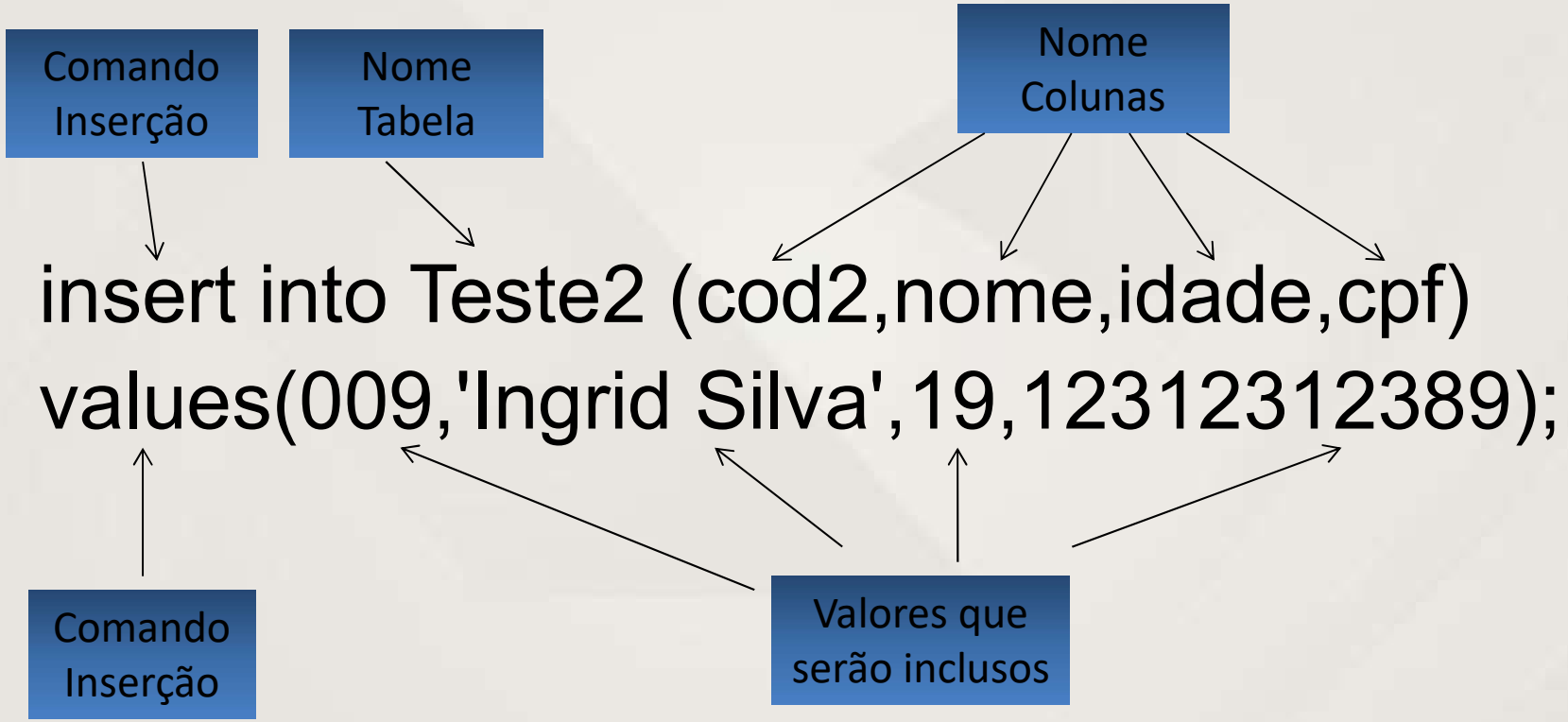
Conceitos Básicos de SQL: DML

DML (*Data manipulation Language*):

- Os principais comandos são:
  - Insert; (Inserir tuplas)
  - Update; (Alterar tuplas)
  - Delete; (Excluir tuplas)
  - Select; (Pesquisar dados nas tuplas)

DML: Comandos para **inserir** dados em uma tabela:

Para a inclusão de dados em uma tabela utilizamos o comando INSERT INTO / VALUES. Veja o exemplo:



Commit Automático Exibição 10

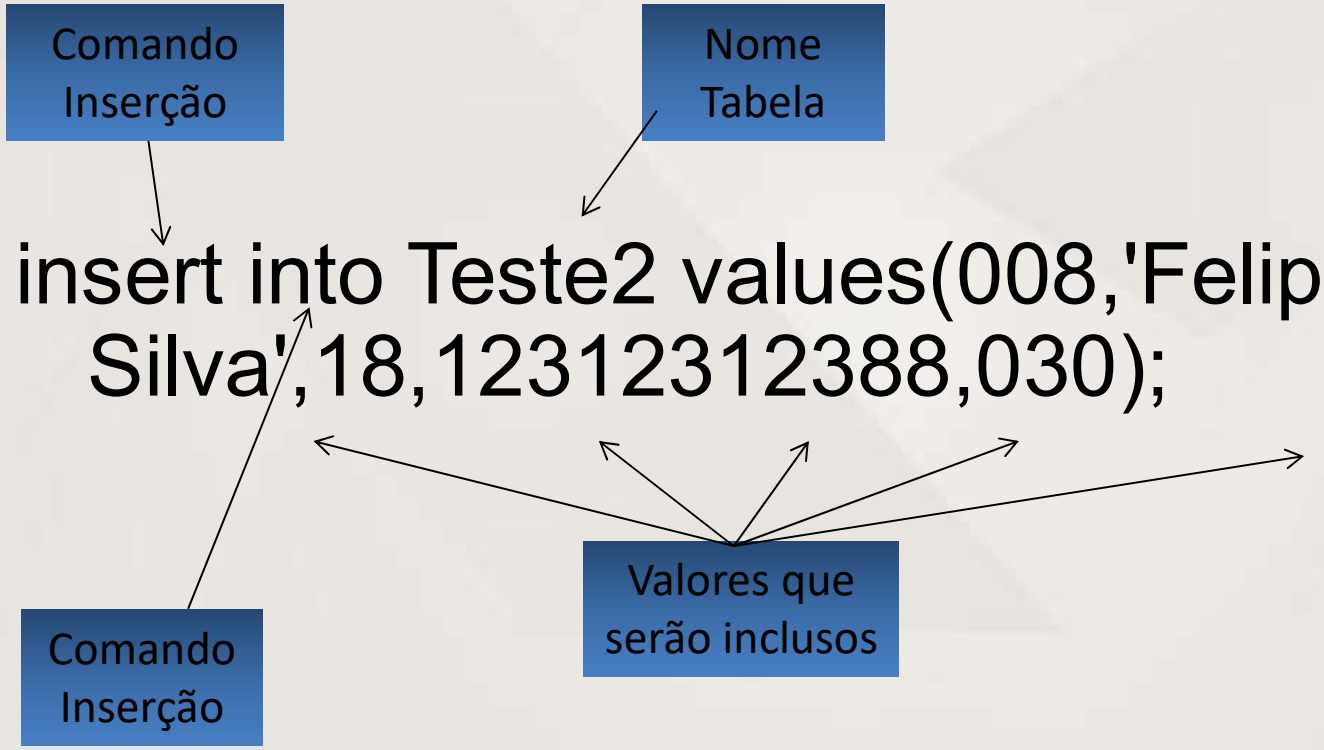
```
insert into Teste2 (cod2,nome,idade,cpf)
values(009,'Ingrid Silva',19,12312312389);
```

Resultados Explicação Descrever Instrução SQL Salva Histórico

```
1 linha(s) inserida(s).
```

0,00 segundos

É possível fazer a inclusão de dados em uma tabela sem especificar as colunas porém os dados devem estar na mesma ordem que serão inclusos. Veja o exemplo:



Commit Automático Exibição 10

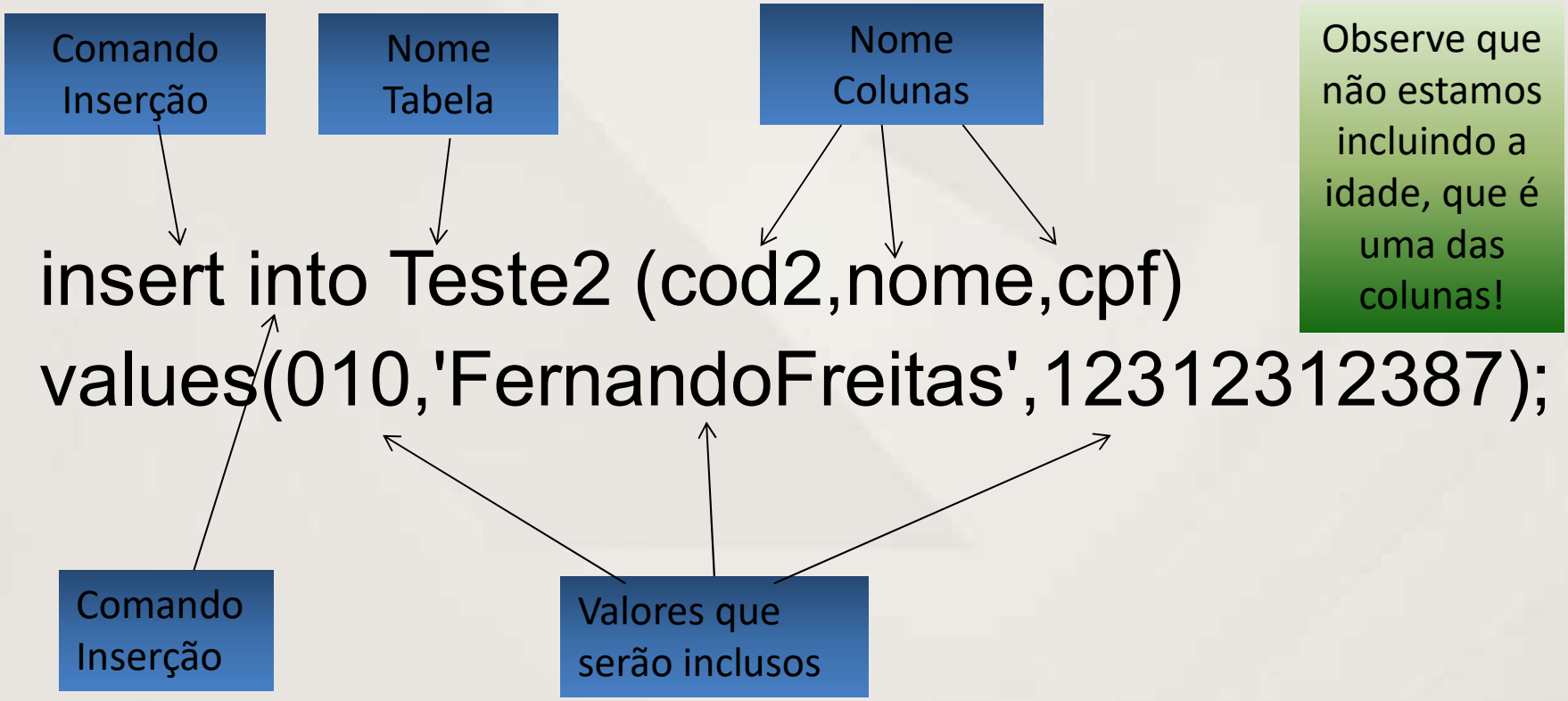
```
insert into Teste2
values(008,'Felipe da Silva',22,12312312388,030);
```

Resultados Explicação Descrever Instrução SQL Salva Histórico

```
1 linha(s) inserida(s).
```

0,00 segundos

Outra característica é a possibilidade de inclusão de dados em uma tabela sem precisar popular todas as colunas, porém para isso precisamos declarar as colunas que irão receber o dado. Veja o exemplo:



Commit Automático Exibição 10

```
insert into Teste2 (cod2,nome,cpf)
values(010,'Fernanda freitas',12312312387);
```

Resultados Explicação Descrever Instrução SQL Salva Histórico

```
1 linha(s) inserida(s).

0,00 segundos
```

Quando incluímos dados em tabelas que possuem chave estrangeiras (FK) referenciando outras tabelas, os valores que estão sendo inseridos nas colunas da chave estrangeira já devem constar na chave primária da tabela referenciada.

