9 – Introdução

- A linguagem SQL oferece literalmente dezenas de funções internas que nos ajudam a solucionar problemas de tratamento de dados.

- Cada sistema gerenciador de banco de dados (SGBD) oferece uma biblioteca interna de funções específica do produto. Muitas destas funções seguem o padrão ANSI SQL, mas sempre existem pequenas variações de sintaxe, dependendo do SGBD utilizado.

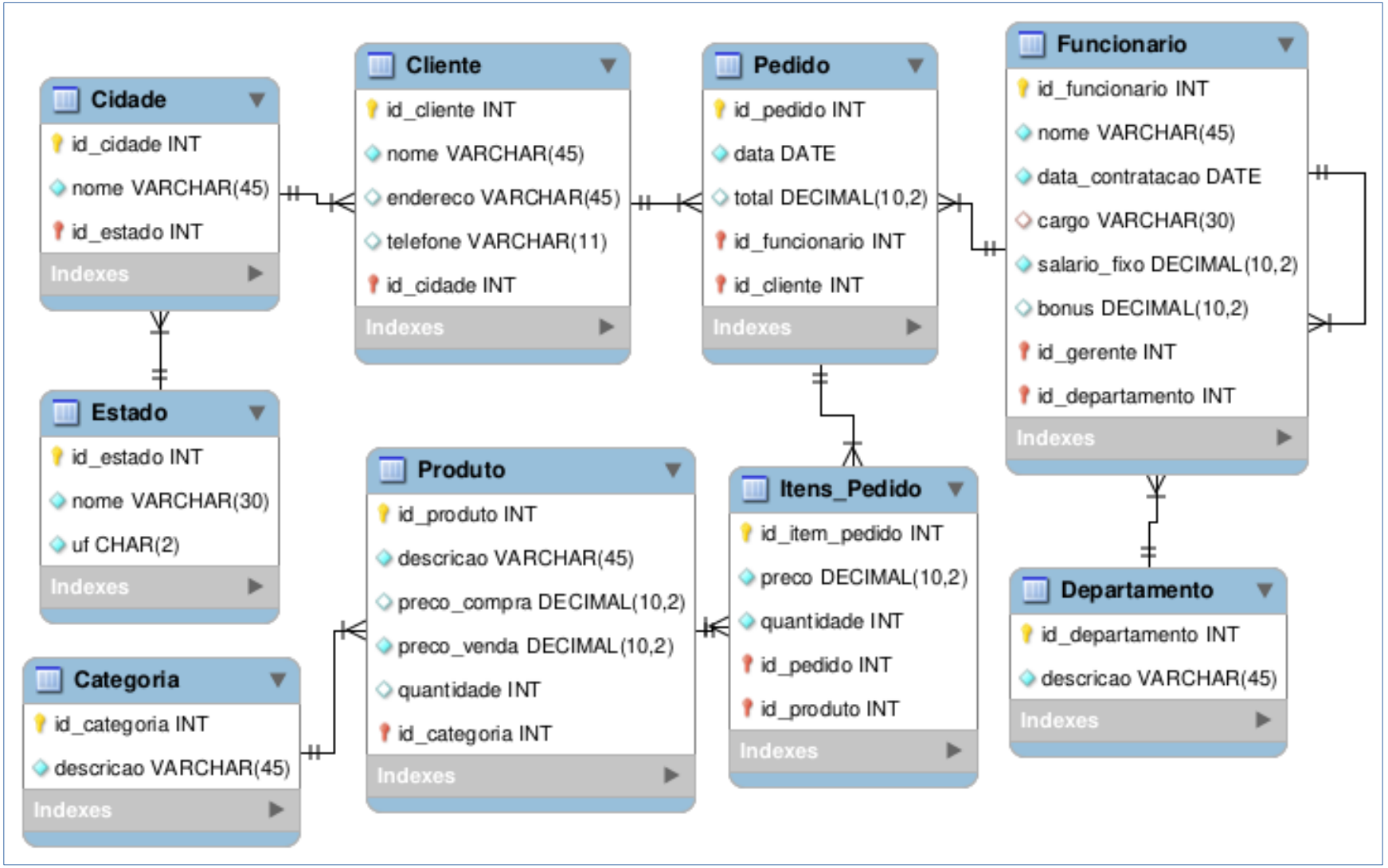
- Além das funções já oferecidas pelo SGBD, pode existir a necessidade de se efetuar operações aritméticas entre os campos das tabelas. A linguagem SQL oferece este recurso.

- Por exemplo, para calcular o valor total do salário de um funcionário, pode ser necessário somar a comissão e o salário fixo. Esse calculo pode ser feito diretamente por uma consulta SQL.

- A linguagem SQL também oferece recursos para operações com datas, horas e textos.

