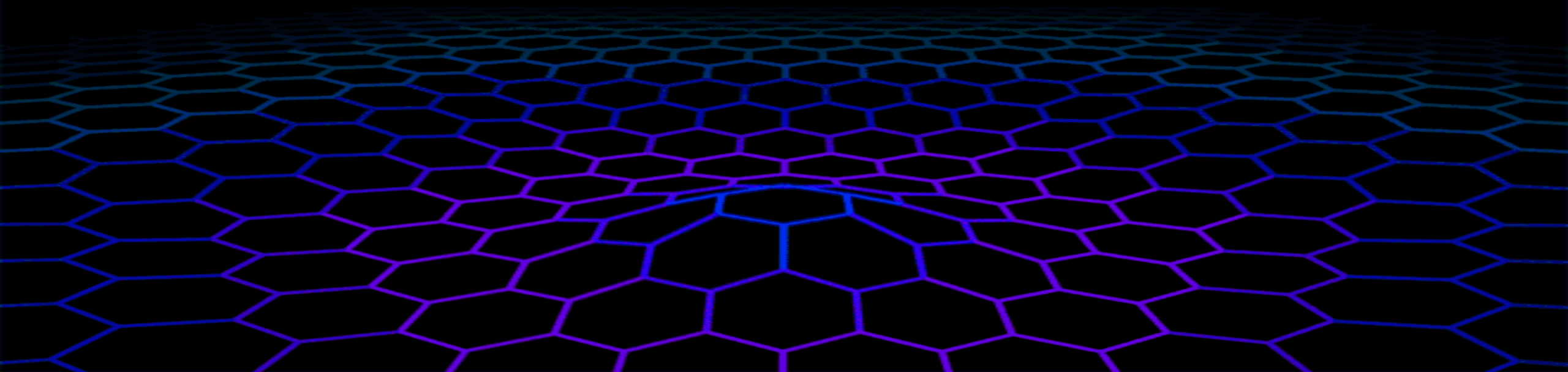
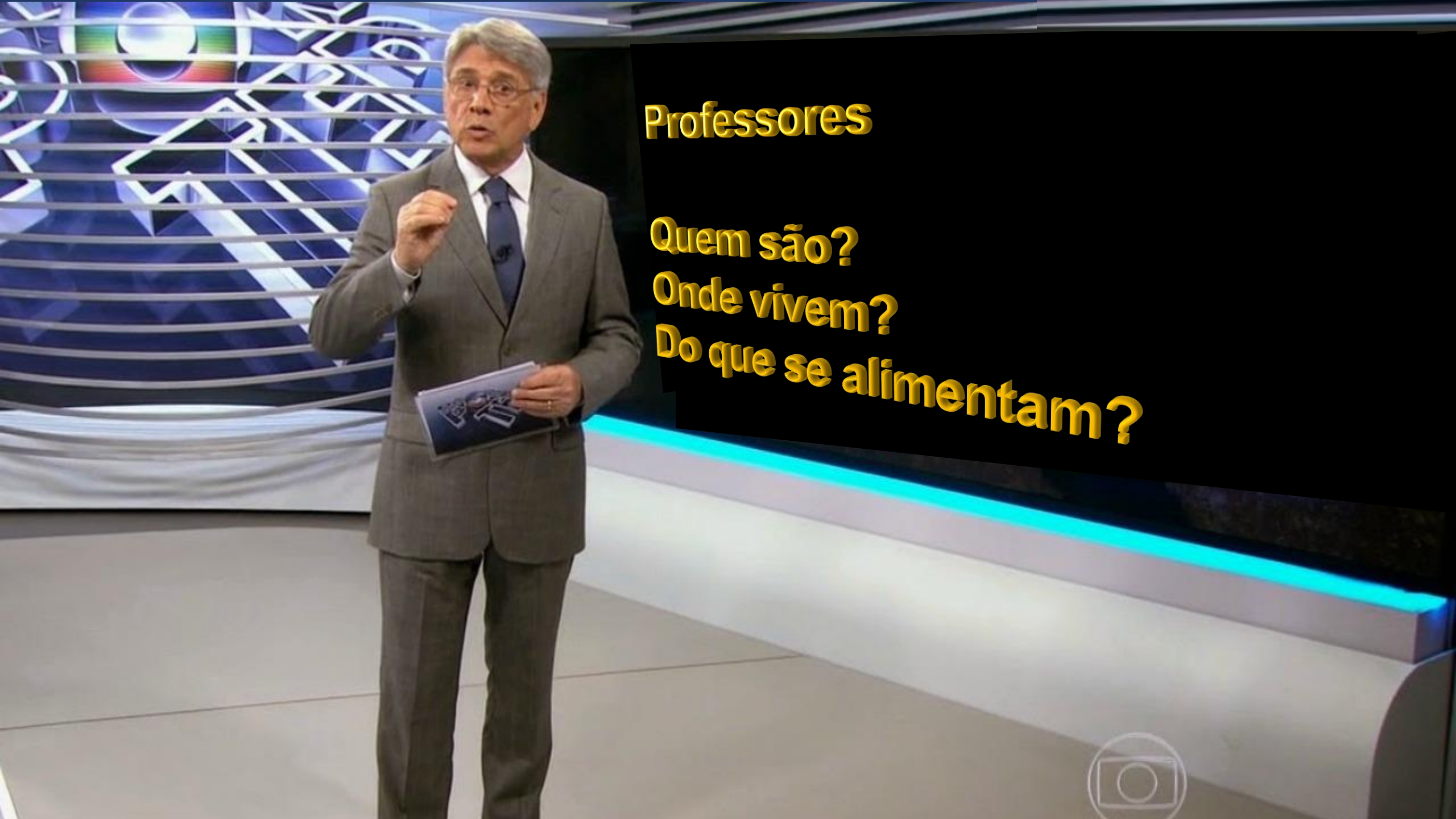
Faculdades senac

Banco de Dados II




BIENVENIDOS
OTRA VEZ.


Professores
Quem são?
Onde vivem?
Do que se alimentam?

Quem é Gladimir?

