



# Linguagem SQL para incluir, alterar e excluir dados

*Miriã Corrêa*

Análise e Desenvolvimento de Sistemas

[miriacoeelho@gmail.com](mailto:miriacoeelho@gmail.com)



# Sumário



INSERT

UPDATE

DELETE

# BD Fornecedor

- Para os exemplos que serão dados nesta aula, considere o esquema do banco de dados de fornecedores mostrado abaixo:

## FORNECEDOR

num_fornecedor	fnome	status	cidade
----------------	-------	--------	--------

## PECA

num_peca	pnome	cor	peso	cidade	preco
----------	-------	-----	------	--------	-------

## PEDIDO

num_fornecedor	num_peca	qde
----------------	----------	-----

# BD Fornecedor

```
CREATE DATABASE fornecedor;
```

```
CREATE TABLE fornecedor(  
  num_fornecedor NUMERIC(4),  
  fnome VARCHAR(30),  
  status VARCHAR(15),  
  cidade VARCHAR(20),  
  CONSTRAINT pk_fornecedor  
  PRIMARY KEY(num_fornecedor));
```

# BD Fornecedor

```
CREATE TABLE peca(  
  num_peca NUMERIC(5),  
  pnome VARCHAR(30),  
  cor VARCHAR(10),  
  peso REAL,  
  cidade VARCHAR(20),  
  preco REAL,  
  CONSTRAINT pk_peca  
  PRIMARY KEY(num_peca));
```

# BD Fornecedor

```
CREATE TABLE pedido(  
  num_fornecedor NUMERIC(4),  
  num_peca NUMERIC(5),  
  qdade integer,  
  CONSTRAINT pk_pedido  
  PRIMARY KEY(num_fornecedor,num_peca),  
  CONSTRAINT fk_fornecedor FOREIGN KEY(num_fornecedor)  
  REFERENCES fornecedor(num_fornecedor),  
  CONSTRAINT fk_peca FOREIGN KEY(num_peca)  
  REFERENCES peca(num_peca));
```

# COMANDOS

- Em SQL três comandos podem ser usados para modificar o banco de dados:
  - INSERT;
  - DELETE;
  - UPDATE.





# INSERT

- Em sua forma mais simples, INSERT é usado para acrescentar uma única tupla a uma relação.
- Temos de especificar o nome da relação e uma lista de valores para a tupla.
- Os valores devem ser listados na mesma ordem em que os atributos correspondentes foram especificados no comando CREATE TABLE.





# INSERT

- Tipos de dados:
  - Do tipo caractere e data devem conter aspas simples;
  - Do tipo numérico devem conter apenas os números, caso o número não seja inteiro, a parte fracionária deste deve vir após o ponto e não após a vírgula. Por exemplo: o salário 700,32 deve ser representado por 700.32.



# INSERT

- A sintaxe SQL utilizada é:

```
INSERT INTO <nome tabela> [(<nome coluna>{, <nome coluna>})]  
VALUES (<valor constante>,{<valor constante>}) | <instrução seleção>
```

```
INSERT INTO peca  
VALUES(1, 'Porca', 'Vermelho', 0.8, 'São Paulo', 1.75);
```

	num_peca	pnome	cor	peso	cidade	preco
	1	Porca	Vermelho	0.8	São Paulo	1.75
▶*	NULL	NULL	NULL	NULL	NULL	NULL

# Exemplo

```
INSERT INTO fornecedor  
VALUES(1, 'Agnaldo', 'disponível', 'São Paulo');
```

	num_fornecedor	fnome	status	cidade
	1	Agnaldo	disponível	São Paulo
*	NULL	NULL	NULL	NULL




# INSERT

- Uma segunda forma da instrução INSERT permite que o usuário especifique nomes de atributo explícitos que correspondem aos valores fornecidos no comando INSERT.

```
INSERT INTO fornecedor(num_fornecedor, fnome, status)  
VALUES(2, 'Ronaldo', 'indisponível');
```

- Exemplo:



	num_fornecedor	fnome	status	cidade
	1	Agnaldo	disponível	São Paulo
	2	Ronaldo	indisponível	NULL
▶*	NULL	NULL	NULL	NULL

# INSERT

- Os atributos não especificados no exemplo mostrado no slide anterior são definidos como seu valor DEFAULT ou NULL, e os valores são listados na mesma ordem que os atributos são listados no próprio comando INSERT.
- Também é possível inserir em uma relação múltiplas tuplas separadas por vírgulas em um único comando INSERT. Os valores do atributo que formam cada tupla ficam entre parênteses.