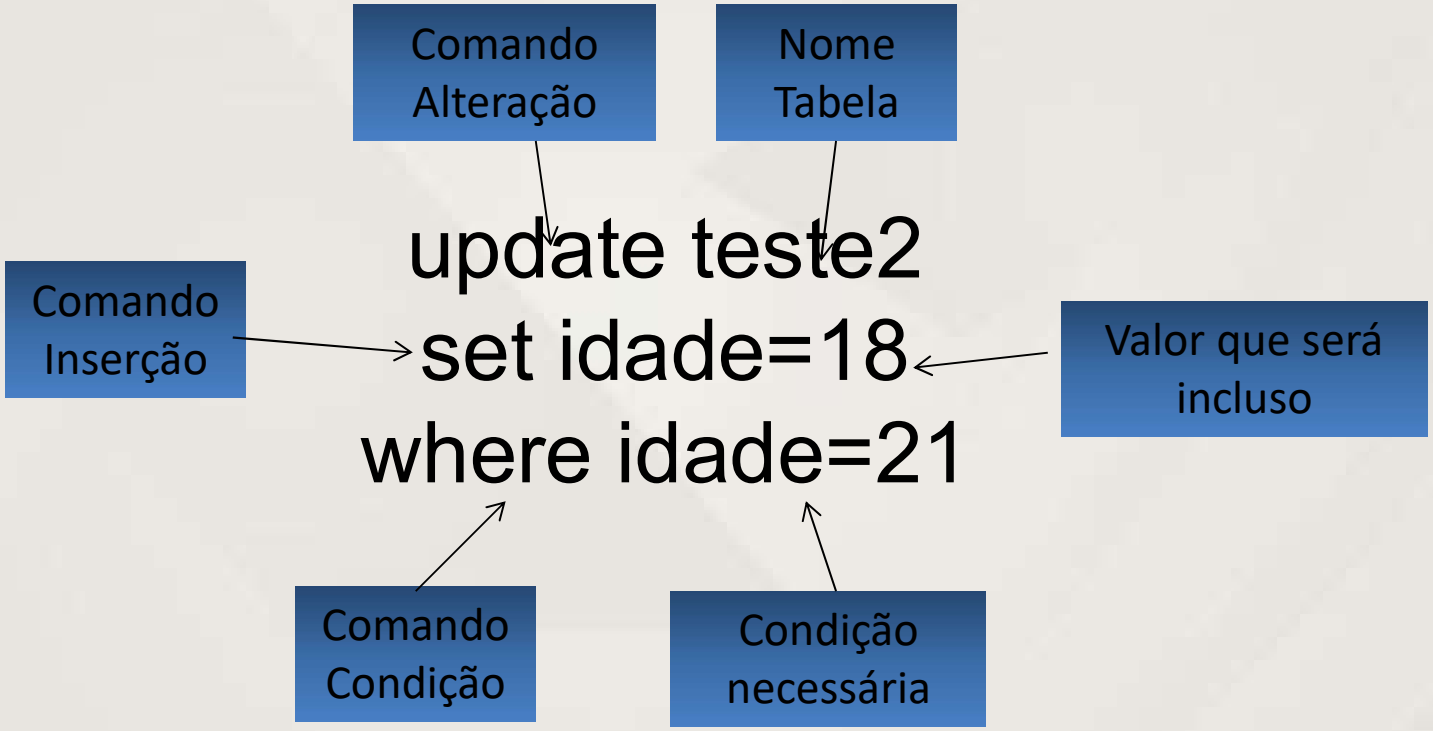
Normalmente o banco apresenta uma mensagem de erro, por exemplo:

insert into teste8 values(045)

```
ORA-02291: restrição de integridade (SYSTEM.FK_TESTE2_COD) violada - chave mãe não
localizada
```

DML: Comandos para **alterar** dados em uma tabela:

Para a alteração de dados em uma tabela utilizamos o comando UPDATE / SET. Veja o exemplo:



Commit Automático Exibição 10

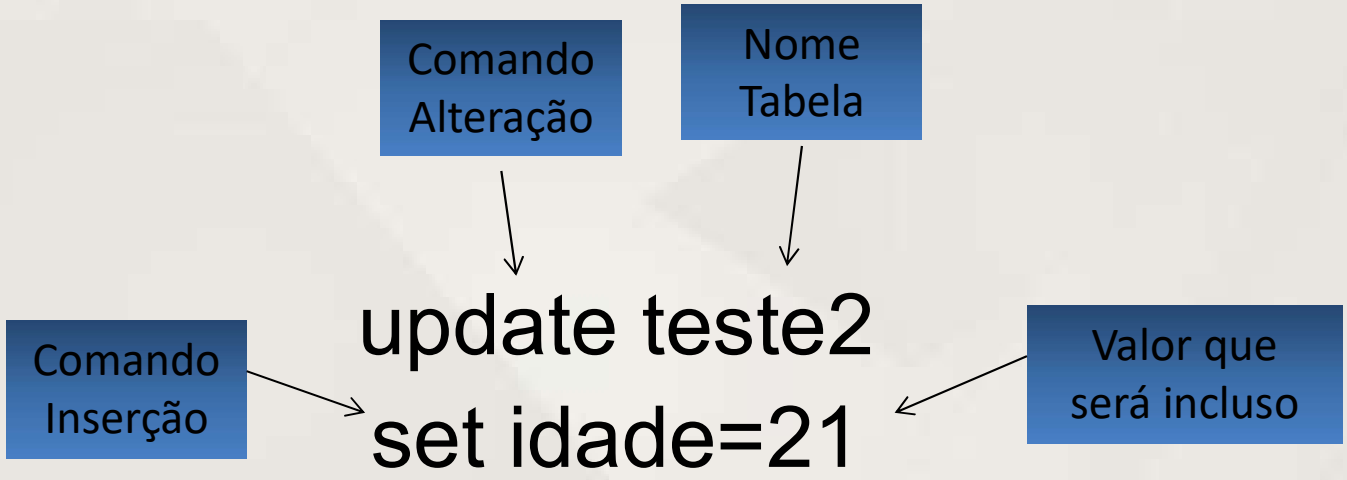
```
update teste2
set idade=18
where idade=21
```

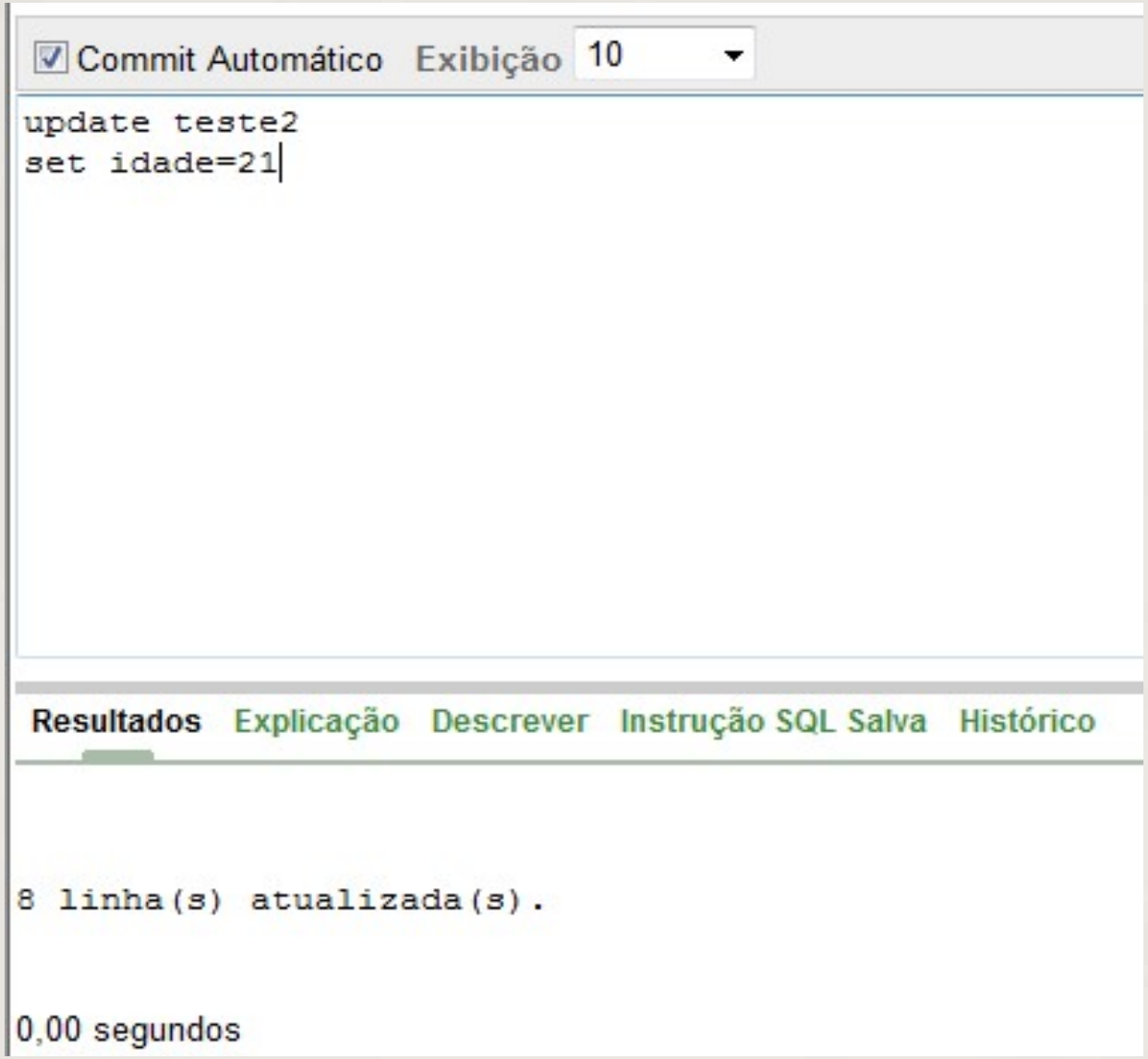
Resultados Explicação Descrever Instrução SQL Salva Histórico

```
3 linha(s) atualizada(s).
```

0,00 segundos

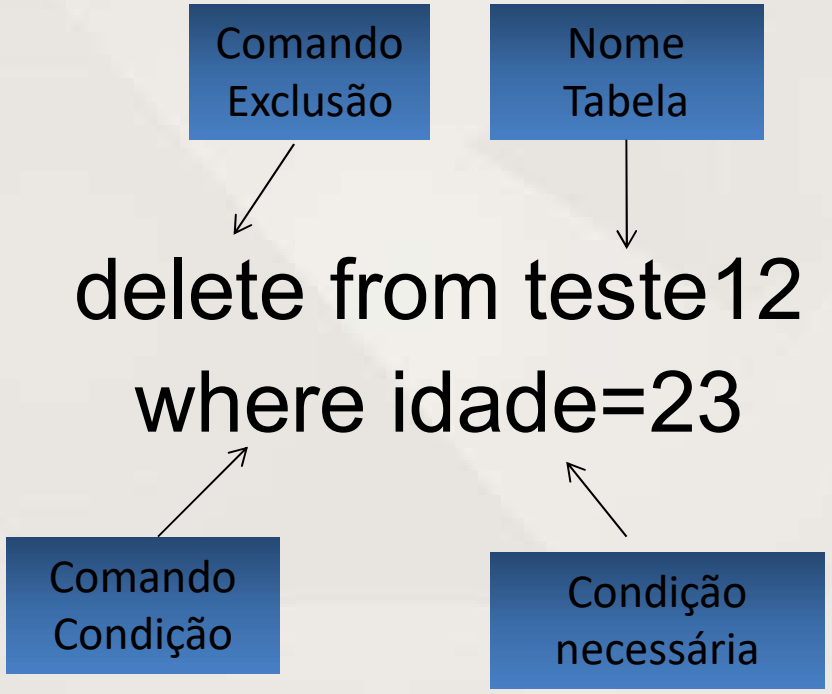
Se não for utilizado a clausula WHERE para determinar a condição, todas as tuplas da tabela sofrerão a mudança.




