8 - BANCO DE DADOS II

8.1 - MÓDULO 01

8.1.1 - ENTENDER A IMPORTANCIA DA JUNÇÃO ENTRE TABELAS, UTILIZANDO AS CHAVES PRIMÁRIAS E ESTRANGEIRAS IMPLEMENTADAS EM SUAS TABELAS E COMO UTILIZÁ-LAS

8.1.1.1 - VÍDEO - Chave primária e estrangeira, o que é preciso saber?

Chave primária(PK) é um identificador único de uma tabela, não pode ser `NULL` e seu conteúdo não deve ser duplicado. Por exemplo, em uma tabela PESSOA, o CPF seria uma chave primária, pois é unica entre as pessoas e não se repete.

A PK pdoe ser dividida em:

- Simples, formado com um único campo da tabela, o cpf por exemplo.
- Composta, formado por mais de um campo, os valores podem ser repetidos mas não combinados.
    - Ex: tabela conta bancária, tem o NUMERO_CONTA como PK e NUMERO_AGENCIA como PK, a agência é única entre os bancos assim como o número da conta é único em cada banco.

> RECOMENDAÇÕES

- Serem únicas;
- Priorizar chave simples;
- Favorecer números;
- Selecionar chaves familiares;

A chave estrangeira é um recurso para criar relacionamento entre tabelas. É possivel otimizar consultas com elas.

Pode verificar que foi adicionada a chave primária CPF da tabela PESSOA na tabela CONTA BANCÁRIA. Assim podendo relacionar a conta à pessoa.

Pode a PK ser formada por mais de uma coluna, assim chamada de `Chave Estrangeira Composta`, assim **uma tabela pode ter mais de uma chave estrangeira**.

Utilizar a chave estrangeira estabelece algumas restrições para EXCLUSÃO ou ATUALIZAÇÃO de registros:

- Restrições para **Exclusão**:
  - *ON DELETE CASCADE*;
    - Ocorre quando uma **tabela referenciada foi apagada**, gerando um efeito cascata para as chaves estrangeiras correspondentes.
  - *ON DELETE SET NULL*;
    - Chaves estrangeiras relacionadas são definidas com valor nulo.
  - *ON DELETE SET DEFAULT*;
    - Chaves estrangeiras relacionadas em outras tabelas são definidas com um valor padrão
- Restrições para **Atualização**:
  - *ON UPDATE CASCADE*;
    - Ao modificar um registro numa tabela que seja relacionada a uma coluna de chave estrangeira em outra tabela, as chaves estrangeiras têm o seu valor automaticamente atualizado.
  - *ON UPDATE SET NULL*;
    - Chaves estrangeiras relacionadas são definidas com valor nulo para que a ação seja um sucesso
  - *ON UPDATE SET DEFAULT*;
    - Chaves estrangeiras relacionadas em outras tabelas são definidas com um valor padrão

8.1.1.2 - Conceito da chave primária-PK e estrangeira-FK, e criação de tabela aplicando a chave primária e estrangeira

A chave primária, ou primary key (`PK`), determina, exclusivamente, um identificador por tabela. Isso **significa que a informação presente nela não pode ser repetida.**

Além disso, uma chave primária não pode ser nula, ou seja, o campo da tabela não pode ficar vazio, **precisa sempre conter a chave.**

A chave primária é categorizada em dois tipos:

- simples, é constituída por um único campo na tabela que não pode ter mais de um registro com o mesmo valor.
- composta, é formada por mais de um campo, que pode repetir valores, mas **sem replicar suas combinações**.

8.1.1.2.1 Exemplo de chave primária simples

Essa tabela tem apenas um campo de chave primária, o CPF de cada pessoa. Portanto, é uma chave primária simples.

Pessoa		
CPF:PK	NOME	ENDEREÇO
123.456.789-11	Carol Dias	Rua XV, 015
120.455.789-12	Carlos Souza	Rua do Barbalho, 56
121.454.789-13	Carol Queiroz	Rua XV, 212

Vale destacar que **uma tabela precisa de um identificador para acessarmos os dados relacionados à chave**. Nessa tabela, que armazena dados de pessoas, cada CPF é uma chave primária. Pessoas podem ter o mesmo nome, podem morar na mesma rua, mas o CPF é algo único, ou seja, cada pessoa tem o seu. Nesse caso, por meio do CPF, **podemos ter acesso ao restante dos dados de cada pessoa**.

A tabela que veremos a seguir tem dois campos de chave primária, sendo NUMERO_CONTA e NUMERO_AGENCIA. Dessa forma, ela tem **chaves primárias compostas**.

CONTA BANCÁRIA		
NUMERO_CONTA: PK	NUMERO_AGENCIA: PK	SALDO
56	1	R$ 200,00
08	2	R$ 1.300,00
56	2	R$ 800,00
25	1	R$ 5.450,00

Nessa tabela, podemos ter mais de uma conta bancária com o mesmo saldo, mais de uma conta bancária com o mesmo número ou mais de uma conta bancária na mesma agência. O que não é possível é ter mais de uma conta com o mesmo número na mesma agência simultaneamente. Observem que a primeira e a terceira conta possuem números de contas iguais mas que são de agências diferentes, e que também que cada agência possui mais de uma conta. Nessa tabela, isso não é um problema, pois os pares das chaves primárias não são iguais. O que não poderia ocorrer é, por exemplo, duas contas iguais na mesma agência.

8.1.1.2.2 Chave estrangeira-FK

Assim como a chave primária, a estrangeira faz parte dos conceitos básicos de Banco de Dados. A chave estrangeira, ou foreign key (FK), **é utilizada na criação dos relacionamentos entre tabelas.**

Ela é chamada assim pois **faz referência à chave primária** *de uma tabela que não é a dela*. Em sua tabela, **a chave estrangeira é a própria chave primária.**

> A chave estrangeira é utilizada para os relacionamentos das tabelas.

Por exemplo, imagine o funcionamento de uma empresa onde existem duas tabelas, uma com os nomes de todos os **departamentos** e outra com os nomes de todos os **funcionários**.

Como saber em qual departamento cada funcionário trabalha? Para isso, precisamos fazer um relacionamento entre as tabelas.

O ID é a chave estrangeira que foi adquirida da tabela departamento. Por meio dela, conseguimos saber em qual departamento cada funcionário trabalha.

Antes de ser chave estrangeira, ela é uma chave primária em sua tabela de origem. Quando passa a ser referenciada em uma outra tabela, ela passa a ser chave estrangeira. Nesse caso, o ID é a chave primária da tabela DEPARTAMENTO, mas, quando é referenciada na tabela FUNCIONÁRIOS, passa a ser uma chave estrangeira.

Com a chave estrangeira, é possível otimizar as consultas no banco de dados a partir de um cruzamento de dados.

[Cruzamento de dados] https://www.site.moki.com.br/post/cruzamento-de-dados#:~:text=Cruzamento de dados%3A entenda o conceito e por que fazer&text=Nos processos que investigam desvios,partes envolvidas em alguma transação.

No nosso exemplo, ao efetuar uma consulta no banco de dados pelo CPF de um funcionário, já conseguimos obter todas as suas informações, incluindo qual departamento ele trabalha.

> O cruzamento de dados é comum para otimizar as consultas em bancos de dados.

Assim, uma chave estrangeira-FK **é uma coluna ou um grupo de colunas**. Ela **pode ou não** ser a chave primária da sua própria tabela, mas, **com certeza, é a chave primária de outra tabela**.

Quando a chave estrangeira-FK é formada por **mais de uma coluna, ela é denominada de composta.**

Para a construção de um banco de dados, é fundamental compreender o uso das diferentes chaves nas tabelas.

Além disso, é importante:
Atentar **quando precisamos utilizar as chaves primárias compostas e as chaves estrangeiras**, pois elas

não são necessárias em todas as tabelas. Porém, **todas as tabelas devem conter uma chave primária**, seja ela simples ou não.