# Exemplo

```
INSERT INTO fornecedor VALUES  
(3, 'Jair', 'indisponível', 'Rio de Janeiro'),  
(4, 'Lindolfo', 'disponível', 'Belo Horizonte'),  
(5, 'Mauro', 'disponível', 'Juiz de Fora');
```

	num_fornecedor	fnome	status	cidade
	1	Agnaldo	disponível	São Paulo
	2	Ronaldo	indisponível	NULL
	3	Jair	indisponível	Rio de Janeiro
	4	Lindolfo	disponível	Belo Horizonte
	5	Mauro	disponível	Juiz de Fora
❏*	NULL	NULL	NULL	NULL

# Exemplo

Peca

	num_peca ▼	pnome	cor ▼	peso	cidade	preco
	1	Porca	Vermelho	0.8	São Paulo	1.75
	2	Pino	Verde	0.5	Rio de Janeiro	1.3
	3	Parafuso	Azul	0.4	Rio de Janeiro	1
	4	Parafuso	Cinza	0.3	Juiz de Fora	0.9
	5	Tubo	Branco	1	Juiz de Fora	2.25
	6	Tubo	Preto	1	Belo Horizonte	3.5
▶*	NULL	NULL	NULL	NULL	NULL	NULL





# Exemplo

Pedido

	num_fornecedor	num_peca	qdade
	1	1	300
	1	2	200
	1	3	50
	1	4	100
	1	5	20
	1	6	68
	4	1	500
	4	2	400
	5	3	450
*	NULL	NULL	NULL

# INSERT

- Um SGBD que implementa totalmente a SQL deve aceitar e impor todas as restrições de integridade que podem ser especificadas na DDL.
- Por exemplo, se emitirmos o comando:

```
INSERT INTO pedido VALUES (6, 1, 300)
```

- O SGBD deve rejeitar a operação, pois não existe uma tupla na tabela FORNECEDOR com num\_fornecedor = 6.



# INSERT

## Fornecedor

	num_fornecedor	fnome	status	cidade
▶	1	Agnaldo	disponível	São Paulo
	2	Ronaldo	indisponível	NULL
	3	Jair	indisponível	Rio de Janeiro
	4	Lindolfo	disponível	Belo Horizonte
	5	Mauro	disponível	Juiz de Fora
*	NULL	NULL	NULL	NULL

## Pedido

	num_fornecedor	num_peca	qdade
	1	1	300
	1	2	200
	1	3	50
	1	4	100
	1	5	20
	1	6	68
	4	1	500
	4	2	400
	5	3	450
*	NULL	NULL	NULL

