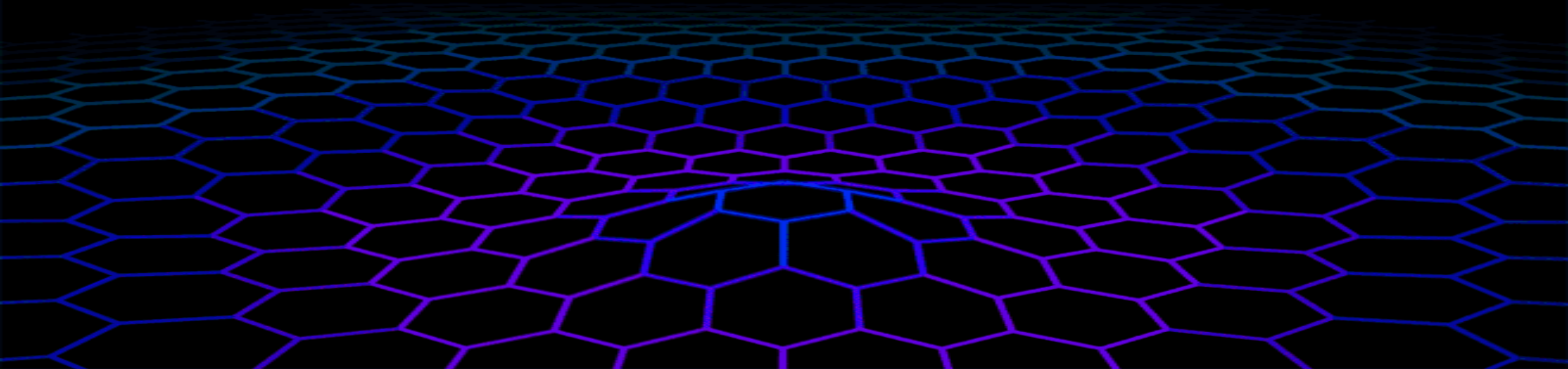


Banco de Dados II



Bit ENIDOS



QUE A FORÇA ESTEJA COM VOCÊS!

gladimircc



gladimir@gmail.com

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

ESCALA DE INTERVALO (NOITE) – UNISENAC & PRÉDIO TECH – 2025/1

DOCENTE	SEGUNDA-FEIRA	TERÇA-FEIRA	QUARTA-FEIRA	QUINTA-FEIRA	SEXTA-FEIRA
Nathália	20h (TECH)	20h15min (UniSenac)			
Edécio	20h (UniSenac)	20h (TECH)	20h30min (UniSenac)	20h (TECH)	20h30min (TECH)
Sérgio	20h15min (UniSenac)	20h15min (UniSenac)		20h (UniSenac)	20h (UniSenac)
Gladimir	20h30min (TECH)	20h15min (TECH)	20h (TECH)	20h15min (TECH)	20h15min (UniSenac)
Augusto	20h (TECH)	20h15min (TECH)	20h15min (UniSenac)	20h15min (UniSenac)	20h30min (UniSenac)
Eduardo	20h (TECH)	20h30min (UniSenac)	20h30min (TECH)		20h (UniSenac)
Angelo				20h (TECH)	
Irapuã		20h30min (UniSenac)	20h15min (UniSenac)	20h30min (UniSenac)	20h30min (UniSenac)
Pablo		20h30min (TECH)	20h30min (TECH)		
Wagner			20h (UniSenac)	20h (UniSenac)	
Leonardo		20h (TECH)	20h (UniSenac)	20h (UniSenac)	20h15min (UniSenac)
Bruna	20h15min (TECH)		20h15min (TECH)	20h15min (UniSenac)	20h (TECH)
Bruna					
Gugliano	20h15min (TECH)		20h15min (UniSenac)	20h (UniSenac)	20h (TECH)