> Leitura Complementar

[BÓSON TREINAMENTOS. Modelagem de Dados - Chave Primária, Estrangeira e outras.] https://www.youtube.com/watch?v=sbIT5UXTEg8

> Referência Bibliografica

[MACÊDO, Diego. Entendendo as Chaves dos Bancos de Dados. Diego Macêdo, 12 dez. 2011.] https://www.diegomacedo.com.br/entendendo-as-chaves-dos-bancos-de-dados/

[MEIRA, Regilan. Banco de Dados. Instituto Federal da Bahia, 2013.] http://www.regilan.com.br/wp-content/uploads/2013/10/Apostila-Banco-de-Dados.pdf

[REGILAN. Aula 05: Relacionamento entre Tabelas - Primary Key e Foreign Key. 34 slides. Blog do Regilan, 22 mar. 2015.] http://www.regilan.com.br/wp-content/uploads/2015/03/banco_de_dados_aula_05-Relacionamento-entre-TabelasPK-e-FK.pdf

[REIS, Fábio. Modelagem de Dados – Tipos de Chaves. BÓSON TREINAMENTOS EM CIÊNCIA E TECNOLOGIA, 16 jun. 2020.]http://www.bosontreinamentos.com.br/modelagem-de-dados/modelagem-de-dados-tipos-de-chaves/

8.1.1.3 - Anotações Exercício

1. Em relação às tabelas de um banco de dados, assinale a alternativa correta sobre a chave estrangeira.
    1. Em uma tabela, a chave estrangeira faz referência a uma chave primária de outra tabela.
        1. Resposta correta!A chave estrangeira cria uma relação entre tabelas.
2. Para que serve uma chave estrangeira na tabela do banco de dados?
    1. Para criar uma relação entre tabelas através da chave primária.
        1. Resposta correta!A chave estrangeira relaciona tabelas por meio da chave primária.
3. A chave estrangeira relaciona tabelas por meio da chave primária.
    1. A chave primária identifica e garante a unicidade de um registro em tabela, enquanto a estrangeira promove o relacionamento entre tabelas.
        1. Resposta correta!A chave primária relaciona-se com a chave estrangeira por ser um dado único associado entre tabelas. Como exemplo, o CPF, um registro único, que é possível relacionar com compras ou pagamentos do seu dono.
4. Assinale a alternativa correta sobre a chave primária.
    1. A chave primária é única e não nula, ou seja, deve haver um valor válido.
        1. Resposta correta!Na chave primária, deve sempre existir um valor válido, pois um valor único é relacionado a esse dado.

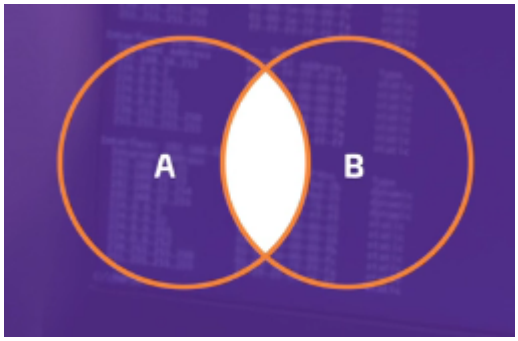
8.1.2 - Compreender e aplicar todas as condições para execução de junções

8.1.2.1 - VÍDEO - Conceitos essenciais de junção entre tabelas

Junções são muito comuns, pois é a relação entre as tabelas que permite isso.

Junções vem da Teoria dos Conjuntos, geralmente apresentada por diagramas de Euler-Venn. Esse diagramas, círculos, são representações gráficas dos conjuntos.

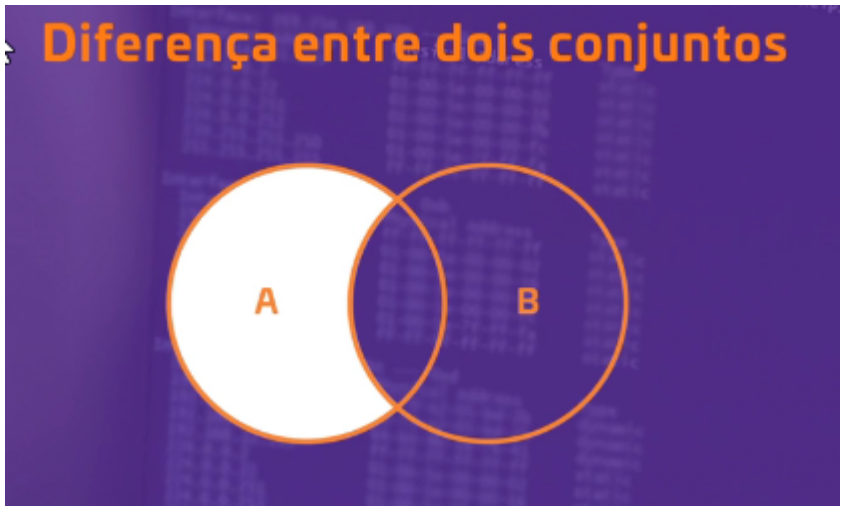
Conjuntos **podem ou não possuir** algum tipo de **interseção**.

- `não possuir interseção` são chamados de **conjuntos distintos**.Situações quem que três conjuntos **não possuem interseção**, são chamados `DISJUNTOS`.
- `possuir interseção`, podem ser representados pelo diagrama de Venn, utilizando simbolos que indicam quando um elemento pertence ou não ao conjunto específico.
  

  conjunto A possui interseção com B
- `contido ou não em outro conjunto`, **relação de inclusão**, é um subconjunto.
  - 

    - Conjunto A contido em B.
- **Operação de União**, une dois conjuntos considerando os seus elementos.
  - 

    - Há a união de conjunto A com B
- **Diferença entre dois conjuntos**, apresenta distinção entre eles.

- 

    - Elementos que pertencem a A que não pertencem a B. São os elementos únicos de A.

8.1.2.2 - Junção entre tabelas (Inner Join, Left Join, Right Join e Full Join)

Junção entre tabelas

O objetivo das junções entre tabelas é relacioná-las a partir de regras específicas.

Por exemplo, para fazer consultas em um banco de dados, é necessário relacionar tabelas.

> A junção entre tabelas otimiza consultas básicas em um banco de dados.

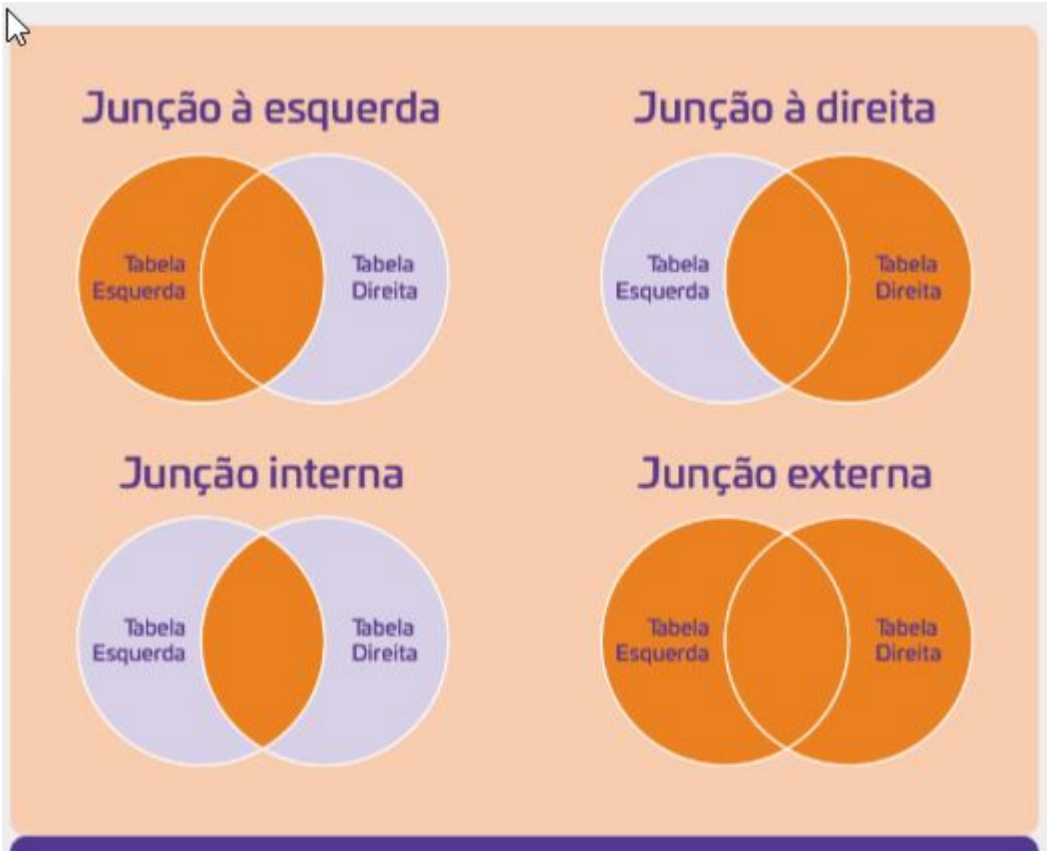
Antes de conceituar a junção entre tabelas, é necessário compreender a teoria dos conjuntos. Ela envolve os diagramas de Euler-Venn, que representam conjuntos graficamente.