9.1 – Operações Aritméticas entre Colunas

Por meio da linguagem SQL é possível efetuar diversos tipos de cálculos aritméticos entre os campos de uma ou mais tabelas. Para exemplificar será usado o modelo de banco abaixo:



9.1 – Operações Aritméticas entre Colunas

- A consulta abaixo ilustra todos os funcionários cadastrados, onde e possível observar os campos salario_fixo e bonus. O total de ganhos do funcionário é a soma destes dois campos:

```
SELECT * FROM funcionario;
```

id_funcionario	nome	data_contratacao	cargo	salario_fixo	bonus	id_departamento	id_gerente
1	Mario Bros	2007-05-21	Presidente	15000.00	2000.00	2	*NULL*
2	Luigi Bros	2009-05-21	Diretor	12000.00	1000.00	2	1
3	Paulo Nobre	2002-07-20	Gerente	13000.00	2000.00	5	2
4	Bill Gates	2015-05-11	Programador	5000.00	500.00	3	2
5	Tony Stark	2002-09-21	Técnico	2000.00	1000.00	3	4
6	José da Silva	2009-05-21	Gerente	2000.00	200.00	4	2
7	João da Garrincha	2001-01-22	Vendedor	1000.00	200.00	4	6
8	John Rambo	2001-04-21	*NULL*	1000.00	*NULL*	4	6
9	Joaquim Fernandez	2004-01-22	Vendedor	1000.00	200.00	4	6
10	Maria da Silva	2003-02-26	Técnico de RH	3000.00	300.00	1	2

9.1 – Operações Aritméticas entre Colunas

- Para exemplificar o cálculo entre colunas, vamos utilizar a tabela de funcionários. Supondo que o rendimento de cada funcionário é formado pela soma das colunas ‘salario_fixo’ e ‘bonus’.

- O rendimento total pode ser calculado por um comando SQL que efetua a soma das colunas, como demonstra o código abaixo:

```
SELECT nome, salario_fixo, bonus, salario_fixo+bonus FROM funcionario ORDER BY salario_fixo+bonus;
```

nome	salario_fixo	bonus	salario_fixo+bonus ▲ 1
John Rambo	1000.00	*NULL*	*NULL*
João da Garrincha	1000.00	200.00	1200.00
Joaquim Fernandez	1000.00	200.00	1200.00
José da Silva	2000.00	200.00	2200.00
Tony Stark	2000.00	1000.00	3000.00
Maria da Silva	3000.00	300.00	3300.00
Bill Gates	5000.00	500.00	5500.00
Luigi Bros	12000.00	1000.00	13000.00
Paulo Nobre	13000.00	2000.00	15000.00
Mario Bros	15000.00	2000.00	17000.00

9.1 – Operações Aritméticas entre Colunas

- Como é possível observar, um campo virtual é gerado mostrando o resultado da operação aritmética envolvendo os demais campos. Por questões visuais, poderíamos criar um rótulo para esse campo virtual, por meio do comando “**AS**”.

- O exemplo abaixo ilustra uma consulta que cria um rótulo chamado “Salario_Total” para a coluna virtual que exibe o resultado da operação para todos os funcionários cujo nome começa com a letra “P”:

```
SELECT nome, salario_fixo, bonus, salario_fixo+bonus AS ‘Salário Total’ FROM funcionario
WHERE nome LIKE 'P%' ORDER BY salario_fixo+bonus;
```

nome	salario_fixo	bonus	Salário Total
Paulo Nobre	13000.00	2000.00	15000.00

9.1 – Operações Aritméticas entre Colunas

- Além de operações aritméticas entre os campos de uma tabela, também é possível efetuar operações entre os campos e uma constante qualquer.

- No caso do exemplo abaixo, será calculado o valor do salário final dos funcionários que ganham um salário liquido superior a 2000. O calculo funciona da seguinte maneira: O salário total é formado pela soma dos campos ‘salario_fixo’ e ‘bonus’. Entretanto, deve ser reduzido 11% de tributação. O comando abaixo faz esse cálculo:

```
SELECT funcionario.nome, departamento.nome, salario_fixo, bonus, (salario_fixo+bonus) AS ‘Bruto’, ((salario_fixo+bonus)*0.11) AS Desconto, ((salario_fixo+bonus)*0.89) AS ‘Liquido’ FROM funcionario JOIN departamento USING(id_departamento) WHERE ((salario_fixo+bonus)*0.89)>2000
```

nome	nome	salario_fixo	bonus	‘Bruto’	Desconto	‘Liquido’
Maria da Silva	RH	3000.00	300.00	3300.00	363.0000	2937.0000
Mario Bros	Diretoria	15000.00	2000.00	17000.00	1870.0000	15130.0000
Luigi Bros	Diretoria	12000.00	1000.00	13000.00	1430.0000	11570.0000
Bill Gates	TI	5000.00	500.00	5500.00	605.0000	4895.0000
Tony Stark	TI	2000.00	1000.00	3000.00	330.0000	2670.0000
Paulo Nobre	Financeiro	13000.00	2000.00	15000.00	1650.0000	13350.0000

