

SQL

Professor Aguinaldo Neto

SQL - STRUCTURED QUERY LANGUAGE

- SQL é uma linguagem de consulta que implementa as operações da álgebra relacional de forma bem amigável;
- É uma linguagem para acesso e manipulação de BD relacionais;
- Padronizada pela ANSI (American National Standards Institute);
- Os comandos em SQL são divididos em
 - DDL (Data Definition Language) – Criação do esquema do BD;
 - DML (Data Manipulation Language) – Manipulação de Dados;
 - DCL (Data Control Language) – Controle de acesso e segurança.
- SQL permite definir estrutura de dados e restrições de integridade, modificar dados no BD, especificar restrições de segurança e controle de transações e utilizar linguagens hospedeiras;

SQL - STRUCTURED QUERY LANGUAGE

- Desenvolvida por pesquisadores da IBM – SYSTEM R;
- Inicialmente chamada de SEQUEL (Structured English Query Language);
- ISO e ANSI lançam em 1986 a primeira versão do padrão da linguagem SQL, o SQL-86.
 - Versão padrão SQL1 (ANSI SQL)
- Em 1992, houve uma revisão e expansão do padrão, gerando a SQL2 (SQL92);
 - Um esquema é definido por um nome e inclui um identificador de autorização para indicar o usuário que é o dono do esquema. CREATE SCHEMA PF AUTHORIZATION agente_silva;
 - Esquema inclui tabelas, restrições, visões e domínios.
- Em 1999, SQL3 trouxe características de BDOO (SQL99)
 - Novos tipos de dados (CLOB), predicados, tipos abstratos de dados;
- SQL:2003, SQL:2006, SQL:2008, SQL:2011 e SQL:2016.

SQL - STRUCTURED QUERY LANGUAGE

LINGUAGEM ESTRUTURADA DE CONSULTA
TIPOS DE LINGUAGEM



DML - LINGUAGEM DE MANIPULAÇÃO DE DADOS

- Existem 4 comandos principais para a manipulação de dados em SQL:

PRINCIPAIS COMANDOS DML

- SELECT – RECUPERAÇÃO/BUSCA DE DADOS
- INSERT – INSERÇÃO DE DADOS
- UPDATE – ATUALIZAÇÃO DE DADOS
- DELETE – EXCLUSÃO DE DADOS

DML – COMANDO SELECT

■ Forma básica:

1. **SELECT** A_1, A_2, \dots, A_n (atributos)
2. **FROM** R_1, R_2, \dots, R_m , (relações)
3. **WHERE** condições

DML – COMANDO SELECT

SELECT Coluna, Coluna, ..., Coluna **FROM** Tabela **WHERE** Condição **ORDER BY** Coluna ASC/DESC

SELECT

Quais colunas se deseja selecionar? (* = todas)

FROM

De qual tabela os dados virão?

WHERE

Condição: like, =, <>, >, >=, <, <=, in, between, and, or, not

ORDER BY

Ordenação: ASC (ascendente); DESC (descendente)

DML – COMANDO SELECT

Function			
Codigo	Nome	Salario	Setor
1	Tadeu	1.500,00	1
2	Ylane	1.200,00	2
3	Julian	1.000,00	1
4	Ewerton	1.000,00	1
5	João	800,00	2
6	Geraldo	1.500,00	3
7	Maria	500,00	

SELECT Codigo, Nome **FROM** Function **WHERE** Setor = 1

Codigo	Nome
1	Tadeu
3	Julian
4	Ewerton

SELECT Codigo, Nome **FROM** Function **WHERE** Setor = 1 **AND** Salario = 1.000,00

Codigo	Nome
3	Julian
4	Ewerton

DML – OPERADORES

- Os operadores podem ser utilizados em conjunto na cláusula **WHERE** para filtrar o resultado da consulta.
- Os operadores podem ser:
 - Relacionais
 - Lógicos
 - Especiais

DML – OPERADORES RELACIONAIS

- São utilizados para realizar comparações entre valores.

Operador	Significado	Exemplo
=	Igual	<code>select * from produto where codigo = 10</code>
<	Menor que	<code>select * from produto where qtde < 5</code>
<=	Menor ou igual a	<code>select * from produto where preco <= 50</code>
>	Maior que	<code>select * from produto where preco > 500</code>
>=	Maior ou igual a	<code>select * from produto where preco >= 500</code>
<>	Diferente	<code>select * from produto where codigo <> 2</code>

DML – OPERADORES LÓGICOS

- São utilizados para realizar operações que tenham um resultado booleano (verdadeiro/falso).