- 54 anos mas com corpinho de 53
- Mais de 34 anos de experiência profissional
- Mais de 25 anos lecionando em ensino superior
- Mestre em Educação e Tecnologia pelo IFSul
- Leciona em todos os cursos da Faculdade Senac
- Prefere Internet das Coisas do que as Coisas da Internet.
- Matou mais de 1.000 no Battlefield 2 (todos na faca)
- É um cara sério, mas faz cosplay de Stormtrooper

QUE A FORÇA ESTEJA COM VOCÊS!

gladimircc

BATTLE.NET

Origin

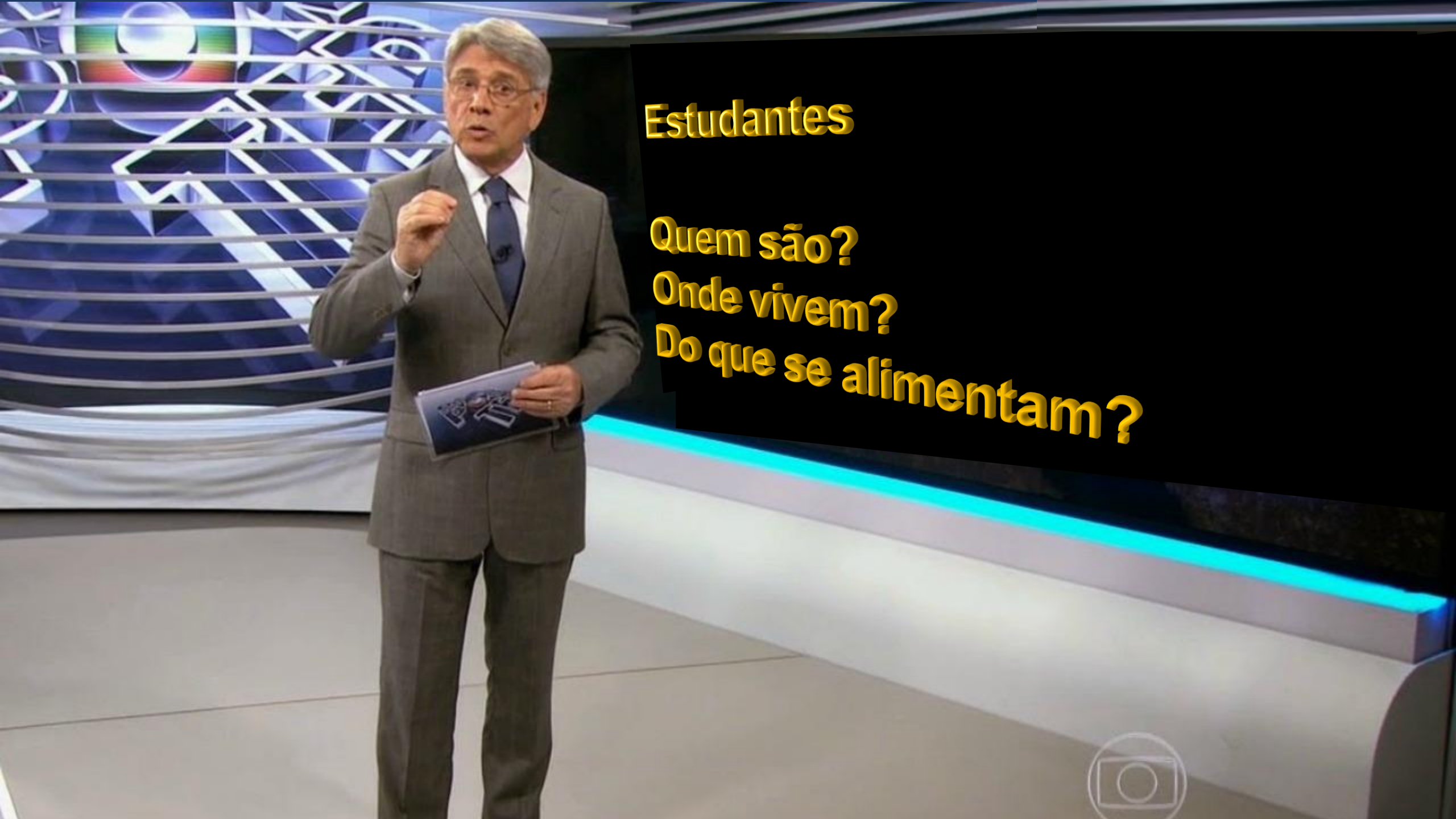
STEAM

XBOX LIVE

EPIC GAMES

UBISOFT

gladimir@gmail.com


Estudantes
Quem são?
Onde vivem?
Do que se alimentam?

Banco de Dados II

Plano de Ensino

Caracterização da Unidade Curricular

Programação em banco de dados objeto relacional utilizando visões, gatilhos, funções e procedimentos.

Conceitos de segurança e permissões em banco de dados SQL.

Utilização de banco de dados não relacional (NoSQL).

Competência Essencial

Identificar problemas e propor soluções para aplicações utilizando diferentes arquiteturas de banco de dados, incluindo técnicas de análise de dados.

Elementos de Competência - Competências Relacionadas

Analisar situações problema e apresentar proposta de solução adequada;

Avaliar os diferentes níveis de isolamento entre transações e aplicar aquele indicado para atender a determinada situação problema;

Elementos de Competência - Competências Relacionadas

Desenvolver visões, funções, procedimentos e gatilhos em banco de dados empregando corretamente;

Linguagem procedural na resolução de problemas;

Classificar os diferentes papéis envolvidos no uso de bancos de dados relacionais;

Elementos de Competência - Competências Relacionadas

Criação e gerenciamento de usuários, papéis e permissões usando linguagem DCL (Data Control Language).

Compreender os conceitos de banco de dados não relacional;

Instalar e utilizar bancos de dados NoSQL na resolução de problemas;

Elementos de Competência - Competências Relacionadas

Expressar-se com clareza, polidez, ética, profissionalismo e objetividade.

Bases Tecnológicas

- Banco de dados objeto Relacional
- Banco de dados não relacional (NoSQL)
- Linguagem procedural para bancos de dados
- Processamento e otimização de consultas
- Processamento de transações.