9.2 – Tratamento de Colunas Null (Coalesce)

- Sempre que um campo com valor **NULL** é utilizado em qualquer tipo de calculo, o resultado sempre será NULL, independente do valor dos demais campos

- O exemplo abaixo ilustra a situação em que um empregado tem o campo “bonus” com valor null. Portanto, a operação “salario+bonus“ também resultará em um valor null.

```
SELECT nome, salario_fixo, bonus, salario_fixo+bonus AS Salario_Total FROM funcionario;
```

nome	salario_fixo	bonus	Salario_Total
Mario Bros	15000.00	2000.00	17000.00
Luigi Bros	12000.00	1000.00	13000.00
Paulo Nobre	13000.00	2000.00	15000.00
Bill Gates	5000.00	500.00	5500.00
Tony Stark	2000.00	1000.00	3000.00
José da Silva	2000.00	200.00	2200.00
João da Garrincha	1000.00	200.00	1200.00
John Rambo	1000.00	*NULL*	*NULL*
Joaquim Fernandez	1000.00	200.00	1200.00
Maria da Silva	3000.00	300.00	3300.00

9.2 – Tratamento de Colunas Null (Coalesce)

- Para solucionar este problema, existe a função **COALESCE**. Ela é capaz de substituir o valor null de um campo, por um valor padrão (somente na consulta, na tabela ainda continuará null).

- No exemplo abaixo, a função COALESCE será usada nas colunas “bonus” e “salario_fixo”, pois caso uma delas tenha valor NULL, o resultado final será NULL também:

```
SELECT nome, COALESCE(salario_fixo, 0), COALESCE(bonus, 0), COALESCE(salario_fixo, 0) +
COALESCE(bonus, 0) AS Salario_Total FROM funcionario;
```

nome	COALESCE(salario_fixo, 0)	COALESCE(bonus, 0)	Salario_Total
Mario Bros	15000.00	2000.00	17000.00
Luigi Bros	12000.00	1000.00	13000.00
Paulo Nobre	13000.00	2000.00	15000.00
Bill Gates	5000.00	500.00	5500.00
Tony Stark	2000.00	1000.00	3000.00
José da Silva	2000.00	200.00	2200.00
João da Garrincha	1000.00	200.00	1200.00
John Rambo	1000.00	0.00	1000.00
Joaquim Fernandez	1000.00	200.00	1200.00
Maria da Silva	3000.00	300.00	3300.00

9.2 – Tratamento de Colunas Null (Coalesce)

- O comando COALESCE também pode ser usado para campos de textos, deixando as consultas com uma aparência melhor. No exemplo abaixo, todos os empregados que possuem o campo “cargo” com valor nulo, terão seu valor substituído por “Desconhecido”:

```
SELECT nome, COALESCE(cargo, 'Desconhecido'), COALESCE(salario_fixo, 0),
COALESCE(bonus, 0), COALESCE(salario_fixo, 0)+COALESCE(bonus, 0) AS Salario_Total FROM
funcionario;
```

nome	COALESCE(cargo, 'Desconhecido')	COALESCE(salario_fixo, 0)	COALESCE(bonus, 0)	Salario_Total
Mario Bros	Presidente	15000.00	2000.00	17000.00
Luigi Bros	Diretor	12000.00	1000.00	13000.00
Paulo Nobre	Gerente	13000.00	2000.00	15000.00
Bill Gates	Programador	5000.00	500.00	5500.00
Tony Stark	Técnico	2000.00	1000.00	3000.00
José da Silva	Gerente	2000.00	200.00	2200.00
João da Garrincha	Vendedor	1000.00	200.00	1200.00
John Rambo	Desconhecido	1000.00	0.00	1000.00
Joaquim Fernandez	Vendedor	1000.00	200.00	1200.00
Maria da Silva	Técnico de RH	3000.00	300.00	3300.00

9.3 – Funções de Arredondamento

- Pode ser necessário, após o término de um cálculo, determinar o número de casas decimais que se deseja exibir o resultado. O MYSQL fornece diversas funções de arredondamento de valores, são elas:

1 - **CEILING**: Arredonda os valores para cima

SELECT CEILING(15.85 + 1.07) – Resultado: 17

2 - **FLOOR**:Arredonda os valores para baixo

SELECT FLOOR(15.85 + 1.07) – Resultado: 16

9.3 – Funções de Arredondamento

3 - **ROUND**: Arredonda para o inteiro mais próximo. É possível determinar o número de casas decimais que se deseja exibir o resultado.