Commit Automático
Exibição
10
update teste2
set idade=21
Resultados
Explicação
Descrever
Instrução SQL Salva
Histórico
8 linha(s) atualizada(s).
0,00 segundos

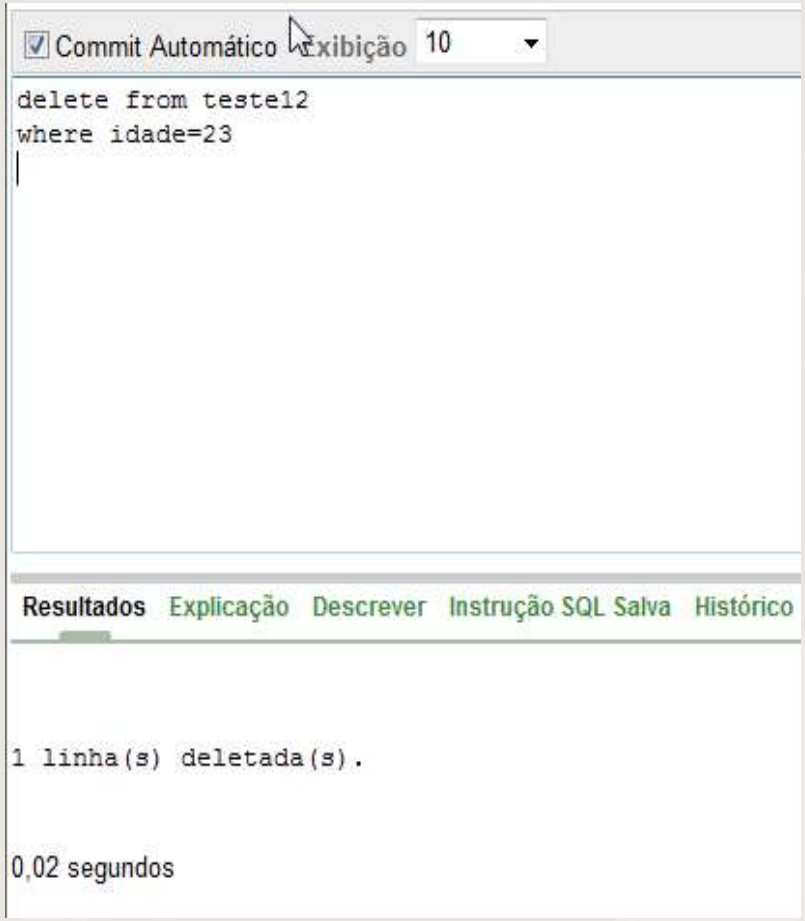
DML: Comandos para **excluir** dados em uma tabela:

Para a alteração de dados em uma tabela utilizamos o comando DELETE. Veja o exemplo:



COD	NOME	IDADE
1	Teste1	21
2	Teste2	22
3	Teste3	21
4	Teste4	23
5	Teste5	26
6	Teste6	27

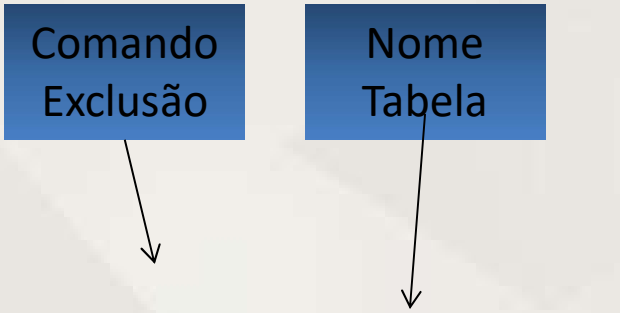
Antes



Depois

COD	NOME	IDADE
1	Teste1	21
2	Teste2	22
3	Teste3	21
5	Teste5	26
6	Teste6	27

Se não for utilizado a clausula WHERE para determinar a condição, todas as tuplas da tabela sofrerão a mudança.



COD	NOME	IDADE
1	Teste1	21
2	Teste2	22
3	Teste3	21
5	Teste5	26
6	Teste6	27

Antes

Commit Automático Exibição 10

```
delete from teste12
```

Resultados Explicação Descrever Instrução SQL Salva Histórico

```
5 linha(s) deletada(s).

0,00 segundos
```

Depois

Resultados Explicação Descrever

dados não encontrados

DML: SELECT

O Comando SELECT é com certeza o comando mais utilizado em sistemas comerciais, é através dele que são feitas pesquisas no banco. O Select simples consiste de duas clausulas:

SELECT e FROM

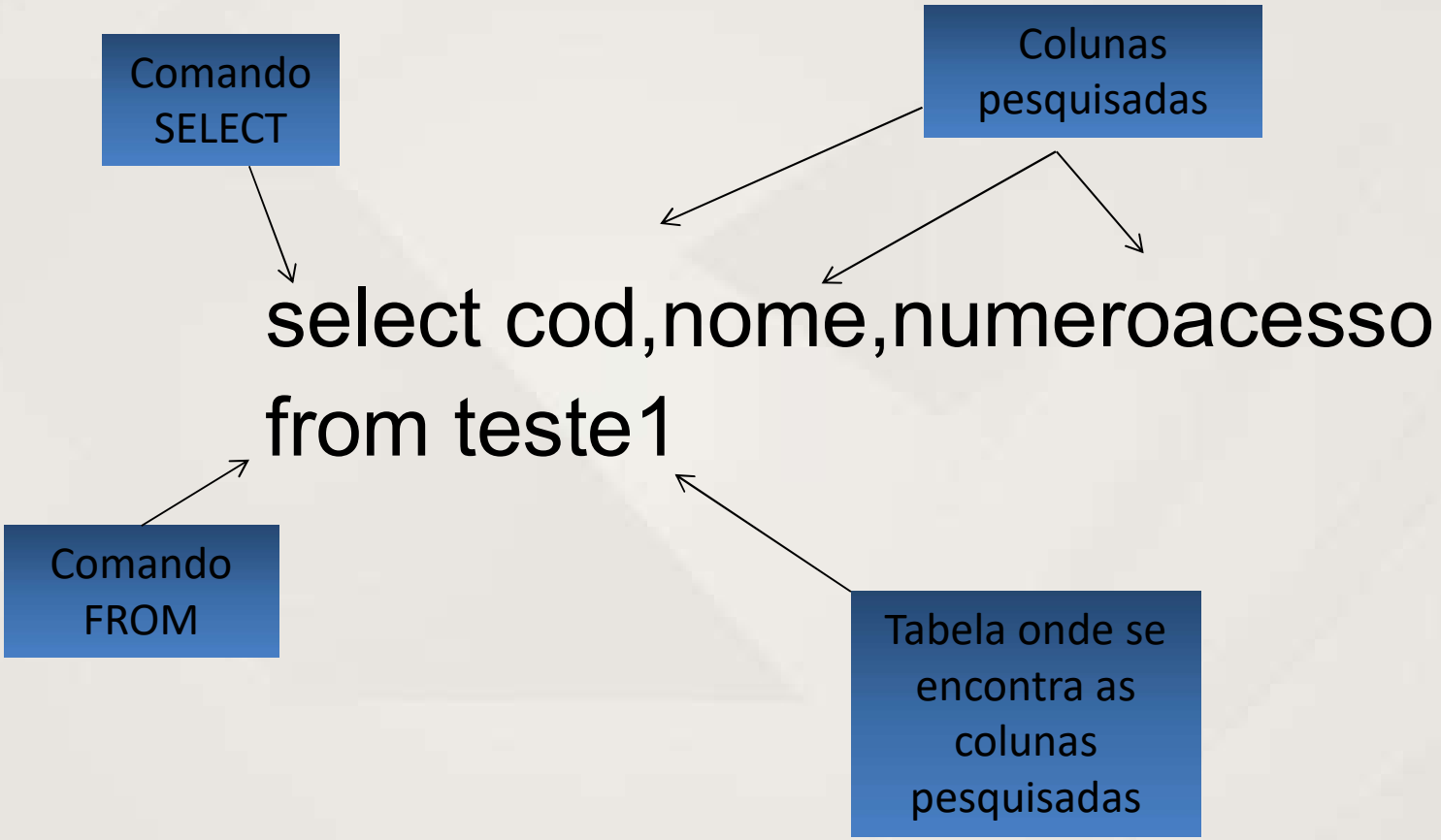
DML: SELECT

SELECT: É o comando de consulta, é aqui onde é definido o que se espera como resposta da pesquisa.

FROM: Identifica onde se encontra as informações pesquisas, ou seja, em que tabelas os dados estão armazenados.

DML: SELECT

Exemplo de um *select* simples:



Commit Automático Exibição 10

```
select cod,nome,numeroacesso
from teste1
```

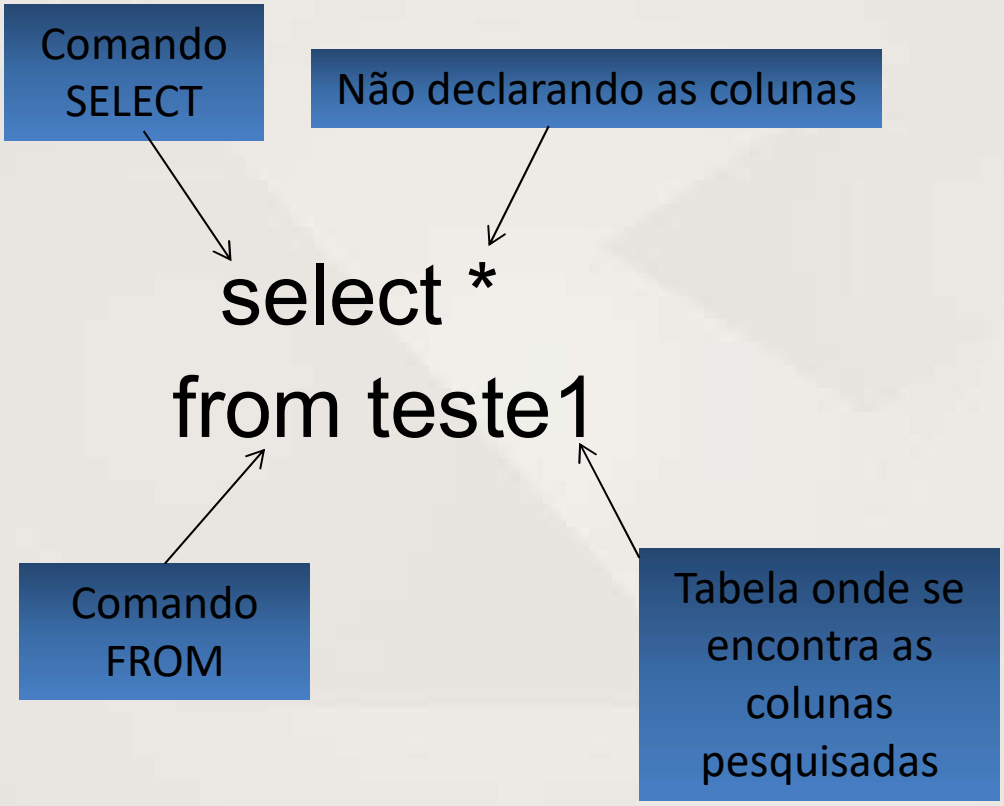
Resultados Explicação Descrever Instrução SQL Salva Histórico