Operador	Significado	Exemplo
AND	E	<code>select * from produto where marca = 'LG' and preco > 1500</code>
OR	OU	<code>select * from produto where qtde < 5 or qtde > 100</code>
NOT	NEGAÇÃO	<code>select * from produto where preco is not null</code>

DML – OPERADORES ESPECIAIS

Operador	Significado	Exemplo
IS NULL ou IS NOT NULL	Testa se o valor é nulo ou não nulo	<code>select * from produto where preco is null</code>
BETWEEN	Delimita um intervalo de valores para a consulta	<code>select * from produto where preco between 10 and 100</code>
LIKE	Define um padrão para uma cadeia de caracteres	<code>select * from produto where nome like '_A%'</code>
IN	Compara o valor de uma coluna com um conjunto	<code>Select * from produto Where codigo in (2, 5, 15, 29)</code>
DISTINCT	Elimina duplicidades	<code>Select distinct categoria from produto</code>

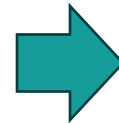
LINGUAGEM SQL - COMANDOS

Produto

Nome	Categoria	Preço	Unidade
Café	Mercearia	R\$ 8,99	KG
Açúcar	Mercearia	R\$ 10,20	5 KG
Sabão em Pó	Limpeza	R\$ 9,90	KG
Vinho	Bebida	R\$ 59,90	750 ML
Refrigerante	Bebida	R\$ 7,90	2 L



```
1. SELECT NOME, PREÇO  
2. FROM PRODUTO  
3. WHERE categoria='BEBIDA'
```



Nome	Preço
Vinho	R\$ 59,90
Refrigerante	R\$ 7,90

LINGUAGEM SQL – (Where)

Produto

Nome	Categoria	Preço	Unidade
Café	Mercearia	R\$ 8,99	KG
Açúcar	Mercearia	R\$ 10,20	5 KG
Sabão em Pó	Limpeza	R\$ 9,90	KG
Vinho	Bebida	R\$ 59,90	750 ML
Refrigerante	Bebida	R\$ 7,90	2 L



```
1. SELECT *  
2. FROM PRODUTO  
3. WHERE PRECO > 10
```



Nome	Categoria	Preço	Unidade
Açúcar	Mercearia	R\$ 10,20	5 KG
Vinho	Bebida	R\$ 59,90	750 ML

LINGUAGEM SQL – (Order By)

Produto

Nome	Categoria	Preço	Unidade
Café	Mercearia	R\$ 8,99	KG
Açúcar	Mercearia	R\$ 10,20	5 KG
Sabão em Pó	Limpeza	R\$ 9,90	KG
Vinho	Bebida	R\$ 59,90	750 ML
Refrigerante	Bebida	R\$ 7,90	2 L



```
1. SELECT *  
2. FROM PRODUTO  
3. ORDER BY NOME ASC
```



Nome	Categoria	Preço	Unidade
Açúcar	Mercearia	R\$ 10,20	5 KG
Café	Mercearia	R\$ 8,99	KG
Refrigerante	Bebida	R\$ 7,90	2 L
Sabão em Pó	Limpeza	R\$ 9,90	KG
Vinho	Bebida	R\$ 59,90	750 ML

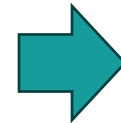
LINGUAGEM SQL – (Distinct)

Produto

Nome	Categoria	Preço	Unidade
Café	Mercearia	R\$ 8,99	KG
Açúcar	Mercearia	R\$ 10,20	5 KG
Sabão em Pó	Limpeza	R\$ 9,90	KG
Vinho	Bebida	R\$ 59,90	750 ML
Refrigerante	Bebida	R\$ 7,90	2 L



```
1. SELECT DISTINCT CATEGORIA  
2. FROM PRODUTO
```



Categoria
Mercearia
Limpeza
Bebida

LINGUAGEM SQL – (Like)

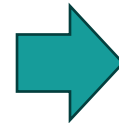
Produto

Nome	Categoria	Preço	Unidade
Café	Mercearia	R\$ 8,99	KG
Açúcar	Mercearia	R\$ 10,20	5 KG
Sabão em Pó	Limpeza	R\$ 9,90	KG
Vinho	Bebida	R\$ 59,90	750 ML
Refrigerante	Bebida	R\$ 7,90	2 L

LIKE - Padrão de strings
% - Qualquer sequência de caracteres
_ - Único caractere



```
1. SELECT *  
2. FROM PRODUTO  
3. WHERE NOME LIKE '_A%'
```