SELECT ROUND(-1.23); Resultado: -1

SELECT ROUND(-1.58); Resultado: -2

SELECT ROUND(1.58); Resultado:2

SELECT ROUND(1.298, 1); Resultado: 1.3

SELECT ROUND(1.298, 0); Resultado: 1

9.3 – Funções de Arredondamento

4 - **TRUNCATE**: Elimina casas decimais, de acordo com o parametro passado.

SELECT TRUNCATE(1.223, 1); Resultado: 1.2

SELECT TRUNCATE(1.999, 1); Resultado: 1.9

SELECT TRUNCATE(1.999, 0); Resultado: 1

SELECT TRUNCATE(-1.999, 1); Resultado: -1.9

5 - **FORMAT**: Formata os valores com o número de casas decimais passado como parâmetro.

SELECT FORMAT(-1.5845, 2); Resultado: -1.58

SELECT FORMAT(1.5845, 2); Resultado: 1.58

9.3 – Funções de Arredondamento

- As funções de arredondamento podem ser usadas para formatar os valores vindos de colunas das tabelas. Como exemplo, a consulta abaixo retorna o salário diário de todos os funcionários, e como o valor não é inteiro, é preciso efetuar arredondamento:

```
SELECT nome, salario_fixo, FORMAT((salario_fixo/30),2) AS 'Salário Diário' FROM funcionario;
```

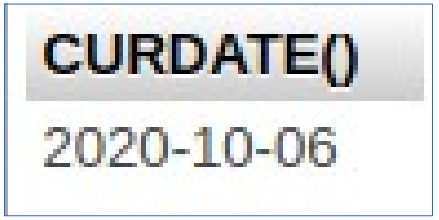
nome	salario_fixo	Salário Diário
Mario Bros	15000.00	500.00
Luigi Bros	12000.00	400.00
Paulo Nobre	13000.00	433.33
Bill Gates	5000.00	166.67
Tony Stark	2000.00	66.67
José da Silva	2000.00	66.67
João da Garrincha	1000.00	33.33
John Rambo	1000.00	33.33
Joaquim Fernandez	1000.00	33.33
Maria da Silva	3000.00	100.00

9.4 – Funções de Data

- Trabalhar com datas em um banco de dados exige o conhecimento de uma série de funções específicas. As funções de manipulação de datas podem sofrer pequenas variações entre os diferentes bancos de dados.

- Para retornar a data atual do servidor de banco de dados deve-se utilizara função **CURDATE()** ou **CURRENT_DATE()**, como ilustra o exemplo abaixo:

```
SELECT CURDATE();
ou
SELECT CURRENT_DATE();
```



9.4 – Funções de Data

- Como é possível observar, o formato da data é diferente do padrão que costumamos utilizar (DD-MM-AAAA). A data exibida como resultado possui primeiro o ano, depois o mês, e por fim o dia (AAAA-MM-DD).

- A função **DATE_FORMAT()** possibilita formatar uma data de forma customizada, de acordo com as necessidades:

Sintaxe

DATE_FORMAT(<Data_para_Formatar>, <Formato_Desejado>)

```
SELECT DATE_FORMAT(CURDATE(), '%d/%m/%Y');
```

DATE_FORMAT(CURDATE(), '%d/%m/%Y
21/06/2017

9.4 – Funções de Data

- O parâmetro “%” é obrigatório antes de informar os caracteres de formato. A tabela abaixo ilustra diversos parâmetros possíveis para formatar datas:

Especificação	Descrição
%d	Dia do mês numérico(00..31)
%D	Dia do mês com sufixo (em Inglês)
%m	Mês, numérico(00..12)
%M	Nome do Mês(em Inglês)
%y	Ano, numérico (dois dígitos)
%Y	Ano, quatro dígitos numéricos
%j	Dia do ano (de 001 a 366)

9.4 – Funções de Data

- A consulta abaixo retorna nome, cargo, departamento e data de contratação (formatada no padrão utilizado no Brasil) de todos os funcionários do departamento ‘Vendas’:

```
SELECT funcionario.nome, cargo, departamento.nome, DATE_FORMAT(data_contratacao, '%d/%m/%Y') AS ‘Data de Contratação’ FROM funcionario JOIN departamento USING(id_departamento) WHERE departamento.nome=’Vendas’;
```

nome	cargo	nome	Data de Contratação
José da Silva	Gerente	Vendas	21/05/2009
João da Garrincha	Vendedor	Vendas	22/01/2001
John Rambo	*NULL*	Vendas	21/04/2001
Joaquim Fernandez	Vendedor	Vendas	22/01/2004

9.4 – Funções de Data

A função **EXTRACT()** possibilita extrair partes da data passada como parâmetro. Para usar a função "EXTRACT()" é preciso passar 2 parâmetros:

1 - Tipo de informação que a função retornará: DAY(Dia), MONTH(Mês) e YEAR(Ano) entre outros.