[Teoria dos Conjuntos]https://www.todamateria.com.br/teoria-dos-conjuntos/

[Diagramas de Euler Venn]https://brasilescola.uol.com.br/matematica/diagrama-de-venn.htm

As junções entre tabelas podem ser realizadas a partir de SQL Joins, como: Inner Join, Left Join, Right Join, Outer Join etc.

> Operações básicas de SQL Joins.



Junção :

- A esquerda,
- A direita,
- interna,
- externa,

Ao menos duas tabelas devem ter sido criadas no banco para se realizar a junção. Criar BD com duas tabelas, disciplina e alunos.

A tabela disciplinas possui os seguintes dados como atributos:

id_disciplina - o identificador da tabela e chave primária;

nome_disciplina - o nome de cada disciplina;

nome_professor - o nome de cada professor das disciplinas.

Para criar essa tabela no banco de dados, usaremos o comando create table.

```
CREATE TABLE disciplinas (
id_disciplina INT AUTO_INCREMENT PRIMARY KEY,
nome_disciplina VARCHAR(50) NOT NULL,
nome_professor VARCHAR(50) NOT NULL
```

```
);
```

Também usaremos o comando create table para criar a tabela alunos no banco de dados.

Ela possui como atributos os dados:

id_alunos - o identificador da tabela e chave primária;

nome_alunos - o nome de cada aluno;

disciplinas_id - atributo que será associado ao id_disciplinas da tabela disciplinas. Portanto, é uma chave estrangeira.

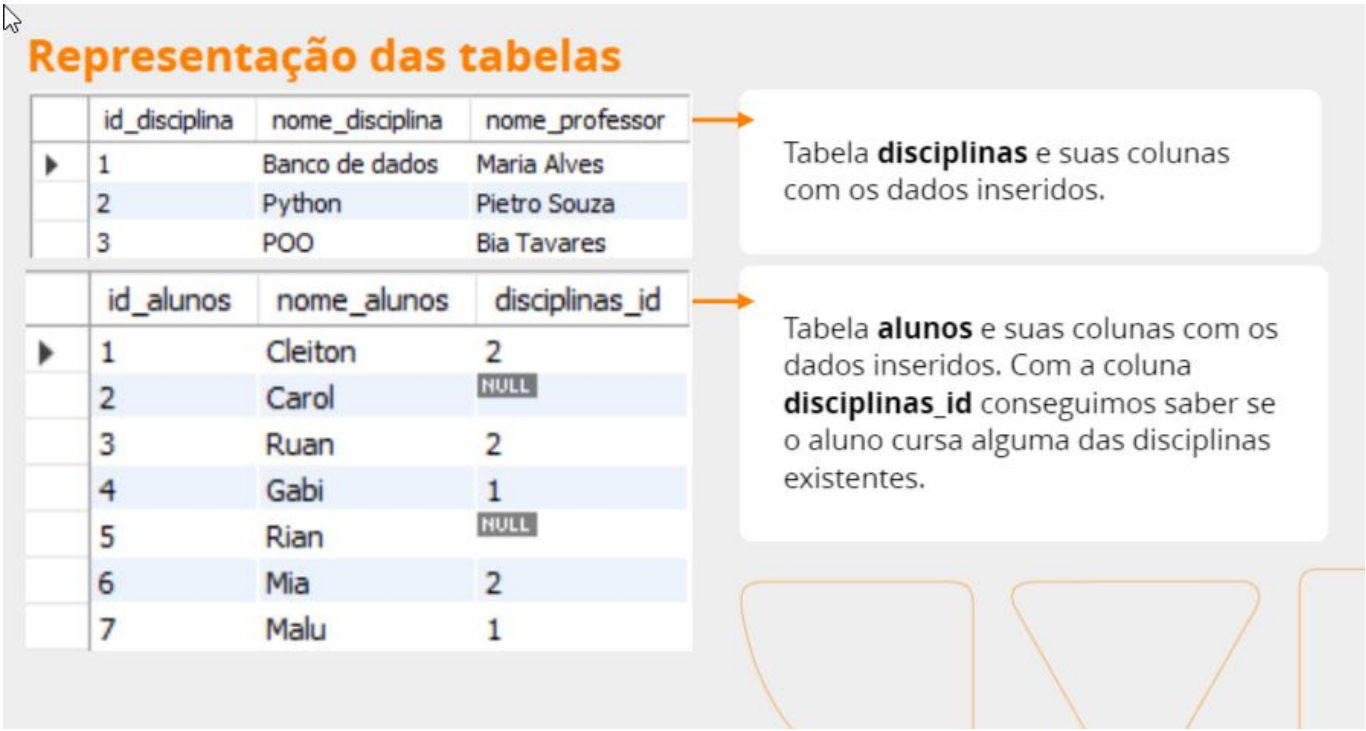
> Criação da tabela alunos com seus atributos.

```
CREATE TABLE alunos (
  id_alunos INT AUTO_INCREMENT PRIMARY KEY,
  nome_alunos VARCHAR(50) NOT NULL,
  disciplinas_id INT,

  CONSTRAINT FOREIGN KEY (disciplinas_id)
    REFERENCES disciplinas (id_disciplina)
  );
```

Insirir dados nas tabelas.

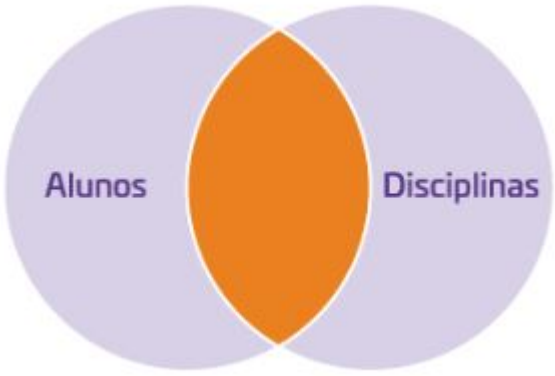
```
INSERT INTO disciplinas(nome_disciplina, nome_professor) VALUES ('Banco de dados',
'Maria Alves' );
INSERT INTO disciplinas(nome_disciplina, nome_professor) VALUES ('Python', 'Pietro
Souza' );
INSERT INTO disciplinas(nome_disciplina, nome_professor) VALUES ('POO', 'Bia
Tavares' );

-- #Dados inseridos na tabela disciplinas, onde informamos o nome da disciplina e
o nome do professor de cada disciplina.


INSERT INTO alunos(nome_alunos, disciplinas_id)
 VALUES
    ('Cleiton',2),
    ('Carol', null),
    ('Ruan', 2),
    ('Gabi',1),
    ('Rian', null),
    ('Mia', 2),
    ('Malu',1)
```

```
-- # Dados inseridos na tabela alunos, onde informamos o nome de cada aluno e o id
de cada disciplina que os alunos fazem parte. Vale destacar que alguns alunos não
fazem parte de nenhuma das disciplinas existentes.
```



8.1.2.3 - INNER JOIN

Esse método **retorna os registros que são comuns entre as tabelas**.



