

# Agenda 03: Gatilhos

## Tecnologia da Informação III

Mônica Zungalo Quintal

1. Desenvolva um gatilho (trigger) que insira um registro de garagem coberta para todo apartamento incluído do tipo cobertura.

a) Exibindo os dados registrados na **tabela apartamento**:

The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' pane shows the database structure, including the 'apartamento' table. The 'Trigger: apartamento\_AFTER\_INSERT' is defined with the following definition:

```
Definition:
Event INSERT
Timing AFTER
```

The 'Query 1' pane shows the SQL query: `select * from apartamento;`. The 'Result Grid' displays the following data:

numero	tipo	codigo_cond	valor
a101	padrao	1	100000.00
a201	padrao	1	115000.00
a301	padrao	1	125000.00
a401	padrao	1	135000.00
a501	cobertura	1	150000.00
b101	padrao	2	220000.00
b201	padrao	2	235000.00
b301	padrao	2	247500.00
b401	padrao	2	258500.00
b501	cobertura	2	275000.00
teste	cobertura	1	10.00

The 'Output' pane shows the 'Action Output' for the trigger, indicating that changes were applied to the 'apartamento' table.

*select \* from apartamento;*

b) Exibindo os dados registrados na **tabela garagem**:

The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' pane shows the database structure, including the 'garagem' table. The 'Trigger: apartamento\_AFTER\_INSERT' is defined with the following definition:

```
Definition:
Event INSERT
Timing AFTER
```

The 'Query 1' pane shows the SQL query: `select * from garagem;`. The 'Result Grid' displays the following data:

numero	tipo	numero_ap
1	padrao	a101
2	padrao	a201
3	padrao	a301
4	padrao	a401
5	coberta	a501
6	padrao	b101
7	padrao	b101
8	padrao	b201
9	padrao	b201
10	padrao	b301
11	padrao	b301
12	padrao	b401
13	padrao	b401
14	coberta	b501
15	coberta	b501

The 'Output' pane shows the 'Action Output' for the trigger, indicating that changes were applied to the 'garagem' table.

*select \* from garagem;*

- c) Na tabela apartamento, realizada implementação do **trigger** de **AFTER INSERT**. Ou seja, após inserção de um novo apartamento, caso este seja do tipo “cobertura”, será inserida uma linha na tabela garagem, definindo-a do tipo “coberta”.

MySQL Workbench interface showing the creation of the trigger `apartamento_AFTER_INSERT` on the `apartamento` table. The trigger is defined as an `AFTER INSERT` trigger. The SQL code is:

```
1 CREATE DEFINER='root'@'localhost' TRIGGER `apartamento_AFTER_INSERT` AFTER INSERT ON `apartamento` FOR EACH ROW BEGIN
2 IF NEW.tipo = 'cobertura' THEN
3     INSERT INTO garagem (tipo, numero_ap) VALUES ('coberta', NEW.numero);
4 END IF;
5 END
```

The output shows the trigger was successfully applied:

#	Time	Action	Message	Duration / Fetch
1	21:31:13	Apply changes to apartamento	Changes applied	
2	21:31:30	select "from apartamento LIMIT 0, 1000	11 row(s) returned	0.000 sec / 0.000 sec
3	21:31:40	select "from garagem LIMIT 0, 1000	15 row(s) returned	0.000 sec / 0.000 sec
4	21:32:17	select "from apartamento LIMIT 0, 1000	11 row(s) returned	0.015 sec / 0.000 sec

```
CREATE DEFINER='root'@'localhost' TRIGGER `apartamento_AFTER_INSERT` AFTER INSERT ON `apartamento` FOR EACH ROW BEGIN
IF NEW.tipo = 'cobertura' THEN
    INSERT INTO garagem (tipo, numero_ap) VALUES ('coberta', NEW.numero);
END IF;
END
```

- d) Para verificar o funcionamento do trigger, é inserida uma nova cobertura, conforme codificação a seguir:

MySQL Workbench interface showing the execution of the following SQL statement:

```
1 insert into apartamento values ('test2', 'cobertura', 2, '1000000');
```

The output shows the statement was successfully executed, affecting 1 row:

#	Time	Action	Message	Duration / Fetch
1	21:38:23	insert into apartamento values ('test2', 'cobertura', 2, '1000000')	1 row(s) affected	0.000 sec

```
insert into apartamento values ('test2', 'cobertura', 2, '1000000');
```

e) Exibindo dados da tabela apartamento, e verificando a nova linha inserida:

The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' pane shows the 'imobiliaria' database selected. The 'apartamento' table is highlighted under the 'Tables' section. The main area displays the 'Query 1' window with the SQL statement `select * from apartamento;`. Below the query, the 'Result Grid' shows the data from the 'apartamento' table. A blue arrow points to the new row 'test2' in the 'cobertura' type, with a value of 2 in the 'codigo\_cond' column and 1000000.00 in the 'valor' column.

numero	tipo	codigo_cond	valor
a101	padrao	1	100000.00
a201	padrao	1	115000.00
a301	padrao	1	125000.00
a401	padrao	1	135000.00
a501	padrao	1	150000.00
b101	padrao	2	220000.00
b201	padrao	2	235000.00
b301	padrao	2	247500.00
b401	padrao	2	258500.00
test2	cobertura	2	1000000.00

The 'Output' pane at the bottom shows the execution results of the query, including the message '12 row(s) returned'.

