou "**D**" é o número máximo de casas decimais Limite 38 dígitos

CHAR(N) : Permite a inclusão de caracteres. "N" corresponde ao tamanho fixo do campo em bytes. Limite =2000 bytes. Se um dado ocupar menos que o definido, será completado com espaços em branco à direita

DATE : Colunas do tipo DATE ocupam sempre 7 bytes internamente. A faixa de datas no Oracle 9i vai de 01-janeiro-4712 a.C até 31-dezembro-9999 d.C. Além da data(dia, mês e ano), o tipo DATE também contém o horário(hora, minuto e segundo). Para precisão de frações de segundo, considerar o tipo de dado TIMESTAMP

Tipos de Dados

EX:

Como exemplo do uso do comando **CREATE TABLE**, imaginemos a necessidade de uma tabela que deva possuir os dados dos clientes de uma loja.

```
CREATE TABLE Cliente     (
                         Codigo          INT NOT NULL,
                         Nome            VARCHAR (60) NOT NULL,
                         Data_Nasc       DATE,
                         Telefone        CHAR (8),
                         PRIMARY KEY     (Codigo)
                         );
```

Tipos de Dados

Neste comando, criamos uma tabela chamada Cliente. Esta tabela contém quatro campos:

- O primeiro campo é o Código do cliente. Este campo será utilizado como chave primária de forma que não poderá se repetir nunca. Desta forma o campo deve ser sempre preenchido (NOT NULL), é numérico do tipo inteiro (INT).

- O campo Nome é do tipo VARCHAR (60), ou seja aceita dados alfanuméricos com até 60 caracteres. No entanto se um nome for inserido com menos de 60 caracteres, o número de bytes consumidos pelo campo será de acordo com o nome inserido.

- O campo de Data_Nasc é do tipo DATE, ou seja, uma data, que no entanto não de preenchimento obrigatório (por isto não foi declarado o NOT NULL).

- O campo Telefone foi determinado como sendo alfanumérico com oito caracteres definidos, e mesmo que sejam utilizados menos caracteres, o número de bytes consumidos serão sempre os mesmos independente dos dados. Isto é útil para dados alfanuméricos que não variam de tamanho, como o caso de UF no Brasil, cuja abreviação sempre são de dois caracteres.

- A instrução PRIMARY KEY define qual dos campos será a chave primária e não pode ser repetido, sendo o diferenciador entre os diversos clientes que sejam inseridos nesta tabela.

- **REGRAS DE INTEGRIDADE**

  Ao criarmos uma tabela dentro de um banco de dados devemos ter em mente as Regras de Integridade, que garantam a consistência, integridade e não redundância dos dados. Entre estas regras podemos englobar as chaves primárias, checagem e chave estrangeira.

PRIMARY KEY

- **CHAVE PRIMÁRIA**

  No exemplo vimos a seguinte declaração na criação da tabela:

  PRIMARY KEY (campo_x,...);

  Esta declaração diz que os campos inseridos entre os parênteses formam a chave primária da tabela. As chaves primárias funcionam como os campos que diferenciam os dados uns dos outros, e que não podem ser repetidos de nenhuma forma. Por exemplo, em nossa tabela Cliente, o código do Cliente funciona como a chave-primária, ou seja, os clientes podem até ter o mesmo nome, endereço ou telefone, mas terão códigos diferentes uns dos outros. Se dois códigos iguais forem inseridos o SGBD retornará erro.

FOREIGN KEY

- **CHAVE ESTRANGEIRA**

  A chave estrangeira é uma cláusula que deve ser incluída quando possuímos mais de duas tabelas em um banco de dados. Através da chave estrangeira estabelecemos as relações entre duas ou mais tabelas. A chave estrangeira desta forma referencia o campo que é chave primária de outra tabela.

  FOREIGN KEY (Campo1, Campo2, Campo3 ...) REFERENCES Nome_Tabela2 (Nome_Chave);

CHECK

- **Checagem**

Podemos inserir em uma tabela depois do campo chave primária e antes do último parêntese a cláusula:

CHECK Nome _Campo IN (valor1 , valor2, valor n);

Esta cláusula força a um campo a aceitar apenas os valores especificados entre os parênteses. Isto pode ser útil para definir, por exemplo, campos como sexo. Desta forma forçamos as opções através de:

CHECK Sexo IN ('M','F');

Onde o campo Sexo só podem assumir a forma M (Masculino) ou F (Feminino).

UNIQUE , NOT NULL

- **Uníco**

  Está Contraint determina que uma coluna não poderá ter 2 linhas com o mesmo valor.