Assim, uma consulta realizada com o método Inner Join retornará o que as tabelas de alunos e disciplinas possuem em comum.

```
-- Para utilizar o ``Inner Join``, é **necessário** informar **os nomes das
colunas que serão utilizadas das duas tabelas**. Após o comando ``from``,
informamos o nome da primeira tabela a ser analisada e, depois do método Inner
Join, adicionamos o nome da segunda tabela a ser analisada.

SELECT nome_alunos, nome_disciplina FROM alunos
INNER JOIN disciplinas
```

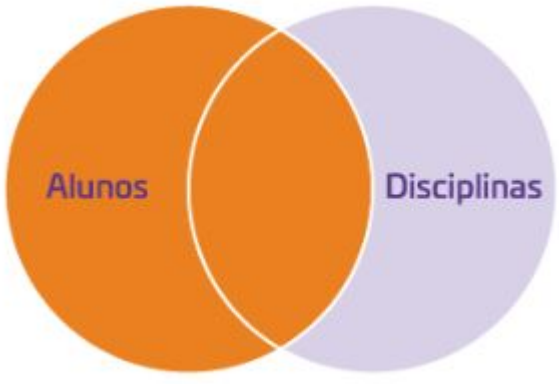
```
    ON disciplinas.id_disciplina = alunos.disciplinas_id;

-- Em seguida, fazemos a ligação entre as chaves de cada tabela. Na tabela
disciplinas, existe uma chave primária, chamada de id_disciplina, que é
referenciada na tabela alunos por meio da chave estrangeira disciplinas_id.
```

O resultado da consulta realizada no código anterior apresenta o nome dos alunos que cursam alguma disciplina. Ou seja, é a interseção entre as tabelas. Os alunos que não cursam nenhuma das disciplinas citadas não são exibidos nessa consulta.

	nome_alunos	nome_disciplina
▶	Gabi	Banco de dados
	Malu	Banco de dados
	Cleiton	Python
	Ruan	Python
	Mia	Python

8.1.2.4 - LEFT JOIN

Esse método apresenta todos os registros que estão em uma tabela, *mesmo que não tenham ligação com a outra*. **Além disso, ele apresenta os registros em comum entre as tabelas**.

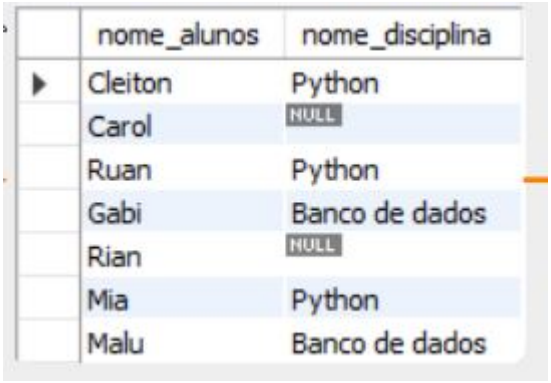


Ao realizar uma consulta com o `Left Join`, todos os dados da tabela à esquerda serão mostrados.

Com o código do `Left Join`, serão apresentados **os dados da tabela alunos** e **os dados da interseção** entre as duas tabelas.

```
SELECT nome_alunos, nome_disciplina
FROM alunos
```

```
LEFT JOIN disciplinas
    ON disciplinas.id_disciplina = alunos.disciplinas_id;
```

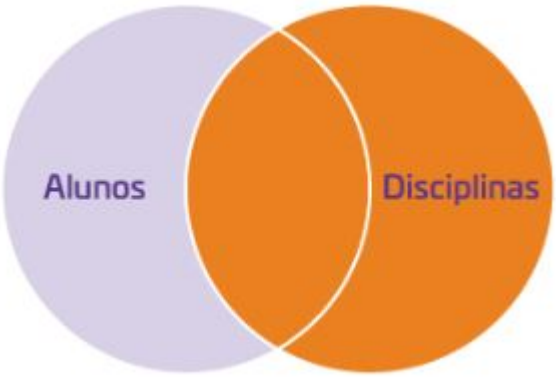


nome_alunos	nome_disciplina
Cleiton	Python
Carol	NULL
Ruan	Python
Gabi	Banco de dados
Rian	NULL
Mia	Python
Malu	Banco de dados

Independente se o aluno cursa ou não uma disciplina, todos os alunos são mostrados como resultado, pois essa é a tabela que se mantém à esquerda.

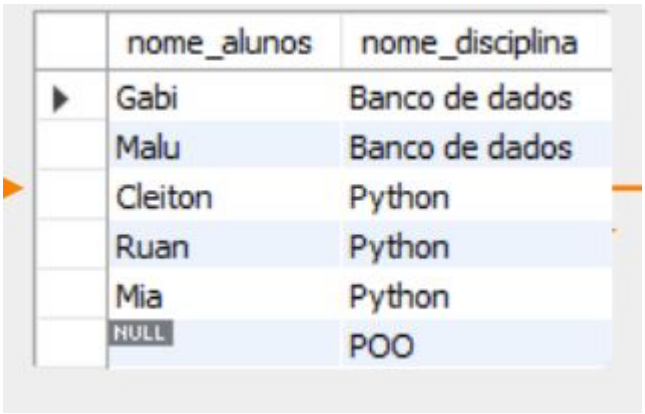
8.1.2.5 - RIGHT JOIN

Esse método apresenta todos os registros que estão em uma tabela, mesmo que não tenham ligação com a outra. Além disso, ele mostra os registros em comum entre as tabelas.



Ao realizar uma consulta com o `Right Join`, **todos os dados da tabela à direita serão mostrados**.

Assim, com o código do `Right Join`, serão apresentados os dados da tabela disciplinas e os dados da interseção entre as duas tabelas.

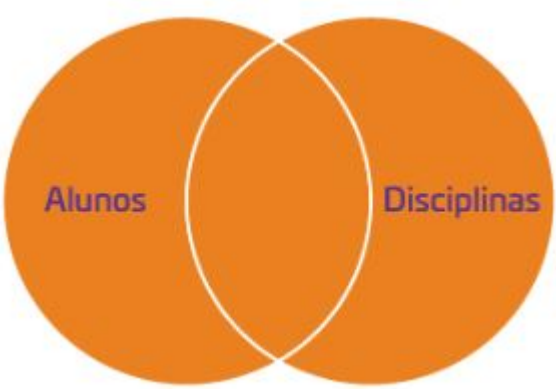


nome_alunos	nome_disciplina
Gabi	Banco de dados
Malu	Banco de dados
Cleiton	Python
Ruan	Python
Mia	Python
NULL	POO

Independente se a disciplina possui ou não alunos, todas as disciplinas são mostradas como resultado, pois essa é a tabela que se mantém à direita.

8.1.2.6 - OUTER JOIN, FULL OUTER JOIN OU FULL JOIN

Esse método **apresenta todos os registros que estão nas duas tabelas.**



Realizando uma consulta com esse método, teremos como resultado todos os dados de ambas as tabelas, **independente se o aluno cursa alguma disciplina ou se a disciplina possui alunos**.

