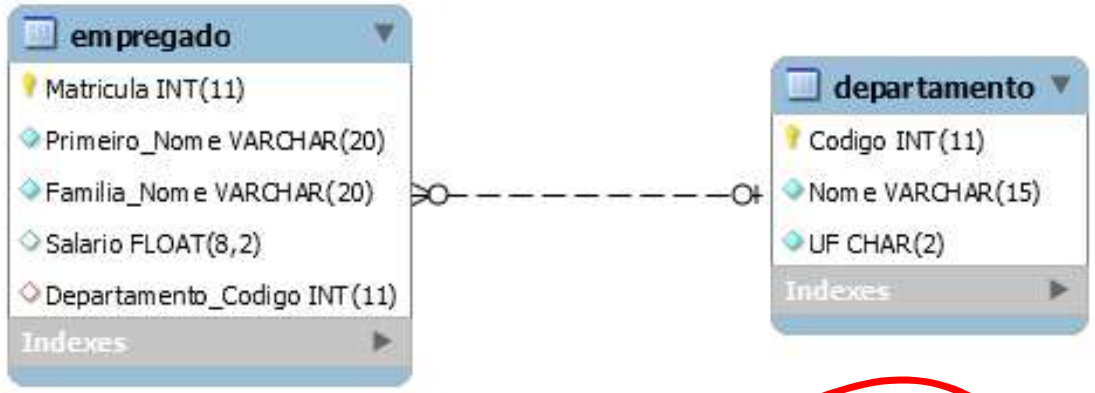
Introdução a Banco de Dados e Linguagem SQL

Marcio Victorino

Linguagem de Manipulação de Dados DML

Estudo de Caso Empresa



Matricula	Primeiro_Nome	Familia_Nome	Salario	Departamento_Codigo
123	Igor	Pereira	100.00	10
159	Denise	Moreno	440.00	40
369	Marcelo	Neiva	900.00	40
456	Ana	Oliveira	200.00	20
789	Clara	Silva	300.00	30
963	Deyse	Silva	330.00	NULL

Codigo	Nome	UF
10	RH	RJ
20	TI	RJ
30	Logística	DF
40	Financeiro	DF
50	Venda	RS
60	Pesquisa	RJ

Estudo de Caso Empresa

```
CREATE TABLE Departamento (
            Codigo      INT NOT NULL,
             Nome       VARCHAR(15) NOT NULL,
            UF          CHAR(2) NOT NULL,
            PRIMARY KEY (Codigo));
```

```
CREATE TABLE Empregado (
  Matricula             INT NOT NULL,
  Primeiro_Nome         VARCHAR(20) NOT NULL,
  Familia_Nome          VARCHAR(20) NOT NULL,
  Salario               FLOAT NULL DEFAULT NULL,
  Departamento_Codigo             INT NULL,
            PRIMARY KEY (Matricula),
            FOREIGN KEY (Departamento_Codigo)
            REFERENCES departamento (Codigo));
```

Inserir Registros (Insert)

Inserir Registros

```
INSERT INTO Departamento (Codigo, Nome, UF)
VALUES (10, 'RH', 'RJ'),
       (20, 'TI', 'RJ'),
       (30, 'Logística', 'DF'),
       (40, 'Financeiro', 'DF'),
       (50, 'Venda', 'RS'),
       (60, 'Pesquisa', 'RJ');
```

```
INSERT INTO Empregado(Matricula, Primeiro_Nome, Familia_Nome, Salario, Departamento_Codigo)
VALUES (123, 'Igor', 'Pereira', 100.00, 10),
(159, 'Denise', 'Moreno', 440.00, 40),
(369, 'Marcelo', 'Neiva', 900.00, 40),
(456, 'Ana', 'Oliveira', 200.00, 20),
(789, 'Clara', 'Silva', 300.00, 30),
(963, 'Deyse', 'Silva', 330.00, null);
```

Inserir Registros

```
CREATE TABLE Dep_Novo
AS SELECT Codigo, Nome, UF
FROM Departamento;
```

