SQL - Primeiros passos

Olá!

Sou a Adriana Leticia

Sou bacharel em Sistemas de Informação pelo Mackenzie com MBA em Finanças pela FGV. Trabalho há 15 anos na Vivo, sempre atuando com BI. Já passei por diferentes áreas - DW, Controladoria, Estratégia e atualmente estou em Planejamento de Marketing B2C Móvel. Amo tocar piano, gatos (tenho dois), viajar e claro, ficar com a família.

Índice

Breve história sobre a linguagem SQL

Antes de 1970, os dados eram armazenados em forma de arquivos indexados, em que cada programador podia definir sua estrutura de dados. O armazenamento e o acesso aos dados eram feitos através do COBOL.

Em 1970, o britânico, Matemático e Cientista da Computação Edgar Francis Codd, em seu trabalho de conclusão de curso propôs o modelo de apresentação de dados na forma de relações (**tabelas**, entidades), que possuem atributos (**colunas**, campos) e registros (**linhas**, tuplas). Este trabalho foi baseado na Teoria dos Conjuntos e Álgebra Relacional.

No mesmo ano a **IBM** desenvolveu o sistema de banco de dados System R, considerado o primeiro SGBDR (Sistema de Gerenciamento de Banco de Dados Relacional). O System R foi baseado no trabalho de E. F. Codd.

Uma linguagem foi desenvolvida para manipular os dados armazenados no System R - a SEQUEL (Structured English Query Language). Posteriormente, condensada para SQL.

Breve história sobre a linguagem SQL

Em 1979, três empresas, Sybase, Microsoft e Ashton-Tate formaram um consórcio para desenvolver um produto que seria concorrente dos servidores de bancos de dados da IBM. Surge o SQL Server 1.0.

Em 1986, surge o primeiro padrão ANSI SQL. Revisado em 1989, 1992, 1999, 2003, 2006, 2008, 2011 e 2016.

Em 1992, a Microsoft lança seu próprio banco de dados, o Microsoft SQL Server 4.2.

Vantagens:

- Pequena quantidade de comandos para realizar uma grande quantidade de operações
- Padrão ANSI facilita migração. É possível executar o mesmo "select" em diferentes tecnologias – Oracle, SQL Server, PostgreSQL, MySql, SQLite, etc.
- Real vantagem está no modelo relacional (como as tabelas se relacionam).

SQL - Structured Query Language

SQL Structured Query Language - Linguagem de pesquisa (consulta) estruturada.

A linguagem SQL faz mais do que uma consulta em uma ou mais tabelas em um banco de dados. Com ela é possível inserir, alterar (modificar) e deletar dados.

Os dados ficam armazenados em uma ou mais tabelas e as tabelas ficam armazenadas em um SGBD (Sistemas Gerenciador de Banco de Dados).

Existem 3 conjuntos de comandos utilizados em SGBD.

- **DML** : Data Manipulation Language (manipulação de dados. Abordados neste Meetup).
- **DDL** : Data Definition Language (manipulação de objetos, como tabelas e colunas)
- **DCL** : Data Control Language (controle de acesso a objetos)

Os conjuntos DDL e DCL normalmente são executados por DBA.

Tipos de dados em uma tabela

VARCHAR(n) - tamanho variável – máximo de n caracteres

CHAR(n) - tamanho fixo - n caracteres

INTEGER | **SMALLINT**

DECIMAL [(precision, scale)] - precision é o número total de dígitos total e scale é o número de dígitos depois do ponto

DOUBLE PRECISION | **FLOAT** | **REAL**

DATE | **TIME** | **TIMESTAMP**

BLOB – Binary Large Object. | **IMAGE**

BOOLEAN

Tabela Funcionários

Column Name	Data Type	Allow Nulls
Emp_ID	int	☐
Name	varchar(50)	☑
Email	varchar(500)	☑
Designation	varchar(50)	☑
City	varchar(50)	☑
State	varchar(50)	☑
Country	varchar(50)	☑
Experience	int	☑