> Atenção: Esse método pode não existir nas consultas de alguns Sistemas de Gerenciamento de Banco de Dados, como o MySQL. Assim, para apresentar os dados da tabela, precisamos informar que queremos os dados da tabela à esquerda, ou seja, da tabela que está em Left Join. Depois, juntamos os dados desta tabela com os dados da tabela Right Join. Por isso, temos o comando union, que une os dados de ambas as tabelas e os apresenta como resultado.

```
SELECT nome_alunos, nome_disciplina
FROM alunos
LEFT JOIN disciplinas
    ON disciplinas.id_disciplina = alunos.disciplinas_id

-- AQUI
UNION
-- AQUI

SELECT nome_alunos, nome_disciplina FROM alunos
```

```
INNER JOIN disciplinas
    ON disciplinas.id_disciplina = alunos.disciplinas_id;
```

Conclusão

O conceito de junção entre tabelas é fundamental para o contexto dos bancos de dados. Isso porque é por meio dos métodos de análise que podemos coletar informações específicas em um conjunto de tabelas.

Para relacionar as tabelas, podemos utilizar a ligação entre chaves estrangeiras ou efetuar uma ligação entre colunas de mesmo nome em tabelas diferentes.

> Leitura Complementar

[DEVMEDIA. SQL JOIN: Entenda como funciona o retorno dos dados.]https://www.devmedia.com.br/sql-join-entenda-como-funciona-o-retorno-dos-dados/31006

> Referência Bibliográfica

[DEVMEDIA. SQL JOIN: Entenda como funciona o retorno dos dados.]https://www.devmedia.com.br/sql-join-entenda-como-funciona-o-retorno-dos-dados/31006

[DEVMEDIA. INNER, CROSS, LEFT, RIGTH E FULL JOINS.]https://www.devmedia.com.br/inner-cross-left-rigth-e-full-joins/21016

[POSTGRESQL. 2.6. Junções entre tabelas.]http://pgdocptbr.sourceforge.net/pg82/tutorial-join.html

8.1.3 - Conceber e saber quando utilizar o Trigger para melhorar uma ação desejada no banco de dados

8.1.3.1 - Vídeo - Triggers

Trigger ou gatinho, é um evento que é automaticamente executado quando ocorre alteração no Banco de Dados. Por exemplo, ele pdoe ser acionado quando uma linha for adicionada a uma tabela, ou quando colunas selecionadas são atualizadas.

Pela Linguagem de Definição de Dados-**DDL**, Triggers são acionados por eventos que afetam a **estrutura do banco de dados**. Adicionar, Alterar ou Remover uma tabela.

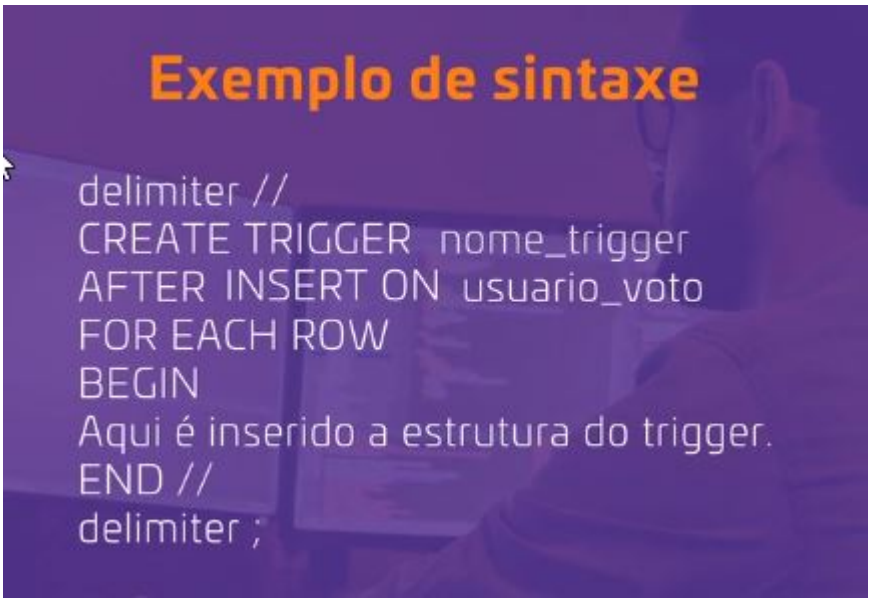
Pela Linguagem de Manipulação de Dados-**DML**, Triggers são os tipos mais comuns, pois a cada inclusão, alteração ou exclusão de linhas(dados) **das tabelas** é desencadeada os eventos dos Triggers. Linguagem DML é a responsável por ativar o trigger(nota do exercício Quiz2 #93168)

Diferentes tipos de Triggers na DML:

- **For ou after**, são executados seja inserindo, atualizando ou excluindo alguma tabela ou coluna via após a conllusão de instrução de trigguer.
    - Inset
    - Update
    - Delete

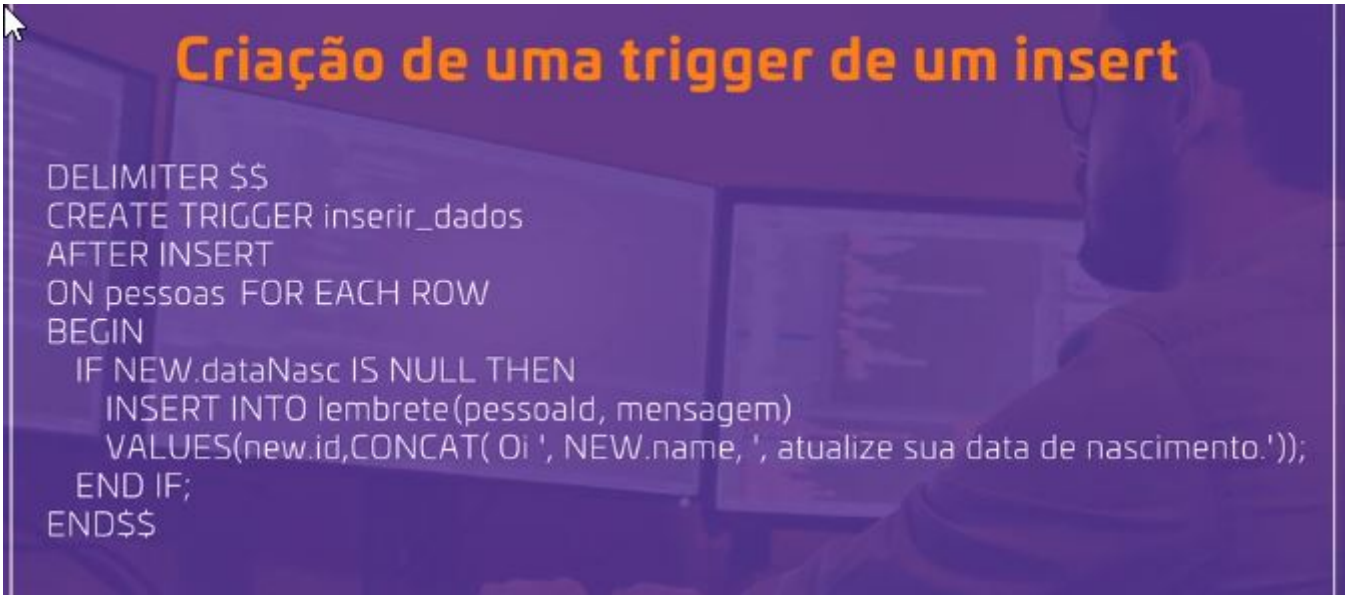
- **Instead of**, ao invés da instrução ser executada é o próprio Trigger que é executado.
    - Inset
    - Update
    - Delete

Construct Trigger, permite criar um novo gatilho que é acionado quando um evento de tabela acontece, ou seja, quando ocorre por exemplo um insert, delete ou update.