Inserir Registros

```
CREATE TABLE Dep ( Codigo      INT NOT NULL,
                   Nome        VARCHAR(15) NOT NULL,
                   UF          CHAR(2) NOT NULL,
                   PRIMARY KEY (Codigo));
```

```
INSERT INTO Dep (Codigo, Nome, UF)
SELECT Codigo, Nome, UF
FROM Departamento;
```

Modificar Registros (Update)

Modificar Registros

- Atualização:

```
UPDATE NOME_TABELA
SET COL1=VAL1, COL2=VAL2,...
WHERE (expressão lógica);
```

- Exemplo:

```
UPDATE Dep
SET UF = 'MG', Nome = 'Finanças'
WHERE Codigo = 40;
```

Excluir Registros (Delete)

Excluir Registros

- Exclusão:

```
DELETE FROM NOME_TABELA
WHERE (expressão lógica);
```

- Exemplo:

```
DELETE FROM Dep
WHERE Codigo = 30;
```

Comandos COMMIT e ROLLBACK

Comandos COMMIT e ROLLBACK

```
START TRANSACTION;
INSERT INTO dep (Codigo, Nome, UF)
VALUES (70, 'Teste', 'UF');
```

- ROLLBACK: desfaz a transação.
- COMMIT: torna permanente os efeitos das transações.

Transação

Transação

- Uma transação é uma unidade lógica do processamento do banco, que inclui uma ou mais operações de acesso ao banco de dados que precisa ser completada (ou desfeita) integralmente para garantir precisão.
- Essas operações podem incluir inclusão, exclusão, modificação ou seleção.
- Iniciar: start transaction (begin).
- Finalizar: commit (work).

Sistemas de Informação

- **Atomicidade:** uma transação é uma unidade de processamento, é realizada integralmente ou não é realizada.
- **Consistência:** uma transação leva um banco de dados de um estado consistente para outro estado consistente.
- **Isolamento:** uma transação deve parecer como se estivesse sendo executada isoladamente.
- **Durabilidade:** as alterações aplicadas a um banco de dados por meio de uma transação confirmada (commited) devem persistir no banco de dados.

Consultar Registros (Select)

Consultar Registros

- Consulta Simples:

```
SELECT COL1, COL2,...,COLN
FROM NOME_TABELA
WHERE (expressão lógica) ;
```

- Exemplo :

```
SELECT Codigo, Nome, UF
FROM Departamento;
```

Consultar Registros com Campos Nulos

- Exemplo:

```
SELECT Primeiro_Nome, Departamento_Codigo
FROM Empregado
WHERE Departamento_Codigo is null;
```

- Exemplo:

```
SELECT Primeiro_Nome, Departamento_Codigo
FROM Empregado
WHERE Departamento_Codigo is not null;
```

Consultar Registros usando o Distinct

- Elimina duplicatas do resultado:

```
SELECT DISTINCT(Departamento_Codigo)
FROM Empregado;
```

Operadores Lógicos

- Usados para validar condições:
  - AND;
  - OR;
  - IN;
  - NOT;
  - BETWEEN;
  - LIKE;
  - ALL;
  - ANY/SOME.

Operador Lógico AND

- Condição 1: Departamento_Codigo = 40.
- Condição 2: Salário > 500,00.

Matricula	Primeiro_Nome	Familia_Nome	Salario	Departamento_Codigo
123	Igor	Pereira	100.00	10
159	Denise	Moreno	440.00	40
369	Marcelo	Neiva	900.00	40
456	Ana	Oliveira	200.00	20
789	Clara	Silva	300.00	30
963	Deyse	Silva	330.00	NULL

C 1	C 2	AND
V	V	V
V	F	F
F	V	F
F	F	F

Operador Lógico AND

- Listar os dados dos empregados que trabalhem no departamento 40 e possuam salário superior à R$ 500,00:

```
SELECT *
FROM Empregado
WHERE Departamento_Codigo = 40 AND Salario > 500.00;
```

Operador Lógico OR

- Condição 1: Departamento_Codigo = 40.
- Condição 2: Salário > 500,00.