Select

```
-- Todas as colunas
SELECT *
FROM Employee

SELECT Emp_ID, Name, Email, Designation, City, State, Country,
Experience
FROM Employee

-- Apenas colunas desejadas
SELECT Country, Name
FROM Employee

-- Usando alias para coluna e para a tabela
SELECT Country as pais, Name as nome_funcionario, City as cidade
FROM Employee as funcionario
```

```
SELECT <colunas >
FROM <tabelas >
WHERE <restricoes >
GROUP BY <colunas >
HAVING <condições>
ORDER BY <colunas > ASC | DESC
```

Tabela Employee

Column Name	Data Type	Allow Nulls
Emp_ID	int	☐
Name	varchar(50)	☑
Email	varchar(500)	☑
Designation	varchar(50)	☑
City	varchar(50)	☑
State	varchar(50)	☑
Country	varchar(50)	☑
Experience	int	☑

SQL – Operadores

Comparação: =, <>, <, >, <=, >=

Lógicos: and, or

Range da valores: between

Conjuntos: in, not in, exists, not exists

Match parcial de texto: like, not like

Nulabilidade : is null, is not null

Select com Filtro (Where)

```
-- Nome e Email dos Funcionarios da Cidade de São Paulo com experiência a partir de 5 anos
SELECT Name as nome, Email as email
FROM Employee as a
WHERE City = 'São Paulo'
AND Experience >= 5

-- Funcionários que são da França e Espanha, que não são das cidades de Paris e Madri
SELECT *
FROM Employee as a
WHERE Country in ('France', 'Spain') AND City not in ('Paris', 'Madri')

-- Funcionários que não tem email cadastrado
SELECT *
FROM Employee as func
WHERE Email is null
```

Tabela Employee

Column Name	Data Type	Allow Nulls
Emp_ID	int	☐
Name	varchar(50)	☑
Email	varchar(500)	☑
Designation	varchar(50)	☑
City	varchar(50)	☑
State	varchar(50)	☑
Country	varchar(50)	☑
Experience	int	☑

Select com Filtro (Where)

```
-- Funcionarios que tem experiencia entre 10 e 15 anos
SELECT *
FROM Employee as a
WHERE Experience between 10 and 15

SELECT *
FROM Employee as a
WHERE Experience >= 10 and Experience <= 15

-- Funcionários que Começam com o nome Adriana
SELECT *
FROM Employee as a
WHERE Name LIKE 'Adriana%'

-- Funcionários que terminam com o sobrenome Silva
SELECT *
FROM Employee as a
WHERE Name LIKE '%Silva'
```

Tabela Employee

Column Name	Data Type	Allow Nulls
Emp_ID	int	☐
Name	varchar(50)	☑
Email	varchar(500)	☑
Designation	varchar(50)	☑
City	varchar(50)	☑
State	varchar(50)	☑
Country	varchar(50)	☑
Experience	int	☑

Select com Filtro (Where)

Tabela Employee

Column Name	Data Type	Allow Nulls
Emp_ID	int	☐
Name	varchar(50)	☑
Email	varchar(500)	☑
Designation	varchar(50)	☑
City	varchar(50)	☑
State	varchar(50)	☑
Country	varchar(50)	☑
Experience	int	☑

```
-- Funcionarios que tem a palavra Maria no nome
SELECT *
FROM Employee as a
WHERE Name LIKE '%Maria%'

-- Funcionários que não contém o Pereira no nome
SELECT *
FROM Employee as a
WHERE Name NOT LIKE '%Pereira%'

-- Funcionários do RJ. Ordernar os funcionários de forma descrescente por tempo de experiência
SELECT *
FROM Employee as a
WHERE State = 'RJ'
ORDER BY Experience DESC

-- Todos os países cadastrados. Mostrar sem repetição e ordenado
SELECT DISTINCT Country
FROM Employee as a
ORDER BY Country

SELECT DISTINCT Country
FROM Employee as a
ORDER BY 1
```