Pré-requisitos

- Banco de Dados I

Bibliografia Básica

GUEDES, Gilleanes T. A. **UML 2 - Uma Abordagem Prática**. Novatec. Edição 3. 2018.

NERY, Felipe e MACHADO, Rodrigues. **Big Data - O Futuro dos Dados e Aplicações**. Érica. Edição 1. 2018.

HOWS, D.; MEMBREY, P. e PLUGGE, E. **Introdução ao MongoDB**. Novatec. 2015.

Bibliografia Complementar

NIELD, Thomas. **Introdução à Linguagem SQL**. Novatec. Edição 1. 2016.
BEIGHLEY, Lynn. Use a Cabeça – SQL. Alta Books. Edição 1. 2008.

CARDOSO, Virgínia. **Sistema de banco de dados: uma abordagem introdutória e aplicada**. Saraiva. 2012.

MANNINO, Michael V. **Projeto, desenvolvimento de aplicações e administração de banco de dados**. Edição 3. AMGH. 2014.

RAMARKRISHNAN, Raghu. **Sistemas de gerenciamento de banco de dados**. Edição 3. AMGH. 2011.

Datas

Datas para lembrar

BD2	
10/03 (sex)	Aula01
17/03 (sex)	Aula02
24/03 (sex)	Aula03
31/03 (sex)	Aula04
07/04 (sex)	Aula05 - Sexta-feira Santa
14/04 (sex)	Aula06
21/04 (sex)	Aula07 - Tiradentes
28/04 (sex)	Aula08
05/05 (sex)	Aula09
12/05 (sex)	**1ª Avaliação**
19/05 (sex)	Aula11
26/05 (sex)	Aula12
02/06 (sex)	Aula13
09/06 (sex)	Aula14
16/06 (sex)	Aula15
23/06 (sex)	Aula16
30/06 (sex)	Aula17
07/07 (sex)	**2ª Avaliação**
14/07 (sex)	**Recuperativa**
21/07 (sex)	*Bancas de TCC*

Revisão (parte 1)

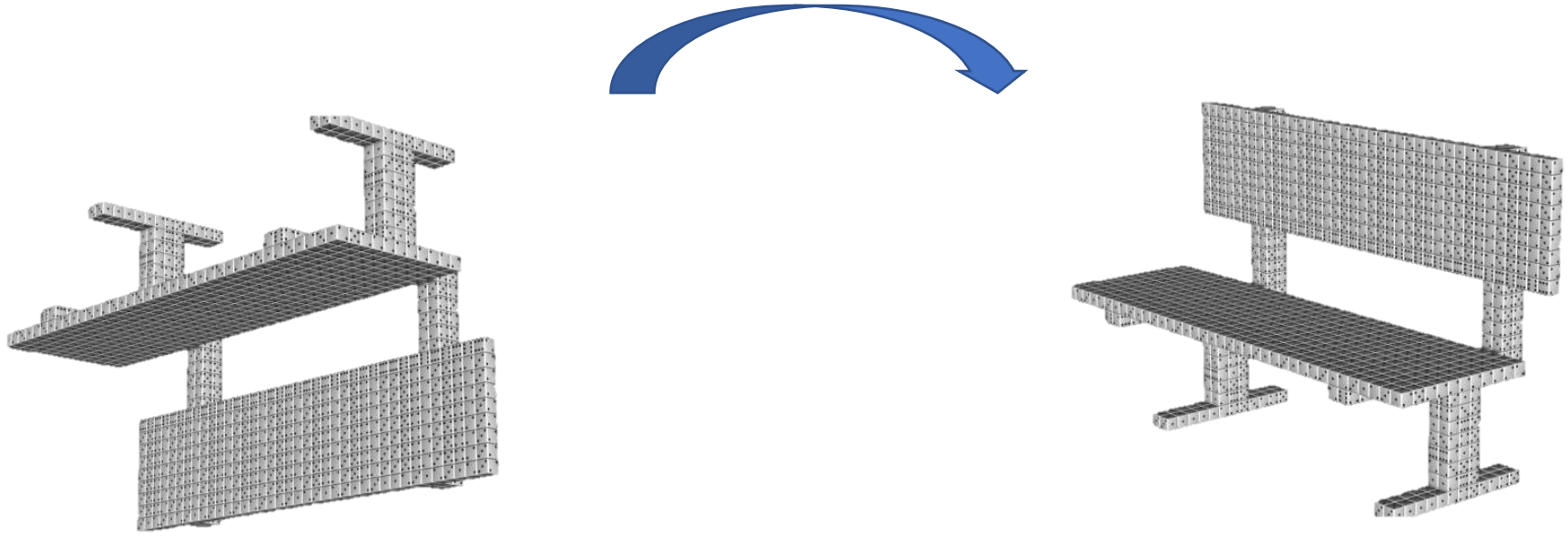
Normalização


EU SOU NORMAL
SARAIVA (FRANCISCO MILANI)

NORMALIZAÇÃO

Normalização de dados é o processo formal e passo a passo que examina os atributos de uma entidade, com o objetivo de evitar anomalias observadas na inclusão, exclusão e alteração de registros.

Fonte: http://www.luis.blog.br/normalizacao-de-dados-e-as-formas-normais.aspx



NORMALIZAÇÃO

Um modelo ER normalizado é convertido facilmente para um Banco de Dados relacional em tempo de projeto.

A **terceira forma normal** geralmente é aceita como boa para projeto de Banco de Dados sem redundância.

Existem formas normais de nível maior, mas que geralmente não são usadas.

