Sistema Fiep

FIEP SESI SENAI IEL

Banco de Dados

Linguagem de Consulta SQL

SQL (Structured Query Language)

SQL (Structured Query Language, com a tradução: Linguagem Estruturada de Consulta), envolve três partes:

- Interface com usuário – permite que o usuário interaja com os dados.
- Conjunto de tabelas armazenadas no banco de dados – cada tabela é independente uma da outra; as linhas de tabelas diferentes são relacionadas com base em valores comuns de atributos comuns.
- Mecanismo de SQL – executa todas as consultas ou solicitações de dados.

SELECT

Trabalhando com SQL em um banco de dados, a tarefa mais comum é solicitar dados de uma ou mais tabelas e exibi-los. A instrução **SELECT** é usada para esse fim. Entretanto, o **SELECT** pode fazer muito mais que simplesmente recuperar e exibir dados. É possível transformar esses dados de forma significativa e construir resumos variáveis a partir de milhares de registros consultados.

```
SELECT CPF, Nome FROM tbl_Clientes

SELECT * FROM tbl_Clientes
```

Expressões em instruções SELECT

Sistema Fiep FIEP SESI SENAI IEL

Na instrução SELECT é possível fazer muito mais que apenas selecionar colunas. Por exemplo, pode-se realizar cálculos em uma ou mais colunas e incluí-los no resultado da consulta.

```
SELECT CodProduto,
       NomeProduto,
       PrecoLista,
       (PrecoLista * 0.05) + PrecoLista AS PrecoNoCartao
FROM tbl_Produtos
```

Concatenação em instruções SELECT

Sistema Fiep FIEP SESI SENAI IEL

Também é possível usar expressões com textos e outros tipos de dados. Um operador que muitas vezes pode ser útil quando utilizado com textos é o de concatenação, que mescla dois ou mais dados.

```
SELECT CPF,
   Nome,
       Endereco1 + ', ' + Bairro as RuaBairro,
       Cidade + ', ' + Estado as CidadeUF
FROM tbl_Clientes
```

```
SELECT CPF,
   Nome,
       concat(Endereco1 + '-', Bairro) as RuaBairro,
       Cidade + ', ' + Estado as CidadeUF
FROM tbl_Clientes
```

Cláusula WHERE

Ao trabalharmos com consultas de dados, uma tarefa muito comum é filtrar os registros de acordo com critérios, o que pode ser realizado com a instrução WHERE.

```
SELECT Nome, DataNasc
FROM tbl_Clientes
WHERE Idade != 18
```

```
SELECT Nome, DataNasc
FROM tbl_Clientes
WHERE Idade < 18
```

```
SELECT Nome, DataNasc
FROM tbl_Clientes
WHERE Idade <> 18
```

```
SELECT Nome, DataNasc
FROM tbl_Clientes
WHERE Idade <= 18
```

Instruções AND, OR e IN

Podemos utilizar as instruções AND, OR e IN para estruturar melhor os resultados das consultas realizadas no banco de dados.

Também podemos qualificar intervalos inclusivos usando a instrução BETWEEN. E uma instrução BETWEEN pode ser expressa alternativamente com o uso das expressões **>=** e **<=** e ainda uma instrução **AND.**

```
SELECT Nome, DataNasc
FROM tbl_Clientes
WHERE Idade BETWEEN 18 AND 20

SELECT Nome, DataNasc
FROM tbl_Clientes
WHERE Idade >= 18 AND Idade <= 20
```

Instruções AND, OR e IN

Ao utilizarmos uma instrução OR, pelo menos um dos critérios deve ser atendido para que o registro seja retornado na consulta.

```
SELECT Nome, DataNasc
FROM tbl_Clientes
WHERE MONTH(DataNasc) = 3 OR
MONTH(DataNasc) = 6 OR
MONTH(DataNasc) = 10
```

Uma forma mais eficiente de escrever a consulta anterior é utilizando uma instrução IN que forneça uma lista dos registros que atendem ao critério conforme a seguir:

```
SELECT Nome, DataNasc
FROM tbl_Clientes
WHERE MONTH(DataNasc) IN (3,6,10)
```