✖ Funções de Agrupamento

Tabela Employee

Column Name	Data Type	Allow Nulls
Emp_ID	int	☐
Name	varchar(50)	☑
Email	varchar(500)	☑
Designation	varchar(50)	☑
City	varchar(50)	☑
State	varchar(50)	☑
Country	varchar(50)	☑
Experience	int	☑

```
-- Contar quantas linhas (registros) contém na tabela
SELECT count(*)
FROM Employee

-- Quantas cidades (sem repetição) a tabela Employee tem
SELECT COUNT(DISTINCT City) AS Quantidade_Cidades
FROM Employee

-- Quantidade de funcionários por país. Ordernar da maior quantidade para a menor
SELECT Country, count(*) AS Quantidade_funcionario_por_pais
FROM Employee as a
GROUP BY Country ORDER BY 2 DESC

-- Quantidade por cidade e cargo dos funcionários do RJ. Somente quantidade maior que 20.
SELECT City AS Cidade, Designation AS Cargo, count(*) AS Quantidade_cargo
FROM Employee
WHERE State = 'RJ'
GROUP BY City, Designation
HAVING COUNT(*) > 20
ORDER BY City
```

Funções de Agrupamento

```
SELECT SUM(Experience) AS Soma_Tempo_de_Experiencia,
MIN(Experience) AS Menor_Tempo_de_Experiencia,
MAX(Experience) AS Maior_Tempo_de_Experiencia,
AVG(Experience) AS Média_Tempo_de_Experiencia
FROM Employee
```

```
-- "Estatísticas" por cargo
SELECT Designation,
SUM(Experience) AS Soma_Tempo_de_Experiencia,
MIN(Experience) AS Menor_Tempo_de_Experiencia,
MAX(Experience) AS Maior_Tempo_de_Experiencia,
AVG(Experience) AS Média_Tempo_de_Experiencia
FROM Employee
GROUP BY Designation
```

Tabela Employee

Column Name	Data Type	Allow Nulls
Emp_ID	int	☐
Name	varchar(50)	☑
Email	varchar(500)	☑
Designation	varchar(50)	☑
City	varchar(50)	☑
State	varchar(50)	☑
Country	varchar(50)	☑
Experience	int	☑

Insert

INSERT INTO <tabela> **VALUES** (<coluna 1>, <coluna 2>, <coluna n>)

Tabela Employee

Column Name	Data Type	Allow Nulls
Emp_ID	int	☐
Name	varchar(50)	☑
Email	varchar(500)	☑
Designation	varchar(50)	☑
City	varchar(50)	☑
State	varchar(50)	☑
Country	varchar(50)	☑
Experience	int	☑

-- Inserir um novo Funcionario em todas as colunas

INSERT INTO Employee **VALUES** (1001, 'Marta Celestino', 'mc@empresa.com', 'Data Science', 'São Paulo', 'SP', 'Brasil', 1)

-- Inserir somente o ID de funcionario, nome e cidade

INSERT INTO Employee (Emp_ID, Name, City) **VALUES** (2020, 'Haydee', 'São Paulo')

Update

```
UPDATE  <tabela>
SET <coluna 1> = 'XXX',
<coluna 2> = 'yyy',
 WHERE  <coluna 1> = 'ZZZ'
```

Tabela Employee

Column Name	Data Type	Allow Nulls
Emp_ID	int	☐
Name	varchar(50)	☑
Email	varchar(500)	☑
Designation	varchar(50)	☑
City	varchar(50)	☑
State	varchar(50)	☑
Country	varchar(50)	☑
Experience	int	☑

```
-- Atualizar o email de todos os funcionários para nulo
UPDATE Employee
SET Email = NULL

-- Atualizar o nome do funcionário Marta Celestino para Marta Vieira Celestino. O Código do funcionário é 10020
UPDATE Employee
SET Name = 'Marta Vieira Celestino'
WHERE Emp_ID = 10020
```

