
Stored Procedures, Functions, and Triggers no MySQL

Prof. Rinaldo Lima

Stored Procedures no MySQL

- Um **stored procedure** contém uma sequência de comandos SQL armazenados no catálogo do BD que pode ser invocada por um programa de aplicação ou pelo próprio BD.

- Sintaxe de uma Stored procedure (procedimento armazenado):

```
CREATE PROCEDURE <proc-name>
    (param_spec1, param_spec2, ..., param_specn)
BEGIN
    -- comandos SQL
END;
```

Onde cada **param_spec** está na forma:

[**in** | **out** | **inout**] <param_name> <param_type>

- **in mode**: permite se passar um parâmetro de leitura para a procedure,
- **out mode**: permite se passar um valor de volta para o programa que chamou o procedimento
- **Inout mode**: ambas opções acima combinadas

Exemplo

```
mysql> select * from employee;
+----+-----+-----+-----+-----+-----+
| id | name | superid | salary | bdate   | dno |
+----+-----+-----+-----+-----+-----+
| 1 | john |      3 | 100000 | 1960-01-01 | 1 |
| 2 | mary |      3 | 50000  | 1964-12-01 | 3 |
| 3 | bob  |     NULL | 80000  | 1974-02-07 | 3 |
| 4 | tom  |      1 | 50000  | 1978-01-17 | 2 |
| 5 | bill |     NULL | NULL   | 1985-01-20 | 1 |
+----+-----+-----+-----+-----+-----+

mysql> select * from department;
+-----+-----+
| dnumber | dname |
+-----+-----+
|      1 | Payroll |
|      2 | TechSupport |
|      3 | Research |
+-----+-----+
```

- Suponha que se deseja manter o **total de salários** dos empregados trabalhando em cada departamento

```
mysql> create table deptsal as
-> select dnumber, 0 as totalsalary from department;
Query OK, 3 rows affected (0.00 sec)
Records: 3 Duplicates: 0 Warnings: 0
```

```
mysql> select * from deptsal;
+-----+-----+
| dnumber | totalsalary |
+-----+-----+
|      1 |          0 |
|      2 |          0 |
|      3 |          0 |
+-----+-----+
```

Agora escreva um procedimento para atualizar os salários na tabela DEPTSAL

Exemplo

```
mysql> delimiter //
mysql> create procedure updateSalary (IN param1 int)
-> begin
->   update deptsal
->   set totalsalary = (select sum(salary) from employee where dno = param1)
->   where dnumber = param1;
-> end; //
Query OK, 0 rows affected (0.01 sec)
```

Passo 1:

1. Defina um procedimento chamado `updateSalary()` que toma como entrada um número de departamento.
2. O corpo do procedimento é um comando SQL para atualizar a coluna `totalsalary` da tabela DEPTSAL;
3. O uso de um **delimitador** é opcional se você não estiver usando o editor de linha de comando do MySQL

Exemplo

Passo 2: Chame o procedimento para atualizar o campo **totalsalary** de cada departamento

```
mysql> call updateSalary(1);
Query OK, 0 rows affected (0.00 sec)

mysql> call updateSalary(2);
Query OK, 1 row affected (0.00 sec)

mysql> call updateSalary(3);
Query OK, 1 row affected (0.00 sec)
```

Exemplo

Passo 3:

Mostre o total de salários atualizados na tabela DEPTSAL

```
mysql> select * from deptsal;
+-----+-----+
| dnumber | totalsalary |
+-----+-----+
|      1 |      100000 |
|      2 |       50000 |
|      3 |      130000 |
+-----+-----+
3 rows in set (0.00 sec)
```

Stored Procedures no MySQL

- Use **show procedure status** para exibir a lista de stored procedures que você criou

```
mysql> show procedure status;
```

Db	Name	Type	Definer	Modified	Created	Security_
type	Comment	character_set_client	collation_connection	Database	Collation	
ptan	updateSalary0	PROCEDURE	ptan@%	2010-03-16 12:21:55	2010-03-16 12:21:55	DEFINER
	latin1		latin1_swedish_ci	latin1_swedish_ci		

```
( row in set (0.02 sec)
```

- Use drop procedure para remover a stored procedure

```
mysql> drop procedure updateSalary;
```

Query OK, 0 rows affected (0.00 sec)