WHERE com texto

As regras para a qualificação de campos de texto seguem a mesma estrutura de campos numéricos, mesmo com diferenças sutis. É possível usar instruções =, AND, OR e IN com texto. Entretanto, quando usamos texto, é preciso inserir os literais (ou os valores textuais que for especificado) entre aspas simples.

```
SELECT * FROM tbl_Clientes
WHERE Nome = 'Eduardo Jorge'

SELECT * FROM tbl_Clientes
WHERE Nome IN ('Eduardo Jorge',
'Abel Silva','Gabriel Araujo')
```

```
--Funcao length no SQL Server
SELECT Nome, LEN(Nome) as Tamanho
FROM tbl_Clientes

SELECT * FROM tbl_Clientes
WHERE LEN (Nome) != 13

SELECT Nome, LENGTH(Nome) as Tamanho
FROM tbl_Clientes
```

Expressão LIKE no WHERE

Sistema Fiep FIEP SESI SENAI IEL

Uma outra operação comum é o uso de “curingas” em expressões LIKE, onde **%** significa qualquer número de caracteres e **_** um único caractere. Os demais caracteres são interpretados literalmente.

```
SELECT * FROM tbl_Clientes
WHERE Nome LIKE 'Eduardo%'

SELECT * FROM tbl_Clientes
WHERE Nome LIKE 'Edua_do%'
```

Expressão booleana no WHERE

Os valores booleanos são do tipo verdadeiro/falso. No universo de banco de dados, normalmente falso é definido como 0 e verdadeiro como 1. Alguns SGBDs permitem o uso implícito das palavras `true` ou `false` na qualificação.

```
-- true = 1 e false = 0
SELECT * FROM tbl_Vendedores
WHERE EmFerias = 'true'
```

```
SELECT * FROM tbl_Vendedores
WHERE EmFerias = 1 OR EmFerias = 0
```

```
-- 1 = true e 0 = false
SELECT * FROM tbl_Vendedores
 WHERE EmFerias = 1
```

```
SELECT * FROM tbl_Vendedores
WHERE NOT EmFerias = 1
```

Manipulando NULL

Valores NULLs são aqueles que não apresentam valor. Ou seja, é a ausência completa de qualquer conteúdo. Os valores nulos não podem ser determinados com “=“. É necessário usar as intruções “IS NULL” ou “IS NOT NULL” para identificar valores nulos.

```
-- IS NULL / IS NOT NULL
SELECT * FROM tbl_Clientes
WHERE Endereco2 IS NOT NULL

-- IS NULL / IS NOT NULL
SELECT * FROM tbl_Clientes
WHERE Endereco2 IS NULL

-- IS NULL / IS NOT NULL
SELECT * FROM tbl_Clientes
WHERE Endereco2 = ''
```

GROUP BY e ORDER BY

A agregação de dados (também chamada de totalização, resumo ou agrupamento de dados) é a criação de algum tipo de totalização a partir de vários registros. A soma “SUM”, máximo “MAX”, contagem “COUNT” e média “AVG” são operações de agregação comuns. É possível agrupar esses totais em qualquer coluna especificada, o que permite controlar facilmente o escopo das agregações.

```
--Contagem com a funcao COUNT()
SELECT COUNT(*) FROM tbl_Produtos

--Agrupar por data, campo DtNF
SELECT DtNF, COUNT(*) AS NFs
FROM tbl_NFs
WHERE Imposto = 0.12
GROUP BY DtNF
```

```
--Agrupar por data e matricula;
--ordenar o resultado por data
SELECT DtNF, Matricula, COUNT(*) AS NFs
FROM tbl_NFs
WHERE Matricula = 00237
GROUP BY DtNF, Matricula
ORDER BY DtNF ASC
```