Nome	Categoria	Preço	Unidade
Café	Mercearia	R\$ 8,99	KG
Sabão em Pó	Limpeza	R\$ 9,90	KG

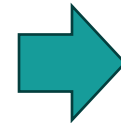
LINGUAGEM SQL – (Between)

Produto

Nome	Categoria	Preço	Unidade
Café	Mercearia	R\$ 8,99	KG
Açúcar	Mercearia	R\$ 10,20	5 KG
Sabão em Pó	Limpeza	R\$ 9,90	KG
Vinho	Bebida	R\$ 59,90	750 ML
Refrigerante	Bebida	R\$ 7,90	2 L



```
1. SELECT *  
2. FROM PRODUTO  
3. WHERE (PRECO BETWEEN 10 AND 100)
```



Nome	Categoria	Preço	Unidade
Açúcar	Mercearia	R\$ 10,20	5 KG
Vinho	Bebida	R\$ 59,90	750 KG

LINGUAGEM SQL – (IN)

Selecione o nome e o preço dos produtos da categoria bebida ou limpeza.

Produto

Nome	Categoria	Preço	Unidade
Café	Mercearia	R\$ 8,99	KG
Açúcar	Mercearia	R\$ 10,20	5 KG
Sabão em Pó	Limpeza	R\$ 9,90	KG
Vinho	Bebida	R\$ 59,90	750 ML
Refrigerante	Bebida	R\$ 7,90	2 L

Categoria

Id_categoria	Nome_categoria
10	Mercearia
20	Limpeza
30	Bebida
40	Açougue



```
1. SELECT nome_prod, preco_prod FROM Produto
2. WHERE id_categoria IN (SELECT id_categoria FROM categoria
3.                         WHERE nome_categoria = 'BEBIDA'
4.                         OR nome_categoria = 'LIMPEZA');
```



Nome_prod	Preco_prod
Sabão em Pó	9,90
Vinho	59,90
Refrigerante	7,90

FUNÇÕES DE AGREGAÇÃO

- São funções que permite agrupar valores:

COUNT – Conta a quantidade de linhas

AVG – realiza a média aritmética

SUM – realiza a soma dos valores da coluna

MAX – retorna o maior valor da coluna

MIN – retorna o menor valor da coluna.

LINGUAGEM SQL – (Agregação - Count)

Produto

Nome	Categoria	Preço	Unidade
Café	Mercearia	R\$ 8,99	KG
Açúcar	Mercearia	R\$ 10,20	5 KG
Sabão em Pó	Limpeza	R\$ 9,90	KG
Vinho	Bebida	R\$ 59,90	750 ML
Refrigerante	Bebida	R\$ 7,90	2 L

Selecione a quantidade de produtos existentes.



```
1. SELECT count(*) as Total FROM Produto
```



Total
5

LINGUAGEM SQL – (Agregação – Avg, Sum, Max, Min)

Produto

Nome	Categoria	Preço
Café	Mercearia	R\$ 8,99
Açúcar	Mercearia	R\$ 10,20
Sabão em Pó	Limpeza	R\$ 9,90
Vinho	Bebida	R\$ 59,90
Refrigerante	Bebida	R\$ 7,90
Detergente	Limpeza	R\$ 2,90

Selecione a média, a soma, o maior e o menor preço de produto por categoria.



```
1. SELECT categoria, AVG(PRECO) as Média, SUM(PRECO) AS Soma,  
2. MAX(Preco) as Maior, MIN(Preco) as Menor  
3. FROM produto  
4. GROUP BY categoria;
```



Categoria	Média	Soma	Maior	Menor
Bebida	33,90	67,80	59,90	7,90
Limpeza	6,40	12,80	9,90	7,90
Mercearia	9,59	19,19	10,20	8,99

LINGUAGEM SQL – (Agregação - Having)

Produto

Nome	Categoria	Preço
Café	Mercearia	R\$ 8,99
Açúcar	Mercearia	R\$ 10,20
Sabão em Pó	Limpeza	R\$ 9,90
Vinho	Bebida	R\$ 59,90
Refrigerante	Bebida	R\$ 7,90
Detergente	Limpeza	R\$ 2,90

Selecione a média de preço de produto por categoria apenas para médias inferiores a R\$ 10,00




```
1. SELECT categoria, AVG(PRECO) as Média
2. FROM produto
3. GROUP BY categoria
4. HAVING AVG(PRECO) < 10;
```



Categoria	Média
Mercearia	R\$ 9,59
Limpeza	R\$ 6,40

LINGUAGEM SQL – (Insert)

```
1. INSERT INTO nome_tabela  
2. VALUES (V1,V2, ..., Vn)
```



■ Ordem dos atributos deve ser mantida.