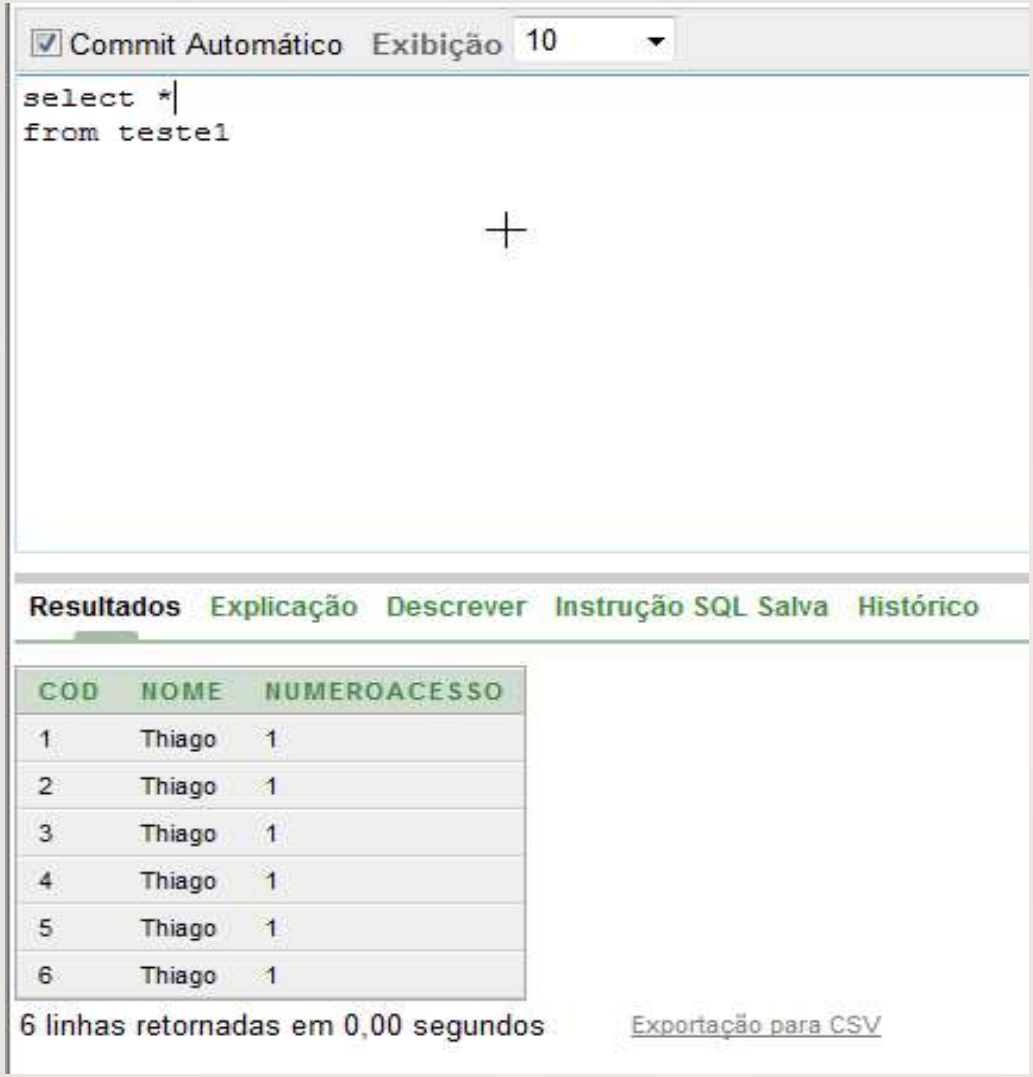
COD	NOME	NUMEROACESSO
1	Thiago	1
2	Thiago	1
3	Thiago	1
4	Thiago	1
5	Thiago	1
6	Thiago	1

6 linhas retornadas em 0,00 segundos Exportação para CSV

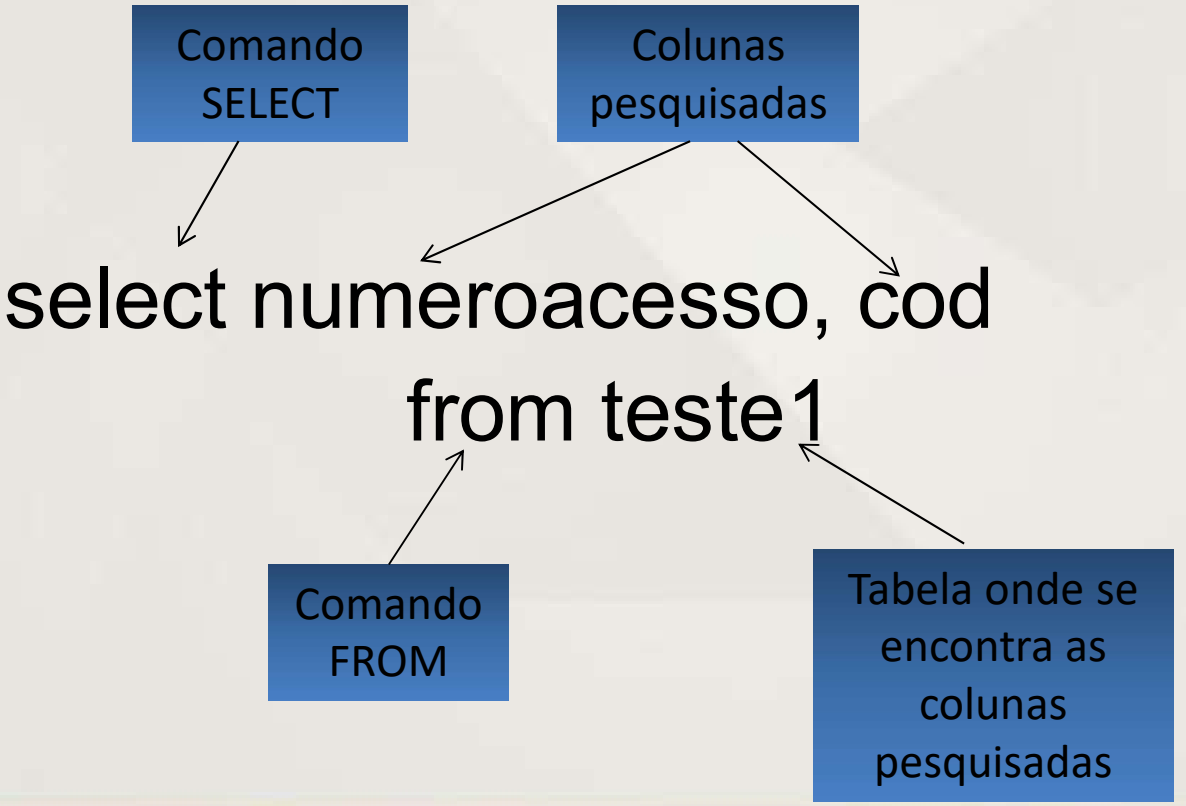
DML: SELECT

Outra forma de fazer a mesma pesquisa é utilizar o * no select ao invés das colunas, veja:




Commit Automático
Exibição
10
select *
from teste1
Resultados
Explicação
Descrever
Instrução SQL Salva
Histórico
COD NOME NUMEROACESSO
1 Thiago 1
2 Thiago 1
3 Thiago 1
4 Thiago 1
5 Thiago 1
6 Thiago 1
6 linhas retornadas em 0,00 segundos
Exportação para CSV

O resultado da pesquisa (select) irá seguir a ordem definida no comando SELECT, você pode mudar a ordem ou eliminar alguma coluna do resultado final. Veja o exemplo:



Observe que não estamos incluindo o nome, que é uma das colunas, e a ordem é diferente da tabela


Commit Automático
Exibição 10
select numeroacesso, cod
from teste1
Resultados
Explicação
Descrever
Instrução SQL Salva
Histórico
NUMEROACESSO COD
1 1
1 2
1 3
1 4
1 5
1 6
6 linhas retornadas em 0,00 segundos
Exportação para CSV

Cláusula WHERE: Os resultados de comandos como apresentados nos slides anteriores, possuem todas as linhas de uma tabela. No entanto, na maioria das consultas, queremos consultar somente as informações referentes a algumas linhas. Para isso utilizamos a cláusula WHERE.

Cláusula WHERE

WHERE é sempre seguida de uma expressão lógica, a qual pode conter operadores comparativos (>,<,=,>=,<=,<>), Operadores lógicos (AND, OR e NOT) e predicados próprios da linguagem SQL, tais como: IS (NOT) NULL, IS (NOT) LIKE, IN e EXISTS.

Os operadores lógicos AND e OR são utilizados para conectar comparações.

Operadores Relacionais:

=	Igual
!= ou <>	Diferente
<	Menor
<=	Menor Igual
>	Maior
>=	Maior Igual

Operadores Lógicos:

AND	E
OR	OU
NOT	Não

Operadores Especiais:

- IS NULL: verifica se o valor é NULO

Ex.: Select * from produtos where estoque is null;

- IS NOT NULL: verifica se o valor NÃO é nulo

Ex.: Select * from produtos where estoque is not null;