Matricula	Primeiro_Nome	Familia_Nome	Salario	Departamento_Codigo
123	Igor	Pereira	100.00	10
159	Denise	Moreno	440.00	40
369	Marcelo	Neiva	900.00	40
456	Ana	Oliveira	200.00	20
789	Clara	Silva	300.00	30
963	Deyse	Silva	330.00	NULL

C 1	C 2	OR
V	V	V
V	F	V
F	V	V
F	F	F

Operador Lógico OR

- Listar os dados dos empregados que trabalhem no departamento 40 ou possuam salário superior à R$ 500,00:

```
SELECT *
FROM Empregado
WHERE Departamento_Codigo = 40 OR Salario > 500.00;
```

Operador Lógico IN

- Quais empregados que trabalham nos departamentos 10 ou 30?

Matricula	Primeiro_Nome	Familia_Nome	Salario	Departamento_Codigo
123	Igor	Pereira	100.00	10
159	Denise	Moreno	440.00	40
369	Marcelo	Neiva	900.00	40
456	Ana	Oliveira	200.00	20
789	Clara	Silva	300.00	30
963	Deyse	Silva	330.00	NULL

Operador Lógico IN

- Listar dados dos empregados que trabalham nos departemantos 10 ou 30:

```
SELECT *
FROM Empregado
WHERE Departamento_Codigo = 10 OR Departamento_Codigo = 30;
```

- Outra forma:

```
SELECT *
FROM Empregado
WHERE Departamento_Codigo IN (10, 30);
```

Operadores Lógicos NOT IN

- Quais empregados que não trabalham nos departamentos 10 ou 30?

Matricula	Primeiro_Nome	Familia_Nome	Salario	Departamento_Codigo
123	Igor	Pereira	100.00	10
159	Denise	Moreno	440.00	40
369	Marcelo	Neiva	900.00	40
456	Ana	Oliveira	200.00	20
789	Clara	Silva	300.00	30
963	Deyse	Silva	330.00	NULL

Operador Lógico NOT IN

- Listar dados dos empregados que trabalham nos departemantos 10 ou 30:

```
SELECT *
FROM Empregado
WHERE (Departamento_Codigo <> 10 AND Departamento_Codigo <> 30);
```

- Outra forma:

```
SELECT *
FROM Empregado
WHERE Departamento_Codigo NOT IN (10, 30);
```

Operador Lógico BETWEEN

- Quais empregados possuem salário entre R$ 250,00 e R$ 800,00?

Matricula	Primeiro_Nome	Familia_Nome	Salario	Departamento_Codigo
123	Igor	Pereira	100.00	10
159	Denise	Moreno	440.00	40
369	Marcelo	Neiva	900.00	40
456	Ana	Oliveira	200.00	20
789	Clara	Silva	300.00	30
963	Deyse	Silva	330.00	NULL

Operador Lógico BETWEEN

- Listar dados dos empregados que possuem salário entre R$ 250,00 e R$ 800,00:

```
SELECT *
FROM Empregado
WHERE salario >= 250.00 AND salario <= 800.00;
```

- Outra forma:

```
SELECT *
FROM Empregado
WHERE salario BETWEEN 250.00 AND 800.00;
```

Operadores Lógicos

Expressão	Explicação
LIKE 'A%'	Todas as palavras que iniciem com a letra A.
LIKE '%A'	Todas que terminem com a letra A.
LIKE '% A%'	Todas que tenham a letra A em qualquer posição.
LIKE 'A_'	String de dois caracteres que tenham a primeira letra A e o segundo caractere seja qualquer outro.
LIKE '_A'	String de dois caracteres cujo primeiro caractere seja qualquer um e a última letra seja A.
LIKE '_A_'	String de três caracteres cuja segunda letra seja A, independentemente do primeiro ou do último caractere.
LIKE '%A_'	Todos que tenham a letra A na panúltima posição e a última seja qualquer outro caractere.
LIKE '_A%'	Todos que tanham a letra A na segunda posição e o primeiro caractere seja qualquer um.