Delete

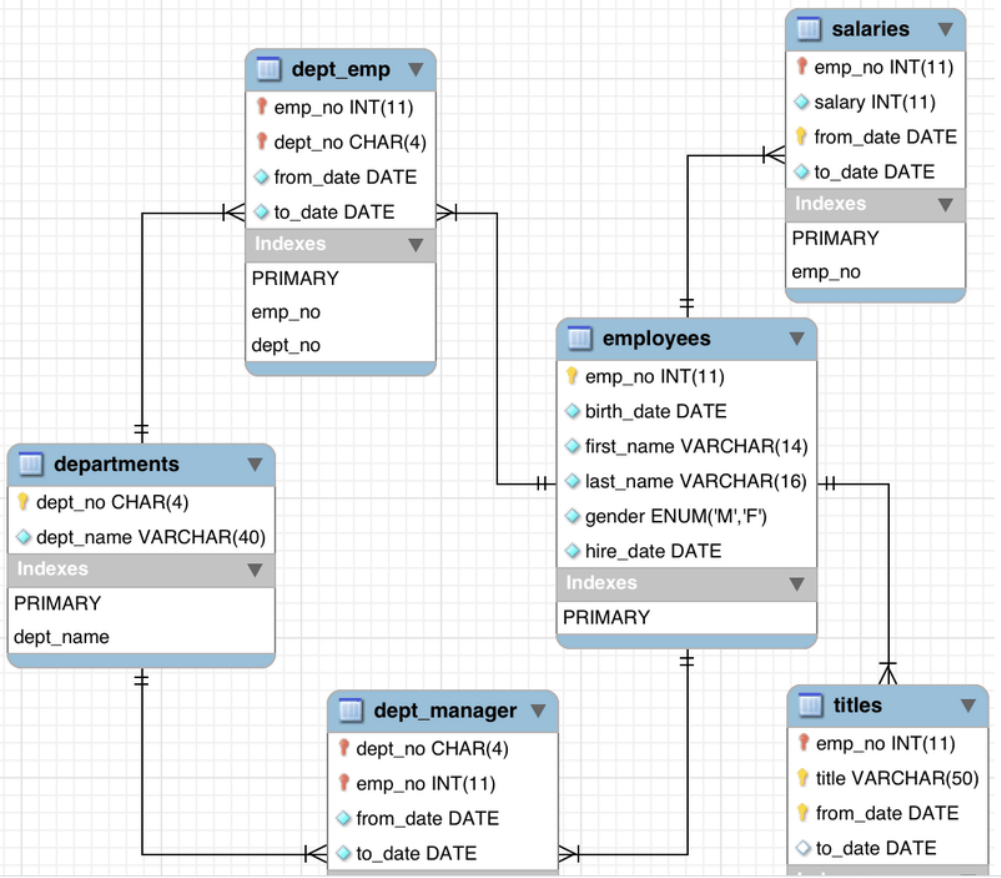
```
DELETE FROM   <tabela>
WHERE  <coluna 1> = 'ZZZ'
```

Tabela Employee

Column Name	Data Type	Allow Nulls
Emp_ID	int	☐
Name	varchar(50)	☑
Email	varchar(500)	☑
Designation	varchar(50)	☑
City	varchar(50)	☑
State	varchar(50)	☑
Country	varchar(50)	☑
Experience	int	☑

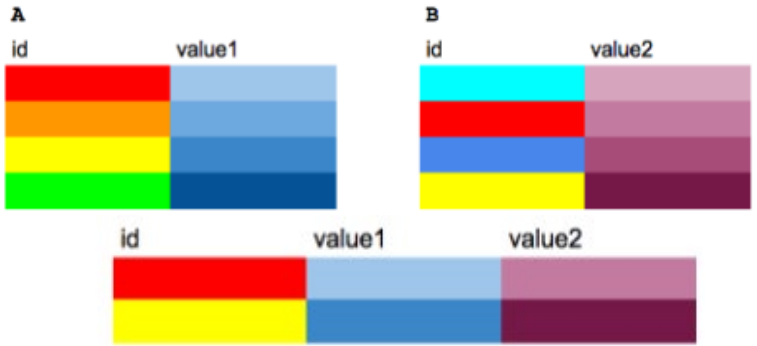
```
-- Excluir todos os registros da tabela Employee
DELETE FROM   Employee

-- Excluir o funcionário de código 10020
DELETE FROM   Employee
WHERE  Emp_ID  = 10020
```