- BETWEEN: “entre”, determina um intervalo entre dois valores

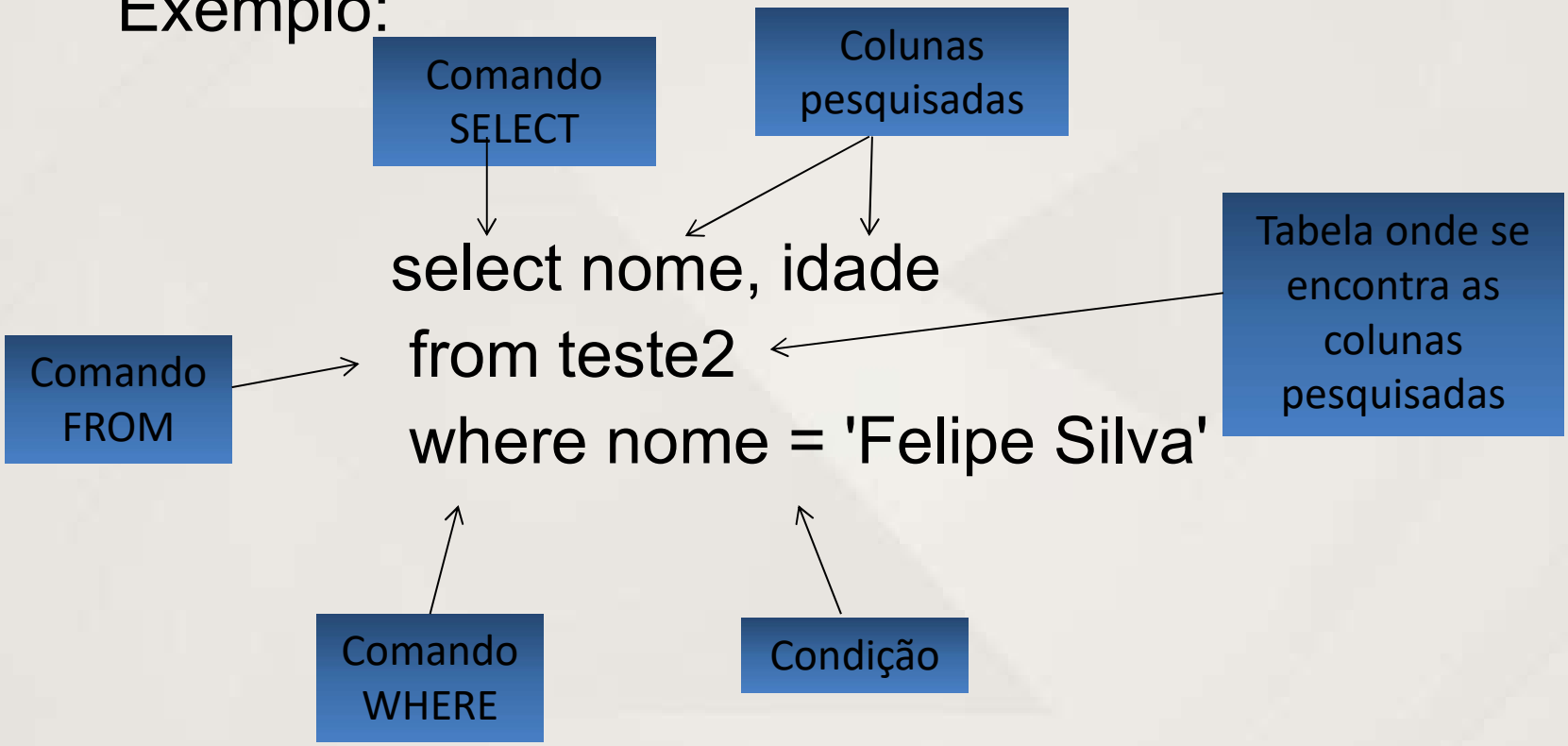
Ex.: Select * from produtos where preco BETWEEN 15 and 20;

- LIKE: utilizado para buscar strings
- [OLIVEIRA, pág. 111]

LIKE ‘A%’	Todas as palavras que iniciem com A
LIKE ‘%A’	Todas as palavras que terminem com A
LIKE ‘%A%’	Todas as palavras que tenham A em qualquer posição
LIKE ‘A_’	String de 2 caracteres, em que o primeiro é A e o segundo seja qualquer outro
LIKE ‘_A’	String de 2 caracteres, em que o primeiro é qualquer caractere e o segundo é A
LIKE ‘_A_’	String de 3 caracteres, em que o caractere central é A
LIKE ‘%A_’	Todas as palavras que possuam A como penúltimo caractere
LIKE ‘_A%’	Todas as palavras que possuam A como segundo caractere

Cláusula WHERE

Exemplo:



COD2	NOME	IDADE	CPF	CODLIVRO
1	Thiago Silva	21	12312312399	-
2	Felipe Silva	21	12312312398	-
3	Fernando Silva	21	12312312397	-
8	Felipe da Silva	21	12312312388	30
5	Fernanda Silva	21	12312312393	25
6	Juliana Silva	21	12312312392	45
9	Ingrid Silva	21	12312312389	-
10	Fernanda freitas	21	12312312387	-

Tabela completa

Commit Automático Exibição 10

```
select nome, idade
from teste2
where nome = 'Felipe Silva'
```

Comando SQL

NOME	IDADE
Felipe Silva	21

1 linhas retornadas em 0,00 segundos

Resultado da Pesquisa

Outros exemplos de Pesquisas utilizando o WHERE:

Commit Automático Exibição

```
select *
from teste2
where cod2>2 AND cod2<9
```

Comando SQL

Resultado da Pesquisa

COD2	NOME	IDADE	CPF	CODLIVRO
3	Fernando Silva	21	12312312397	-
8	Felipe da Silva	21	12312312388	30
5	Fernanda Silva	21	12312312393	25
6	Juliana Silva	21	12312312392	45

Commit Automático Exibição 10

```
select *
from teste2
where nome <> 'Thiago Silva' or cpf <= 12312312398
```

Comando SQL

Resultado da Pesquisa

COD2	NOME	IDADE	CPF	CODLIVRO
2	Felipe Silva	21	12312312398	-
3	Fernando Silva	21	12312312397	-
8	Felipe da Silva	21	12312312388	30
5	Fernanda Silva	21	12312312393	25
6	Juliana Silva	21	12312312392	45
9	Ingrid Silva	21	12312312389	-
10	Fernanda freitas	21	12312312387	-

Outros exemplos de Pesquisas utilizando o WHERE:

Commit Automático Exibição

```
select *
from teste2
where cod2>2 AND cod2<9
```

Comando SQL

Resultado da Pesquisa

COD2	NOME	IDADE	CPF	CODLIVRO
3	Fernando Silva	21	12312312397	-
8	Felipe da Silva	21	12312312388	30
5	Fernanda Silva	21	12312312393	25
6	Juliana Silva	21	12312312392	45

Commit Automático Exibição 10

```
select *
from teste2
where nome <> 'Thiago Silva' or cpf <= 12312312398
```

Comando SQL

Resultado da Pesquisa

COD2	NOME	IDADE	CPF	CODLIVRO
2	Felipe Silva	21	12312312398	-
3	Fernando Silva	21	12312312397	-
8	Felipe da Silva	21	12312312388	30
5	Fernanda Silva	21	12312312393	25
6	Juliana Silva	21	12312312392	45
9	Ingrid Silva	21	12312312389	-
10	Fernanda freitas	21	12312312387	-

Funções Agregadas:

Na linguagem SQL são definidas várias funções que operam sobre grupos de dados. Tais funções, usualmente, realizam operações ou comparações sobre um conjunto de dados e retornam como resultado, uma relação de apenas uma linha e uma coluna. São chamadas funções Agregadas, comumente, recebem apenas um parâmetro.

Funções Agregadas:

- AVG: calcula a média;
- COUNT: retorna o total de tuplas;
- MAX: retorna o maior valor;
- MIN: retorna o menor valor;
- SUM: somatório;

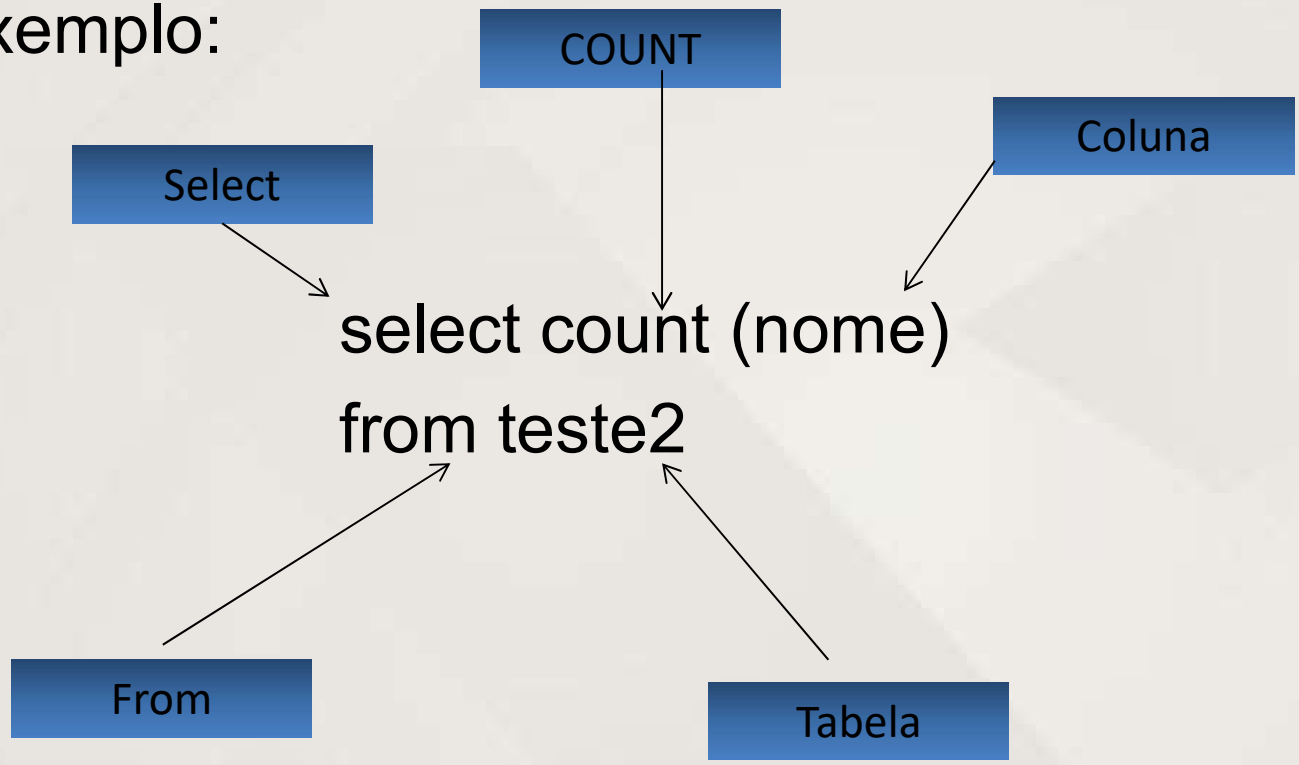
COUNT

Muitas vezes é necessário contar a quantidade de linhas que satisfazem determinada condição. Para isso é utilizado a função COUNT.

A função COUNT recebe um parâmetro, que pode ser o nome da coluna ou *, e retorna um número.