AFTER, define quando uma trigger será executada; INSERT ON, especifica o nome da tabela que será associada a trigger; FOR EACH ROW, determina que cada modificação realizada pela pela trigger seja feita em cada linha afetada

Exemplo:



`inserir_dados` é o nome da Trigger; Será comparado se na tabela `pessoas` ao inserir um registro se data de nascimento foi adicionada. Caso não seja inserida, o id dessa pessoa será adicionada na tabela `lembrete`, junto com uma mensagem.
Assim o gatilho será executado quando ao inserir uma pessoa não informarem sua data de nascimento.

Criar BD, Criar Tabelas e seus atributos e Criar Triggers.

8.1.3.2 - Exercício

1. Qual o procedimento que se deve usar para atualizar uma tabela de banco de dados, automaticamente, sempre que um produto for cadastrado?
    1. Trigger.
        1. Resposta correta!Isso mesmo! Com o trigger, a tabela sempre será atualizada de forma automática.
2. Quando um trigger pode ser ativado?
    1. Quando um procedimento DML é executado.
        1. Resposta correta!A linguagem DML é a responsável por ativar o trigger.
3. Assinale a alternativa que apresenta corretamente a sintaxe de um trigger. 1.

```
CREATE TRIGGER nome evento
ON tabela
FOR EACH ROW
BEGIN
/*corpo do código*/
END
```

1. Resposta correta!Isso mesmo! Assim, basta definir se o trigger será de insert, delete ou update. Depois, ele será executado.

8.2 - MÓDULO 02

8.2.1 - Conceitos de Function e Procedure

A rotina é composta por tarefas inevitáveis, como tirar o lixo de casa e forrar a cama.

Assim como nós, os banco de dados também tem rotinas importantes que são identificadas por um nome definido, que são: **função**, ou `function`, e **procedimento**, ou `procedure`

A **função é uma rotina armazenada.** Ela executa uma determinada ação, cujo **resultado retorna algum valor.**

Para criar uma função, é necessário entender como ela funciona, ou seja, compreender a sua sintaxe básica.

A sintaxe é :

```
CREATE FUNCTION nome_função (parâmetros)

RETURNS tipos_dados

Código_da_funcao;
```

[função]
https://imasters.com.br/banco-de-dados/voce-sabe-o-que-e-uma-function#:~:text=Uma user defined function (função,(único) ou uma tabela.

[Criar função]
http://www.bosontreinamentos.com.br/mysql/mysql-rotinas-armazenadas-funcoes-create-function-33/#:~:text=Rotinas Armazenadas – Funções (CREATE FUNCTION) em MySQL&text=Uma Rotina Armazenada é um,SQL e Lógica de Programação.

8.2.2 - Entendendo a `function`

As palavras usadas na criação de uma função são:

- `create function`, essa rotina precisa de um nome para ser referenciada. Além disso, é necessário apresentar os parâmetros que serão utilizados na função dentro dos parênteses.
- `returns`, indica o dado que retornou pela função.

```
CREATE FUNCTION teste (a DECIMAL(4,2), b INT)
 RETURNS int
 DETERMINISTIC
 return a * b;
```

> Nesta função `teste`, temos dois parâmetros: a, que é do tipo decimal, e b, que é do tipo inteiro.

> Através do `returns`, o valor do resultado será retornado e ele deve ser inteiro.

> Em seguida, declaramos se a função é `deterministic` (determinística) ou `not deterministic` (não determinística). Uma função é **determinística** *se produzir o mesmo resultado para os mesmos parâmetros de entrada*. Caso contrário, ela é não determinística. `return` informa qual é o valor da função que deve ser retornado. Nesse exemplo, será o resultado da multiplicação dos parâmetros a e b.

8.2.3 - Utilizando uma `function`

Para invocar uma função, usamos a seguinte linha de código:

```
SELECT nome_função(parâmetros);
```

No exemplo, veremos como invocar a função do exemplo anterior. Utilizaremos o `select` para selecionar a função **teste** e passaremos os valores para `a` e `b` separados por vírgula. O resultado da multiplicação ficará guardado em resultado.

```
SELECT teste (5.5, 4) AS resultado;
```

Visto que os valores são a = 5.5 e b = 4, o resultado da multiplicação será 22.

8.2.4 - Excluindo uma `function`

```
DROP FUNCTION nome_função;

--Exclui a função "teste" do exemplo anterior
DROP FUNCTION teste;
```

8.2.5 - Definindo uma `procedure`

Um procedimento, ou procedure, **é uma instrução ou um grupo de instruções organizadas para executar tarefas**. Ele pode ser usado através de uma simples invocação, que executa uma série de outros comandos.

[Procedimentos]
https://www.devmedia.com.br/stored-procedures-e-functions-no-mysql-com-phpmyadmin/30837
http://www.bosontreinamentos.com.br/mysql/mysql-procedimentos-armazenados-stored-procedures-basico-34/

> O procedimento é uma instrução para executar tarefas

A sintaxe é :

```
CREATE PROCEDURE nome_procedimento (parâmetros)

declarações;
```

As palavras usadas para criar um procedimento são `create procedure`. Ele precisa de um nome para ser referenciado. Além disso, é necessário apresentar os parâmetros que serão utilizados dentro dos parênteses. Por fim, as declarações serão definidas. Vamos observar o exemplo detalhado.

```
CREATE PROCEDURE saberPreço (alimento smallint)
```

```
SELECT concat('O valor é ', preço) AS preços

FROM tabela_alimento

WHERE id_alimento = alimento;
```

> Temos acima a criação do procedure `saberPreço`. Entre os parênteses, temos a variável `alimento`, um parâmetro do tipo `smallint`, que são números inteiros.

> O `id_alimento` indica o alimento que corresponde ao preço exibido.

> Esse procedimento `select concat` concatena duas informações. Nesse caso, ele junta o texto `O valor é` com o `preço do alimento`. Essa informação será apresentada em uma coluna chamada `preços` em que `where id_alimento = alimento;`

> Os dados de `alimentos` e `preço` estão na tabela de nome `tabela_alimento`.

Para invocar um procedimento, utilizamos o seguinte código:

```
CALL nome_procedimento(parâmetros);

CALL saberPreço (3);
```

Assim, conseguimos saber qual é o preço do alimento que se encontra na posição **3** da tabela. Nesse caso, é a **goiaba**, com o preço **4**.

8.2.6 - Excluindo uma `procedure`

```
DROP PROCEDURE nome_procedimento;

--Excluindo o procedimento do exemplo anterior
DROP PROCEDURE saberPreço;
```

8.2.7 - Diferença entre `function` e `procedure`

As principais diferenças entre eles são que a `function` **sempre retorna um valor**, **não necessita de parâmetros** de entrada e saída e **pode ser chamada dentro de um procedimento**.

Já o `procedure` **nem sempre retorna um valor**, **pois é opcional**, **necessita de parâmetros** de entrada e saída e **não pode ser chamado dentro de uma função**.

[diferenças entre function e procedure]https://codigosimples.net/2016/02/24/principais-diferencas-entre-stored-procedures-e-functions/#:~:text=Stored Procedure Procedimentos armazenados são,vez que ela é chamada.