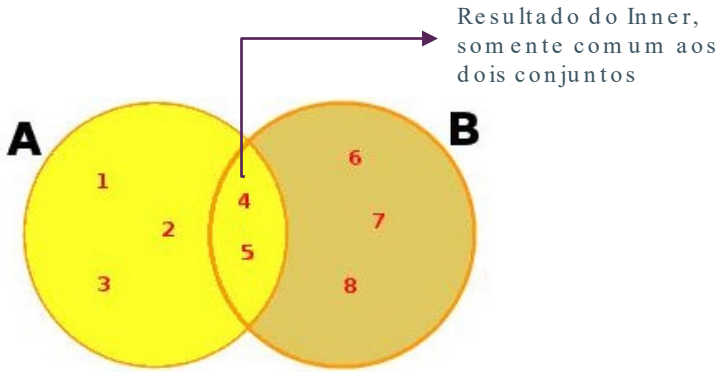
SGBDR - Sistema de Gerenciamento de Banco de Dados Relacional



Inner join

```
SELECT * FROM <tabela_A> INNER JOIN <tabela_B>
ON <tabela_A.coluna 1> = <tabela_B.coluna 1>
```

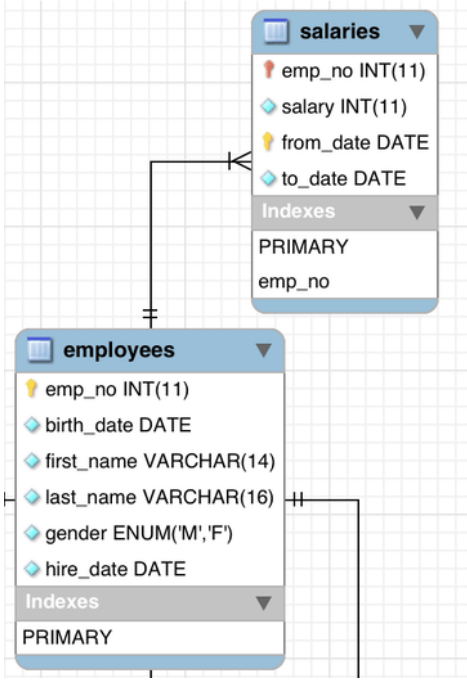




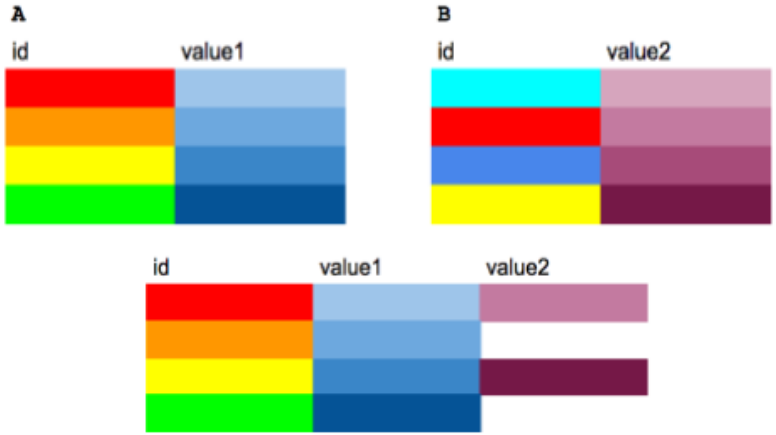
Fonte: https://rpubs.com/CristianaFreitas/311735

Inner join

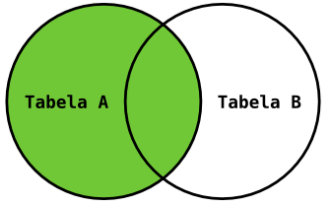
```
-- Salário dos funcionários

SELECT *
FROM employees a INNER JOIN  salaries b
ON a.emp_no = b.emp_no


-- Todas as colunas da tabela employees e somente o salario

SELECT a.*, b.salary
FROM employees a INNER JOIN  salaries b
ON a.emp_no = b.emp_no


-- Funcionários que ganham mais do que R$ 10.000,00

SELECT a.emp_no, a.first_name, a.last_name, b.salary
FROM employees a INNER JOIN salaries b
ON a.emp_no = b.emp_no
WHERE b.salary > 10000
```