GROUP BY e ORDER BY

A soma é outra operação comum na agregação. Para calcular a soma de vários valores é possível utilizar a função SUM(), para obter-se o valor máximo de um campo a função MAX().

```
--Soma de itens da NF
SELECT Numero, SUM(Qtd*Preco) AS Total
FROM tbl_Itens_NF
WHERE Numero = 100
GROUP BY Numero
```

```
--Maior Qtd levada na NF
SELECT Numero,
SUM(Preco) AS Total,
MAX(Qtd) AS MaiorQtdLevada
FROM tbl_Itens_NF
WHERE Numero = 100
GROUP BY Numero
```

```
--Produto mais caro
SELECT MAX(PrecoLista) ASVlrProdMaisCaro
FROM tbl_Produtos
```

Instrução HAVING

Para filtrar campos agregados não é possível utilizar WHERE. Para isso é preciso usar a instrução HAVING. A agregação funciona com o software do SGBD processando registro a registro e encontrando o que ele deseja manter de acordo com a cláusula WHERE. Em seguida, ele agrupa os registros no GROUP BY e executa as funções de agregação, por exemplo SUM( ).

Para filtramos pelo valor de SUM( ), é preciso que a filtragem ocorra após o cálculo, e é nesta ocasião que utilizamos o HAVING.

```
SELECT Numero, SUM(Qtd*Preco) AS Total
FROM tbl_Itens_NF
--WHERE Numero = 100
--WHERE SUM(Qtd*Preco) = 3080.168
GROUP BY Numero
HAVING SUM(Qtd*Preco) = 3080.168
```

Instrução HAVING

Para filtrar campos agregados não é possível utilizar WHERE. Para isso é preciso usar a instrução HAVING. A agregação funciona com o software do SGBD processando registro a registro e encontrando o que ele deseja manter de acordo com a cláusula WHERE. Em seguida, ele agrupa os registros no GROUP BY e executa as funções de agregação, por exemplo SUM( ).

Para filtramos pelo valor de SUM( ), é preciso que a filtragem ocorra após o cálculo, e é nesta ocasião que utilizamos o HAVING.

```
SELECT Numero, SUM(Qtd*Preco) AS Total
FROM tbl_Itens_NF
--WHERE Numero = 100
--WHERE SUM(Qtd*Preco) = 3080.168
GROUP BY Numero
HAVING SUM(Qtd*Preco) = 3080.168
```

Instrução CASE

Uma instrução CASE permite mapear uma ou mais condições para um valor correspondente a cada condição. Ela deve ser iniciada com a palavra reservada CASE e finalizada com END. Entre essas palavras-chave devemos especificar cada condição com a sintaxe WHEN [condição] e THEN [valor] em que a [condição] e o [valor] correspondente, precisamos informar.

Após especificar os pares condição/valor, podemos estabelecer um valor geral a ser usado como padrão se nenhuma das condições for atendida, que deve ser definido em ELSE.

```
CASE
        WHEN <condição01> THEN <valor01>
        WHEN <condição01> THEN <valor01>
        <...>
        WHEN <condição01> THEN <valor01>
        ELSE <valorELSE>
END
```

Operadores Relacionais

PROJECTION+SELECTION: Apresentar o ultimo nome e o salário (Projeção) de todos os funcionários do departamento 3 com o salário maior que R$ 600,00 (Seleção).

π Ult_Nome, Salario

σ (Cod_Depto=3) and (Salario>600) **(Funcionario)**

```
select Ult_Nome,Salario,Cod_Depto ⟶ π
from Funcionario
where Cod_Depto = 3 and Salario > 600; ⟶ σ
```

INNER JOIN

O operador INNER JOIN nos permite mesclar duas tabelas. Entretanto para poder mesclar duas tabelas é preciso definir um ou mais atributos comuns entre estas tabelas.

```
SELECT * FROM tbl_Clientes
INNER JOIN tbl_NFs
ON tbl_Clientes.CPF = tbl_NFs.CPF

SELECT tbl_Clientes.Nome, COUNT(*)
FROM tbl_Clientes
INNER JOIN tbl_NFs
ON tbl_Clientes.CPF = tbl_NFs.CPF
GROUP BY tbl_Clientes.Nome
```

Junção Externa (OUTER JOIN)

OUTER JOIN:

Utilizado quando desejamos manter no resultado da consulta todos os registros de uma das duas tabelas ou todos os registros das duas tabelas.