Produto (ID, NOME, PRECO, CATEG)

Exemplo:

INSERT INTO PRODUTO

VALUES (1, 'TV LCD 50', 3.500, 'ELETRÔNICOS')

```
1. INSERT INTO nome_tabela (A1, A2, ..., An)  
2. VALUES (V1,V2, ..., Vn)
```




■ Ordem dos atributos não precisa ser mantida.

INSERT INTO PRODUTO (ID, CATEG, NOME)

VALUES (1, 'ELETRÔNICOS' , 'TV LCD 50');

```
1. INSERT INTO nome_tabela  
2. SELECT  
3. FROM TABELA  
4. WHERE (V1,V2, ..., Vn)
```



■ Tupla sresultantes da cláusula SELECT serão inseridas na tabela.

INSERT INTO PRODUTO

SELECT * FROM PRODUTO2

WHERE CATEG='HIGIENE';

LINGUAGEM SQL – (Delete)

```
1. DELETE FROM nome_tabela  
2. WHERE PREDICADO
```



■ Produto (ID, NOME, PRECO, CATEG).

Excluir todos os produtos que estão sem preço:

Exemplo:

DELETE FROM PRODUTO WHERE PRECO IS NULL;

■ Excluir todos os produtos

DELETE FROM PRODUTO;

LINGUAGEM SQL – (Update)

Produto (ID, NOME, PRECO, CATEG)

```
1. UPDATE nome_tabela
2. SET coluna1=valor1,
3.     coluna2=valor2,
4.     ...
5.     colunaN=valorN
6. WHERE PREDICADO
```



- Aplicar um reajuste de 10% em todos os produtos:
Exemplo:

```
UPDATE PRODUTO
SET PRECO = PRECO * 1.1;
```

- Aplicar um reajuste de 10% em todos os produtos nos
produto de limpeza

```
UPDATE PRODUTO
SET PRECO = PRECO * 1.1
WHERE CATEGORIA = 'LIMPEZA';
```

LINGUAGEM SQL – (Inner Join)

Produto

Nome_Prod	ID_Categoria	Preco_Prod	Und_Prod
Café	10*	R\$ 8,99	KG
Açúcar	10	R\$ 10,20	5 KG
Sabão em Pó	20	R\$ 9,90	KG
Vinho	30	R\$ 59,99	750 ML
Refrigerante		R\$ 7,90	2 L

Selecione o nome dos produtos e das respectivas categorias

```
SELECT p.nome_prod as produto,  
       c.nome_categoria as categoria  
FROM produto as p INNER JOIN categoria as  
       c ON (p.id_categoria = c.id_categoria);
```

Categoria

Id_Categoria	Nome_Categoria
10	Mercearia
20	Limpeza
30	Bebida
40	Açougue

Produto	Categoria
Café	Mercearia*
Açúcar	Mercearia
Sabão em Pó	Limpeza
Vinho	Bebida

LINGUAGEM SQL – (Left Outer Join)

Produto

Nome_Prod	ID_Categoria	Preco_Prod	Und_Prod
Café	10*	R\$ 8,99	KG
Açúcar	10	R\$ 10,20	5 KG
Sabão em Pó	20	R\$ 9,90	KG
Vinho	30	R\$ 59,99	750 ML
Refrigerante		R\$ 7,90	2 L

Categoria

Id_Categoria	Nome_Categoria
10	Mercearia
20	Limpeza
30	Bebida
40	Açougue

```
1. SELECT p.nome_prod as produto,  
2.      c.nome_categoria as categoria  
3. FROM produto as p LEFT OUTER JOIN categoria as c  
4. ON (p.id_categoria = c.id_categoria);
```

Produto	Categoria
Café	Mercearia
Açúcar	Mercearia
Sabão em Pó	Limpeza
Vinho	Bebida
Refrigerante	<i>null</i>

LINGUAGEM SQL – (Right Outer Join)

Produto

Nome_Prod	ID_Categoria	Preco_Prod	Und_Prod
Café	10*	R\$ 8,99	KG
Açúcar	10	R\$ 10,20	5 KG
Sabão em Pó	20	R\$ 9,90	KG
Vinho	30	R\$ 59,99	750 ML
Refrigerante		R\$ 7,90	2 L

Categoria

Id_Categoria	Nome_Categoria
10	Mercearia
20	Limpeza
30	Bebida
40	Açougue

```
1. SELECT p.nome_prod as produto,  
2.      c.nome_categoria as categoria  
3. FROM produto as p RIGHT OUTER JOIN  
      categoria as c ON (p.id_categoria =  
      c.id_categoria);
```