Operador %

- Listar os dados dos empregados cujo nome inicia com a letra ‘M’:

```
SELECT * FROM Empregado
WHERE Primeiro_Nome Like 'M%’;
```

- Listar os dados dos empregados cujo nome termina com a letra ‘a’:

```
SELECT * FROM Empregado
WHERE Primeiro_Nome Like ‘%a’;
```

- Listar os dados dos empregados cujo nome tenha como segunda letra o ‘e’:

```
SELECT * FROM Empregado
WHERE Primeiro_Nome Like '_e%’;
```

- Listar os dados dos empregados cujo nome tenha como penúltima letra o ‘n’:

```
SELECT * FROM Empregado
WHERE Primeiro_Nome Like '%n_’;
```

Operador %

- Listar os dados dos empregados cujo nome tenha a letra ‘s’:

```
SELECT * FROM Empregado
WHERE Primeiro_Nome Like '%s%’;
```

- Listar os dados dos empregados cujo nome tenha três letras, sendo que a letra do meio seja o ‘n’:

```
SELECT * FROM Empregado
WHERE Primeiro_Nome Like '_n_’;
```

- Listar os dados dos empregados cujo nome não inicia com a letra ‘M’:

```
SELECT * FROM Empregado
WHERE Primeiro_Nome Not Like 'M%';
```

Operadores Lógicos ALL

- Quais empregados do departamento 40 possuem salário maior que todos os empregados do departamento 30?

Matricula	Primeiro_Nome	Familia_Nome	Salario	Departamento_Codigo
123	Igor	Pereira	100.00	10
159	Denise	Moreno	440.00	40
369	Marcelo	Neiva	900.00	40
456	Ana	Oliveira	200.00	20
789	Clara	Silva	300.00	30
855	Edu	Snatos	600.00	30
963	Deyse	Silva	330.00	NULL

Operadores Lógicos ALL

```
INSERT INTO empregado(Matricula, Primeiro_Nome, Familia_Nome, Salario, Departamento_Codigo)
VALUES (855, 'Edu', 'Santos', 600.00, 30);
```

- Listar o nome e o salário dos empregados do departamento 40 que possuem salário maior que todos os empregados do departamento 30.

```
SELECT Primeiro_Nome, Salario
FROM Empregado
WHERE Departamento_Codigo = 40 AND
Salario > ALL (SELECT Salario
               FROM Empregado
               WHERE Departamento_Codigo = 30);
```

Operadores Lógicos ALL

- Listar o nome e o salário dos empregados do departamento 40 que possuem salário maior que todos os empregados do departamento 30.

- Utilizando MAX():

```
SELECT Primeiro_Nome, Salario
FROM Empregado
WHERE Departamento_Codigo = 40 AND
Salario > (SELECT MAX(Salario)
            FROM Empregado
            WHERE Departamento_Codigo = 30);
```

Operadores Lógicos ANY/SOME

- Quais empregados do departamento 40 possuem salário maior que pelo menos um dos empregados do departamento 30?

Matricula	Primeiro_Nome	Familia_Nome	Salario	Departamento_Codigo
123	Igor	Pereira	100.00	10
159	Denise	Moreno	440.00	40
369	Marcelo	Neiva	900.00	40
456	Ana	Oliveira	200.00	20
789	Clara	Silva	300.00	30
855	Edu	Snatos	600.00	30
963	Deyse	Silva	330.00	NULL

Operadores Lógicos ANY/SOME

- Listar o nome e o salário dos empregados do departamento 40 que possuem salário maior que pelo menos um dos empregados do departamento 30.

```
SELECT Primeiro_Nome, Salario
FROM Empregado
WHERE Departamento_Codigo = 40 AND
Salario > ANY (SELECT Salario
                FROM Empregado
                WHERE Departamento_Codigo = 30);
```

Operadores Lógicos ANY/SOME

- Listar o nome e o salário dos empregados do departamento 40 -- que possuem salário maior que pelo menos um dos empregados do departamento 30.
- Utilizando MIN():

```
SELECT Primeiro_Nome, Salario
FROM Empregado
WHERE Departamento_Codigo = 40 AND
Salario > (SELECT MIN(Salario)
            FROM Empregado
            WHERE Departamento_Codigo = 30);
```

Operadores Aritméticos

- Operadores Aritméticos:
  - +;
  - -;
  - *;
  - /;
  - %.