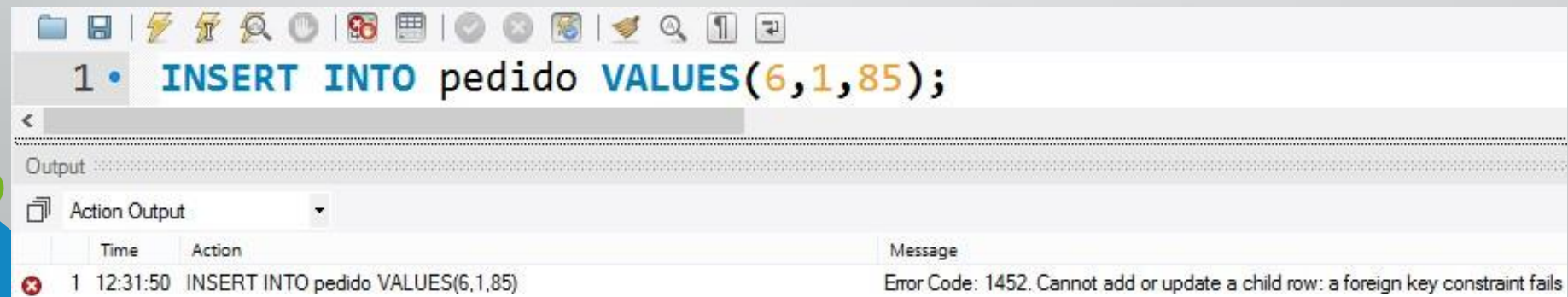


# INSERT

```
INSERT INTO pedido VALUES(6,1,85);
```



Executou o insert



# INSERT

- De modo semelhante o comando:  

```
INSERT INTO pedido (num_fornecedor, qdade)  
VALUES(5,250);
```
- Seria rejeitado, porque nenhum valor de num\_peca é fornecido e essa faz parte da chave primária juntamente com num\_fornecedor e não pode ser NULL.



# INSERT

```
INSERT INTO pedido (num_fornecedor, qdade)  
VALUES(5,250);
```



Executou o insert



```
1 • INSERT INTO pedido (num_fornecedor, qdade)  
2 VALUES(5,250);
```

Output

Action Output

	Time	Action	Message
✖ 1	12:37:14	INSERT INTO pedido (num_fornecedor, qdade) VALUES(5,250)	Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails

# DELETE– estrutura

- A forma básica do comando DELETE é composta pelas três cláusulas DELETE, FROM e WHERE, e tem a seguinte forma:

**DELETE**

**FROM** <tabela>

**WHERE** <condição>;





# DELETE– estrutura

<tabela> é o nome de tabela exigido para processar a deleção.

<condição> é uma expressão condicional (booleana) que identifica as tuplas a serem deletadas.



# DELETE

- Dependendo do número de tuplas selecionadas pela condição na cláusula WHERE, zero, uma ou várias tuplas podem ser excluídas por um único comando DELETE.
- Porém a tabela permanece no banco de dados como uma tabela vazia.
- Temos de usar o comando DROP TABLE para remover a definição de tabela.



# Operadores lógicos básicos

- Os operadores lógicos básicos são:

Operador	Significado
AND	'E' lógico
OR	'OU' lógico
IS NOT NULL	Testa se um valor é não nulo
IS NULL	Testa se um valor é nulo
IS NOT	Testa se o valor não é o valor comparado
IS	Testa se o valor é o valor comparado

# Operadores lógicos básicos

- Os operadores básicos de comparação lógicos para comparar valores de atributo entre si e com constantes literais são:

Operador	Significado
=	Igual a
>	Maior do que
>=	Maior do que ou igual
<	Menor do que
<=	Menor do que ou igual
!= ou <>	Diferente de

# MySQL - Proteção

- O MySQL possui proteção contra exclusões(DELETE) e atualizações(UPDATE), sendo assim, precisamos desativar o modo de proteção para realizar essas operações.
- Desativando o modo de segurança do MySQL: SET SQL\_SAFE\_UPDATES=0;



# DELETE

- A sintaxe SQL é:

```
DELETE FROM <nome tabela> [WHERE <condição seleção>]
```

- Deleção de nenhum registro:

```
delete from fornecedor where num_fornecedor=6;
```



# Exemplo

SQL File 1\* x

delete from fornecedor where num\_fornecedor=6;

Output

Action Output

	Time	Action	Message
✓	1 13:45:16	delete from fornecedor where num_fornecedor=6	0 row(s) affected





# Exemplo

- Deleção de nenhum registro:

```
delete from fornecedor where  
cidade='São Paulo' AND status='indisponível';
```

	num_fornecedor	fnome	status	cidade
	1	Aginaldo	disponível	São Paulo
	2	Ronaldo	indisponível	NULL
	3	Jair	indisponível	Rio de Janeiro
	4	Lindolfo	disponível	Belo Horizonte
	5	Mauro	disponível	Juiz de Fora

```
1 • delete from fornecedor where  
2 cidade='São Paulo' AND status='indisponível';
```