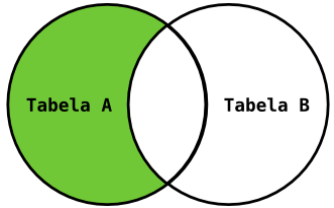
Left join




```
SELECT * FROM <tabela_A> LEFT JOIN <tabela_B>
ON <tabela_A.coluna 1> = <tabela_B.coluna 1>
```




```
SELECT * FROM <tabela_A> LEFT JOIN <tabela_B>
ON <tabela_A.coluna 1> = <tabela_B.coluna 1>
WHERE tabela_B.coluna 1 IS NULL
```



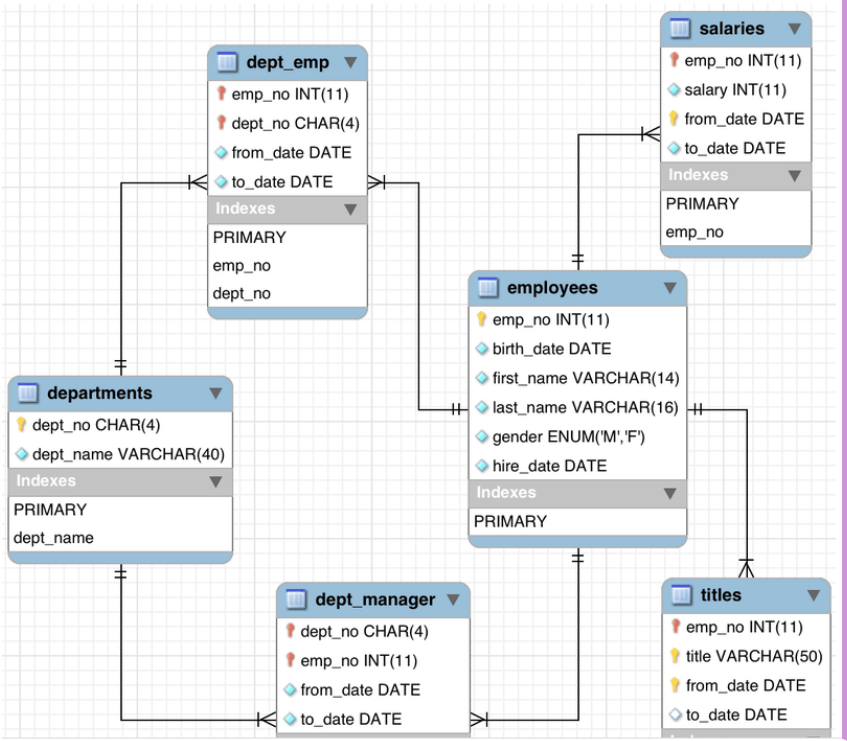
Fonte: https://rpubs.com/CristianaFreitas/311735

Left join

```
-- Funcionários e seus departamentos

SELECT *
FROM dept_emp a LEFT JOIN employees b
ON a.emp_no = b.emp_no
LEFT JOIN departments c
ON a.dept_no = c.dept_no
```

```
-- Departamentos sem gerentes

SELECT a.*
FROM departments a LEFT JOIN dept_mananger b
ON a.dept_no = b.dept_no
WHERE B.dept_no IS NULL
```

```
-- Funcionários que não são gerentes

SELECT a.*
FROM employees a LEFT JOIN dept_mananger b
ON a.emp_no = b.emp_no
WHERE b.emp_no IS NULL
```



Obrigada!

Contato:

Website RLadies Global: https://rladies.org/

MeetUp : https://www.meetup.com/pt -BR/R -Ladies -Sao -Paulo

Git: drimack

Linkedin : linkedin.com/in/reis -al