Pode ser:

- LEFT JOIN (Junção Externa à Esquerda).]X|
- RIGHT JOIN (Junção Externa à Direita). |X[
- FULL JOIN (Junção Externa Completa).]X[

LEFT JOIN

Utilizado quando desejamos manter no resultado da consulta todos os registros da tabela à esquerda.

Funcionario |×| Departamento

Funcionario				
Mat	Pri_Nome	Ult_Nome	Salario	Cod_Depto
1234	Paulo	Santos	1000	NULL
4321	Laura	Silva	1200	1
1122	Joana	Albuquerque	3000	2
3344	Carlos	Pavão	1300	3
2233	Vanessa	Fernandes	2000	4
4422	Carla	Gular	1100	NULL
4433	Ricardo	Doi	8000	2
2143	Rogerio	Faria	2000	1

Departamento		
Cod	Nome	UF
1	TI	PR
2	RH	SP
3	Financeiro	SP
4	Juridico	SP

LEFT JOIN

Utilizado quando desejamos manter no resultado da consulta todos os registros da tabela à esquerda.

Funcionario |×| Departamento

Funcionario				
Mat	Pri_Nome	Ult_Nome	Salario	Cod_Depto
1234	Paulo	Santos	1000	NULL
4321	Laura	Silva	1200	1
1122	Joana	Albuquerque	3000	2
3344	Carlos	Pavão	1300	3
2233	Vanessa	Fernandes	2000	4
4422	Carla	Gular	1100	NULL
4433	Ricardo	Doi	8000	2
2143	Rogerio	Faria	2000	1

Departamento		
Cod	Nome	UF
1	TI	PR
2	RH	SP
3	Financeiro	SP
4	Juridico	SP

LEFT JOIN

Utilizado quando desejamos manter no resultado da consulta todos os registros da tabela à esquerda.

Funcionario ⟕ Departamento

Funcionario					Departamento		
Mat	Pri_Nome	Ult_Nome	Salario	Cod_Depto	Cod	Nome	UF
1234	Paulo	Santos	1000	NULL	NULL	NULL	NULL
4321	Laura	Silva	1200	1	1	TI	PR
1122	Joana	Albuquerque	3000	2	2	RH	SP
3344	Carlos	Pavão	1300	3	3	Financeiro	SP
2233	Vanessa	Fernandes	2000	4	4	Juridico	SP
4422	Carla	Gular	1100	NULL	NULL	NULL	NULL
4433	Ricardo	Doi	8000	2	2	RH	SP
2143	Rogerio	Faria	2000	1	1	TI	PR

LEFT JOIN

Utilizado quando desejamos manter no resultado da consulta todos os registros da tabela à esquerda.

```
SELECT tbl_Clientes.Nome, COUNT(*)
FROM tbl_Clientes
LEFT JOIN tbl_NFs
ON tbl_Clientes.CPF = tbl_NFs.CPF
GROUP BY tbl_Clientes.Nome
```

RIGHT JOIN

Utilizado quando desejamos manter no resultado da consulta todos os registros tabela à direita.

Funcionario |×| Departamento

Funcionario				
Mat	Pri_Nome	Ult_Nome	Salario	Cod_Depto
1234	Paulo	Santos	1000	NULL
4321	Laura	Silva	1200	1
1122	Joana	Albuquerque	3000	2
3344	Carlos	Pavão	1300	3
2233	Vanessa	Fernandes	2000	4
4422	Carla	Gular	1100	NULL
4433	Ricardo	Doi	8000	2
2143	Rogerio	Faria	2000	1

Departamento		
Cod	Nome	UF
1	TI	PR
2	RH	SP
3	Financeiro	SP
4	Juridico	SP
5	Fiscal	SP

RIGHT JOIN

Utilizado quando desejamos manter no resultado da consulta todos os registros tabela à direita.