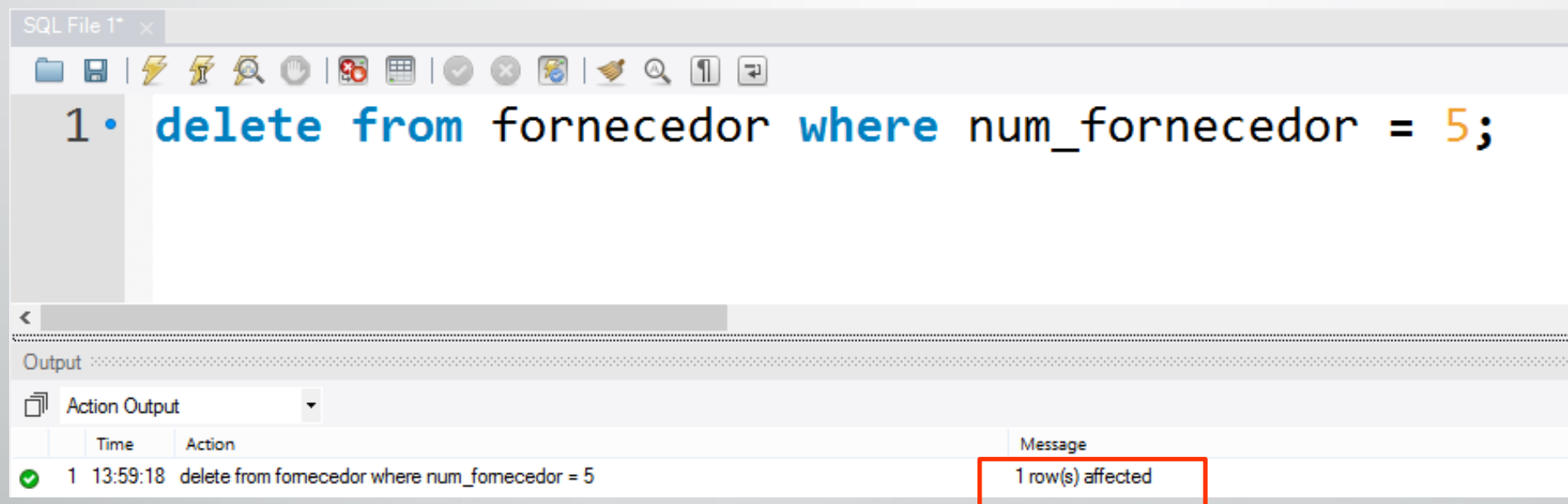
## Output

Action Output

	Time	Action	Message
✓	1 14:49:55	delete from fornecedor where cidade='São Paulo' AND status='indisponível'	0 row(s) affected

# Exemplo

```
delete from fornecedor where num_fornecedor = 5;
```



The screenshot shows a SQL IDE window titled "SQL File 1\* x". The main editor contains the SQL statement: `1 • delete from fornecedor where num_fornecedor = 5;`. Below the editor is an "Output" pane with a dropdown menu set to "Action Output". The output table shows the execution of the statement:

	Time	Action	Message
✓	1 13:59:18	delete from fornecedor where num_fornecedor = 5	1 row(s) affected

The "Message" column for the executed statement is highlighted with a red box.