Datas e feriados

2025-1	Datas importantes
10/03 (seg)	Início do Semestre
18/04 (sex)	Feriado - Sexta-feira Santa
21/04 (seg)	Feriado - Tiradentes
01/05 (qui)	Feriado - Dia do trabalhador
12/05 (seg)	Semana Acadêmica
13/05 (ter)	
14/05 (qua)	
15/05 (qui)	
16/05 (sex)	
19/06 (qui)	Feriado - Corpus Christi
19/07 (sáb)	Encerramento do Semestre

2025-2	Datas importantes
04/08 (seg)	Início do Semestre
20/09 (sáb)	Feriado - Revolução Farroupilha
11/10 (sáb)	Hackaton*
13/10 (seg)	Feriado - Dia do professor (antecipação)
15/11 (sáb)	Feriado - Proclamação da República
20/11 (qui)	Feriado - Consciência Negra
05/12 (sex)	Dia não letivo - Reunião geral
13/12 (sáb)	Encerramento do Semestre

Cronograma de Estudos

BD2	
12/mar - qua	Aula 01
19/mar - qua	Aula 02
26/mar - qua	Aula 03
02/abr - qua	Aula 04
09/abr - qua	Aula 05
16/abr - qua	Aula 06
23/abr - qua	Aula 07
30/abr - qua	Aula 08
07/mai - qua	Avaliação
14/mai - qua	Aula 10 - Semana Acadêmica
21/mai - qua	Aula 11
28/mai - qua	Aula 12
04/jun - qua	Aula 13
11/jun - qua	Aula 14
18/jun - qua	Aula 15
25/jun - qua	Aula 16
02/jul - qua	Avaliação
09/jul - qua	Bancas PI/PD1/PD2
16/jul - qua	Recuperativa
19/jul - sáb	Feedback -Aula NÃO presencial. NÃO síncrona. Material no BB

Avaliação

75% de frequência

Desenvolvimento de trabalhos em aula

Avaliações



MUSSUM

Arquivos

ISOs

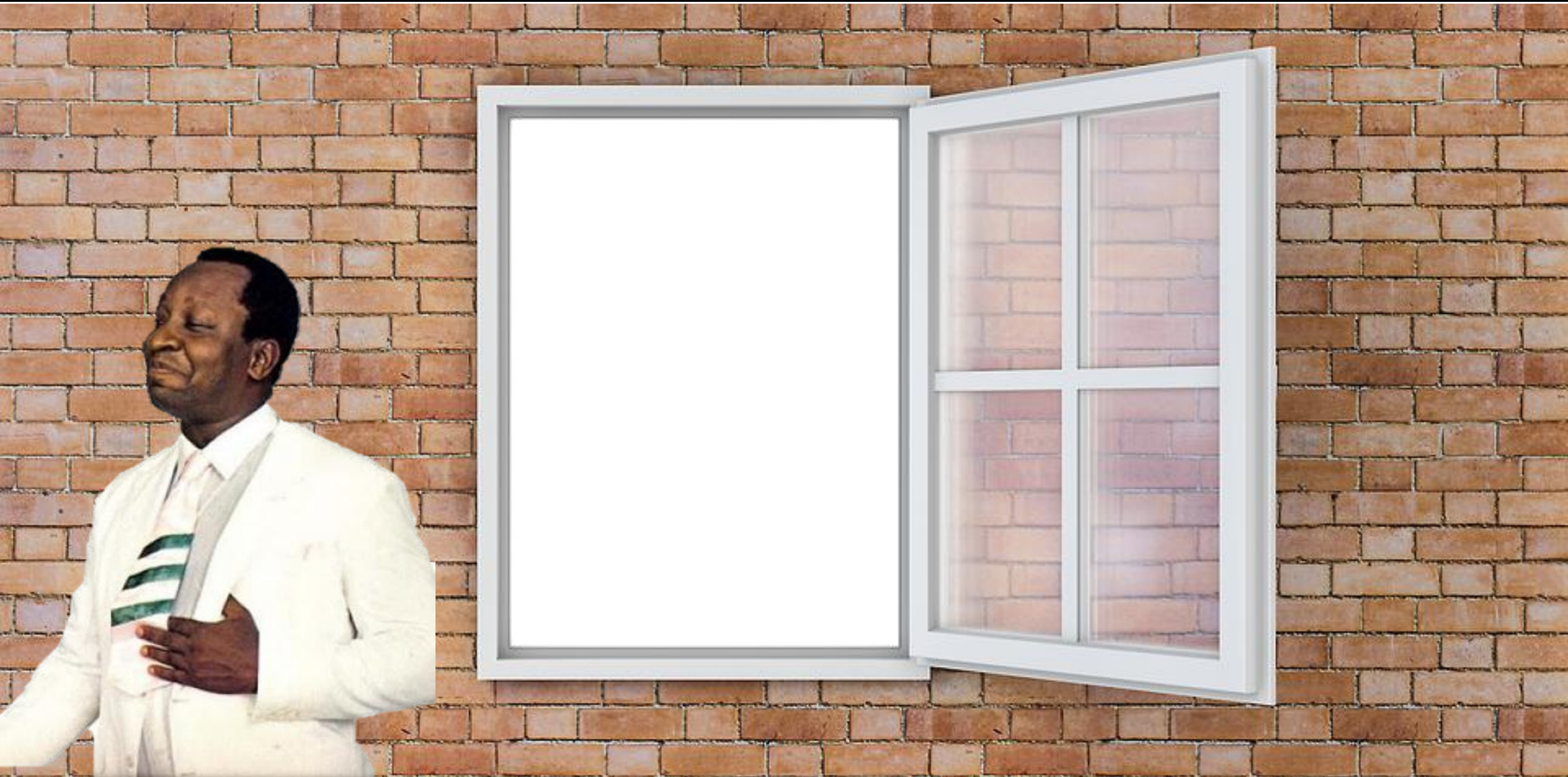
...