- Operador % (módulo):
  - Retorna o resto inteiro de uma divisão.
  - Por exemplo, 13% 5 = 3 porque o resto de 13 dividido por 5 é 3.

Operadores Aritméticos

- Listar o nome e o salário dos empregados acrescidos de R$ 50.00.

```
SELECT Primeiro_Nome, Salario + 50.00
FROM Empregado;
```

- Utilizando Alias:

```
SELECT Primeiro_Nome, Salario + 50.00 as Novo_Salario
FROM Empregado;
```

- Listar o nome e o salário dos empregados acrescidos de 10%.

```
SELECT Primeiro_Nome, Salario * 1.1
FROM Empregado;
```

- Utilizando Alias:

```
SELECT Primeiro_Nome, Salario * 1.1 as Novo_Salario
FROM Empregado;
```

Operadores Aritméticos

- Listar o nome e a metade do salário dos empregados.

```
SELECT Primeiro_Nome, Salario / 2
FROM Empregado;
```

- round(15.654, 2) = 15.65
- round(15.655, 2) = 15.66
- truncate(15.654, 2) = 15.65
- truncate(15.655, 2) = 15.65

Utilizando Alias:

```
SELECT Primeiro_Nome, trunc(Salario/2, 2) as Novo_Salario
FROM Empregado;
```

Operadores Aritméticos

- Listar o nome e salário dos empregados cujo salário seja divisível por 3.

```
SELECT Primeiro_Nome, Salario
FROM Empregado
WHERE MOD(Salario,3) = 0;
```

Operador Concatenação

- Concatena Strings:
  - CONCAT(COL1, COL2,...,COLN).

```
SELECT Concat(Primeiro_Nome, ` ` , Familia_Nome)
FROM Empregado;
```

Funções Agregadas

- Funções Agregadas:
  - COUNT();
  - SUM();
  - AVG();
  - MAX();
  - MIN().

Funções Agregadas

- Listar o total de empregados que estão alocados em deparamentos:

```
SELECT COUNT(Departamento_Codigo)
FROM Empregado;
```

- Listar o total de empregados que estão alocados ao deparamento 40:

```
SELECT COUNT(*)
FROM Empregado
WHERE Departamento_Codigo = 40;
```

- Listar o total dos salários dos empregados:

```
SELECT SUM(salario)
FROM Empregado;
```

- Listar o total dos salários dos empregados que estão alocados ao deparamento 40:

```
SELECT SUM(Salario)
FROM Empregado
WHERE Departamento_Codigo = 40;
```

Funções Agregadas

- Listar a média dos salários dos empregados:

```
SELECT AVG(salario)
FROM Empregado;
```

- Listar a média dos salários dos empregados que estão alocados ao deparamento 40:

```
SELECT AVG(Salario)
FROM Empregado
WHERE Departamento_Codigo = 40;
```

- Listar o maior salário dos empregados:

```
SELECT MAX(salario)
FROM Empregado;
```

Funções Agregadas

- Listar o maior salário dos empregados que estão alocados ao deparamento 40:

```
SELECT MAX(Salario)
FROM Empregado
WHERE Departamento_Codigo = 40;
```

- Listar o menor salário dos empregados:

```
SELECT MIN(salario)
FROM Empregado;
```

- Listar o menor salário dos empregados que estão alocados ao deparamento 40:

```
SELECT MIN(Salario)
FROM Empregado
WHERE Departamento_Codigo = 40;
```

Order By

- Ordenação de Resultados:

```
SELECT Matricula, Primeiro_Nome, Salario
FROM Empregado
ORDER BY Primeiro_Nome ASC;

SELECT Matricula, Primeiro_Nome, Salario
FROM Empregado
ORDER BY Primeiro_Nome DESC;
```

Group By

Listar o código do departamento e o maior salário desse departamento.