Stored Procedures no MySQL

- Pode-se declarar variáveis nas stored procedures
- Pode-se também usar o controle de fluxo condicional (IF-THEN-ELSE ou loops tais como WHILE e o REPEAT)
- MySQL também fornece o conceito de **CURSOR** nas stored procedures.
 - Um cursor é usado para iterar através de um conjunto de registros retornado por uma consulta.
 - Dessa forma, se pode processar cada linha do resultado individualmente

Exemplo de Procedures usando Cursores

- O procedimento anterior atualiza uma linha na tabela DEPTSAL baseado no parâmetro de entrada
- Agora, suponha que se deseja atualizar todas as linhas da tabela DEPTSAL **de uma vez**
- Primeiro, reseta-se a coluna **totalsalary** da tabela DEPTSAL para 0

```
mysql> update deptsal set totalsalary = 0;
Query OK, 0 rows affected (0.00 sec)
Rows matched: 3  Changed: 0  Warnings: 0
```

```
mysql> select * from deptsal;
+-----+-----+
| dnumber | totalsalary |
+-----+-----+
|      1 |          0 |
|      2 |          0 |
|      3 |          0 |
+-----+-----+
3 rows in set (0.00 sec)
```

Exemplo usando Cursores

```
mysql> delimiter $$
mysql> drop procedure if exists updateSalary$$
Query OK, 0 rows affected (0.00 sec)
```

Drop o procedimento antigo

```
mysql> create procedure updateSalary()
-> begin
->     declare done int default 0;
->     declare current_dnum int;
->     declare dnumcur cursor for select dnumber from deptsal;
->     declare continue handler for not found set done = 1;
->
->     open dnumcur;
->
->     repeat
->         fetch dnumcur into current_dnum;
->         update deptsal
->         set totalsalary = (select sum(salary) from employee
->                             where dno = current_dnum)
->         where dnumber = current_dnum;
->     until done
->     end repeat;
->
->     close dnumcur;
-> end$$
Query OK, 0 rows affected (0.00 sec)
```

Use cursor para iterar as linhas do resultado

Fecha o cursor. Libera recursos alocados por ele

```
mysql> delimiter ;
```

Exemplo usando Cursores

- Chame a procedure

```
mysql> select * from deptsal;
+-----+-----+
| dnumber | totalsalary |
+-----+-----+
|      1 |           0 |
|      2 |           0 |
|      3 |           0 |
+-----+-----+
3 rows in set (0.01 sec)

mysql> call updateSalary;
Query OK, 0 rows affected (0.00 sec)

mysql> select * from deptsal;
+-----+-----+
| dnumber | totalsalary |
+-----+-----+
|      1 |      100000 |
|      2 |       50000 |
|      3 |      130000 |
+-----+-----+
3 rows in set (0.00 sec)
```

Outro Exemplo

- Cria uma procedure para dar um aumento para todos os empregados

```
mysql> select * from emp;
+-----+-----+-----+-----+-----+-----+
| id | name  | superid | salary | bdate      | dno |
+-----+-----+-----+-----+-----+-----+
| 1 | john  |      3 | 100000 | 1960-01-01 | 1 |
| 2 | mary  |      3 |  50000 | 1964-12-01 | 3 |
| 3 | bob   |    NULL |  80000 | 1974-02-07 | 3 |
| 4 | tom   |      1 |  50000 | 1978-01-17 | 2 |
| 5 | bill  |    NULL |    NULL | 1985-01-20 | 1 |
| 6 | lucy  |    NULL |  90000 | 1981-01-01 | 1 |
| 7 | george |    NULL |  45000 | 1971-11-11 | NULL |
+-----+-----+-----+-----+-----+-----+
7 rows in set (0.00 sec)
```

Outro Exemplo

```
mysql> delimiter |
mysql> create procedure giveRaise (in amount double)
-> begin
->     declare done int default 0;
->     declare eid int;
->     declare sal int;
->     declare emprec cursor for select id, salary from employee;
->     declare continue handler for not found set done = 1;
->
->     open emprec;
->     repeat
->         fetch emprec into eid, sal;
->         update employee
->         set salary = sal + round(sal * amount)
->         where id = eid;
->     until done
->     end repeat;
-> end |
Query OK, 0 rows affected (0.00 sec)
```