2 - Data que será extraída as informações.

O exemplo abaixo fragmenta a data retornada pela função CURDATE() em dia , mês e ano.

```
SELECT EXTRACT(DAY FROM CURDATE()) AS DIA, EXTRACT(MONTH FROM CURDATE())
AS MES, EXTRACT(YEAR FROM CURDATE()) AS ANO;
```

DIA	MES	ANO
21	6	2017

9.4 – Funções de Data

O exemplo abaixo seleciona todos os funcionários que foram contratados no ano de 2001:

```
SELECT funcionario.nome, cargo, departamento.nome, DATE_FORMAT(data_contratacao,
'%d/%m/%Y') AS ‘Data de Contratação’ FROM funcionario JOIN departamento
USING(id_departamento) WHERE EXTRACT(YEAR FROM data_contratacao)=’2001’;
```

nome	cargo	nome	Data de Contratação
João da Garrincha	Vendedor	Vendas	22/01/2001
John Rambo	*NULL*	Vendas	21/04/2001

9.4 – Funções de Data

- As funções **DAY(), MONTH() e YEAR()** possuem funcionalidades parecidas, retornando o dia, mês e ano, respectivamente, de uma terminada data, como ilustra o exemplo abaixo:

```
SELECT nome, cargo, DAY(data_contratacao) AS Dia, MONTH(data_contratacao) AS Mes FROM funcionario WHERE YEAR(data_contratacao)=2001;
```

nome	cargo	Dia	Mes
João da Garrincha	Vendedor	22	1
John Rambo	*NULL*	21	4

9.4 – Funções de Data

- Caso seja necessário adicionar determinada quantidade de dias em uma data, deve-se usar a função "**DATE_ADD()**". Esta função recebe dois parâmetros:

  1 - O primeiro é a data à qual desejamos acrescentar

  2 - Definimos o número de dias, meses ou anos que se deseja acrescentar. Deve-se colocar **DAY** (no caso de acrescentar dias), **MONTH** (no caso de meses) e **YEAR** (no caso de anos). Deve-se utilizar o termo **INTERVAL** para o intervalo.

```
SELECT CURDATE() AS Data_Atual, DATE_ADD(CURDATE(), INTERVAL 30 DAY) AS
Soma_30_dias, DATE_ADD(CURDATE(), INTERVAL 3  MONTH) AS Soma_3_Meses,
DATE_ADD(CURDATE(), INTERVAL 2 YEAR)AS Soma_2_Anos;
```

Data_Atual	Soma_30_dias	Soma_3_Meses	Soma_2_Anos
2017-06-27	2017-07-27	2017-09-27	2019-06-27

9.4 – Funções de Data

- O exemplo abaixo mostra uma listagem de pedidos feitos em 2016, onde é exibida uma coluna virtual com a data de entrega, que ocorre 10 dias após o pedido ser feito:

```
SELECT id_pedido, nome, data, DATE_ADD(data, INTERVAL 10 DAY) AS 'Data de Entrega', total
FROM pedido JOIN cliente USING(id_cliente) WHERE YEAR(data)='2016' ORDER BY data;
```

id_pedido	nome	data ▲ 1	Data de Entrega	total
4	William Bigode	2016-02-12	2016-02-22	1400.00
3	Roger Guedes	2016-03-22	2016-04-01	1400.00

9.4 – Funções de Data

- A função **DATE_SUB**() efetua a subtração (em dias, meses ou anos) de uma data passada como parâmetro. Como ilustra o exemplo abaixo:

```
SELECT CURDATE() AS Data_Atual, DATE_SUB(CURDATE(), INTERVAL 30 DAY) AS Sub_30_dias,
DATE_SUB(CURDATE(), INTERVAL 3 MONTH) AS Sub_3_Meses, DATE_SUB(CURDATE(),
INTERVAL 2 YEAR) AS Sub_2_Anos;
```

Data_Atual	Sub_30_dias	Sub_3_Meses	Sub_2_Anos
2017-06-27	2017-05-28	2017-03-27	2015-06-27

9.4 – Funções de Data

- Uma função, que embora não seja muito utilizada, pode ser útil em determinadas situações, é a **LAST_DAY()**, que retorna o último dia do mês de qualquer data passada como parâmetro. Como ilustra o exemplo abaixo:

```
SELECT LAST_DAY(CURDATE());
```