Formas normais

1 FN (1o Forma Normal)

2 FN (2o Forma Normal)

3 FN (3o Forma Normal) “Normalizado”

---

4 FN (4o Forma Normal)

FNBC (Forma Normal de Boyce e Codd) (Raymond F. **Boyce** e Edgar F. **Codd**)

5 FN (5o Forma Normal)

MODELO NÃO NORMALIZADO

codFornecedor	nomeFornecedor	Tel1	Tel2	Endereco	codProduto	nomeProduto	precoUnitario	qtdPedida
1	Treichel	3232	5454	Rua X	100	Sabonete	6,00	50
1	Treichel	3232	5454	Rua X	200	Saboneteira	8,00	30
1	Treichel	3232	5454	Rua X	300	Talco	5,00	40
2	Krolow	6677	8899	Av. Y	100	Sabonete	6,00	30
2	Krolow	6677	8899	Av. Y	200	Saboneteira	8,00	15

Problemas de Exclusão:

Caso sejam deletadas todas as solicitações de um fornecedor, seus dados cadastrais também serão apagados.

MODELO NÃO NORMALIZADO

codFornecedor	nomeFornecedor	Tel1	Tel2	Endereco	codProduto	nomeProduto	precoUnitario	qtdPedida
1	Treichel	3232	5454	Rua X	100	Sabonete	6,00	50
1	Treichel	3232	5454	Rua X	200	Saboneteira	8,00	30
1	Treichel	3232	5454	Rua X	300	Talco	5,00	40
2	Krolow	6677	8899	Av. Y	100	Sabonete	6,00	30
2	Krolow	6677	8899	Av. Y	200	Saboneteira	8,00	15

Problemas de Inserção:

- Só é possível inserir um novo fornecedor quando o mesmo vender produtos;
- Só é possível inserir um novo produto quando o mesmo for vendido por um fornecedor.

MODELO NÃO NORMALIZADO

codFornecedor	nomeFornecedor	Tel1	Tel2	Endereco	codProduto	nomeProduto	precoUnitario	qtdPedida
1	Treichel	3232	5454	Rua X	100	Sabonete	6,00	50
1	Treichel	3232	5454	Rua X	200	Saboneteira	8,00	30
1	Treichel	3232	5454	Rua X	300	Talco	5,00	40
2	Krolow	6677	8899	Av. Y	100	Sabonete	6,00	30
2	Krolow	6677	8899	Av. Y	200	Saboneteira	8,00	15

Problemas de Atualização:

- Para atualizar o endereço do fornecedor, todos os registros desse fornecedor deverão ser atualizados.
- Para atualizar o preço do produto, todos os registros desse produto deverão ser atualizados.

MODELO NÃO NORMALIZADO

A normalização permite eliminar atributos:

- Com mais de um valor
- Duplicados ou repetidos
- Que contém dados derivados de outros atributos

Dependências

Dependência Funcional (Dependência Funcional Total/Completa)

Quando um atributo não identificador depende do(s) atributo(s) identificador(es).

Dependência Funcional Parcial

Quando um atributo não identificador depende de parte dos atributos identificadores.

Dependência Funcional Transitiva

Quando um atributo não identificador depende de outro atributo também não identificador.

Dependência Funcional

Um relacionamento entre dois ou mais atributos de forma que o valor de um atributo identifique o valor para cada um dos outros atributos, ou seja, um atributo está relacionado a outro.

CLIENTE
(PK) codigo
nomeCliente
tipoLogradouro
logradouro
numero
complemento
bairro
cidade
UF

Para cada **codigo** temos somente um valor para **nomeCliente**, **tipoLogradouro**, **logradouro**, **numero**, **complemento**, **bairro**, **cidade** e **UF**.

Os atributos **nomeCliente**, **tipoLogradouro**, **logradouro**, **numeroComplemento**, **bairro**, **cidade** e **UF** são funcionalmente dependentes de **codigo**.

Dependência Funcional Parcial

(PK) matriculaAluno	(PK) semestre	(PK) codUnidade	nomeUnidade	conceito
8784321	2	11	Banco de Dados 2	C
8784321	1	15	IoT	B
8784321	1	10	Gestão da TI	A
8784321	1	11	Banco de Dados 2	B

O campo **nomeUnidade** depende somente de **codUnidade** e não depende de **matriculaAluno** e **semestre**.

Dependência Funcional Transitiva

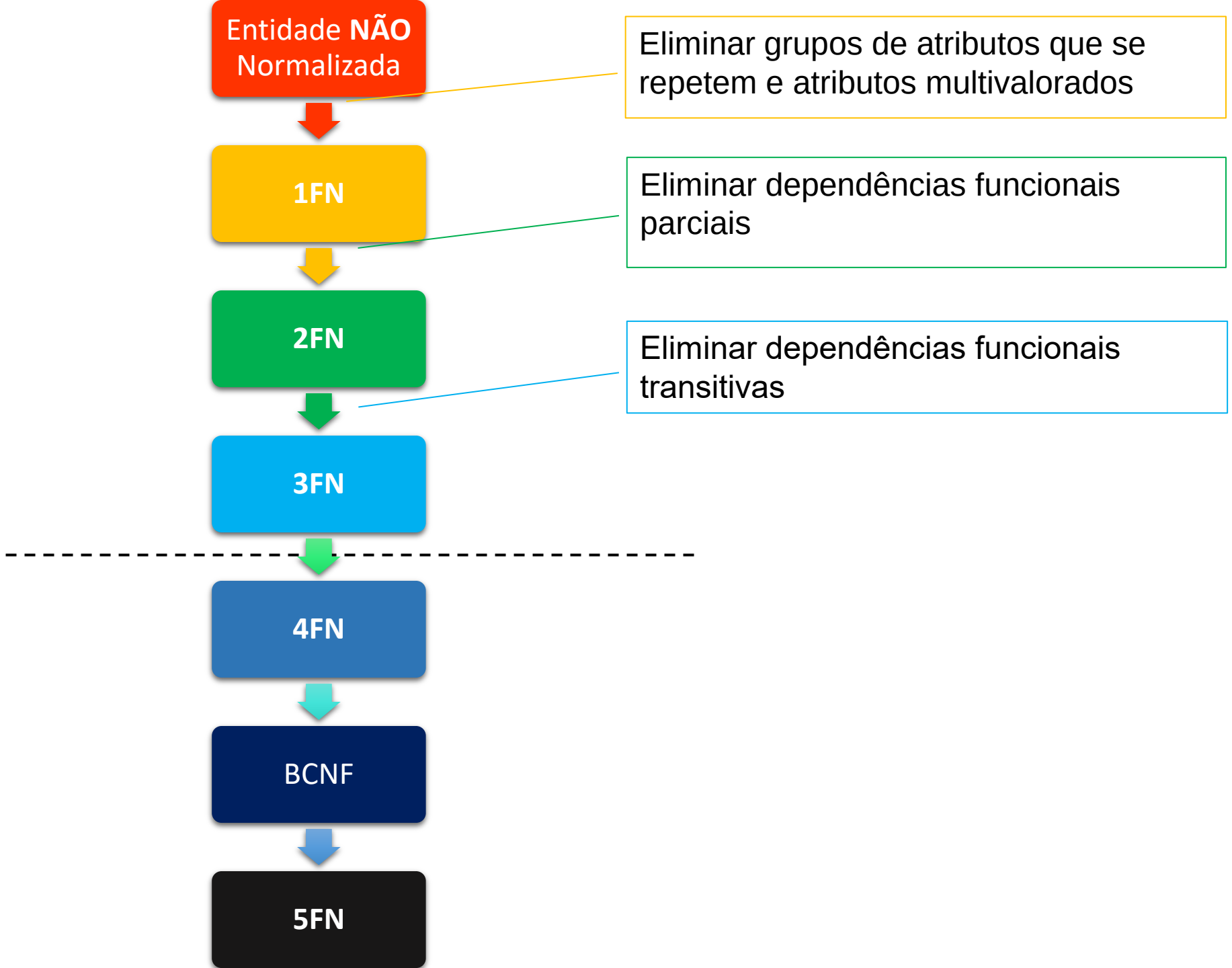
(PK) matricula	nomeFuncionario	codCargo	nomeCargo	salarioCargo
4432	Edécio	11	Professor	R$ 985,00
3321	Angelo	22	Secretário	R$ 400,00
2213	Cícero	33	Advogado	R$ 9.999,99
1124	Kelly	44	Analista de Sistemas	R$ 8.888,88
8940	Gladimir	11	Professor	R$ 985,00