MUSSUM – INTERNO - 192.168.200.3



MUSSUM – EXTERNO - 177.101.203.139



Uso do laboratório



Não é permitido o consumo de alimentos e/ou bebidas no interior do laboratório.

Computadores (Desktops e Notebooks)



Preferencial

Usuário: **cpf@senacrs.edu.br** Senha: **ddmmaaaa#Educ**

Alternativa

Usuário: **.\aluno** Senha: **senacrs**



Wi-Fi

Usuário: **visitantes** Senha: **trijuntos**

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

<u>codFornecedor</u>	nomeFornecedor	Tel1	Tel2	Endereco	<u>codProduto</u>	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

<u>codFornecedor</u>	nomeFornecedor	Tel1	Tel2	Endereco	<u>codProduto</u>	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

<u>codFornecedor</u>	nomeFornecedor	Tel1	Tel2	Endereco	<u>codProduto</u>	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

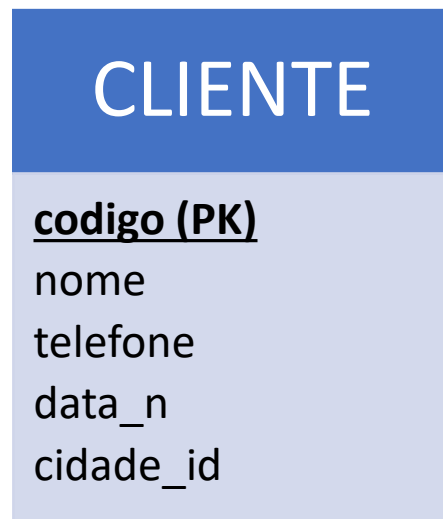
Quando um atributo não identificador depende de parte dos atributos identificadores.

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.



Para cada **codigo** temos somente um valor para **nome**, **telefone**, **data_n** e **cidade_id**.

Os atributos **nome**, **telefone**, **data_n** e **cidade_id** são funcionalmente dependentes de **codigo**.