LAST_DAY(CURDATE())
2017-06-30

9.4 – Funções de Data

- A função **DATEDIFF**() retorna a diferença em dias entre duas datas passadas como parâmetro. O exemplo abaixo mostra como a função pode ser utilizada:

```
SELECT DATEDIFF('2017-01-19', '2016-01-12');
```

DATEDIFF('2017-01-19', '2016-01-12')
373

9.4 – Funções de Data

- O exemplo abaixo mostra uma consulta que retorna todos os funcionários que já possuem mais de 15 anos de trabalho na empresa. Para isso, é preciso encontrar a diferença entre a data atual e data de contratação dos funcionários com a função **DATEDIFF**():

```
SELECT id_funcionario, nome, FORMAT(DATEDIFF(CURDATE(), data_contratacao)/365, 1) AS 'Tempo' FROM funcionario WHERE DATEDIFF(CURDATE(), data_contratacao)/365>15 ORDER BY nome;
```

id_funcionario	nome ▲ 1	Tempo
7	João da Garrincha	19.7
9	Joaquim Fernandez	16.7
8	John Rambo	19.5
10	Maria da Silva	17.6
3	Paulo Nobre	18.2
5	Tony Stark	18.1

9.4 – Funções de Data

- Também é possível encontrar a diferença entre duas datas em meses, usando a função "**PERIOD_DIFF**()". Para isso é preciso passar duas datas no formato "YYMM" ou "YYYYMM" como parâmetro. Não é aceito o uso de separadores"-", o que faz esta função ser pouco utilizada:

```
SELECT PERIOD_DIFF('201712','201708');
```

PERIOD_DIFF('201712','201708')
4

9.4 – Funções de Data

- A função **DAYOFWEEK**() retorna, em forma de índice, o dia da semana da data passada como parâmetro (1 = Domingo, 2 = Segunda, ... 7 = Sábado). O exemplo abaixo ilustra o uso da função, onde a data 2017-06-28 é passada como parâmetro e o resultado é 4, o que indica que a data é uma quarta-feira.

```
SELECT DAYOFWEEK(‘2017-06-28’);
```

9.4 – Funções de Data

- O exemplo abaixo mostra todos os pedidos realizados somente nas quartas-feiras do mês de fevereiro de 2017:

```
SELECT id_pedido, data, nome, total FROM pedido JOIN cliente USING(id_cliente) WHERE YEAR(data)='2017' AND MONTH(data)='02' AND DAYOFWEEK(data)=4
```

id_pedido	data	nome	total
2	2017-02-22	Roger Guedes	1400

9.4 – Funções de Data

- A função **WEEKDAY**() retorna o índice do dia da semana para data passada com o parâmetro (0 = Segunda, 1 = Terça, ... 6 = Domingo). No caso do exemplo abaixo, o resultado indica que o dia da semana para a data especificada é quarta-feira:

```
SELECT WEEKDAY('2017-06-28');
```

WEEKDAY('2017-06-28')
2

9.4 – Funções de Data

- A função **DAYOFYEAR**() retorna o dia do ano para a data passada como parâmetro, variando de 1 até 366. No caso do exemplo abaixo, o resultado indica que a data passado como parâmetro representa o dia 34 do ano.

```
SELECT DAYOFYEAR(‘2017-02-03’);
```

DAYOFYEAR('2017-02-03')
34

9.5 – Funções de Hora

- A função **CURTIME**() retorna a hora atual como um valor no formato 'HH:MM:SS‘, como ilustra o exemplo abaixo:

```
SELECT CURTIME();
```

CURTIME()
17:14:38

9.5 – Funções de Hora

- As funções **HOUR**(), **MINUTE**(), **SECOND**() E **MICROSECOND**(), retornam a hora, minuto, segundo e microssegundo, respectivamente, de uma hora passada como parâmetro. O exemplo abaixo ilustra essa situação:

```
SELECT HOUR(CURTIME()) AS Hora, MINUTE(CURTIME()) AS Minutos,
SECOND(CURTIME()) AS Segundo;
```

Hora	Minutos	Segundo
17	16	22

9.5 – Funções de Hora

- A função **ADDTIME**() é capaz de fazer a soma entre dois valores do tipo “time”. Os microssegundos são separados dos segundos por ponto. O exemplo abaixo ilustra a soma de dois valores de tempo:

```
SELECT ADDTIME("01:00:00.999999", "02:00:00.999998");
```

ADDTIME("01:00:00.999999", "02:00:0
03:00:01.999997

9.5 – Funções de Hora

- Para subtrair dois valores do tipo “time” utiliza-se a função **SUBTIME**(). O calculo será feito da seguinte forma: O valor do primeiro parâmetro menos o valor do segundo parâmetro, como no exemplo abaixo:

```
SELECT SUBTIME(“23:01:01","02:00:00");
```