COUNT

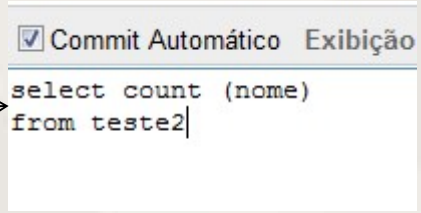
Exemplo:



COD2	NOME	IDADE	CPF	CODLIVRO
1	Thiago Silva	21	12312312399	-
2	Felipe Silva	21	12312312398	-
3	Fernando Silva	21	12312312397	-
8	Felipe da Silva	21	12312312388	30
5	Fernanda Silva	21	12312312393	25
6	Juliana Silva	21	12312312392	45
9	Ingrid Silva	21	12312312389	-
10	Fernanda freitas	21	12312312387	-
11	Thiago Silva	15	12312312386	30

TABELA

SQL



Resultado do COUNT

COUNT(NOME)
9

1 linhas retornadas em 0,00 segundos

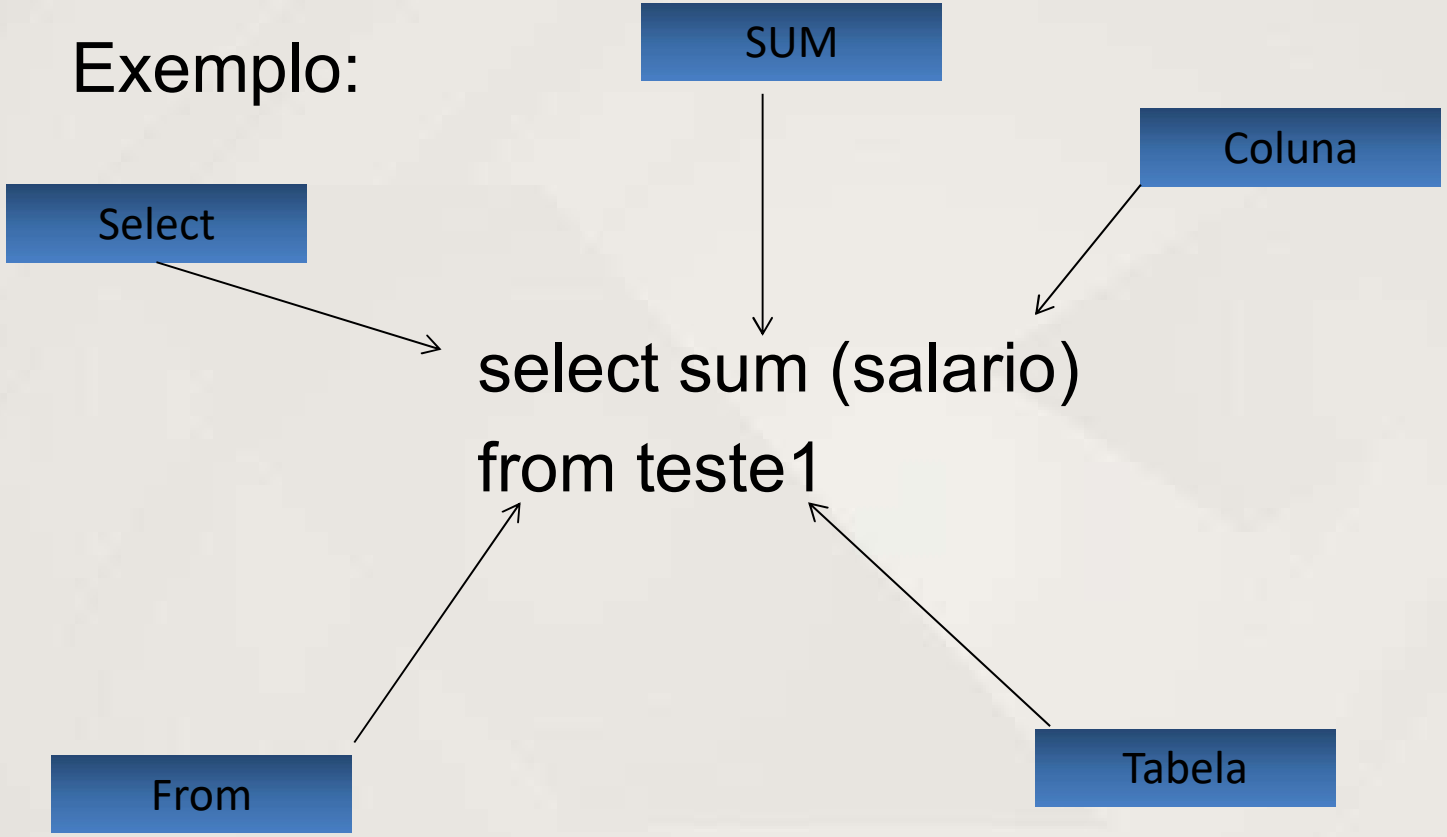
SUM

Outra operação comumente utilizada é a soma. Para realizar a soma dos valores de uma coluna para um grupo de dados, utilizamos a função SUM.

A função SUM retornará os somatório dos valores não-nulos da coluna utilizada como parâmetro. Retornará NULL se todos os valores desta coluna forem nulos ou se nenhum valor atender ao critério de seleção.

SUM

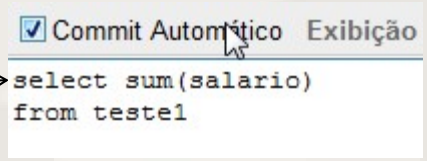
Exemplo:


SUM
Coluna
Select
select sum (salario)
from teste1
From
Tabela

COD	NOME	NUMEROACESSO	SALARIO
1	Thiago	1	600
2	Thiago	1	1500
3	Thiago	1	1000
4	Thiago	1	1000
5	Thiago	1	1500
6	Thiago	1	600

TABELA

SQL



Resultado do SUM

SUM(SALARIO)
6200

1 linhas retornadas em 0,00 segundos

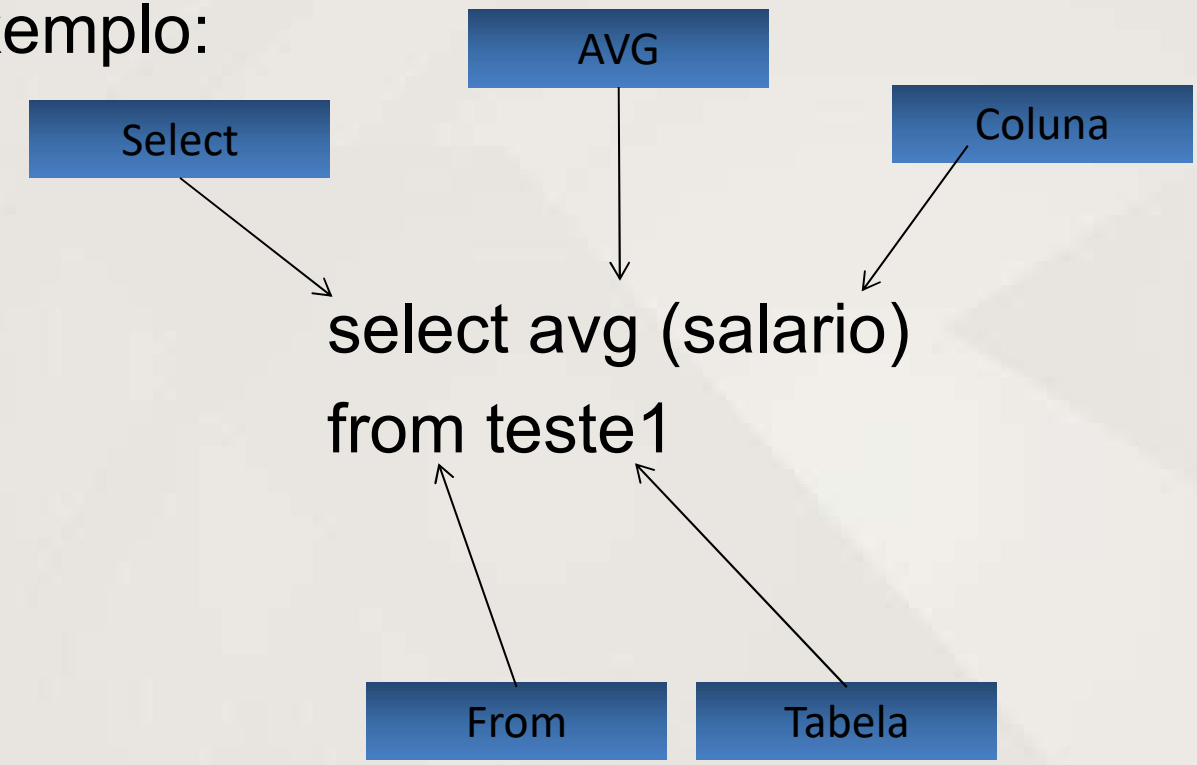
AVG

Para obter a média aritmética dos valores de uma coluna, utilizamos a função AVG, informando, como parâmetro para a mesma, o nome da coluna para a qual desejamos obter a média.

A função AVG retorna a média considerando apenas os valores não-nulos da coluna especificada.

AVG

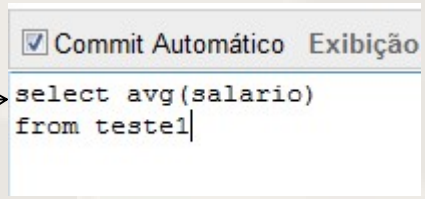
Exemplo:


AVG
Select
Coluna
select avg (salario)
from teste1
From
Tabela

COD	NOME	NUMEROACESSO	SALARIO
1	Thiago	1	600
2	Thiago	1	1500
3	Thiago	1	1000
4	Thiago	1	1000
5	Thiago	1	1500
6	Thiago	1	600

TABELA

SQL



Resultado do AVG

AVG(SALARIO)
1033,33333333333333333333333333333333333

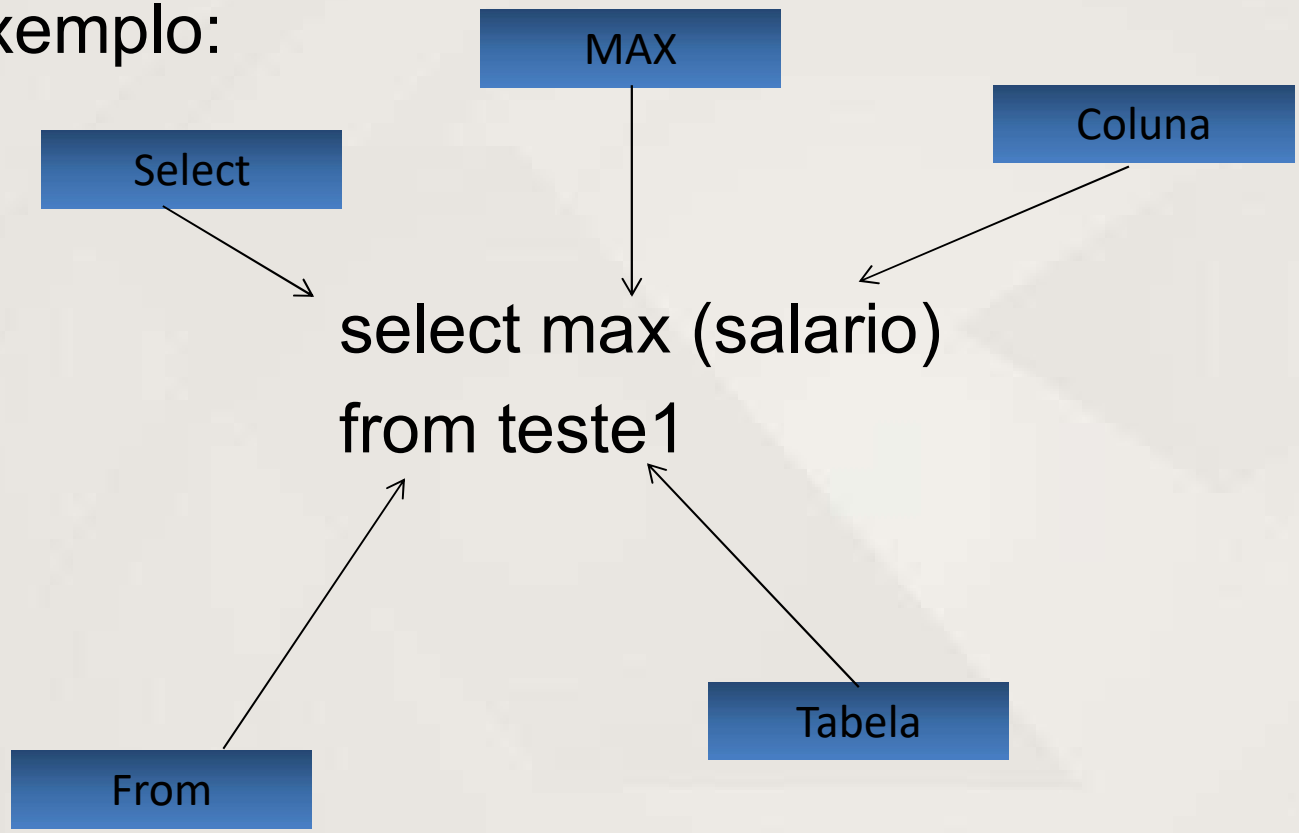
1 linhas retornadas em 0,02 segundos

MAX

Para obter o valor máximo de uma coluna em um conjunto de dados, utilizamos a função MAX. Assim como a função AVG, MAX recebe um parâmetro e retorna o valor NULL se em todas as linhas da tabela consultada o valor da coluna for nulo, ou se nenhuma linha atender ao critério de seleção.

