
UNITAU
Universidade de Taubaté


SQL UPDATE
DELETE

UPDATE

O comando **UPDATE** altera os registros de uma tabela.

```
UPDATE [ ONLY ] table [ [ AS ] alias ]
  SET { column = { expression | DEFAULT } |
    ( column [,...] ) =( {expression | DEFAULT} [,...] )} [,...]
    [ FROM fromlist ]
    [ WHERE condition | WHERE CURRENT OF cursor_name ]
    [ RETURNING * | output_expression [AS output_name] [,...] ]
```

As cláusulas **FROM** e **RETURNING** não são parte do padrão da linguagem SQL e são extensões do PostgreSQL.

UPDATE

```
SELECT codigo,fabricante,modelo,pais FROM automovel;
 codigo | fabricante |     modelo       |   pais
--------+------------+------------------+-----------
 01     | 01         | Gol              | Brasil
 02     | 01         | Golf             | Argentina
 03     | 04         | Ford Ka          | Brasil
 04     | 04         | Fiesta           | Mexico
 05     | 03         | Corsa Sedan      | Brasil
 06     | 04         | Fiesta           | Argentina
 07     | 03         | Corsa Sedan      | Argentina
 08     | 05         | Palio            | Brasil
 09     | 01         | Golf             | Mexico
 10     | 05         | Siena            | Brasil
 11     | 01         | Polo             | Brasil
 12     | 04         | Fiesta           | Argentina
 13     | 05         | Palio            | Brasil
 14     | 03         | Corsa Sedan      | Argentina
 15     | 01         | Polo             | Mexico
```

UPDATE

Quando utilizado sem nenhuma condição, o comando **UPDATE** altera todos os registros da tabela.

```
UPDATE automovel SET pais='Brasil';

SELECT codigo,fabricante,modelo,pais FROM automovel;
```

codigo	fabricante	modelo	pais
01	01	Gol	Brasil
02	01	Golf	Brasil
03	04	Ford Ka	Brasil
04	04	Fiesta	Brasil
05	03	Corsa Sedan	Brasil
06	04	Fiesta	Brasil
07	03	Corsa Sedan	Brasil
08	05	Palio	Brasil
09	01	Golf	Brasil
10	05	Siena	Brasil
11	01	Polo	Brasil
12	04	Fiesta	Brasil
13	05	Palio	Brasil
14	03	Corsa Sedan	Brasil
15	01	Polo	Brasil

UPDATE

A cláusula **WHERE** permite selecionar quais registros serão alterados.

```
UPDATE automovel SET pais='Argentina' WHERE fabricante='04';

SELECT codigo,fabricante,modelo,pais FROM automovel;
```

codigo	fabricante	modelo	pais
01	01	Gol	Brasil
02	01	Golf	Brasil
05	03	Corsa Sedan	Brasil
07	03	Corsa Sedan	Brasil
08	05	Palio	Brasil
09	01	Golf	Brasil
10	05	Siena	Brasil
11	01	Polo	Brasil
13	05	Palio	Brasil
14	03	Corsa Sedan	Brasil
15	01	Polo	Brasil
03	04	Ford Ka	Argentina
04	04	Fiesta	Argentina
06	04	Fiesta	Argentina
12	04	Fiesta	Argentina

UPDATE

A cláusula **FROM** permite relacionar a tabela que será atualizada com outras tabelas utilizadas na cláusula **WHERE**, a tabela que será atualizada não deve constar da lista de tabelas da cláusula **FROM** a não ser que seja necessário fazer um auto-join, nesse caso é necessário usar um alias para a tabela.

```
UPDATE automovel SET pais='Mexico' FROM fabricante WHERE
fabricante.nome='Chevrolet' AND
automovel.fabricante=fabricante.codigo;
```

UPDATE

```
SELECT codigo,fabricante,modelo,pais FROM automovel;
```

codigo	fabricante	modelo	pais
01	01	Gol	Brasil
02	01	Golf	Brasil
08	05	Palio	Brasil
09	01	Golf	Brasil
10	05	Siena	Brasil
11	01	Polo	Brasil
13	05	Palio	Brasil
15	01	Polo	Brasil
03	04	Ford Ka	Argentina
04	04	Fiesta	Argentina
06	04	Fiesta	Argentina
12	04	Fiesta	Argentina
05	03	Corsa Sedan	Mexico
07	03	Corsa Sedan	Mexico
14	03	Corsa Sedan	Mexico