SUBTIME("23:01:01","02:00:00")
21:01:01

9.5 – Funções de Hora

- A função **TIMEDIFF**() retorna o tempo entre uma hora final e inicial passadas como parâmetro, como ilustra o exemplo abaixo:

```
SELECT TIMEDIFF('23:01:01', '20:01:01');
```

TIMEDIFF('23:01:01', '20:01:01')
03:00:00

9.6 – Funções de Data e Hora

- Como visto anteriormente, existem uma série de funções para manipular data e hora n banco de dados. Entretanto, também é possível trabalhar com data e hora dentro da mesma coluna.

- O MySQL possui os campos **TIMESTAMP** e **DATETIME**, que possuem a capacidade de armazenar data e hora. A função **NOW**() retorna a data e hora local, como ilustra o exemplo abaixo:

```
SELECT NOW();
```

now()
2017-07-04 18:04:22

9.6 – Funções de Data e Hora

- Caso seja necessário exibir apenas a data ou a hora de um campo DATETIME, pode-se utilizar as funções **DATE**() e **TIME**(), respectivamente:

```
SELECT NOW() AS Data_Hora, DATE(NOW()) AS Data, TIME(NOW()) AS Hora;
```

Data_Hora	Data	Hora
2017-07-04 18:06:07	2017-07-04	18:06:07

9.7 – Funções de Manipulação de Strings

- A função **ASCII**() determina o número referente a tabela ASCII de um caractere qualquer. Basta passar o caractere como parâmetro que a função retorna seu respectivo código ASCII.

- Já a função **CHAR**() faz exatamente o contrário. Deve-se passar como parâmetro o código ASCII, e a função retorna o caractere, como ilustra o exemplo abaixo:

```
SELECT ASCII('P'), CHAR(80);
```

```
+-------------+-----------+
| ASCII('P') | CHAR(80) |
+-------------+-----------+
|          80 | P         |
+-------------+-----------+
```

9.7 – Funções de Manipulação de Strings

- A função **BIN**() retorna o binário do caractere que for passado como parâmetro, como ilustra o exemplo abaixo:

```
SELECT BIN('15');
```

BIN(15)
1111

9.7 – Funções de Manipulação de Strings

- A função **CONCAT**() efetua a concatenação (junção) das colunas ou caracteres passados como parâmetro, como ilustra o exemplo abaixo:

```
SELECT CONCAT('My', 'S', 'QL');
```

CONCAT('My', 'S', 'QL')
MySQL

9.7 – Funções de Manipulação de Strings

- A função **CONCAT**() pode ser usada para concatenar campos das tabelas, como no exemplo abaixo, onde são concatenados os nomes e salário dos funcionários:

```
SELECT CONCAT(nome, ' - Salário: ', salario_fixo) FROM funcionario WHERE
salario_fixo>2000;
```

CONCAT(nome, ' - Salário: ', salario_fix(
Mario Bros - Salário: 15000
Luigi Bros - Salário: 12000
Paulo Nobre - Salário: 13000
Bill Gates - Salário: 5000
John Rambo - Salário: 3000

9.7 – Funções de Manipulação de Strings

- Uma outra função de concatenação é o **CONCAT_WS**(), que significa CONCAT With separator (CONCAT com separador). O primeiro argumento é o separador para os outros argumentos. O separador é adicionado entre as strings que serão concatenadas, como no exemplo abaixo:

```
SELECT CONCAT_WS('-','My','SQL');
```

CONCAT_WS('-','My','SQL')
My-SQL

9.7 – Funções de Manipulação de Strings

- A função **CONV**() converte caracteres entre diferentes bases. Os parâmetros devem ser colocados na seguinte ordem: Caractere que se deseja converter, base atual, base para qual se deve converter. No exemplo abaixo, deseja-se converter o caractere “45” da base decimal para hexadecimal.

```
SELECT CONV(45,10,16);
```

CONV(45,10,16)
2D

9.7 – Funções de Manipulação de Strings

- A função **INSTR**()retorna a posição da primeira ocorrência de uma determinada substring dentro de uma string qualquer. No caso do exemplo abaixo, a posição de “nc” se inicia na posição 3 da string “Banco de Dados”.

```
SELECT INSTR(‘Banco de Dados', ‘nc’);
```

INSTR('Banco de Dados', 'nc')
3

9.7 – Funções de Manipulação de Strings

- As funções **LOWER**() ou **LCASE**() retornam o conteúdo de seu parâmetro em minúsculo. Assim como as funções **UCASE**() ou **UPPER**() retornam a string passada como parâmetro em maiúsculo.

```
SELECT LOWER(nome) AS 'Nome em Minusculo', UPPER(nome) AS 'Nome em Maiusculo' FROM funcionario;
```