Produto	Categoria
Café	Mercearia
Açúcar	Mercearia
Sabão em Pó	Limpeza
Vinho	Bebida
<i>null</i>	Açougue

LINGUAGEM SQL – (Full Outer Join)

Produto

Nome_Prod	ID_Categoria	Preco_Prod	Und_Prod
Café	10*	R\$ 8,99	KG
Açúcar	10	R\$ 10,20	5 KG
Sabão em Pó	20	R\$ 9,90	KG
Vinho	30	R\$ 59,99	750 ML
Refrigerante		R\$ 7,90	2 L

Categoria

Id_Categoria	Nome_Categoria
10	Mercearia
20	Limpeza
30	Bebida
40	Açougue

```
1. SELECT p.nome_prod as produto,  
2.      c.nome_categoria as categoria  
3. FROM produto as p FULL OUTER JOIN categoria as c  
4. ON (p.id_categoria = c.id_categoria);
```

Produto	Categoria
Café	Mercearia
Açúcar	Mercearia
Sabão em Pó	Limpeza
Vinho	Bebida
<i>null</i>	Açougue
Refrigerante	<i>null</i>

LANGUAGE SQL – (JOIN)

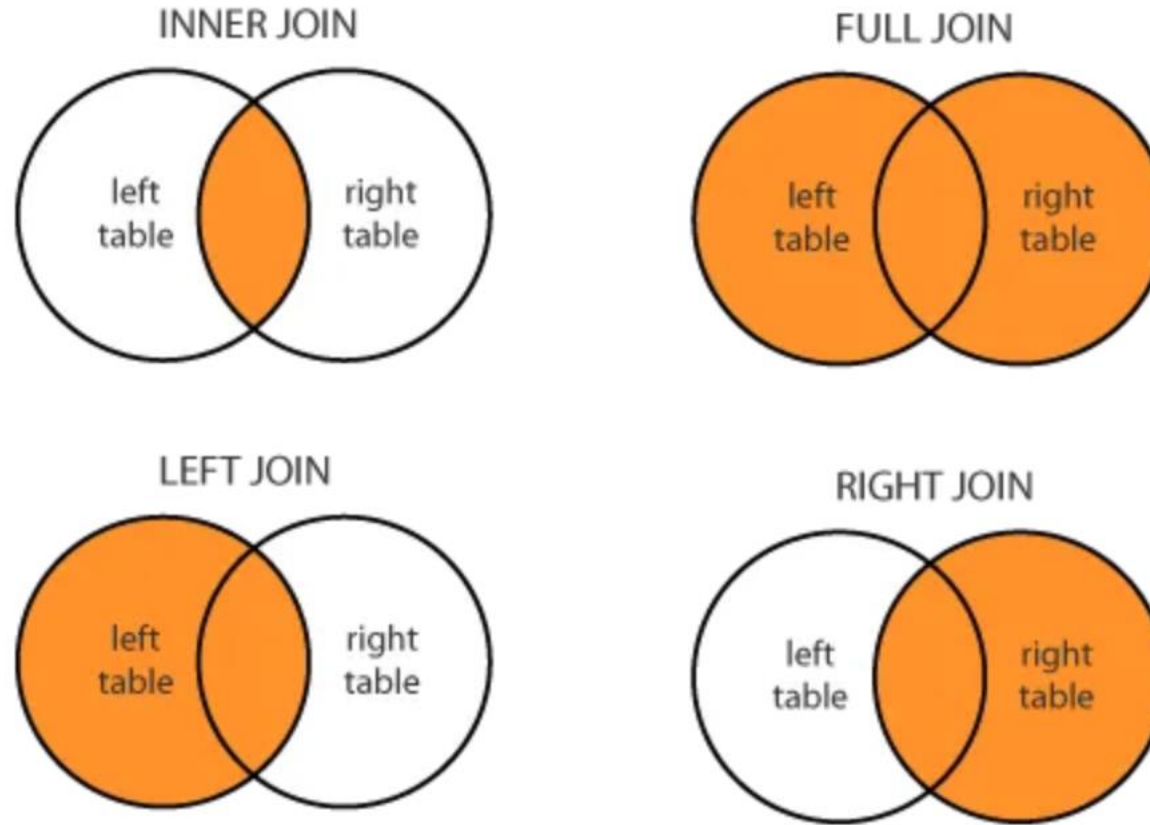


IMAGEM de <https://www.csestack.org/sql-join/>

LINGUAGEM SQL – (AII)