Dependência Funcional Parcial

<u>matriculaAluno</u> (PK)	<u>semestre</u> (PK)	<u>codUnidade</u> (PK)	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

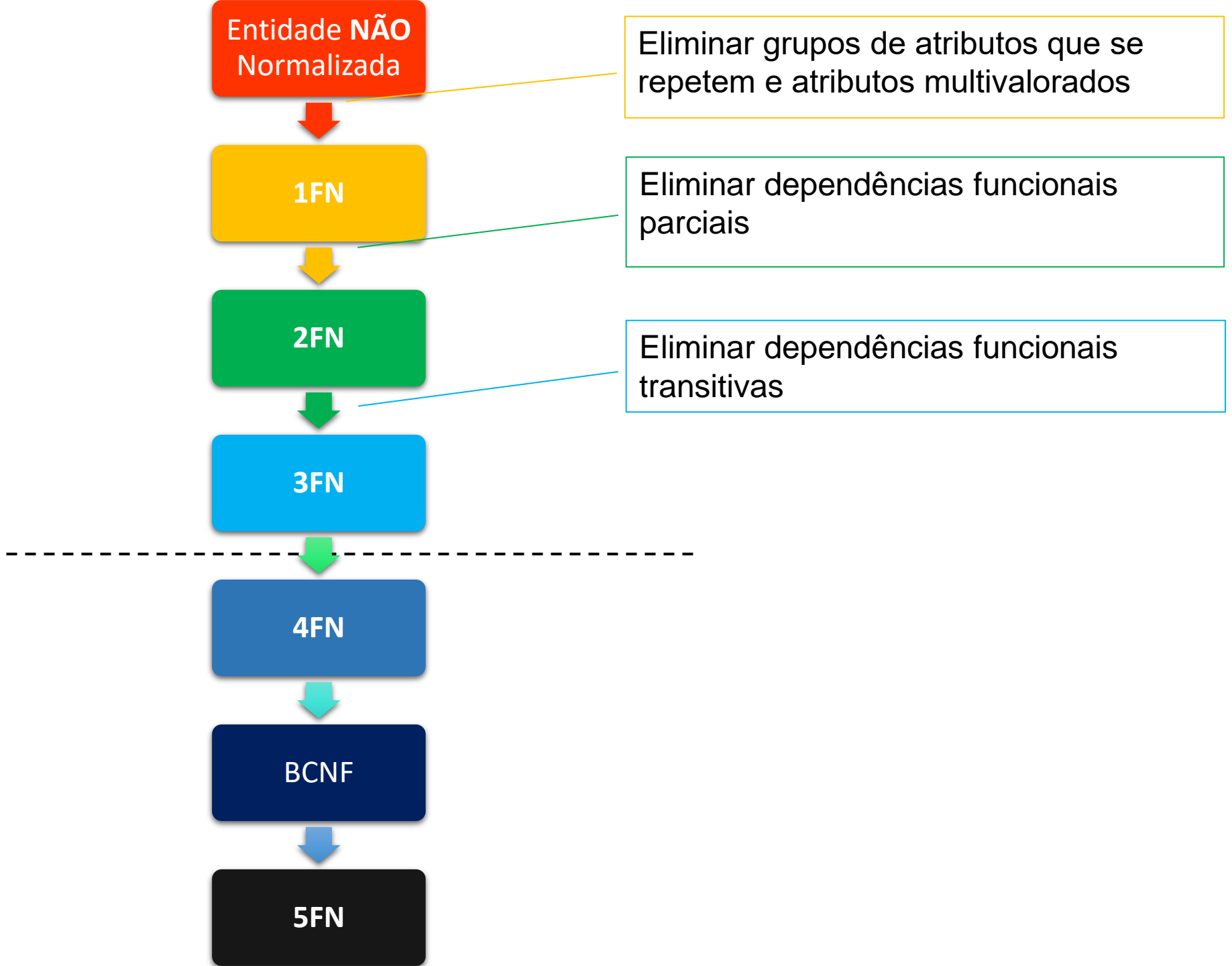
<u>matricula (PK)</u>	nomeFuncionario	codCargo	nomeCargo	salarioCargo
4432	Edécio	11	Professor	R\$ 985,00
3321	Angelo	22	Secretário	R\$ 400,00
2213	Cícero	33	Advogado	R\$ 9.999,99
1124	Kelly	44	Analista de Sistemas	R\$ 8.888,88
8940	Gladimir	11	Professor	R\$ 985,00