Outro Exemplo

```
mysql> delimiter ;
mysql> call giveRaise(0.1);
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> select * from employee;
+----+-----+-----+-----+-----+-----+
| id | name | superid | salary | bdate       | dno |
+----+-----+-----+-----+-----+-----+
| 1  | john |      3  | 110000 | 1960-01-01  | 1   |
| 2  | mary |      3  |  55000 | 1964-12-01  | 3   |
| 3  | bob  |    NULL |  88000 | 1974-02-07  | 3   |
| 4  | tom  |      1  |  55000 | 1978-01-17  | 2   |
| 5  | bill |    NULL |    NULL | 1985-01-20  | 1   |
+----+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)
```

Funções

- Funções são declaradas usando a seguinte sintaxe:

```
FUNCTION <function-name> (param_spec1, ..., param_speck)  
    returns <return_type>  
    [not] deterministic           permite a otimização se houver a  
                                   mesma saída para a mesma entrada  
                                   (use RAND para não determinístico)  
  
BEGIN  
    -- execution code  
END;
```

Onde param_spec é:

```
[in | out | in out] <param_name> <param_type>
```

Exemplo de Funções

```
mysql> select * from employee;  
+----+-----+-----+-----+-----+-----+  
| id | name  | superid | salary | bdate      | dno |  
+----+-----+-----+-----+-----+-----+  
| 1  | john  | 3       | 100000 | 1960-01-01 | 1   |  
| 2  | mary  | 3       | 50000  | 1964-12-01 | 3   |  
| 3  | bob   | NULL    | 80000  | 1974-02-07 | 3   |  
| 4  | tom   | 1       | 50000  | 1970-01-17 | 2   |  
| 5  | bill  | NULL    | NULL   | 1985-01-20 | 1   |  
+----+-----+-----+-----+-----+-----+  
5 rows in set (0.00 sec)  
  
mysql> delimiter ;  
mysql> create function giveRaise (oldval double, amount double  
-> returns double  
-> deterministic  
-> begin  
->     declare newval double;  
->     set newval = oldval * (1 + amount);  
->     return newval;  
-> end ;  
Query OK, 0 rows affected (0.00 sec)  
  
mysql> delimiter ;
```


Exemplo de Funções

```
mysql> select name, salary, giveRaise(salary, 0.1) as newsal
-> from employee;
```

name	salary	newsal
john	100000	110000
mary	50000	55000
bob	80000	88000
tom	50000	55000
bill	NULL	NULL

```
5 rows in set (0.00 sec)
```

SQL Triggers (Gatilhos)

- Triggers (ou gatilhos) são usados para **monitorar o BD** e tomar uma ação corretiva ou de acordo com as **regras de negócio** quando uma certa condição ocorre
- Exemplos:
 - ◆ Adicione uma tarifa de R\$10 se o saldo de uma conta após um saque ficar negativo
 - ◆ Limite o aumento de salário de um empregado, para no máximo 5%

```
CREATE TRIGGER trigger-name
    trigger-time trigger-event
    ON table-name
    FOR EACH ROW
        trigger-action;
```

- trigger-time ∈ {BEFORE, AFTER}
- trigger-event ∈ {INSERT,DELETE,UPDATE}

SQL Triggers: Exemplo

```
mysql> select * from employee;
```

id	name	superid	salary	bdate	dno
1	john	3	100000	1960-01-01	1
2	mary	3	50000	1964-12-01	3
3	bob	NULL	80000	1974-02-07	3
4	ton	1	50000	1970-01-17	2
5	bill	NULL	NULL	1985-01-20	1

```
5 rows in set (0.00 sec)
```



```
mysql> select * from deptsal;
```

dnumber	totalsalary
1	100000
2	50000
3	130000

```
3 rows in set (0.00 sec)
```

- Deseja-se criar uma trigger para atualizar o salário total de um departamento quando um **novo empregado é contratado**

SQL Triggers: Exemplo

Trigger para atualizar o salário total de um departamento quando um novo empregado é **contratado**

```
mysql> delimiter ;
mysql> create trigger update_salary
-> after insert on employee
-> for each row
-> begin
->     if new.dno is not null then
->         update deptsal
->         set totalsalary = totalsalary + new.salary
->         where dnumber = new.dno;
->     end if;
-> end ;
Query OK, 0 rows affected (0.06 sec)
mysql> delimiter ;
```