Nome em Minusculo	Nome em Maiusculo
mario bros	MARIO BROS
luigi bros	LUIGI BROS
paulo nobre	PAULO NOBRE
bill gates	BILL GATES
tony stark	TONY STARK
josé da silva	JOSÉ DA SILVA
joão da garrincha	JOÃO DA GARRINCHA
john rambo	JOHN RAMBO
joaquim fernandez	JOAQUIM FERNANDEZ
maria da silva	MARIA DA SILVA

9.7 – Funções de Manipulação de Strings

- A função **LENGTH()** retorna o tamanho da string passada como parâmetro, como ilustra o exemplo abaixo:

```
SELECT LENGTH(‘Bando de Dados’);
```

LENGTH('Bando de Dados')
14

9.7 – Funções de Manipulação de Strings

- A função **REPEAT()** retorna uma determinada string passada como parâmetro repetidas vezes, de acordo com o número passado no segundo parâmetro, como ilustra o exemplo abaixo, onde a string “Banco de Dados” é repetida 3 vezes:

```
SELECT REPEAT(‘Bando de Dados ’, 3);
```

REPEAT('Bando de Dados ', 3)
Bando de Dados Bando de Dados Bando de Dados

9.7 – Funções de Manipulação de Strings

- A função **REPLACE()** serve para substituir partes de uma string por uma outra substring qualquer passada como parâmetro. Como ilustra o exemplo, onde em cada ocorrência da substring “w”, está será substituída por “W”.

```
SELECT REPLACE('www.mysql.com', 'w', 'W');
```

REPLACE('www.mysql.com', 'w', 'W')
WWW.mysql.com

9.7 – Funções de Manipulação de Strings

- A função **REVERSE()**inverte a ordem de uma string passada como parâmetro, como ilustra o exemplo abaixo:

```
SELECT REVERSE('abc');
```

REVERSE('abc')
cba

9.7 – Funções de Manipulação de Strings

- Pode ocorrer, principalmente quando as strings são geradas a partir de formulários onde o próprio usuário digita a informação, de existirem espaços em branco no inicio e no final da string.

- Para retirar os espaços em branco temos as seguintes funções:

1 - TRIM(): Retira os espaços em branco do inicio e do final da string.

2 - LTRIM(): Tira os espaços em branco do inicio da string.

3 - RTRIM(): Tira os espaços em branco do final da string.

9.7 – Funções de Manipulação de Strings

- Vejamos os exemplos abaixo:

```
SELECT TRIM('   abc   ');
```

TRIM(' abc ')
abc

```
SELECT LTRIM('   abc   ');
```

LTRIM(' abc ')
abc

```
SELECT RTRIM('   abc   ');
```

RTRIM(' abc ')
abc

9.7 – Funções de Manipulação de Strings

- Também pode ser necessário tirar algum outro caractere do inicio e do final da string. Para isso podemos usar o complemento **BOTH** para função **TRIM**(), como ilustra o exemplo abaixo:

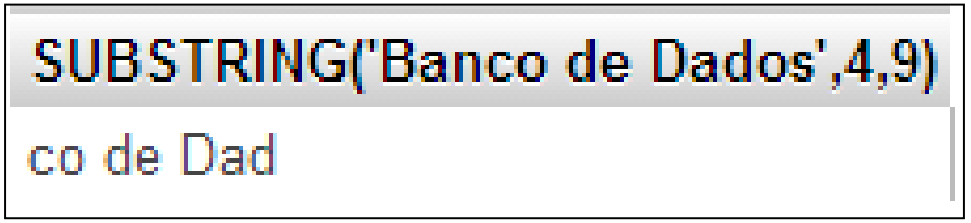
```
SELECT TRIM(BOTH'xy' FROM'xyxybarxyxy');
```

TRIM(BOTH 'xy' FROM 'xyxybarxyxy')
bar

9.7 – Funções de Manipulação de Strings

- A função **SUBSTRING**() é utilizada para retornar partes da string. Deve-se passar como parâmetro a string, a posição inicial e a quantidade de caracteres que se deve retornar, como ilustra o exemplo abaixo:

```
SELECT SUBSTRING(‘Banco de Dados’, 4, 9);
```



9.7 – Funções de Manipulação de Strings

- Caso o último parâmetro, que indica o tamanho da substring, não seja informado, a função **SUBSTRING**() irá retornar toda a string da posição inicial indicada até o final, como ilustra o exemplo abaixo:

```
SELECT SUBSTRING(‘Banco de Dados’, 4);
```

SUBSTRING('Banco de Dados',4)
co de Dados