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



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						
<u>codProj</u>	descProj	<u>codFunc</u>	nomeFunc	cargoFunc	salFunc	dtInicio
11	Alfa	1001	Antonio	Analista Sr	1.800,00	02/01/2022
11	Alfa	1004	Daniela	Analista Pl	1.200,00	05/01/2022
12	Beta	1003	Claudio	Analista Sr	1.800,00	10/02/2022

As tabelas a seguir estão na 1FN:

PROJETO	
<u>codProj</u>	descProj
11	Alfa
12	Beta

PROJFUNC					
<u>codProj</u>	<u>codFunc</u>	nomeFunc	cargoFunc	salFunc	dtInicio
11	1001	Antonio	Analista Sr	1.800,00	02/01/2022
11	1004	Daniela	Analista Pl	1.200,00	05/01/2022
12	1003	Claudio	Analista Sr	1.800,00	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					
<u>codProj</u>	<u>codFunc</u>	nomeFunc	cargoFunc	salFunc	dtInicio
11	1001	Antonio	Analista Sr	1.800,00	02/01/2022
11	1004	Daniela	Analista Pl	1.200,00	05/01/2022
12	1003	Claudio	Analista Sr	1.800,00	10/02/2022

Passando para a 2FN:

FUNCIONARIO			
<u>codFunc</u>	nomeFunc	cargoFunc	salFunc
1001	Antonio	Analista Sr	1.800,00
1004	Daniela	Analista Pl	1.200,00
1003	Claudio	Analista Sr	1.800,00

PROJFUNC		
<u>codProj</u>	<u>codFunc</u>	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.

PRODUTO
<u>codProduto</u> (PK)
nomeProduto
codCor
nomeCor
peso

3FN – Terceira Forma Normal

Passos para normalização da relação:

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

PRODUTO
<u>codProduto</u> (PK)
nomeProduto
codCor
peso

COR
<u>codCor</u> (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			
<u>codFunc</u>	nomeFunc	cargoFunc	salFunc
1001	Antonio	Analista Sr	1.800,00
1004	Daniela	Analista Pl	1.200,00
1003	Claudio	Analista Sr	1.800,00

codFunc → cargoFunc → salFunc



Dependência Funcional Transitiva

Passando para a 3FN:

FUNCIONARIO		
<u>codFunc</u>	nomeFunc	cargoFunc
1001	Antonio	Analista Sr
1004	Daniela	Analista Pl
1003	Claudio	Analista Sr

CARGOSALARIO	
<u>cargoFunc</u>	salFunc
Analista Sr	1.800,00
Analista Pl	1.200,00

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



PostgreSQL



SQL



ORACLE®
PL/SQL



DDL – Data Definition Language

Create, Alter e Drop

DML – Data Manipulation Language

Select, Insert, Delete e Update

DCL – Data Control Language (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	} criados na instalação do MySQL
mysql	
performance_schema	
sys	
exemplo	

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 cláusula 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.

```
Mysqlq> USE exemplo;
```

Criação de Tabela

```
CREATE TABLE teste (  
  codigo INT(11) ,  
  nome CHAR(15) ,  
  email VARCHAR(25)  
) ;
```

Nome da tabela

Tipos dos campos

Nome dos campos

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.

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)  
);
```

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) DEFAULT "XBZ4",
  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 RESTRICT ON UPDATE CASCADE
);
```

Comandos SQL - DDL

Chave estrangeira

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

Comandos SQL - DDL

ON DELETE e ON UPDATE

Ação	MySQL	PostgreSQL	Observações
RESTRICT	✓ Sim	✓ Sim	Impede a atualização/exclusão se houver linhas relacionadas.
NO ACTION	✓ Sim	✓ Sim	Semelhante ao RESTRICT, é o padrão em ambas as bases.
CASCADE	✓ Sim	✓ Sim	Atualiza/exclui automaticamente linhas relacionadas.
SET NULL	✓ Sim	✓ Sim	Define a chave estrangeira como NULL (necessita permitir NULL).
SET DEFAULT	✗ Não	✓ Sim	Define a chave estrangeira para um valor padrão previamente configurado (não suportado no MySQL).