Empregado

ID_Emp	Nome	Depto	Idade
1	Joao	TI	25
2	Maria	FIN	38
3	Marcos	FIN	44
4	Renata	JUR	27
5	Joana	JUR	22
6	Juliana	TI	24
7	Marcelo	TI	28
8	Artur	ENG	35
9	Gustavo	ENG	37
10	Marta	ENG	39

Selecione os empregados que são mais velhos que todos empregados do departamento de Engenharia

```
1. SELECT * FROM Empregado
2. WHERE idade > ALL (SELECT idade FROM Empregado
3.                      WHERE depto = 'ENG');
```



Id_Emp	Nome	Depto	Idade
3	Marcos	FIN	44

LINGUAGEM SQL – (Exists)

Empregado

CPF	Nome	Depto
111	Joao	TI
222	Maria	FIN
333	Marcos	FIN
444	Renata	JUR
555	Joana	JUR

Dependente

CPF	Nome
111	Joao Jr.
111	Joao Neto
333	Marcos Jr
444	Renato

Selecione os empregados que não possuem dependente.

```
SELECT * FROM Empregado E
WHERE NOT EXISTS (SELECT * FROM Dependente D
                  WHERE E.CPF = D.CPF);
```



CPF	Nome	Depto
222	Maria	FIN
555	Joana	JUR

LINGUAGEM SQL – (Union)

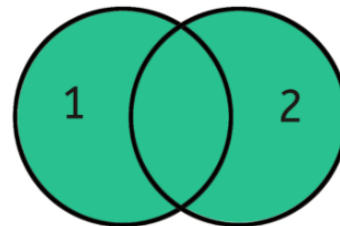
Servidor

Id	Nome	Depto
1	Joao	TI
2	Maria	FIN
3	Marcos	FIN
4	Renata	JUR
5	Joana	JUR

Terceirizado

Id	Nome	Depto
1	Vinicius	TI
2	Maria	FIN
3	Miguel	FIN
4	Andre	JUR

```
1. SELECT Nome, Depto FROM Servidor UNION  
2. SELECT Nome, Depto FROM Terceirizado;
```



Nome	Depto
Joao	TI
Maria	FIN
Marcos	FIN
Renata	JUR
Joana	JUR
Vinicius	TI
Miguel	FIN
Andre	JUR

LINGUAGEM SQL – (Union All)

Servidor

Id	Nome	Depto
1	Joao	TI
2	Maria	FIN
3	Marcos	FIN
4	Renata	JUR
5	Joana	JUR

Terceirizado

Id	Nome	Depto
1	Vinicius	TI
2	Maria	FIN
3	Miguel	FIN
4	Andre	JUR

```
1 SELECT Nome, Depto FROM Servidor UNION ALL
2 SELECT Nome, Depto FROM Terceirizado;
```



Nome	Depto
Joao	TI
Maria	FIN
Marcos	FIN
Renata	JUR
Joana	JUR
Maria	FIN
Vinicius	TI
Miguel	FIN
Andre	JUR

LINGUAGEM SQL – (Intersect)

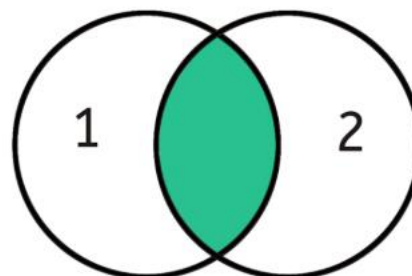
Conta

AG	Num	Nome*
10	123	Joao
10	245	Maria
50	456	Rafael
50	789	Julio
30	999	Renata

Emprestimo

Id	Nome*	Valor
1	Maria	5.000,00
2	Rafael	3.500,00
3	Julio	2.000,00
4	Antonio	2.500,00

```
1 SELECT Nome FROM Conta INTERSECT  
2 SELECT Nome FROM Emprestimo;
```



Nome
Maria
Rafael
Julio

LINGUAGEM SQL – (Except)

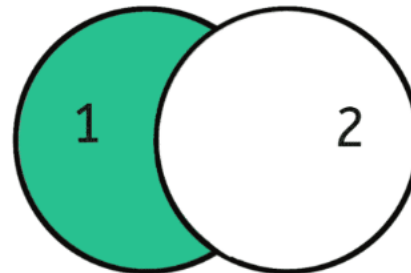
Conta

AG	Num	Nome*
10	123	Joao
10	245	Maria
50	456	Rafael
50	789	Julio
30	999	Renata

Emprestimo

Id	Nome*	Valor
1	Maria	5.000,00
2	Rafael	3.500,00
3	Julio	2.000,00
4	Antonio	2.500,00

```
1. SELECT Nome FROM Conta EXCEPT
2. SELECT Nome FROM Emprestimo;
```