UPDATE

A cláusula **FROM** não é um padrão da linguagem SQL e o mesmo efeito pode ser obtido utilizando subconsultas na cláusula **WHERE**.

```
UPDATE automovel SET pais='Uruguai' WHERE ( SELECT nome FROM
fabricante WHERE codigo=automovel.fabricante ) =
'Volkswagen';
```

UPDATE

```
SELECT codigo,fabricante,modelo,pais FROM automovel;
 codigo | fabricante |     modelo       |   pais
--------+------------+------------------+-----------
 08     | 05         | Palio            | Brasil
 10     | 05         | Siena            | Brasil
 13     | 05         | Palio            | Brasil
 03     | 04         | Ford Ka          | Argentina
 04     | 04         | Fiesta           | Argentina
 06     | 04         | Fiesta           | Argentina
 12     | 04         | Fiesta           | Argentina
 05     | 03         | Corsa Sedan      | Mexico
 07     | 03         | Corsa Sedan      | Mexico
 14     | 03         | Corsa Sedan      | Mexico
 01     | 01         | Gol              | Uruguai
 02     | 01         | Golf             | Uruguai
 09     | 01         | Golf             | Uruguai
 11     | 01         | Polo             | Uruguai
 15     | 01         | Polo             | Uruguai
```

UPDATE

A cláusula **RETURNING** permite o retorno de campos dos registros que foram atualizados.

```
UPDATE automovel SET pais='Uruguai' WHERE ( SELECT nome FROM
fabricante WHERE codigo=automovel.fabricante ) =
'Volkswagen' RETURNING codigo,pais;
 codigo |   pais
--------+----------
 01     | Uruguai
 02     | Uruguai
 09     | Uruguai
 11     | Uruguai
 15     | Uruguai
```

UPDATE

É possível atualizar mais de um campo na mesma sentença SQL.

```
UPDATE automovel SET modelo='Corsa Hatch', preco=10500 WHERE codigo='06';
SELECT codigo,modelo,preco FROM automovel;
```

codigo	modelo	preco
08	Palio	15000.00
10	Siena	26000.00
13	Palio	23000.00
03	Ford Ka	15000.00
04	Fiesta	20000.00
12	Fiesta	18000.00
05	Corsa Sedan	12500.00
07	Corsa Sedan	10000.00
14	Corsa Sedan	16000.00
01	Gol	25000.00
02	Golf	39000.00
09	Golf	37000.00
11	Polo	29000.00
15	Polo	27500.00
06	Corsa Hatch	10500.00

UPDATE

Pode ser utilizada uma sintaxe semelhante a utilizada no comando INSERT para especificar os campos e valores a serem alterados.

```
UPDATE automovel SET (modelo,preco) = ('Celta',10200) WHERE codigo='06';
```

```
SELECT codigo,modelo,preco FROM automovel;
```

codigo	modelo	preco
08	Palio	15000.00
10	Siena	26000.00
13	Palio	23000.00
03	Ford Ka	15000.00
04	Fiesta	20000.00
12	Fiesta	18000.00
05	Corsa Sedan	12500.00
07	Corsa Sedan	10000.00
14	Corsa Sedan	16000.00
01	Gol	25000.00
02	Golf	39000.00
09	Golf	37000.00
11	Polo	29000.00
15	Polo	27500.00
06	Celta	10200.00

DELETE

O comando **DELETE** remove os registros de uma tabela.

```
DELETE FROM [ ONLY ] table [ [ AS ] alias ]
    [ USING usinglist ]
    [ WHERE condition | WHERE CURRENT OF cursor_name ]
    [ RETURNING * | output_expression [AS output_name] [,...] ]
```

As cláusulas **USING** e **RETURNING** não são parte do padrão da linguagem SQL e são extensões do PostgreSQL.