> Leitura Complementar

[BÓSON TREINAMENTOS. MySQL - Rotinas Armazenadas - Funções (CREATE FUNCTION)] https://www.youtube.com/watch?v=mzd2W3cwohM

> Referências Bibliográficas

[DEVMEDIA. Stored Procedures e Functions no MySQL com PhpMyAdmin. [s/d].]https://www.devmedia.com.br/stored-procedures-e-functions-no-mysql-com-phpmyadmin/30837

[KINGHOST. Functions e Procedures no MySQL. [s/d]]https://king.host/wiki/artigo/functions-e-procedures-no-mysql/

[REIS, Fábio. MySQL – Rotinas Armazenadas – Funções (CREATE FUNCTION). Bóson Treinamentos,]https://www.devmedia.com.br/stored-procedures-e-functions-no-mysql-com-phpmyadmin/30837

[_____.MySQL – Procedimentos Armazenados (Stored Procedures) Básico – 34. Bóson Treinamentos, 13 fev. 2014]http://www.bosontreinamentos.com.br/mysql/mysql-procedimentos-armazenados-stored-procedures-basico-34/

[SOARES, Jhonathan. Principais diferenças entre Stored Procedures e Functions. Código Simples, 24 fev. 2016.]https://codigosimples.net/2016/02/24/principais-diferencas-entre-stored-procedures-e-functions/#:~:text=Stored Procedure Procedimentos armazenados são,vez que ela é chamada

8.2.8 - Anotações Exercícios

1. Qual é o principal papel da function?
    1. Sempre retornar um valor.
        1. Resposta correta!A function é importante quando for necessário retornar um dado e requisições como essa são frequentes. Por exemplo, quando se extrai o número de itens que foram vendidos no dia e seus nomes.
2. Para que usamos um select em uma function?
    1. Para executar uma `function` onde precisamos passar o nome da função e os parâmetros que devem ser executados.
        1. Resposta correta!Isso mesmo! Dos comandos SQL, o select é um dos mais simples. Sua utilização na function também não é complicada, basta informar o nome da função e os parâmetros.
3. Quando se usa um procedure?
    1. Quando um update, insert ou delete for recorrente numa tabela
        1. Procedure é uma das principais ferramentas utilizadas do banco de dados. Com ela, é possível otimizar o desempenho e a segurança.

8.3.1 - Entender a funcionalidade de uma Procedure e aplicá-la no dia a dia

Procedure, é um *conjunto de comandos SQL* que podem ser executados de uma só vez.

Assim como uma função, ele **armazena tarefas repetitivas**, aceitando ou não **parâmetros de entrada** para realizar tarefas **de acordo com as necessidades individuais de cada banco de dados**.

Stored procedures(Procedimentos armazenados), Podem:

- Reduzir tráfegos de rede,
- Melhorar o desempenho do banco de dados,
- Criar tarefas agendadas,
- Reduzir riscos,
- Construir rotinas de processamento.

Nos `Stored procedures`, é **possível** ESCREVER COMANDOS DE MANIPULAÇÃO DE DADOS e de DEFINIÇÃO DE DADOS. Podendo :

- Inserir dados,
- Atualizar dados,
- Mesclar dados,
- Criar dados,
- Deletar dados,
- Alterar dados.

Ajuda na lógica do negócios onde o BD está sendo projetado.

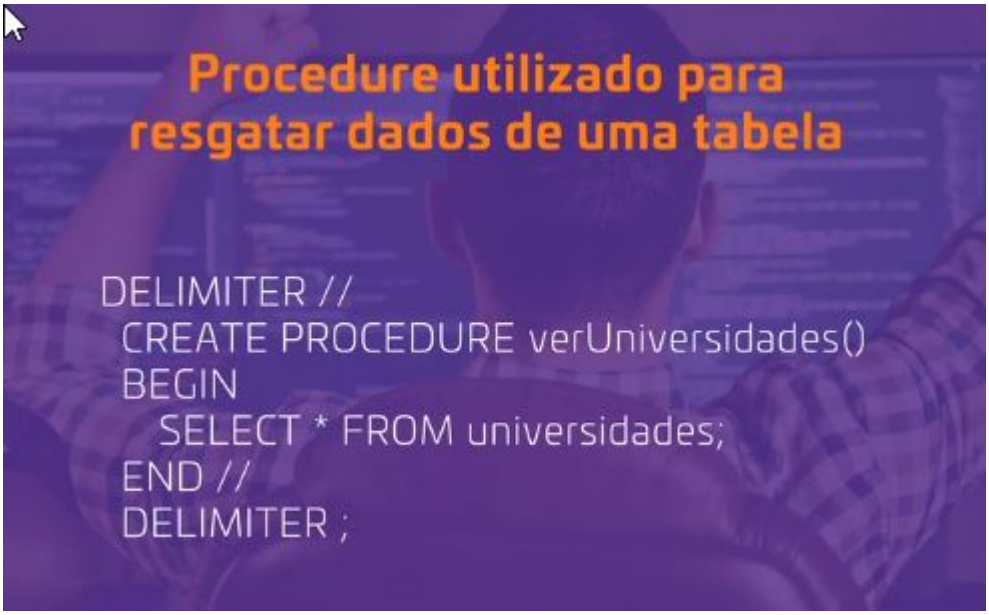
> E é de extrema importância para os administradores de Banco de Dados e Desenvolvedores

Pode-se CRIAR 5 tipos **básicos** de `Stored Procedures`:

1. Procedimentos locais;
    - Criados a partir do próprio banco de dados do usuário.
2. Procedimentos temporários;
    - Local:
        - Deve começar com `#` uma hasttag
    - Global:
    - Deve começar com `##` duas hasttag
3. Procedimentos de sistema;
    1. Realizam tarefas administrativas, podem ser executados a partir de qulquer banco de dados. identificados com a sigla `SP`, no SQL Server por exemplo.
4. Procedimentos remotos;
    1. Utilizam consultas distribuidas, aplicadas só pra compatibilidade
5. Procedimentos estendidos;
    1. Executados **fora do SGBD** e identificados com o perfixo `XP`.
        - Usados quando:
            - Temos várias aplicações em linguagens diferentes;
            - Funcionando em outras plataformas, mesmo executando a mesma função.
            - Queremos priorizar consistência e segurança;
                - Já que eles só podem executar procedimentos armazenados que executam operações específicas determinadas pelo BDA e pelo desenvolvedor.

8.3.2 - Como criar um procedure utilizando comandos SQL

Nomear; Parametros são opcionais, caso em que deve ficar vazio; entre BEGIN e AND, qual a função da criação do procedimento, TUDO que será executado.



No exemplo acima temos a `procedure` veruniversidades(), qual é responsavel em selecionar todas as informações das univerdades. `SELECT * FROM universidades;`

`DELIMITER //` `DELIMITER ;` Eles informam onde o procedimento inicia e finaliza

Para usar o procedimento criado usamos o `CALL`, seguido do nome do procedimento.

> LEMBRE-SE Ao utilizar um procedimento de armazenamento OUTROS USUÁRIOS E APLICAÇÕES NÃO CONSEGUEM TER ACESSO DE FORMA DIRETA as tabelas do banco de dados.

8.3.3 - Anotações Exercício

1. Por que usar um procedure é mais seguro?

    1. Porque as aplicações e os usuários não conseguem acessar as tabelas do banco de dados de forma direta.
        1. Resposta correta!Apenas o nome do procedure e os parâmetros são disponibilizados.

2. O que é um procedure?

    1. É uma biblioteca de comandos em SQL que são utilizados para executar uma ou mais tarefas.
        1. Resposta correta!Isso mesmo! Procedure armazena tarefas repetitivas e aceita parâmetros de entrada para que a tarefa seja efetuada de acordo com a necessidade individual.

3. Assinale a alternativa que apresenta a sintaxe correta para criar um procedure.

    1.