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;`



Atividade

```
-- Excluir banco de dados, caso exista
DROP DATABASE IF EXISTS revisao1;

-- Criar o banco de dados
CREATE DATABASE revisao1;

-- Usar o banco de dados
USE revisao1;

-- Criar tabela aluno
CREATE TABLE aluno (
    id          INT AUTO_INCREMENT PRIMARY KEY,
    nome        VARCHAR(50),
    sobrenome    VARCHAR(50),
    status      INT,
    email       VARCHAR(100)
);

-- Inserir dados na tabela alunos
INSERT INTO aluno (nome, sobrenome, status, email)
VALUES
('João', 'Silva', 20, 'joao@senacrs.com.br'),
('Maria', 'Santos', 21, 'maria@senacrs.com.br'),
('Pedro', 'Oliveira', 22, 'pedro@senacrs.com.br'),
('Ana', 'Ferreira', 19, 'ana@senacrs.com.br'),
('Carlos', 'Almeida', 20, 'carlos@senacrs.com.br'),
('Mariana', 'Pereira', 23, 'mariana@senacrs.com.br'),
('Lucas', 'Rodrigues', 24, 'lucas@senacrs.com.br'),
('Juliana', 'Gomes', 25, 'juliana@senacrs.com.br'),
('Fernanda', 'Martins', 18, 'fernanda@senacrs.com.br'),
('Rafael', 'Barbosa', 20, 'rafael@senacrs.com.br');
```



```
-- Criar tabela curso
CREATE TABLE curso (
    id          INT AUTO_INCREMENT PRIMARY KEY,
    nome        VARCHAR(50),
    descricao   VARCHAR(255)
);
```

```
-- Inserir dados na tabela curso
INSERT INTO curso (nome, descricao)
VALUES
('Matemática', 'Curso de matemática básica'),
('Física', 'Curso de física aplicada'),
('Química', 'Curso de química orgânica'),
('História', 'Curso de história contemporânea'),
('Geografia', 'Curso de geografia política'),
('Inglês', 'Curso de inglês avançado'),
('Programação', 'Curso de programação web'),
('Artes', 'Curso de artes plásticas'),
('Literatura', 'Curso de literatura mundial'),
('Educação Física', 'Curso de educação física');
```

```
-- Criar tabela matricula
CREATE TABLE matricula (
    id          INT AUTO_INCREMENT PRIMARY KEY,
    aluno_id    INT,
    curso_id    INT,
    DataMatricula DATE,
    FOREIGN KEY (aluno_id) REFERENCES Aluno(id),
    FOREIGN KEY (curso_id) REFERENCES Curso(id)
);
```



```
-- Inserir dados na tabela matricula
INSERT INTO matricula (aluno_id, curso_id, dataMatricula)
VALUES
(1, 1, '2025-01-15'),
(2, 3, '2025-02-20'),
(3, 5, '2025-03-10'),
(4, 7, '2025-04-05'),
(5, 9, '2025-05-12'),
(6, 2, '2025-06-18'),
(7, 4, '2025-07-22'),
(8, 6, '2025-08-30'),
(9, 8, '2025-09-14'),
(10, 10, '2025-10-25');
```

Exercícios:

- a) Escreva uma consulta para exibir todos os alunos matriculados em um curso específico, por exemplo, "Matemática".
- b) Altere o nome do curso com o id 3 para "Química Orgânica".
- c) Adicione uma nova coluna chamada "telefone" à tabela aluno do tipo VARCHAR(15).
- d) Tente remover o aluno com o id 7 da tabela Alunos. Houve um erro? Explique.
- e) Liste todos os cursos juntamente com a contagem de alunos matriculados em cada curso.
- f) Altere o status do aluno com id 1 para 21.
- g) Adicione uma nova coluna chamada "nível" à tabela curso do tipo VARCHAR(20).
- h) Remova a matrícula com o id 5 da tabela matricula.
- i) Renomeie a tabela curso para "disciplina".
- j) Altere o e-mail do aluno com id 9 para 'fernanda_novo@senacrs.com.br'.

Exercícios de revisão