# Exemplo

- Deleção de um registro:

```
delete from fornecedor where cidade is null;
```

	num_fornecedor	fnome	status	cidade
	1	Aginaldo	disponível	São Paulo
	2	Ronaldo	indisponível	NULL
	3	Jair	indisponível	Rio de Janeiro
	4	Lindolfo	disponível	Belo Horizonte
	5	Mauro	disponível	Juiz de Fora

SQL File 1\* x

1 • `delete from fornecedor where cidade is null;`

Output

Action Output

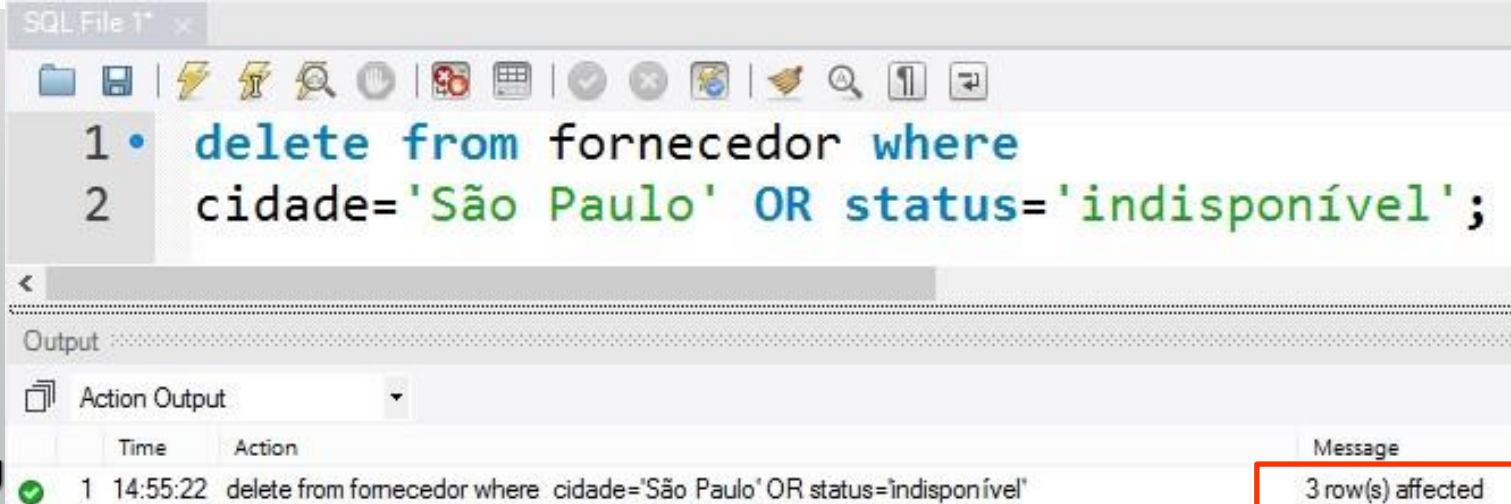
	Time	Action	Message
✓ 1	14:59:01	delete from fornecedor where cidade is null	1 row(s) affected

# Exemplo

- Deleção de mais de um registro:

```
delete from fornecedor where  
cidade='São Paulo' OR status='indisponível';
```

	num_fornecedor	fnome	status	cidade
	1	Agnaldo	disponível	São Paulo
	2	Ronaldo	indisponível	NULL
	3	Jair	indisponível	Rio de Janeiro
	4	Lindolfo	disponível	Belo Horizonte
	5	Mauro	disponível	Juiz de Fora



SQL File 1\* x

1 • delete from fornecedor where  
2 cidade='São Paulo' OR status='indisponível';

Output

Action Output

	Time	Action	Message
✓	1 14:55:22	delete from fornecedor where cidade='São Paulo' OR status='indisponível'	3 row(s) affected

# Exemplo

**DELETE FROM peca where preco>=2.00**

	num_peca	pnome	cor	peso	cidade	preco
	1	Porca	Verm...	0.8	São Paulo	1.75
	2	Pino	Verde	0.5	Rio de Janeiro	1.3
	3	Parafuso	Azul	0.4	Rio de Janeiro	1
	4	Parafuso	Cinza	0.3	Juiz de Fora	0.9
	5	Tubo	Branco	1	Juiz de Fora	2.25
	6	Tubo	Preto	1	Belo Horizonte	3.5

SQL File 1\* x

1 • **delete from peca where preco>=2.00**

Output

Action Output

	Time	Action	Message
✓	1 14:23:29	delete from peca where preco>=2.00	2 row(s) affected