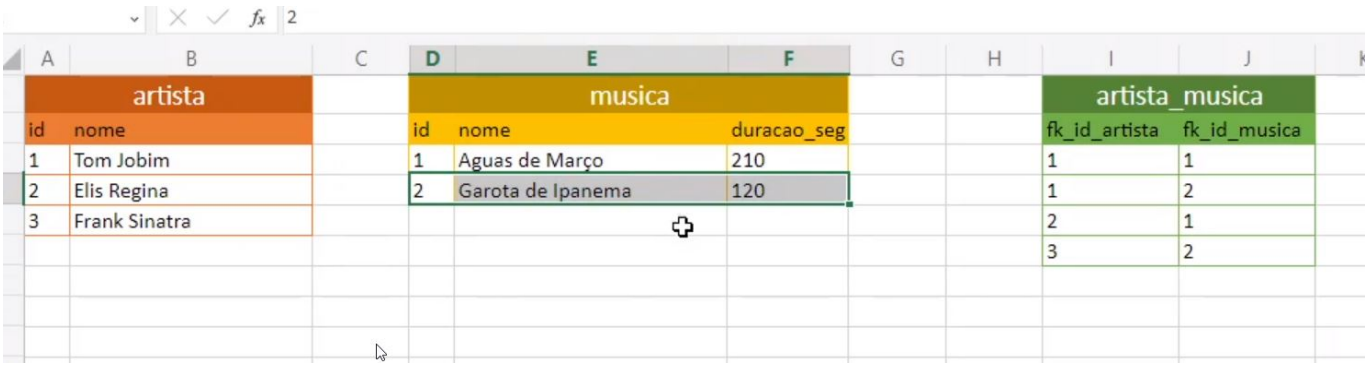
    DELIMITER $$ CREATE PROCEDURE nome_procedimento (parâmetros) BEGIN */CORPO DO PROCEDIMENTO/* END $$ DELIMITER ; ``` 1. Resposta correta!Correto! Essa é a sintaxe correta para criar um procedure.

8.4 - Relacionamento N:N em SQL - Luíz

Muitos : Muitos, tabela auxiliar leva o nome das duas tabelas, nessa tabela se guarda relacionamentos com no mínimo duas tabelas. Na tablema auxiliar deve ainda conter uma chave composta, uma vez que não se repete os dados de relação.

artista_musica	
fk_id_artista	fk_id_musica
1	1
1	2
2	1
3	2

```sql
CREATE TABLE artista_musica (
  fk_id_artista INT NOT NULL,
  fk_id_musica INT NOT NULL,
  CONSTRAINT artista_musica_pk PRIMARY KEY (fk_id_artista, fk_id_musica),
  CONSTRAINT fk_artista FOREIGN KEY (fk_id_artista) REFERENCES artista(ID),
  CONSTRAINT fk_musica FOREIGN KEY (fk_id_musica) REFERENCES musica(ID)
)
```



artista	
id	nome
1	Tom Jobim
2	Elis Regina
3	Frank Sinatra

musica		
id	nome	duracao_seg
1	Aguas de Março	210
2	Garota de Ipanema	120

artista_musica	
fk_id_artista	fk_id_musica
1	1
1	2
2	1
3	2

Controle de Estadias Hotel

Cenário 01 - Hotel

Precisamos de um sistema para gerenciar o aluguel de quartos do nosso hotel. Cada quarto tem um número, valor da diária, e vista pro mar ou pra cidade. Para fazer uma reserva, nossos clientes precisam enviar apenas o nome completo do responsável, RG e telefone de contato. Precisamos que cada aluguel gere um registro associando a pessoa com o quarto que deseja alugar. **Seria interessante**, incluir nesse registro as datas de entrada e saída, algém de ter talvez algum tipo de controle dos pagamentos (valor total, meio de pagamento) de cada estadia.

Muitos pra Muitos requerem duas tabelas se relacionando.

Tabela Quarto:

- número do quarto,
- valor da diária,
- vita pro mar ou cidade
    - se fosse mais de dois valores poderiam ser feitas mais uma tabela. Como tem apenas 2 alternativas, pode-se optar uma única coluna chamada "vista_mar"

```
CREATE TABLE quarto (
  numero INT NOT NULL PRIMARY KEY,
  valor_diaria FLOAT NOT NULL,
  vista_mar BOOLEAN NOT NULL

);
```

Tabela Clientes:

- Nome completo do responsável,
- RG
- Telefone

```
CREATE TABLE cliente (
  ID SERIAL PRIMARY KEY,
  nome VARCHAR(50) NOT NULL,
  rg VARCHAR(12) NOT NULL,
  telefone VARCHAR(12) NOT NULL


);
--  `;` DEFINE ONDE TERMINA O COMANDO
```

```
-- INSERIR VALOR
INSERT INTO quarto (numero, valor_diaria, vista_mar) VALUES
  (101, 74.99, TRUE),
  (102, 54.50, FALSE),
  (201, 110, TRUE)


INSERT INTO cliente (nome, rg, telefone) VALUES
  ('João Pereira', '12345678', '(11)98765432'),
  ('Rafael Alves', '98765432', '(11)23456789'),
```

Tabela Estadia(tabela auxiliar):

- Registro associando a pessoa ao quarto que deseja alugar,
- Data de entrada, CAMPO CRIADO APÓS A CRIAÇÃO DA TABELA
- Data de saída, CAMPO CRIADO APÓS A CRIAÇÃO DA TABELA
- Valor total da estadia, CAMPO CRIADO APÓS A CRIAÇÃO DA TABELA
- Meio de pagamento de cada estadia, CAMPO CRIADO APÓS A CRIAÇÃO DA TABELA

```
CREATE TABLE estadia (
  fk_id_quarto INT NOT NULL,
  fk_id_cliente INT NOT NULL,
  --SE GERARMOS UMA CHAVE COMPOSTA DO ID_QUARTO E ID_CLIENTE, ficará impossivil um mesmo cliente ficar no mesmo quarto em outro período. Pois não aceitará a mesma chave. Por exemplo em Janeiro Wat ficou no quarto 01, gerou a chave composta única Wat01, se em dezembro ele ficar no mesmo quarto dará erro, pois o banco não aceitará novamente o Wat01. Por isso devemos criar a chave abaixo
  ID INT SERIAL PRIMARY KEY,

  CONSTRAINT fk_quarto FOREIGN KEY (fk_id_quarto) REFERENCES quarto(numero),

  CONSTRAINT fk_cliente FOREIGN KEY (fk_id_cliente) REFERENCES cliente(ID)
);
--  `;` DEFINE ONDE TERMINA O COMANDO
```

```
INSERT INTO estadia (fk_id_quarto, fk_id_cliente) VALUES
  (101, 1),
  (102, 2),
  (102, 1),
  (201, 2)
```

id	fk_id_quarto	fk_id_cliente
9	101	1
10	102	2
11	201	1
12	102	2

```
ALTER TABLE estadia
  ADD COLUMN data_entrada VARCHAR(48),
  ADD COLUMN data_saida, VARCHAR(48),
  -- COLUNAS DATA DEVERIAM SER DATE, PORÉM PRO EXEMPLO FICAR SIMPLES SE USOU VARCHAR
  ADD COLUMN valor_total, FLOAT(48),
  -- AQUI PODERIA TER SIDO CRIADO UMA FUNÇÃO QUE GERASSE O VALOR TOTAL AUTOMÁTICO USANDO DATA DE ENTRADA, DATA SAÍDA E VALOR DIÁRIA.
```

```
  ADD COLUMN fk_id_meio_pagamento INT,
  ADD CONSTRAINT fk_meio_pagamento FOREIGN KEY (fk_id_meio_pagamento)
    REFERENCES meio_pagamento(ID);
```

```
CREATE TABLE meio_pagamento (
  id SERIAL PRIMARY KEY,
  nome VARCHAR(18) NOT NULL
);

INSERT INTO meio_pagamento (nome) VALUES
  ('dinheiro'),
  ('cartão de crédito'),
  ('cartão de débito')
```

```
--UPDATE AS TABELAS

UPDATE estadia
  SET data_entrada = '2022-08-26t08:54:00'
  --PADRÃO DE DATA VIA TEXTO CONFORME NORMA ISO
  WHERE ID = 9;


UPDATE estadia
  SET
    data_saida = '2022-08-26t08:54:00',
    valor_total = 148.99,
    fk_id_pagamento = 1,
  WHERE ID = 9;


UPDATE estadia
  SET
    data_entrada = '2022-08-26T08:54:00',
    data_saida = '2022-08-30T08:54:00',
    valor_total = 750.50,
    fk_id_pagamento = 2,
  WHERE ID = 10;
```