Matricula	Primeiro_Nome	Familia_Nome	Salario	Departamento_Codigo
123	Igor	Pereira	100.00	10
159	Denise	Moreno	440.00	40
369	Marcelo	Neiva	900.00	40
456	Ana	Oliveira	200.00	20
789	Clara	Silva	300.00	30
855	Edu	Snatos	600.00	30
963	Deyse	Silva	330.00	NULL

Group By

```
SELECT Departamento_Codigo, MAX(Salario)
FROM Empregado
GROUP BY Departamento_Codigo;
```

Group By

Listar o código do departamento e a média salarial desse departamento.

Matricula	Primeiro_Nome	Familia_Nome	Salario	Departamento_Codigo
123	Igor	Pereira	100.00	10
159	Denise	Moreno	440.00	40
369	Marcelo	Neiva	900.00	40
456	Ana	Oliveira	200.00	20
789	Clara	Silva	300.00	30
855	Edu	Snatos	600.00	30
963	Deyse	Silva	330.00	NULL

Group By

```
SELECT Departamento_Codigo,  AVG(Salario)
FROM Empregado
GROUP BY Departamento_Codigo;
```

Group By

Listar o código do departamento e a quantidade de empregados desse departamento.

Matricula	Primeiro_Nome	Familia_Nome	Salario	Departamento_Codigo
123	Igor	Pereira	100.00	10
159	Denise	Moreno	440.00	40
369	Marcelo	Neiva	900.00	40
456	Ana	Oliveira	200.00	20
789	Clara	Silva	300.00	30
855	Edu	Snatos	600.00	30
963	Deyse	Silva	330.00	NULL

Group By

```
SELECT Departamento_Codigo, COUNT(*)
FROM Empregado
GROUP BY Departamento_Codigo;
```

Having

Listar o código do departamento e a quantidade de empregados desse departamento quando o departamento possuir mais de um empregado.

Matricula	Primeiro_Nome	Familia_Nome	Salario	Departamento_Codigo
123	Igor	Pereira	100.00	10
159	Denise	Moreno	440.00	40
369	Marcelo	Neiva	900.00	40
456	Ana	Oliveira	200.00	20
789	Clara	Silva	300.00	30
855	Edu	Snatos	600.00	30
963	Deyse	Silva	330.00	NULL

Having

```
SELECT Departamento_Codigo, COUNT(*)
FROM Empregado
GROUP BY Departamento_Codigo
HAVING COUNT(*) > 1;
```

Having

Listar o código do departamento e a média salarial dos empregados desse departamento quando a média salarial do departamento for maior que R$ 500.00.

Matricula	Primeiro_Nome	Familia_Nome	Salario	Departamento_Codigo
123	Igor	Pereira	100.00	10
159	Denise	Moreno	440.00	40
369	Marcelo	Neiva	900.00	40
456	Ana	Oliveira	200.00	20
789	Clara	Silva	300.00	30
855	Edu	Snatos	600.00	30
963	Deyse	Silva	330.00	NULL

Having

```
SELECT Departamento_Codigo, AVG(Salario)
FROM Empregado
GROUP BY Departamento_Codigo
HAVING AVG(Salario) > 500.00;
```

Having

Listar o código do departamento e a média salarial dos empregados desse departamento quando a média salarial for maior que R$ 500.00. Considerar apenas empregados com salário maior que R$ 350.00.

Matricula	Primeiro_Nome	Familia_Nome	Salario	Departamento_Codigo
159	Denise	Moreno	440.00	40
369	Marcelo	Neiva	900.00	40
855	Edu	Snatos	600.00	30
				NULL

Having

```
SELECT Departamento_Codigo, AVG(Salario)
FROM Empregado
WHERE Salario > 350.00
GROUP BY Departamento_Codigo
HAVING AVG(Salario) > 500.00;
```

Matricula	Primeiro_Nome	Familia_Nome	Salario	Departamento_Codigo
159	Denise	Moreno	440.00	40
369	Marcelo	Neiva	900.00	40
855	Edu	Snatos	600.00	30
				NULL

Consulta Completa

SELECT COL1, COL2,...,COLN
FROM NOME_TABELA
WHERE (expressão lógica)
GROUP BY (atributos de agrupamento)
HAVING (condição de agrupamento)
ORDER BY (lista de atributos)

Fim