Nome
Joao
Renata

DDL - LINGUAGEM DE DEFINIÇÃO DE DADOS

PRINCIPAIS COMANDOS DDL

- CREATE – CRIAÇÃO DE ESTRUTURAS E ELEMENTOS
- ALTER – ALTERAÇÃO DE ESTRUTURAS E ELEMENTOS
- DROP – EXCLUSÃO DE ESTRUTURA E ELEMENTOS
- TRUNCATE – REDEFINIÇÃO DE TABELAS

DDL - LINGUAGEM DE DEFINIÇÃO DE DADOS

■ [Create | Drop | Alter] Table

■ A linguagem DDL permite especificar um conjunto de relações, bem como o esquema, o domínio de valores, as restrições de integridade e o conjunto de índices de cada relação.

■ O comando **CREATE TABLE** é usado para especificar uma nova relação (tabela), incluindo seus atributos e restrições.

■ Para remover uma relação de um banco de dados, SQL usa o comando **DROP TABLE**;

■ O comando **ALTER TABLE** é usado para adicionar ou remover atributos e restrições de uma relação, e para alterar a definição de um atributo.

DDL – Create Table

Constraints

```
1. CREATE TABLE Empregado
2. ( Nome VARCHAR(15) NOT NULL,
3.   CodEmpregado CHAR(9) NOT NULL,
4.   DataNascimento DATE,
5.   Endereco VARCHAR(30),
6.   Sexo CHAR,
7.   Salario DECIMAL(10,2),
8.   CodSupervisor CHAR(9),
9.   CodDeppto INT NOT NULL DEFAULT 0,
10.  CONSTRAINT PK_Emp PRIMARY KEY (CodEmpregado),
11.  CONSTRAINT FK_NumSup FOREIGN KEY (CodSupervisor)
12.  REFERENCES Empregado (CodEmpregado),
13.  CONSTRAINT FK_EmpDep FOREIGN KEY (CodDeppto)
14.  REFERENCES Departamento (NumDeppto) ON DELETE CASCADE
15. );
```

Annotations:

- CREATE TABLE Empregado** → **Tabela**
- CodEmpregado CHAR(9) NOT NULL** → **Restrição de Atributo**
- Endereco VARCHAR(30)** → **Tipo de Dado**

DDL – Alter Table e Drop Table

■ Exemplo Alter:

1. **ALTER TABLE Produto ADD Marca VARCHAR(30);**
2. **ALTER TABLE Produto ALTER Preco SET DEFAULT 0;**

■ Exemplo Drop:

1. **DROP TABLE Produto;**

DCL – LINGUAGEM DE CONTROLE DE DADOS

PRINCIPAIS COMANDOS DCL

- GRANT – CONCEDE PERMISSÃO
- REVOKE– REVOGA/RETIRA PERMISSÃO
- ALTER PASSWORD – ALTERA SENHA

DCL – GRANT

O comando GRANT no MySQL é utilizado para conceder permissões a usuários para acessar e manipular bancos de dados.

```
CREATE USER 'neto'@'localhost' IDENTIFIED BY '12345';
```

```
GRANT ALL PRIVILEGES ON *.* TO 'neto'@'localhost';
```

```
FLUSH PRIVILEGES;
```

Após conceder permissões com GRANT, é importante rodar:

```
FLUSH PRIVILEGES;
```

DCL – GRANT

Inserir e atualizar dados (INSERT, UPDATE):

```
GRANT INSERT, UPDATE ON db_fametro.* TO 'neto'@'localhost';
```

Somente leitura (SELECT):

```
GRANT SELECT ON db_fametro.* TO 'neto'@'localhost';
```

Permissões em tabelas específicas:

```
GRANT SELECT, INSERT ON db_fametro.tb_aluno TO 'neto'@'localhost';
```

DCL – REVOKE

Para revogar permissões, você pode usar o comando REVOKE:

```
REVOKE ALL PRIVILEGES ON db_fametro.* FROM 'neto'@'localhost';
```

DCL – ALTER PASSWORD

Alterar a senha do usuário:

```
ALTER USER 'neto'@'localhost' IDENTIFIED BY 'nova_senha';
```

```
FLUSH PRIVILEGES;
```