O campo **matrícula** determina apenas os atributos **nomeFuncionario** e **codCargo**.

O campo **codCargo** (que não é chave primária) determina **nomeCargo** e o **salarioCargo**.

Os campos **nomeCargo** e o **salarioCargo** não dependem diretamente da chave (**matricula**).

PASSOS


Entidade NÃO Normalizada
1FN
2FN
3FN
4FN
BCNF
5FN
Eliminar grupos de atributos que se repetem e atributos multivalorados
Eliminar dependências funcionais parciais
Eliminar dependências funcionais transitivas

1FN – Primeira Forma Normal

- Uma entidade está na primeira forma normal se não tem atributos com mais de um valor, nem atributos que ocorrem mais de uma vez.

- Uma relação está 1FN se e somente se todos os seus domínios só contém valores atômicos;

- O modelo relacional exige que as relações estejam pelo menos na 1FN

1FN – Primeira Forma Normal

Os atributos da tabela não contêm grupos de repetição (tabelas aninhadas)

A tabela a seguir NÃO está na 1FN:

PROJETO						
codProj	**descProj**	**codFunc**	**nomeFunc**	**cargoFunc**	**salFunc**	**dtInicio**
11	Alfa	1001	Antonio	Analista Sr	1800	02/01/2022
		1004	Daniela	Analista Pl	1200	05/01/2022
12	Beta	1003	Claudio	Analista Sr	1800	10/02/2022

As tabelas a seguir estão na 1FN:

PROJETO	
codProj	**descProj**
11	Alfa
12	Beta

PROJFUNC					
codProj	**codFunc**	**nomeFunc**	**cargoFunc**	**salFunc**	**dtInicio**
11	1001	Antonio	Analista Sr	1800	02/01/2022
11	1004	Daniela	Analista Pl	1200	05/01/2022
12	1003	Claudio	Analista Sr	1800	10/02/2022

1FN – Primeira Forma Normal

A tabela a seguir NÃO está na 1FN:

Tabela: Cliente			
ID	CPF	Nome	Telefone
1	987654321	José Antônio	(31) 3333-4444
2	987654321	José Antônio	(31) 9999-8888
3	123456789	Carlos Alberto	(31) 8979-5969
4	512346789	Ricardo Roberto	(31) 8889-6325

As tabelas a seguir estão na 1FN:

Tabela: Cliente		
ID	CPF	Nome
1	987654321	José Antônio
3	123456789	Carlos Alberto
4	512346789	Ricardo Roberto

Tabela: Telefone		
ID Telefone	ID Cliente	Telefone
1	1	(31) 3333-4444
2	1	(31) 9999-8888
3	3	(31) 8979-5969
4	4	(31) 8889-6325

2FN – Segunda Forma Normal

- Uma entidade está na segunda forma normal se está na primeira forma normal e todos os seus atributos não identificadores são dependentes do atributo identificador da entidade.

- Uma relação está na 2 FN se e somente se está na 1FN e todos os atributos não chave são **totalmente** dependentes da chave primária;

- Diz respeito às chaves primárias compostas;

- Se a chave primária de uma relação não é composta e a relação está na 1FN, ela está também na 2FN;

- Uma relação que está na 1 FN pode não estar na 2FN se sua chave for composta.

2FN – Segunda Forma Normal

Condição: Chave Primária (PK) composta por mais de uma coluna

Todas as colunas que não fazem parte da PK dependem de todas as colunas que compõem a PK.