# Exemplo

- Deleção de mais de um registro:

```
delete from fornecedor where cidade is not null;
```

	num_fornecedor	fnome	status	cidade
	1	Agnaldo	disponível	São Paulo
	2	Ronaldo	indisponível	NULL
	3	Jair	indisponível	Rio de Janeiro
	4	Lindolfo	disponível	Belo Horizonte
	5	Mauro	disponível	Juiz de Fora

SQL File 1\* x

1 • delete from fornecedor where cidade is not null;

Output

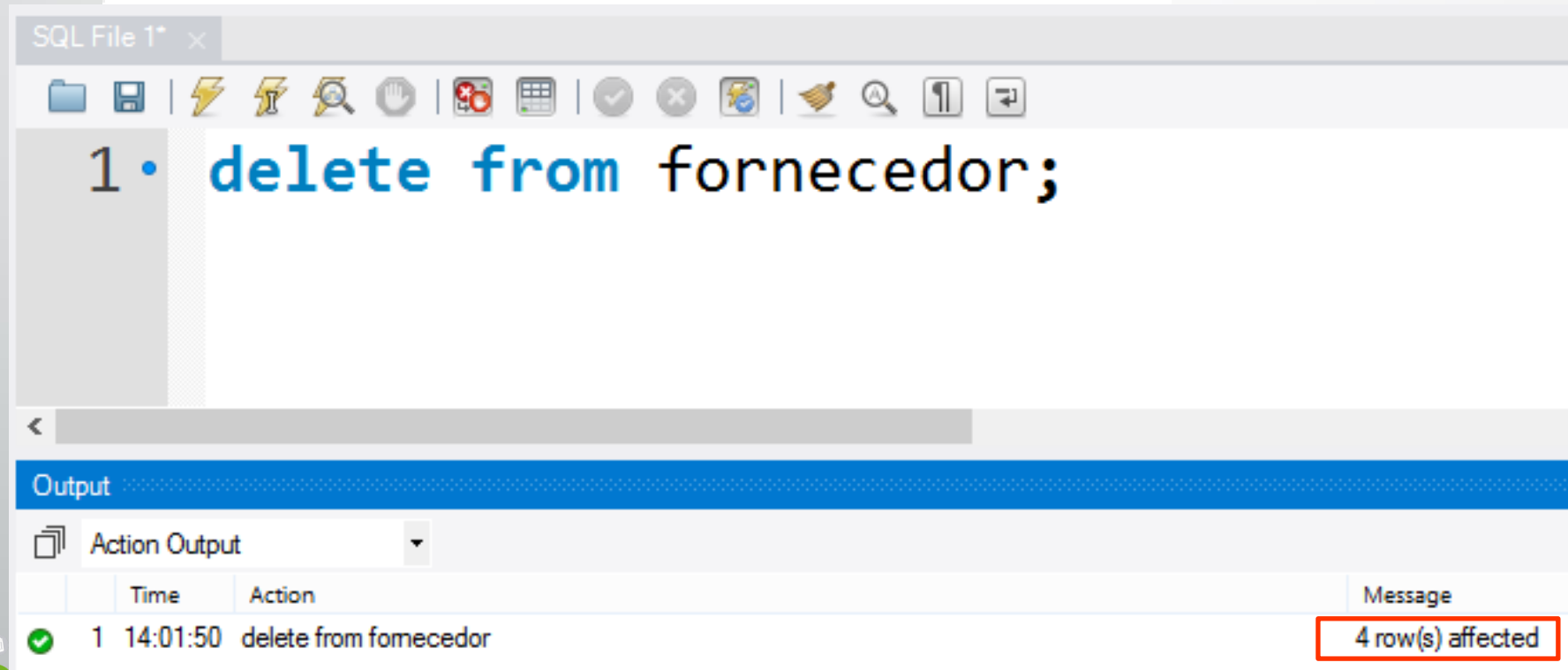
Action Output

	Time	Action	Message
✓	1 15:02:24	delete from fornecedor where cidade is not null	4 row(s) affected

# Exemplo

- Deleção de todos os registros:

```
delete from fornecedor;
```





# UPDATE

- O comando UPDATE é usado para modificar valores de atributo de uma ou mais tuplas selecionadas.
- Assim como no comando DELETE, uma cláusula WHERE no comando UPDATE seleciona as tuplas a serem modificadas em uma única relação.