- A palavra chave “new” refere-se a nova linha (ou registro) a ser inserido

SQL Triggers: Exemplo

```
mysql> select * from deptsal;
+-----+-----+
| dnumber | totalsalary |
+-----+-----+
| 1       | 100000      |
| 2       | 50000       |
| 3       | 130000      |
+-----+-----+
3 rows in set (0.00 sec)

mysql> insert into employee values (6,'lucy',null,90000,'1981-01-01',1);
Query OK, 1 row affected (0.08 sec)

mysql> select * from deptsal;
+-----+-----+
| dnumber | totalsalary |
+-----+-----+
| 1       | 190000      |
| 2       | 50000       |
| 3       | 130000      |
+-----+-----+
3 rows in set (0.00 sec)

mysql> insert into employee values (7,'george',null,45000,'1971-11-11',null);
Query OK, 1 row affected (0.02 sec)

mysql> select * from deptsal;
+-----+-----+
| dnumber | totalsalary |
+-----+-----+
| 1       | 190000      |
| 2       | 50000       |
| 3       | 130000      |
+-----+-----+
3 rows in set (0.00 sec)

mysql> drop trigger update_salary;
Query OK, 0 rows affected (0.00 sec)
```

← totalsalary aumentou

totalsalary não mudou!
Porque?

SQL Triggers: Exemplo

- Trigger para atualizar o total de salários de um departamento quando um registro da tabela EMPREGADO é **modificado**

```
mysql> delimiter !
mysql> create trigger update_salary2
-> after update on employee
-> for each row
-> begin
->     if old.dno is not null then
->         update deptsal
->         set totalsalary = totalsalary - old.salary
->         where dnumber = old.dno;
->     end if;
->     if new.dno is not null then
->         update deptsal
->         set totalsalary = totalsalary + new.salary
->         where dnumber = new.dno;
->     end if;
-> end !
Query OK, 0 rows affected (0.06 sec)
```

SQL Triggers: Exemplo

```
mysql> delimiter ;
mysql> select * from employee;
```

id	name	superid	salary	bdate	dno
1	john	3	100000	1960-01-01	1
2	nary	3	50000	1964-12-01	3
3	bob	NULL	80000	1974-02-07	3
4	tom	1	50000	1970-01-17	2
5	bill	NULL	NULL	1985-01-20	1
6	lucy	NULL	90000	1981-01-01	1
7	george	NULL	45000	1971-11-11	NULL

```
7 rows in set (0.00 sec)

mysql> select * from deptsal;
```

dnumber	totalsalary
1	190000
2	50000
3	130000

```
3 rows in set (0.00 sec)

mysql> update employee set salary = 100000 where id = 6;
Query OK, 1 row affected (0.03 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from deptsal;
```

dnumber	totalsalary
1	200000
2	50000
3	130000

```
3 rows in set (0.00 sec)
```

SQL Triggers: Exemplo

- Trigger para atualizar o total de salários de um departamento quando um registro da tabela EMPREGADO é **deletado**:

```
mysql> delimiter ;
mysql> create trigger update_salary3
-> before delete on employee
-> for each row
-> begin
->     if (old.dno is not null) then
->         update deptsal
->         set totalsalary = totalsalary - old.salary
->         where dnumber = old.dno;
->     end if;
-> end ;
Query OK, 0 rows affected (0.08 sec)

mysql> delimiter ;
```

SQL Triggers: Exemplo

```
mysql> select * from employee;
```

id	name	superid	salary	bdate	dno
1	john	3	100000	1960-01-01	1
2	mary	3	50000	1964-12-01	3
3	bob	NULL	80000	1974-02-07	3
4	tom	1	50000	1970-01-17	2
5	bill	NULL	NULL	1985-01-20	1
6	lucy	NULL	100000	1981-01-01	1
7	george	NULL	45000	1971-11-11	NULL

```
7 rows in set (0.00 sec)
```

```
mysql> select * from deptsal;
```

dnumber	totalsalary
1	200000
2	50000
3	130000

```
3 rows in set (0.00 sec)
```

```
mysql> delete from employee where id = 6;  
Query OK, 1 row affected (0.02 sec)
```

```
mysql> delete from employee where id = 7;  
Query OK, 1 row affected (0.03 sec)
```

```
mysql> select * from deptsal;
```

dnumber	totalsalary
1	100000
2	50000
3	130000

```
3 rows in set (0.00 sec)
```

SQL Triggers

- Comando para listar todas as triggers que você criou

```
mysql> show triggers;
```