Funcionario |×| Departamento

Funcionario				
Mat	Pri_Nome	Ult_Nome	Salario	Cod_Depto
1234	Paulo	Santos	1000	NULL
4321	Laura	Silva	1200	1
1122	Joana	Albuquerque	3000	2
3344	Carlos	Pavão	1300	3
2233	Vanessa	Fernandes	2000	4
4422	Carla	Gular	1100	NULL
4433	Ricardo	Doi	8000	2
2143	Rogerio	Faria	2000	1

Departamento		
Cod	Nome	UF
1	TI	PR
2	RH	SP
3	Financeiro	SP
4	Juridico	SP
5	Fiscal	SP

RIGHT JOIN

Utilizado quando desejamos manter no resultado da consulta todos os registros tabela à direita.

Funcionario | × [Departamento

Funcionario					Departamento		
Mat	Pri_Nome	Ult_Nome	Salario	Cod_Depto	Cod	Nome	UF
4321	Laura	Silva	1200	1	1	TI	PR
1122	Joana	Albuquerque	3000	2	2	RH	SP
3344	Carlos	Pavão	1300	3	3	Financeiro	SP
2233	Vanessa	Fernandes	2000	4	4	Juridico	SP
4433	Ricardo	Doi	8000	1	1	TI	PR
2143	Rogerio	Faria	2000	2	2	RH	SP
NULL	NULL	NULL	NULL	NULL	5	Fiscal	SP

RIGHT JOIN

Utilizado quando desejamos manter no resultado da consulta todos os registros tabela à direita.

```
SELECT DISTINCT tbl_Clientes.Nome,
tbl_NFs.Matricula
FROM tbl_Clientes
RIGHT JOIN tbl_NFs
ON tbl_Clientes.CPF = tbl_NFs.CPF
```

FULL JOIN

Utilizado quando desejamos manter no resultado da consulta todos os registros das duas tabelas.

Funcionario] × [Departamento

Funcionario				
Mat	Pri_Nome	Ult_Nome	Salario	Cod_Depto
1234	Paulo	Santos	1000	NULL
4321	Laura	Silva	1200	1
1122	Joana	Albuquerque	3000	2
3344	Carlos	Pavão	1300	3
2233	Vanessa	Fernandes	2000	4
4422	Carla	Gular	1100	NULL
4433	Ricardo	Doi	8000	2
2143	Rogerio	Faria	2000	1

Departamento		
Cod	Nome	UF
1	TI	PR
2	RH	SP
3	Financeiro	SP
4	Juridico	SP
5	Fiscal	SP

FULL JOIN

Utilizado quando desejamos manter no resultado da consulta todos os registros das duas tabelas.

Funcionario] × [Departamento

Mat	Pri_Nome	Ult_Nome	Salario	Cod_Depto	Cod	Nome	UF
1122	Joana	Albuquerque	3000.00	2	2	RH	SP
1234	Paulo	Santos	1000.00	NULL	NULL	NULL	NULL
2143	Rogerio	Faria	2000.00	2	2	RH	SP
2233	Vanessa	Fernandes	2000.00	4	4	Juridico	SP
3344	Carlos	Pavão	1300.00	3	3	Financeiro	SP
4321	Laura	Silva	1200.00	1	1	TI	PR
4422	Carla	Gular	1100.00	NULL	NULL	NULL	NULL
4433	Ricardo	Doi	8000.00	1	1	TI	PR
NULL	NULL	NULL	NULL	NULL	5	Fiscal	SP

FULL JOIN

Utilizado quando desejamos manter no resultado da consulta todos os registros das duas tabelas.

```
SELECT tbl_Vendedores.BAIRRO AS
BairroVendedores,
tbl_Vendedores.Nome AS NomeVendedor,
tbl_Clientes.BAIRRO AS BairroCliente,
tbl_Clientes.Nome AS NomeCliente
FROM tbl_Vendedores
FULL JOIN tbl_Clientes
ON tbl_Vendedores.BAIRRO = tbl_Clientes.BAIRRO
```

Sistema Fiep

FIEP
SESI
SENAI
IEL

nosso i é de indústria.