# UPDATE

- No entanto, a atualização de uma chave primária pode ser propagada para os valores de chave estrangeira das tuplas em outras relações se tal ação de disparo referencial for especificada nas restrições de integridade referencial da DDL.
- Uma cláusula **SET** adicional no comando UPDATE especifica os atributos a serem modificados e seus novos valores.



# UPDATE

- A sintaxe SQL utilizada é:

```
UPDATE <nome tabela>  
SET <nome coluna> = <expressão valor> {,<nome coluna> =  
<expressão valor>}  
[WHERE <condição seleção>]
```



# Exemplo

```
update fornecedor set fnome = 'Pedro'  
where fnome='Jair'
```

	num_fornecedor	fnome	status	cidade
	1	Agnaldo	disponível	São Paulo
	2	Ronaldo	indisponível	NULL
	3	Jair	indisponível	Rio de Janeiro
	4	Lindolfo	disponível	Belo Horizonte
	5	Mauro	disponível	Juiz de Fora

SQL File 1\* x

1 • update fornecedor set fnome = 'Pedro'  
2 where fnome='Jair'

Output

Action Output

	Time	Action	Message
✓	1 16:12:27	update fornecedor set fnome = 'Pedro' where fnome='Jair'	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0

# UPDATE

	num_fornecedor	fnome	status	cidade
	1	Agnaldo	disponível	São Paulo
	2	Ronaldo	indisponível	NULL
	3	Jair	indisponível	Rio de Janeiro
	4	Lindolfo	disponível	Belo Horizonte
	5	Mauro	disponível	Juiz de Fora

`update fornecedor set fnome = 'Pedro'`  
`where fnome='Jair'`

	num_fornecedor	fnome	status	cidade
	1	Agnaldo	disponível	São Paulo
	2	Ronaldo	indisponível	NULL
	3	Pedro	indisponível	Rio de Janeiro
	4	Lindolfo	disponível	Belo Horizonte
	5	Mauro	disponível	Juiz de Fora

# UPDATE

- Várias tuplas podem ser modificadas com um único comando UPDATE.
- Um exemplo é dar a todas as peças de nome tubo um aumento de 10 por cento no preço, como mostra o exemplo abaixo.

```
update peca set preco = 1.1*preco  
where pnome='Tubo'
```

# UPDATE

	num_peca	pnome	cor	peso	cidade	preco
	1	Porca	Vermelho	0.8	São Paulo	1.75
	2	Pino	Verde	0.5	Rio de Janeiro	1.3
	3	Parafuso	Azul	0.4	Rio de Janeiro	1
	4	Parafuso	Cinza	0.3	Juiz de Fora	0.9
	5	Tubo	Branco	1	Juiz de Fora	2.25
	6	Tubo	Preto	1	Belo Horizonte	3.5

`update peca set preco = 1.1*preco  
where pnome='Tubo'`

	num_peca	pnome	cor	peso	cidade	preco
	1	Porca	Vermelho	0.8	São Paulo	1.75
	2	Pino	Verde	0.5	Rio de J...	1.3
	3	Parafuso	Azul	0.4	Rio de J...	1
	4	Parafuso	Cinza	0.3	Juiz de F...	0.9
	5	Tubo	Branco	1	Juiz de F...	2.475
	6	Tubo	Preto	1	Belo Hori...	3.8500000000000005

# Exemplo

SQL File 1\* x

1 • `update peca set preco = 1.1*preco`  
2 `where pnome='Tubo'`

Output

Action Output

	Time	Action	Message
✓	1 16:30:14	update peca set preco = 1.1*preco where pnome='Tubo'	2 row(s) affected Rows matched: 2 Changed: 2 Warnings: 0





# Dúvidas?