*select \* from apartamento;*

f) Exibindo dados da tabela garagem, e verificando a nova linha inserida pelo trigger:

The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' pane shows the 'imobiliaria' database selected. The 'garagem' table is highlighted under the 'Tables' section. The main area displays the 'Query 1' window with the SQL statement `select * from garagem;`. Below the query, the 'Result Grid' shows the data from the 'garagem' table. A blue arrow points to the new row 'test2' in the 'cobertura' type, with a value of 2 in the 'codigo\_cond' column and 1000000.00 in the 'valor' column.

numero	tipo	numero_ap
2	padrao	a201
3	padrao	a301
4	padrao	a401
5	padrao	a501
6	padrao	b101
7	padrao	b101
8	padrao	b201
9	padrao	b201
10	padrao	b301
11	padrao	b301
12	padrao	b401
13	padrao	b401
14	padrao	b501
15	padrao	b501
16	cobertura	test2

The 'Output' pane at the bottom shows the execution results of the query, including the message '16 row(s) returned'.

*select \* from garagem;*

2. Desenvolva um gatilho (trigger) que desvalorize o preço do apartamento em 30% quando uma garagem vinculada a ele for excluída.

a) Exibindo os dados registrados na **tabela apartamento**:

The screenshot shows the MySQL Workbench interface. The 'Schemas' pane on the left displays the database structure, including the 'imobiliaria' database and its tables: 'apartamento', 'condominio', and 'garagem'. The 'Query' pane shows the SQL query: `select * from apartamento;`. The 'Result Grid' displays the data from the 'apartamento' table. A green arrow points to the row with 'numero' 1000000.00, 'tipo' 'cobertura', and 'valor' 1000000.00.

numero	tipo	codigo_cond	valor
a101	padrao	1	100000.00
a201	padrao	1	115000.00
a301	padrao	1	125000.00
a401	padrao	1	135000.00
a501	cobertura	1	150000.00
b101	padrao	2	220000.00
b201	padrao	2	235000.00
b301	padrao	2	247500.00
b401	padrao	2	258500.00
b501	cobertura	2	270000.00
test2	cobertura	2	1000000.00

*select \* from apartamento;*

b) Exibindo os dados registrados na **tabela garagem**:

The screenshot shows the MySQL Workbench interface. The 'Schemas' pane on the left displays the database structure, including the 'imobiliaria' database and its tables: 'apartamento', 'condominio', and 'garagem'. The 'Query' pane shows the SQL query: `select * from garagem;`. The 'Result Grid' displays the data from the 'garagem' table. A green arrow points to the row with 'numero' 16, 'tipo' 'cobertura', and 'numero\_ap' 'test2'.

numero	tipo	numero_ap
1	padrao	a101
2	padrao	a201
3	padrao	a301
4	padrao	a401
5	cobertura	a501
6	padrao	b101
7	padrao	b101
8	padrao	b201
9	padrao	b201
10	padrao	b301
11	padrao	b301
12	padrao	b401
13	padrao	b401
14	cobertura	b501
16	cobertura	test2

*select \* from garagem;*

- c) Na tabela garagem, realizada implementação do **trigger** de **AFTER DELETE**. Ou seja, após exclusão de um registro da tabela garagem, o campo “valor” da tabela apartamento será atualizado, com uma desvalorização de 30%.

The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' pane shows the 'imobiliaria' database selected. The 'Triggers' pane for the 'garagem' table shows the 'garagem\_AFTER\_DELETE' trigger. The main editor displays the SQL code for creating the trigger:

```
1 CREATE DEFINER='root'@'localhost' TRIGGER `garagem_AFTER_DELETE` AFTER DELETE ON `garagem` FOR EACH ROW BEGIN
2
3     update apartamento
4     set valor = valor * 0.7
5     where numero = OLD.numero_ap;
6
7 END
```

The 'Output' pane at the bottom shows the execution results, including a message: "Error Code: 1109. Unknown table 'apto' in field list".

```
CREATE DEFINER='root'@'localhost' TRIGGER `garagem_AFTER_DELETE` AFTER DELETE ON `garagem` FOR EACH ROW BEGIN
update apartamento
set valor = valor * 0.7
where numero = OLD.numero_ap;
END
```

- d) Para verificar o funcionamento do trigger, realizada exclusão da garagem do **exemplo “test2”** e exibição dos valores da tabela garagem:

The screenshot shows the MySQL Workbench interface. The main editor displays the SQL code for deleting a record from the 'garagem' table and selecting all records:

```
1 delete from garagem where numero_ap="test2";
2 select * from garagem;
```

The 'Result Grid' pane shows the data returned by the second query, displaying 15 rows of data with columns 'numero', 'tipo', and 'numero\_ap'. The 'Output' pane at the bottom shows the execution results, including a message: "1 row(s) affected" and "15 row(s) returned".

```
delete from garagem where numero_ap='test2';
select * from garagem;
```

e) Exibindo os registros da tabela apartamento; em destaque, o apartamento desvalorizado:

The screenshot shows the MySQL Workbench interface. The left sidebar displays the database schema, including the 'imobiliaria' database and its tables. The main window shows a query result for the 'apartamento' table. The query is `select * from apartamento;`. The result grid shows 12 rows. The row for 'test2' is highlighted in green, indicating it is the 'desvalorizado' (devalued) apartment. A green arrow points to this row.

numero	tipo	codigo_cond	valor
a101	padrao	1	100000.00
a201	padrao	1	115000.00
a301	padrao	1	125000.00
a401	padrao	1	135000.00
a501	cobertura	1	150000.00
b101	padrao	2	220000.00
b201	padrao	2	236500.00
b301	padrao	2	247500.00
b401	padrao	2	258500.00
test2	cobertura	2	700000.00
teste	cobertura	1	10.00

The bottom panel shows the Action Output, indicating that the query returned 12 rows.

*select \* from apartamento;*