DELETE

Quando utilizado sem nenhuma condição, o comando DELETE remove todos os registros da tabela.

```
SELECT * FROM venda;
 cliente | revenda | automovel |    data    |  valor
---------+---------+-----------+------------+----------
 02      | 01      | 03        | 2010-02-05 | 17500.00
 04      | 02      | 01        | 2010-01-07 | 28000.00
 01      | 04      | 10        | 2010-02-15 | 28000.00
 02      | 04      | 02        | 2010-03-12 | 42000.00
 03      | 06      | 07        | 2010-02-06 | 11500.00
 06      | 07      | 15        | 2010-01-10 | 29500.00
 03      | 02      | 06        | 2010-01-25 | 22100.00
 01      | 01      | 05        | 2010-01-21 | 15500.00
 01      | 06      | 13        | 2010-03-10 | 24500.00
 06      | 03      | 09        | 2010-03-02 | 39500.00
 04      | 01      | 11        | 2010-02-23 | 31000.00
 02      | 01      | 14        | 2010-02-19 | 17500.00
 01      | 05      | 04        | 2010-03-11 | 21500.00
DELETE FROM venda;
SELECT * FROM venda;
 cliente | revenda | automovel | data | valor
---------+---------+-----------+------+-------
(0 rows)
```

DELETE

A cláusula **WHERE** permite selecionar quais registros serão deletados e a cláusula **RETURNING** permite retornar campos dos registros que foram deletados.

```
DELETE FROM fabricante WHERE NOT EXISTS ( SELECT * FROM
automovel WHERE fabricante=fabricante.codigo ) RETURNING *;
 codigo |    nome
--------+------------
 02     | Peugeot
SELECT * FROM fabricante;
 codigo |    nome
--------+------------
 01     | Volkswagen
 03     | Chevrolet
 04     | Ford
 05     | Fiat
```

DELETE

A cláusula **USING** permite relacionar a tabela da qual os registros serão deletados com outras tabelas utilizadas na cláusula **WHERE**, a tabela que será atualizada não deve constar da lista de tabelas da cláusula **USING** a não ser que seja necessário fazer um auto-join, nesse caso é necessário usar um alias para a tabela.

```
DELETE FROM funcionario USING secao WHERE secao.codigo =
funcionario.secao AND secao.descricao = 'Ct Receber'
RETURNING matricula;

 matricula
-----------
 38283
 36392
```

MERGE

O comando MERGE atualiza a tabela de destino baseada em uma fonte de dados.

```
MERGE INTO [ ONLY ] target_table_name [ * ] [ [ AS ] target_alias ]
USING {[ONLY] source_table_name [*] | ( source_query )} [ [AS] source_alias ]
ON join_condition
{WHEN MATCHED [AND condition] THEN {merge_update|merge_delete|DO NOTHING} |
WHEN NOT MATCHED [AND condition] THEN {merge_insert|DO NOTHING} }

merge_insert:
INSERT [( column_name [, ...] )] [ OVERRIDING {SYSTEM | USER} VALUE ]
       {VALUES ( {expression | DEFAULT} [, ...] ) | DEFAULT VALUES}
merge_update:
UPDATE SET { column_name = {expression | DEFAULT} |
        ( column_name [,...] ) = ( {expression | DEFAULT} [,...] ) } [,...]
merge_delete:
DELETE
```

MERGE

O comando **MERGE** executa um **JOIN** entre a tabela de destino e a fonte de dados.

Para os registros que atenderem a condição de junção é executada a cláusula **WHEN MATCHED** que não tenha condição adicional ou que a condicação adicional seja avaliada como **TRUE**.

Para os registros da fonte de dados que não atenderem a condição de junção é executada a cláusula **WHEN NOT MATCHED** que não tenha condição adicional ou que a condição adicional seja avaliada como **TRUE**.

Apenas uma cláusula **MATCHED** ou **NOT MATCHED** é executada para cada registro da fonte de dados.

É obrigatório que a condição de junção relacione cada registro da tabela destino com um único registro da fonte de dados.