MAX

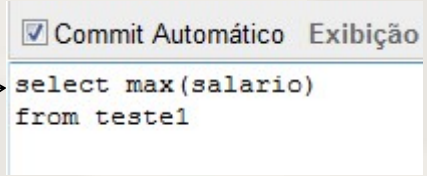
Exemplo:


MAX
Coluna
Select
select max (salario)
from teste1
Tabela
From

COD	NOME	NUMEROACESSO	SALARIO
1	Thiago	1	600
2	Thiago	1	1500
3	Thiago	1	1000
4	Thiago	1	1000
5	Thiago	1	1500
6	Thiago	1	600

TABELA

SQL



Resultado do MAX

MAX(SALARIO)
1500

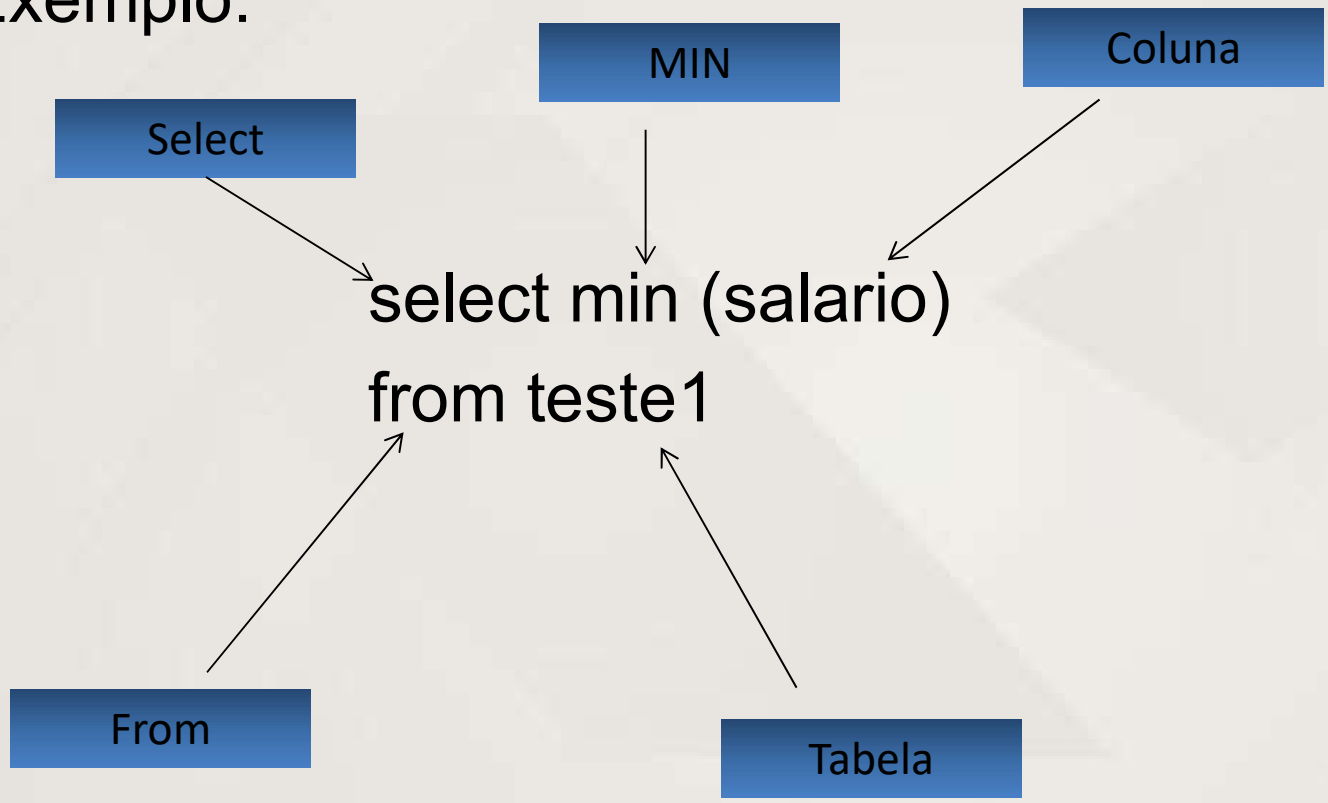
1 linhas retornadas em 0,00 segundos

MIN

Em oposição a função MAX, temos a função MI, que retorna o menor valor de uma coluna para a tabela específica.

MIN

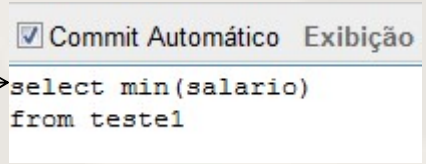
Exemplo:


MIN
Coluna
Select
select min (salario)
from teste1
From
Tabela

COD	NOME	NUMEROACESSO	SALARIO
1	Thiago	1	600
2	Thiago	1	1500
3	Thiago	1	1000
4	Thiago	1	1000
5	Thiago	1	1500
6	Thiago	1	600

TABELA

SQL



Resultado do MIN

MIN(SALARIO)
600

1 linhas retornadas em 0,00 segundos

Comando estudados nesta aula:

- GROUP BY
- HAVING

GROUP BY

O GROUP BY é a função de agregação em conjuntos baseado nos dados contidos nas colunas das tabelas.

Se utilizarmos as funções agregadas conseguimos uma linha de resposta de toda a coluna da tabela, porém não conseguimos filtrar por tipo de informação, por exemplo. Nesta situação utilizamos o GROUP BY.

GROUP BY

Exemplo: baseado nas informações da tabela livros abaixo, faça uma pesquisa informando a quantidade de livros por editora.

CODIGO	TITULO	PRECO	LANCAMENTO	ASSUNTO	EDITORA
1	Banco de Dados 1	200,5	22/12/08	B	1
2	Introdução a Banco de Dados 2	150,34	18/05/06	B	1
3	Engenharia de Software	140,4	05/02/01	E	2
4	Principios de Engenharia de Software	90,8	30/05/99	E	3
5	Sistemas de Informação	128,83	23/09/95	S	1
6	Basico de Sistemas de Informaçao	40,39	15/10/99	S	3
7	SQL	50,4	18/09/03	B	4
8	SQL for dummies	80,4	28/03/04	B	2
9	O pequeno principe	30,5	19/02/50	F	5
10	Princípios de Hardware	800,33	21/03/90	H	3

GROUP BY

Para que sua pesquisa tenha sucesso é necessário utilizar o GROUPY BY, veja:

```
select count(titulo), editora
from livros
group by editora
```

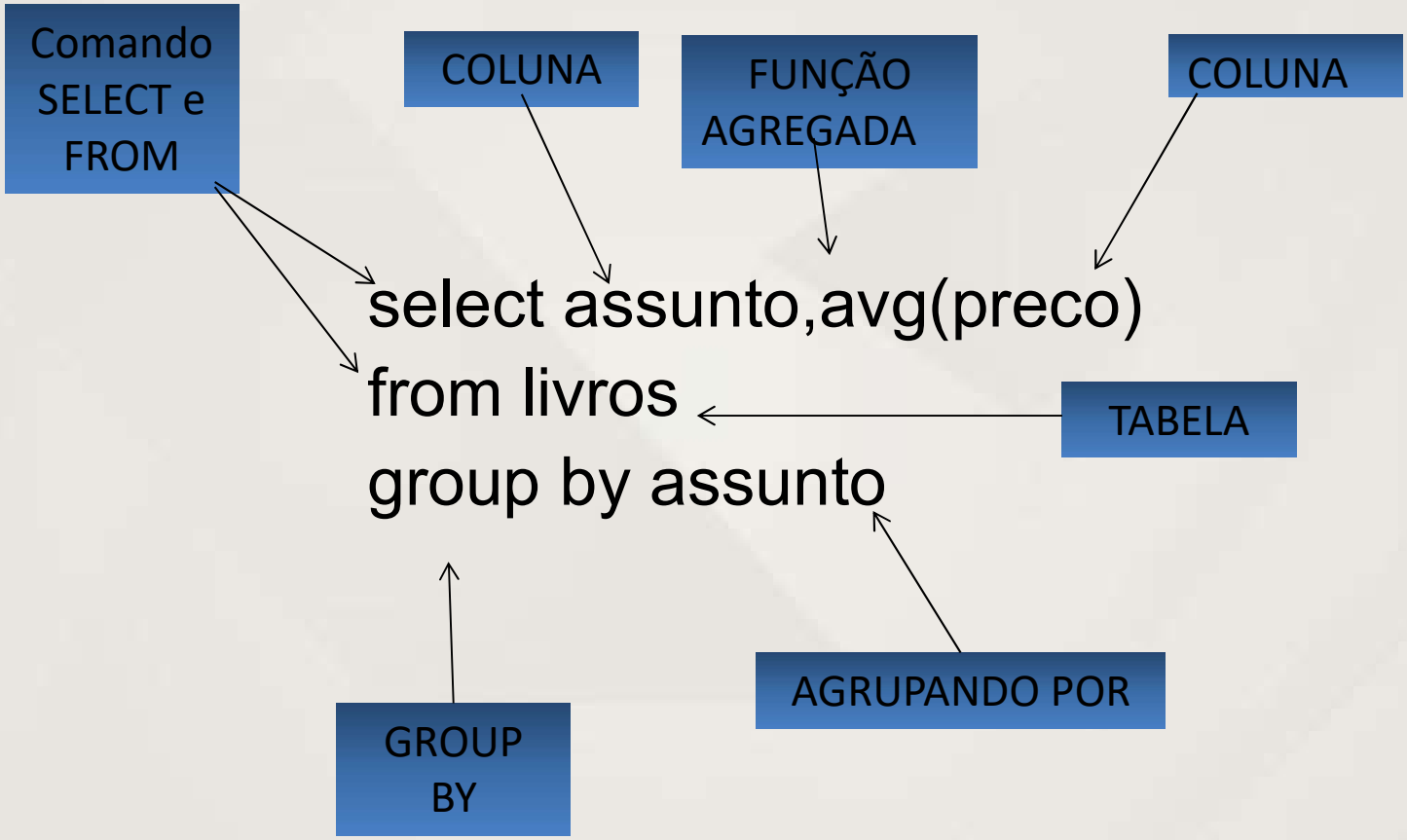
Comando SELECT utilizando GROUP BY

RESULTADO da Pesquisa

COUNT(TITULO)	EDITORA
3	1
2	2
1	4
1	5
3	3

GROUP BY

A montagem do SQL é da seguinte forma:



GROUP BY

A utilização da cláusula GROUP BY faz com que os dados sejam sumarizados pelas colunas que são especificadas na mesma.Assim, somente valores distintos destas colunas farão parte do resultado. Neste caso torna-se possível utilizar funções agregadas, as quais irão operar sobre as linhas que foram utilizadas para montar cada grupo(sumarização) de dados.

GROUP BY

UNIP
UNIVERSIDADE PAULISTA

Exemplos: Qual o preço médio dos livros para cada assunto?