A tabela a seguir NÃO está na 2FN:

PROJFUNC					
codProj	**codFunc**	**nomeFunc**	**cargoFunc**	**salFunc**	**dtInicio**
11	1001	Antonio	Analista Sr	1800	02/01/2022
11	1004	Daniela	Analista Pl	1200	05/01/2022
12	1003	Claudio	Analista Sr	1800	10/02/2022

Passando para a 2FN:

FUNCIONARIO			
codFunc	**nomeFunc**	**cargoFunc**	**salFunc**
1001	Antonio	Analista Sr	1800
1004	Daniela	Analista Pl	1200
1003	Claudio	Analista Sr	1800

PROJFUNC		
codProj	**codFunc**	**dtInicio**
11	1001	02/01/2022
11	1004	05/01/2022
12	1003	10/02/2022

3FN – Terceira Forma Normal

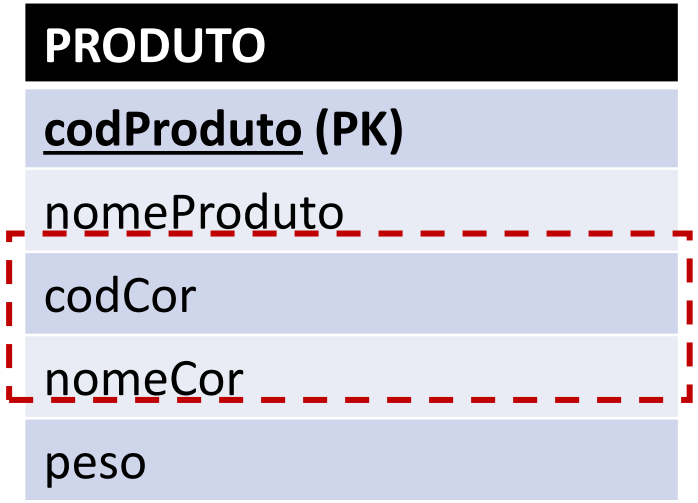
- Uma entidade está na terceira forma normal se está na primeira e na segunda forma normal e não contém atributos não identificadores dependentes de outros atributos não identificadores

- Uma relação está na 3FN se e somente se ela está na 2FN e seus atributos não chave são mutuamente independentes;

- A título de simplificação a terceira forma normal considera que a relação terá somente uma chave candidata, ou seja, a chave primária.

3FN – Terceira Forma Normal

- Os atributos cod_cor e cor são dependentes um do outro;
- Dependência indesejada durante as atualizações de dados;
- Não está na 3FN.



3FN – Terceira Forma Normal

Passos para normalização da relação:

- Eliminar o atributo cod_cor;
- Criar uma relação distinta para as informações sobre cor.

PRODUTO
codProduto (PK)
nomeProduto
codCor
peso

COR
codCor (PK)
nomeCor

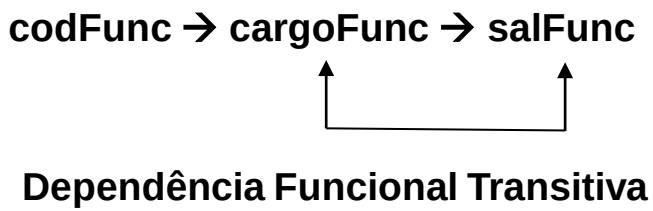
3FN – Terceira Forma Normal

Não há dependências funcionais transitivas.

Cada coluna não PK depende DIRETAMENTE da PK.

A tabela a seguir NÃO está na 3FN:

FUNCIONARIO			
codFunc	**nomeFunc**	**cargoFunc**	**salFunc**
1001	Antonio	Analista Sr	1800
1004	Daniela	Analista Pl	1200
1003	Claudio	Analista Sr	1800



Passando para a 3FN:

FUNCIONARIO		
codFunc	**nomeFunc**	**cargoFunc**
1001	Antonio	Analista Sr
1004	Daniela	Analista Pl
1003	Claudio	Analista Sr

CARGOSALARIO	
cargoFunc	**salFunc**
Analista Sr	1800
Analista Pl	1200

RESUMÃO

PRIMEIRA FORMA NORMAL

Uma entidade está na primeira forma normal se não tem atributos com mais de um valor, nem atributos que ocorrem mais de uma vez.

RESUMÃO

SEGUNDA FORMA NORMAL

- Uma entidade está na segunda forma normal se está na primeira forma normal e todos os seus atributos não identificadores são dependentes do atributo identificador da entidade.

RESUMÃO

TERCEIRA FORMA NORMAL

- Uma entidade está na terceira forma normal se está na primeira e na segunda forma normal e não contém atributos não identificadores dependentes de outros atributos não identificadores Observações

- Um modelo de E x R normalizado é convertido facilmente para um Banco de Dados relacional em tempo de projeto.

- A terceira forma normal geralmente é aceita como boa para projeto de Banco de Dados sem redundância.

REVISÃO (parte 2)

SQL


SQL


PostgreSQL

SQL


Microsoft
SQL Server


ORACLE
PL/SQL


MySQL

SQL - Structured Query Language - Linguagem de Consulta Estruturada

DDL – Data **D**efinition **L**anguage

Create, Alter e Drop

DML – Data **M**anipulation **L**anguage

Select, Insert, Delete e Update

DCL – Data **C**ontrol **L**anguage (subgrupo da DML)

Grant e Revoke

Bancos de Dados

O MySQL, por exemplo, já vem com alguns bancos de dados criados.

Para visualizá-los basta executar o comando **SHOW DATABASES**;

```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
4 rows in set (0.00 sec)

mysql>
```

Criação de Banco de Dados

```
Mysql> CREATE DATABASE exemplo;
```

ou

```
Mysql> CREATE SCHEMA exemplo;
```

Visualizar Bancos de Dados

```
Mysql> SHOW DATABASES; [Enter]
+------------------------------------+
|    Databases
+------------------------------------+
|    information_schema
|    mysql
|    performance_schema
|    sys
|    exemplo
+------------------------------------+
```

criados na instalação do MySQL

Excluindo banco de dados

Ao apagar um banco de dados, todas as tabelas e dados também serão excluídos.

```
Mysql> DROP DATABASE exemplo;
```

ou

```
Mysql> DROP SCHEMA exemplo;
```

Com DROP também podemos usar a clausula IF EXISTS:

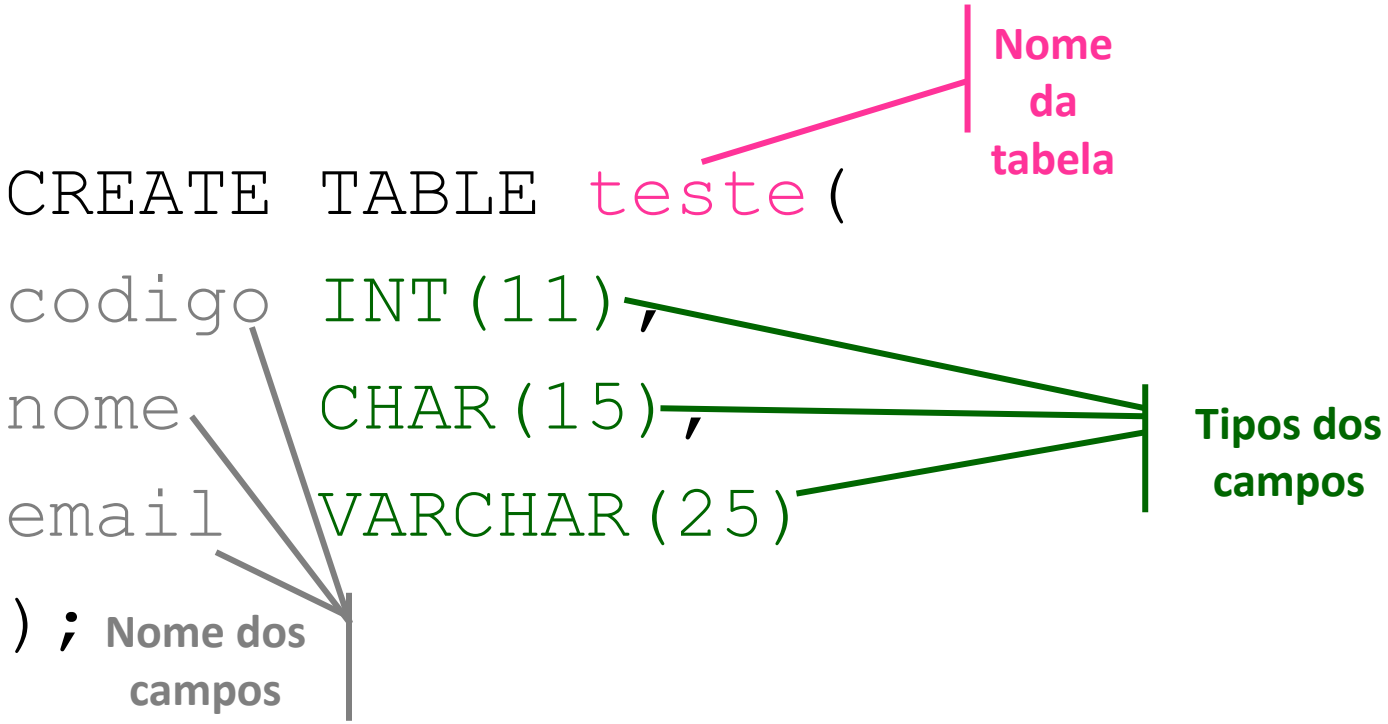
```
Mysql> DROP DATABASE IF EXISTS exemplo;
```

Selecionando um banco de dados

Podemos ter vários bancos de dados, mas só podemos manipular um por vez.

```
Myslq> USE  exemplo;
```

Criação de Tabela



Restrições

Uma maneira de limitar os dados que podem ser inseridos em uma tabela é a definição de tipo.

Restrições

Não Nulo (**NOT NULL**):

- Define que uma coluna não pode conter valor nulo.
- A cláusula NOT NULL especifica que uma coluna não admite valor vazio.

- Exemplo: `nome CHAR(15)` **`NOT NULL;`**
  - O campo nome não pode ser vazio, é de preenchimento obrigatório.

Restrições

Unicidade (UNIQUE):

- Define que os dados contidos na coluna (ou grupo de colunas) sejam únicos (não se repitam) em relação a todas as outras linhas da tabela.

- A cláusula **UNIQUE** especifica que os dados não se repetem.

Restrições

```
CREATE TABLE alunos (
  mat     char(5) UNIQUE,
  nome    char(50),
  nasc    date
);
```

ou

```
CREATE TABLE alunos (
  mat     char(5),
  nome    char(50),
  nasc    date,
  UNIQUE (mat)
);
```

Restrições

Se uma **restrição de unicidade** faz referencia a um grupo de colunas, elas são separadas por vírgula:

```
CREATE TABLE exemplo (
  a  char(5),
  b  char(50),
  c  date,
  UNIQUE (a,c)
);
```

Restrições

Chave primária:

- A chave primária indica que a coluna, ou grupo de colunas, pode ser utilizado como identificador único para as linhas da tabela

- É a junção de restrição de unicidade com a restrição de não nulo. Apenas a unicidade não garante identificador único pois não exclui os valores nulos.

- Uma tabela pode ter no máximo uma chave primária, mas pode ter várias restrições de unicidade e de não nulo.

Restrições

```
CREATE TABLE alunos (
  mat   char(5) PRIMARY KEY,
  nome char(50),
  nasc date);
```

A chave primária pode ser composta de vários atributos:

```
CREATE TABLE exemplo (
   a   char(5),
   b   char(50),
   c   date,
   PRIMARY KEY (a,c)
);
```

Restrições

Auto incremento:

- Este recurso, faz com que conforme novos registros são criados, automaticamente estes obtém valores que correspondem ao valor deste mesmo campo no registro anterior, somado a 1.

- Exemplo: `codigo INT` **`AUTO_INCREMENT`**;

  - Soma um a cada registro automaticamente neste campo.
  - Começando de 1, com inserção subsequente.