- **Não Nulo**

  Determina que a coluna tem preenchimento obrigatório

  Nome_Campo VARCHAR2(40) NOT NULL UNIQUE

Verificando a Estrutura

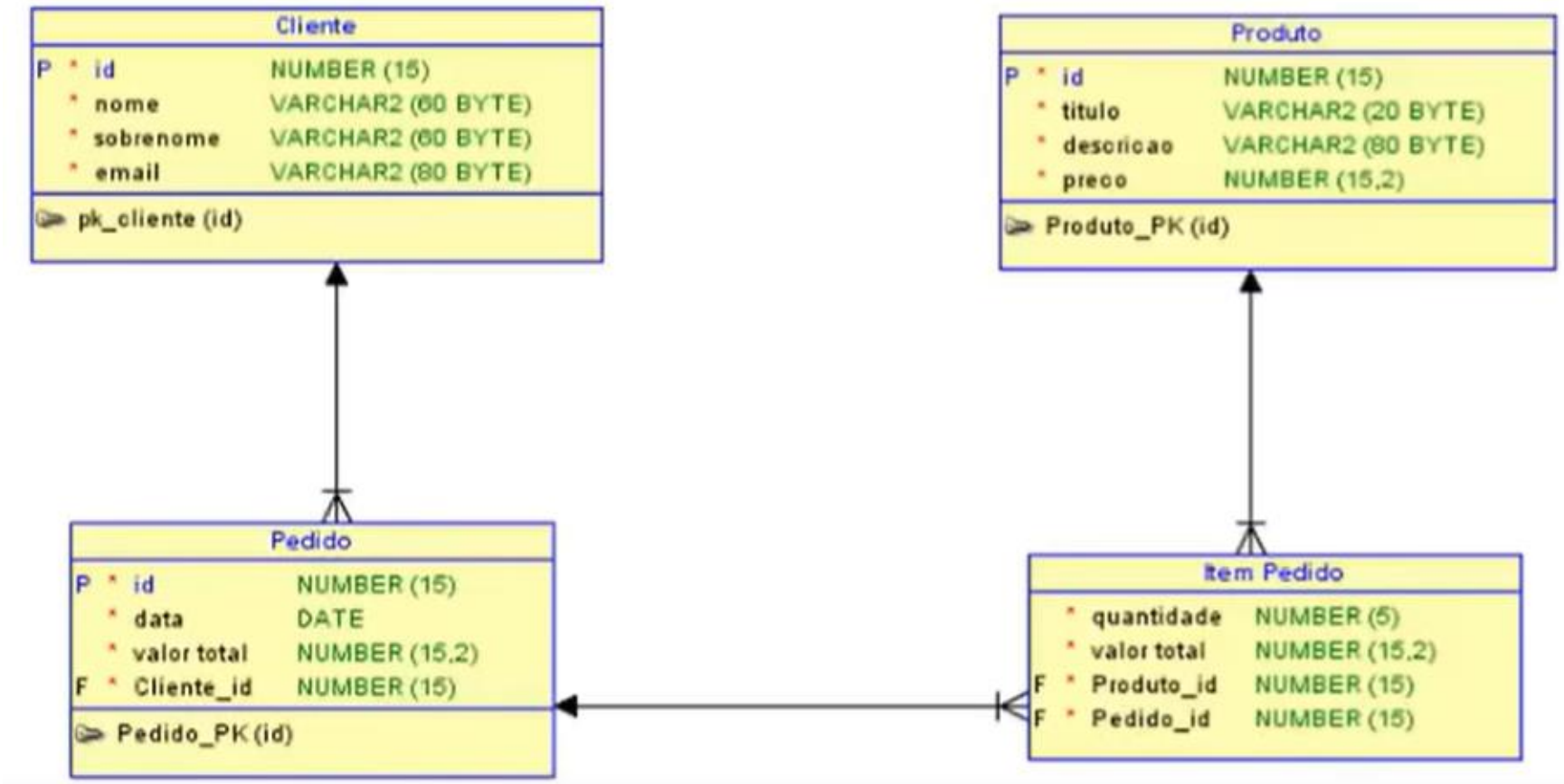
- **DESCRIBE**

  Após criarmos uma tabela, elas podem ser manipuladas de diversas formas, inserindo, alterando ou excluindo dados. Podemos visualizar a estrutura e campos de uma tabela criada utilizando o comando DESCRIBE ou DESC:

  DESCRIBE Nome_Tabela;

  Ou:

  DESC Nome_Tabela;


Cliente
P * id NUMBER (15)
* nome VARCHAR2 (60 BYTE)
* sobrenome VARCHAR2 (60 BYTE)
* email VARCHAR2 (80 BYTE)
pk_cliente (id)
Produto
P * id NUMBER (15)
* titulo VARCHAR2 (20 BYTE)
* descricao VARCHAR2 (80 BYTE)
* preco NUMBER (15,2)
Produto_PK (id)
Pedido
P * id NUMBER (15)
* data DATE
* valor total NUMBER (15,2)
F * Cliente_id NUMBER (15)
Pedido_PK (id)
Item Pedido
* quantidade NUMBER (5)
* valor total NUMBER (15,2)
F * Produto_id NUMBER (15)
F * Pedido_id NUMBER (15)