TABELA

CODIGO	TITULO	PRECO	LANCAMENTO	ASSUNTO	EDITORA
1	Banco de Dados 1	200,5	22/12/08	B	1
2	Introdução a Banco de Dados 2	150,34	18/05/06	B	1
3	Engenharia de Software	140,4	05/02/01	E	2
4	Principios de Engenharia de Software	90,8	30/05/99	E	3
5	Sistemas de Informação	128,83	23/09/95	S	1
6	Basico de Sistemas de Informaçao	40,39	15/10/99	S	3
7	SQL	50,4	18/09/03	B	4
8	SQL for dummies	80,4	28/03/04	B	2
9	O pequeno principe	30,5	19/02/50	F	5
10	Princípios de Hardware	800,33	21/03/90	H	3

Commit Automático Exibição

```
select assunto,avg(preco)
from livros
group by assunto
```

Comando SELECT utilizando GROUP BY

RESULTADO da Pesquisa

ASSUNTO	AVG(PRECO)
H	800,33
B	120,41
E	115,6
S	84,61
F	30,5

GROUP BY

UNIP
UNIVERSIDADE PAULISTA

Exemplos: Quantos livros existem para cada assunto?

TABELA

CODIGO	TITULO	PRECO	LANCAMENTO	ASSUNTO	EDITORA
1	Banco de Dados 1	200,5	22/12/08	B	1
2	Introdução a Banco de Dados 2	150,34	18/05/06	B	1
3	Engenharia de Software	140,4	05/02/01	E	2
4	Principios de Engenharia de Software	90,8	30/05/99	E	3
5	Sistemas de Informação	128,83	23/09/95	S	1
6	Basico de Sistemas de Informaçao	40,39	15/10/99	S	3
7	SQL	50,4	18/09/03	B	4
8	SQL for dummies	80,4	28/03/04	B	2
9	O pequeno principe	30,5	19/02/50	F	5
10	Princípios de Hardware	800,33	21/03/90	H	3

Commit Automático Exibição

```
select assunto, count(*)
from livros
group by assunto
```

Comando SELECT utilizando GROUP BY

RESULTADO da Pesquisa

ASSUNTO	COUNT(*)
H	1
B	4
E	2
S	2
F	1

HAVING

A cláusula WHERE nos permite fazer restrições quanto aos dados que serão considerados para a montagem dos grupos de dados, porém não permite realizar restrições com base nos resultados das funções agregadas. Para isso, devemos utilizar a cláusula HAVING.

HAVING

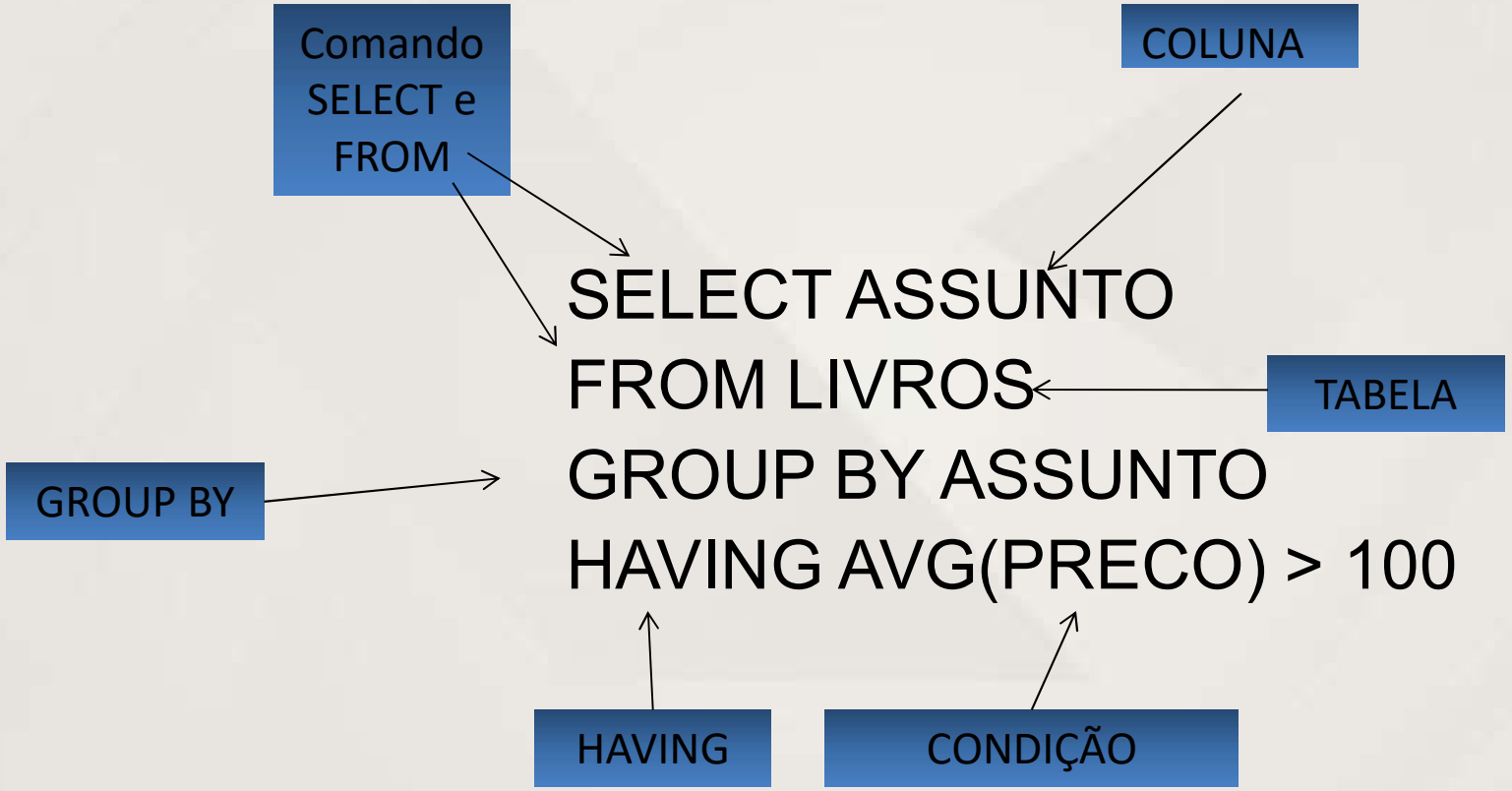
A principal diferença entre estas cláusulas se dá no fato de que, no caso da cláusula WHERE, o filtro é aplicado quando as linhas são recuperadas do banco de dados, fazendo com que estas nem cheguem a ser consideradas quando da realização de agrupamentos ou na execução de função de agregação.

HAVING

Já as restrições descritas na cláusula HAVING serão aplicadas somente após a recuperação das linhas do banco de dados, da montagem dos grupos e da execução de funções agregadas.Por isso é possível utilizar funções agregadas em expressões lógicas da cláusula.

HAVING

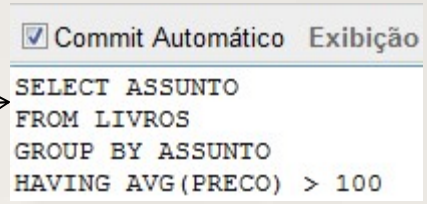
A montagem do SQL é da seguinte forma:



HAVING

"*Quais assuntos cujo o preço médio dos livros ultrapassa R$100,00*",veja:

Comando SELECT utilizando HAVING



RESULTADO da Pesquisa

ASSUNTO
H
B
E

HAVING

UNIP
UNIVERSIDADE PAULISTA

Exemplos: Quais os assuntos que possuem preço menor que 100 reais?

TABELA

CODIGO	TITULO	PRECO	LANCAMENTO	ASSUNTO	EDITORA
1	Banco de Dados 1	200,5	22/12/08	B	1
2	Introdução a Banco de Dados 2	150,34	18/05/06	B	1
3	Engenharia de Software	140,4	05/02/01	E	2
4	Principios de Engenharia de Software	90,8	30/05/99	E	3
5	Sistemas de Informação	128,83	23/09/95	S	1
6	Basico de Sistemas de Informaçao	40,39	15/10/99	S	3
7	SQL	50,4	18/09/03	B	4
8	SQL for dummies	80,4	28/03/04	B	2
9	O pequeno principe	30,5	19/02/50	F	5
10	Princípios de Hardware	800,33	21/03/90	H	3

Commit Automático Exibição

```
select assunto
from livros
group by assunto
having min(preco) < 100
```

Comando SELECT utilizando HAVING

RESULTADO da Pesquisa

ASSUNTO
B
E
S
F

HAVING

UNIP
UNIVERSIDADE PAULISTA

Exemplos: Quais assuntos possuem pelo menos 2 livros?

TABELA

CODIGO	TITULO	PRECO	LANCAMENTO	ASSUNTO	EDITORA
1	Banco de Dados 1	200,5	22/12/08	B	1
2	Introdução a Banco de Dados 2	150,34	18/05/06	B	1
3	Engenharia de Software	140,4	05/02/01	E	2
4	Principios de Engenharia de Software	90,8	30/05/99	E	3
5	Sistemas de Informação	128,83	23/09/95	S	1
6	Basico de Sistemas de Informaçao	40,39	15/10/99	S	3
7	SQL	50,4	18/09/03	B	4
8	SQL for dummies	80,4	28/03/04	B	2
9	O pequeno principe	30,5	19/02/50	F	5
10	Princípios de Hardware	800,33	21/03/90	H	3

Commit Automático Exibição

```
select assunto, count(*)
from livros
group by assunto
having count(*) > 1
```

Comando SELECT utilizando HAVING

RESULTADO da Pesquisa

ASSUNTO	COUNT(*)
B	4
E	2
S	2

BIBLIOGRAFIA:

• [SILBERSCHATZ,1999] SILBERSCHATZ,A., KORTH,H. F., SUDARSHAN,S.; Sistema de Banco de Dados, 3ª ed.,Editora Makron Books, São Paulo, 1999. (Capítulo 4).

• [ELMASRI, 2005] ELMASRI, R., NAVATHE, S. B. ; Sistema de Banco de Dados, 4ª ed., Editora Makron Books, São Paulo, 2005. (Capítulo 8).

• [DATE, 2004] DATE, C. J.; Introdução a Sistemas de Banco de Dados, 8ª ed., Editora Campus, São Paulo 2004. (Capítulo 4).

UNIP
UNIVERSIDADE PAULISTA

BIBLIOGRAFIA:

• [COSTA,2007] COSTA,Rogério Luís de C.; SQL Guia Prático, 2ª ed.,Editora Brasport, Rio de Janeiro, 2007. (Capítulos 3,4,5,6,7 e 8).

•[MACHADO, 2008] MACHADO,Felipe, ABREU,Maurício; Projeto de Banco de Dados, 15ª ed., Editora ERICA, São Paulo, 2008. (Capítulo 14).

• [OLIVEIRA, 2002] OLIVEIRA,Celso Henrique.; SQL Guia Prático, Editora Novatec, São Paulo 2002, Módulo 2 e Capitulos 4,5,6 e 7).

BIBLIOGRAFIA:

• [RAMALHO,2005] RAMALHO,José Antônio; ORACLE 10*g*, Editora Thomson, São Paulo, 2005. (Capítulo 2 e 7).

• [REZENDE, 2008] REZENDE, Alexandre; SQL: Apostila treinamento SQL MEDCO, 2008. (Capitulos 3,5 e 8).

• [TAYLOR, 2002] TAYLOR, Allen G.; SQL for Dummies, 3rd Edition, Editora IDG Books, New York 2002, Part 2 and 3).