Mais exemplos

Antes de criar uma tabela, precisamos selecionar o banco de dados dentro do qual ela será criada, para isso utilize o comando USE nome do banco;

Criar tabela:

```
CREATE TABLE teste(
codigo    INTEGER AUTO_INCREMENT,
nome      CHAR(15) NOT NULL,
email     CHAR(30),
telefone CHAR(8),
PRIMARY KEY(codigo));
```

Comandos relativos as tabelas:

Mostrar tabelas - Lista todas as tabelas existentes no banco de dados atual.

```
mysql>SHOW tables;
```

Mostrar colunas - Mostra as colunas da tabela.

```
mysql>SHOW COLUMNS FROM teste;
```

Mostrar estrutura - Mostra a estrutura da tabela.

```
mysql>DESCRIBE teste;
mysql>DESC teste;
```

Comandos SQL – DDL – exemplo empregado

```
CREATE TABLE empregado
 (nome VARCHAR(15) NOT NULL,
  matricula CHAR(9),
  dataNasc DATE,
  endereco VARCHAR(30),
  sexo CHAR(1),
  salario NUMERIC(10,2),
  supervisor CHAR(9),
  depto INT(11) NOT NULL,
  PRIMARY KEY (matricula),
  CHECK (salario >= 0),
  FOREIGN KEY (supervisor) REFERENCES empregado(matricula),
  FOREIGN KEY (depto) REFERENCES departamento(codDep)
);
```

Comandos SQL – DDL – exemplo departamento

```
CREATE TABLE departamento
(codDep INT(11),
 nomeDep VARCHAR(15) NOT NULL,
 gerente CHAR(9) NOT NULL,
 dataInicioGer DATE,
 PRIMARY KEY(codDep),
 UNIQUE (nomeDep),
 FOREIGN KEY (gerente) REFERENCES empregado(matricula)
);
```

Comandos SQL - DDL

Exercício:

Defina as tabelas abaixo usando SQL

- **FORNECEDOR** (**codigo**, nome, cidade)
- **VENDA** (**codForn**, **codPeca**, quantidade, dataVenda)
- **PECA** (**codPeca**, nome, descricao)

É obrigatório que:

- nome da peça não seja nulo

Comandos SQL - DDL

```
CREATE TABLE Fornecedor
(
 codigo CHAR(10),
 nome VARCHAR(50) NOT NULL,
 cidade VARCHAR(80),
 PRIMARY KEY(codigo)
);
```

Comandos SQL - DDL

```
CREATE TABLE Peca
(
 codPeca CHAR(10),
 nome VARCHAR(50) NOT NULL,
 descricao VARCHAR(100),
 PRIMARY KEY (codPeca)
);
```

Comandos SQL - DDL

```
CREATE TABLE Venda
(
 codForn CHAR(10),
 codPeca CHAR(10),
 quantidade INT NOT NULL,
 data DATE NOT NULL,
 PRIMARY KEY (codForn, codPeca),
 FOREIGN KEY (codForn) REFERENCES Fornecedor(codigo) ON DELETE RESTRICT ON UPDATE CASCADE,
 FOREIGN KEY (codPeca) REFERENCES Peca(codPeca) ON DELETE SET NULL ON UPDATE SET DEFAULT
);
```

Comandos SQL - DDL

Chave estrangeira

- É definida com a cláusula **FOREIGN KEY**

Comandos SQL - DDL

ON DELETE

- **RESTRICT**: (default) significa que uma tentativa de se remover uma linha de T1 falhará se alguma linha em T2 combina com a chave
- **CASCADE**: remoção de uma linha de T1 implica em remoção de todas as linhas de T2 que combina com a chave de T1
- **SET NULL**: remoção de T1 implica em colocar NULL em todos os atributos da chave estrangeira de cada linha de T2 que combina
- **SET DEFAULT**: remoção de linha em T1 implica em colocar valores DEFAULT nos atributos da chave estrangeira de cada linha de T2 que combina

Comandos SQL - DDL

ON UPDATE

- **RESTRICT**: (default) update de um atributo de T1 falha se existem linhas em T2 combinando

- **CASCADE**: update de atributo em T1 implica que linhas que combinam em T2 também serão
  Atualizadas

- **SET NULL**: update de T1 implica que valores da chave estrangeira em T2 nas linhas que combinam são postos para NULL

- **SET DEFAULT**: update de T1 implica que valores da chave estrangeira de T2 nas linhas que combinam terão valores default aplicados

Comandos SQL - DDL

```
CREATE TABLE empregado (
  matricula CHAR(9),
  nome VARCHAR(15) NOT NULL,
  depto INT NOT NULL DEFAULT 1,
  supervisor CHAR(9),

  CONSTRAINT empPK
    PRIMARY KEY(matricula),

  CONSTRAINT empSuperFK
    FOREIGN KEY(supervisor) REFERENCES empregado(matricula),

  CONSTRAINT deptoFK
    FOREIGN KEY (depto) REFERENCES departamento(codigo)
);
```

Comandos SQL - DDL

ALTER TABLE

- Permite que se altere os atributos de uma determinada tabela ou que se adicione novos atributos (evolução de esquemas)

- Os novos atributos terão valores nulos em todas as linhas

- Ao incluirmos uma coluna devemos especificar o seu tipo de dado, não podendo esta coluna ser NOT NULL

Comandos SQL - DDL

Sintaxe:

```
ALTER TABLE tabela_base ADD atributo tipo_dado
```

Exemplo:

```
ALTER TABLE Peca ADD espessura INT;
```

Comandos SQL - DDL

Podemos remover um atributo usando a sintaxe

```
ALTER TABLE tabela_base
DROP atributo [CASCADE|RESTRICT]
```

CASCADE: remove todas as restrições relativas ao atributo e visões que contêm o atributo

RESTRICT: não permite a remoção do atributo se este é usado numa visão ou como chave estrangeira numa outra tabela

Comandos SQL - DDL

Exemplos:

```
ALTER TABLE empregado DROP endereco CASCADE;

ALTER TABLE departamento ALTER gerente SET DEFAULT “333444555”;

ALTER TABLE departamento ALTER gerente DROP DEFAULT;

ALTER TABLE empregado DROP CONSTRAINT empSuperFK;

ALTER TABLE empregado ADD CONSTRAINT empSuperFK FOREIGN KEY (supervisor) REFERENCES
empregado (matricula);
```

Comandos SQL - DDL

DROPTABLE

- Remove uma tabela-base do BD. Remove tanto os dados quanto a definição da tabela

Sintaxe

- `DROP TABLE <nomeTabela>`

Exemplo

- `DROP TABLE peca;`