MERGE

```
CREATE TEMPORARY TABLE movimento AS
    ( SELECT produto,data,SUM(quantidade) AS entradas,
             null AS saidas, SUM(quantidade) AS saldo
      FROM entrada
      WHERE produto='04'
      GROUP BY produto, data);
SELECT * FROM movimento;
```

produto	data	entradas	saidas	saldo
04	2010-03-01	25		25
04	2010-03-05	10		10
04	2010-03-25	5		5

MERGE

```
MERGE INTO movimento
USING (SELECT produto,data,SUM(quantidade) AS saidas
       FROM saida GROUP BY produto, data) AS saidas
ON movimento.produto=saidas.produto AND movimento.data=saidas.data
WHEN NOT MATCHED AND produto='04' THEN
     INSERT (produto, data,entradas,saidas,saldo)
     VALUES (produto, data, 0, -saidas, -saidas)
WHEN MATCHED THEN
     UPDATE SET saidas=saidas.saidas, saldo=entradas-saidas.saidas;
```

```
SELECT * FROM movimento ORDER BY data;
```

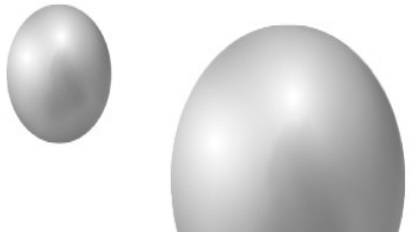
produto	data	entradas	saidas	saldo
04	2010-03-01	25		25
04	2010-03-02	0	-8	-8
04	2010-03-05	10	1	9
04	2010-03-25	5	12	-7
04	2010-03-29	0	-5	-5

TRUNCATE

O comando **TRUNCATE** remove todos os registros de uma tabela.

```
TRUNCATE [ TABLE ] name [, ...] [ CASCADE | RESTRICT ]
    [ RESTART IDENTITY | CONTINUE IDENTITY ]
```

O comando **TRUNCATE** não executa os gatilhos **ON DELETE** para os registros deletados. Truncar a tabela é mais rápido do que deletar todos os registros porque não deleta registro por registro.



TRUNCATE

A cláusula **CONTINUE IDENTITY** mantém os valores das sequências dos campos da tabela. Esse é o comportamento padrão.

```
TRUNCATE entrada CONTINUE IDENTITY;
SELECT * FROM entrada;
 codigo | data | produto | quantidade
--------+------+---------+------------
(0 rows)
INSERT INTO entrada ( data, produto, quantidade ) VALUES
( '2010/03/01', '01', '15' );
SELECT * FROM entrada;
 codigo |    data    | produto | quantidade
--------+------------+---------+------------
     19 | 2010-03-01 | 01      |         15
```

TRUNCATE

A cláusula RESTART IDENTITY reinicia as sequências dos campos da tabela.

```
TRUNCATE entrada RESTART IDENTITY;
SELECT * FROM entrada;
 codigo | data | produto | quantidade
--------+------+---------+------------
(0 rows)
INSERT INTO entrada ( data, produto, quantidade ) VALUES
( '2010/03/01', '01', '15' );
SELECT * FROM entrada;
 codigo |    data    | produto | quantidade
--------+------------+---------+------------
      1 | 2010-03-01 | 01      |         15
```

TRUNCATE

Se a tabela é referencia por outras tabelas, pode ser utilizada a cláusula CASCADE para truncar todas a cadeia de dependência.

```
TRUNCATE fabricante CASCADE;
NOTICE:  truncate cascades to table "automovel"
NOTICE:  truncate cascades to table "venda"
TRUNCATE TABLE
```

Exercícios

Escreva a sentença SQL para:

a) Aumentar em 10% o salario dos funcionários com salário menor que R$1500,00

b) Reduzir em 5% o maior salario

c) Mudar os alunos da 1a série B do curso de Tecnologia da Informação para a 1a série A

d) Aumentar em 15% os preços dos automóveis fabricados na Argentina pela Ford, exibindo o código, modelo e novo valor dos automóveis

e) Equiparar o salário dos técnicos II ao maior salário da seção onde trabalha

f) Deletar todos os alunos sem nenhum curso

g) Deletar as salas que não tem aluno

h) Deletar as diretorias que não tem funcionário

i) Deletar as vendas de carros da Ford realizadas em SP