DTL – LINGUAGEM DE TRANSAÇÃO DE DADOS

PRINCIPAIS COMANDOS DTL

- START TRANSACTION – INICIA TRANSAÇÃO
- COMMIT (TRANSACTION/WORD) – CONFIRMA TRANSAÇÃO
- ROLLBACK – DESFAZ TRANSAÇÃO

CHAVES



Elementos usados na construção de relacionamentos entre tabelas

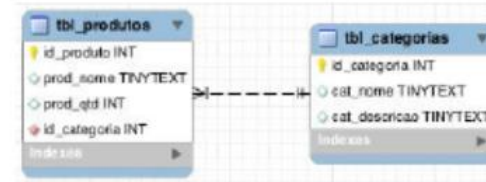


Aplicam restrições aos dados



Impõem integridade referencial

TIPOS DE CHAVES



**CHAVE
PRIMÁRIA
PK**

Atributos que identificam unicamente os registros de uma tabela.

SIMPLES X COMPOSTA

**CHAVE
ESTRANGEIRA
FK**

Identifica os registros da tabela de origem (T1) na tabela de destino (T2).

$FK(T2) = PK(T1)$

CHAVE PRIMÁRIA

*CPF	NOME	GÊNERO	SETOR	CARGO
549.539.155-49	Ricardo	Masculino	Informática	Técnico
743.111.000-22	Ana Cláudia	Feminino	Recursos Humanos	Supervisor
654.888.765-45	Gabriella	Feminino	Cartório	Analista
123.456.789-32	Luciana	Feminino	Recursos Humanos	Analista
456.789.123-98	Carlos	Masculino	Informática	Técnico
321.543.765-78	André	Masculino	Informática	Analista
987.654.321-09	Raquel	Feminino	Cartório	Supervisor
345.678.012-54	Lucas	Masculino	Recursos Humanos	Técnico
749.049.163-36	Roberta	Feminino	Informática	Supervisor
708.576.254-12	Luiz	Masculino	Cartório	Técnico

```
CREATE TABLE Persons (  
    ID int NOT NULL,  
    LastName varchar(255) NOT NULL,  
    FirstName varchar(255),  
    Age int,  
    PRIMARY KEY (ID)  
);
```

CHAVE PRIMÁRIA COMPOSTA

ANO	SEDE	VENCEDOR	TIMES
1930	Uruguai	Uruguai	13
1934	Itália	Itália	16
1938	França	Itália	16
1950	Brasil	Uruguai	16
1954	Suíça	Alemanha	16
1958	Suécia	Brasil	16
1962	Chile	Brasil	16
1966	Inglaterra	Inglaterra	16
1970	México	Brasil	16
1974	Alemanha	Alemanha	16

```
CREATE TABLE Persons (  
    ID int NOT NULL,  
    LastName varchar(255) NOT NULL,  
    FirstName varchar(255),  
    Age int,  
    CONSTRAINT PK_Person PRIMARY KEY (ID,LastName)  
);
```

CHAVE ESTRANGEIRA

AUTOMÓVEL			
Placa	Marca	Modelo	Cor
A1	Honda	Fit	Prata
A2	Chevrolet	Astra	Branca

COMPRA		
PLACA	IDENTIDADE	DATA
A1	P1	D1
A2	P2	D2

PESSOA			
IDENTIDADE	NOME	ENDEREÇO	GÊNERO
P1	A	E1	M
P2	B	E2	F

```
CREATE TABLE Orders (  
    OrderID int NOT NULL,  
    OrderNumber int NOT NULL,  
    PersonID int,  
    PRIMARY KEY (OrderID),  
    FOREIGN KEY (PersonID) REFERENCES Persons(PersonID)  
);
```

CHAVE SURROGADA



**Chave primária
composta de uma
coluna numérica,
autoincremental**

**Serve de identificador
ÚNICO para o registro**

#ID	NOME	GÊNERO	SETOR
1	Ricardo	Masculino	Informática
2	Ana Cláudia	Feminino	Recursos Humanos
3	Gabriella	Feminino	Cartório
4	Luciana	Feminino	Recursos Humanos
5	Carlos	Masculino	Informática
6	André	Masculino	Informática
7	Raquel	Feminino	Cartório
8	Lucas	Masculino	Recursos Humanos
9	Roberta	Feminino	Informática
10	Luiz	Masculino	Cartório

w3school

<https://www.w3schools.com/sql/>

■ Referências

DATE, Christopher J. Introdução a sistemas de bancos de dados. Elsevier Brasil, 2004.

Elmars, R., & NAVATHE, S. B. (2011). Sistemas de banco de dados. Fundamentals of database systems.

SILBERSCHATZ, Abraham; SUNDARSHAN, S.; KORTH, Henry F. Sistema de banco de dados. Elsevier Brasil, 2016