

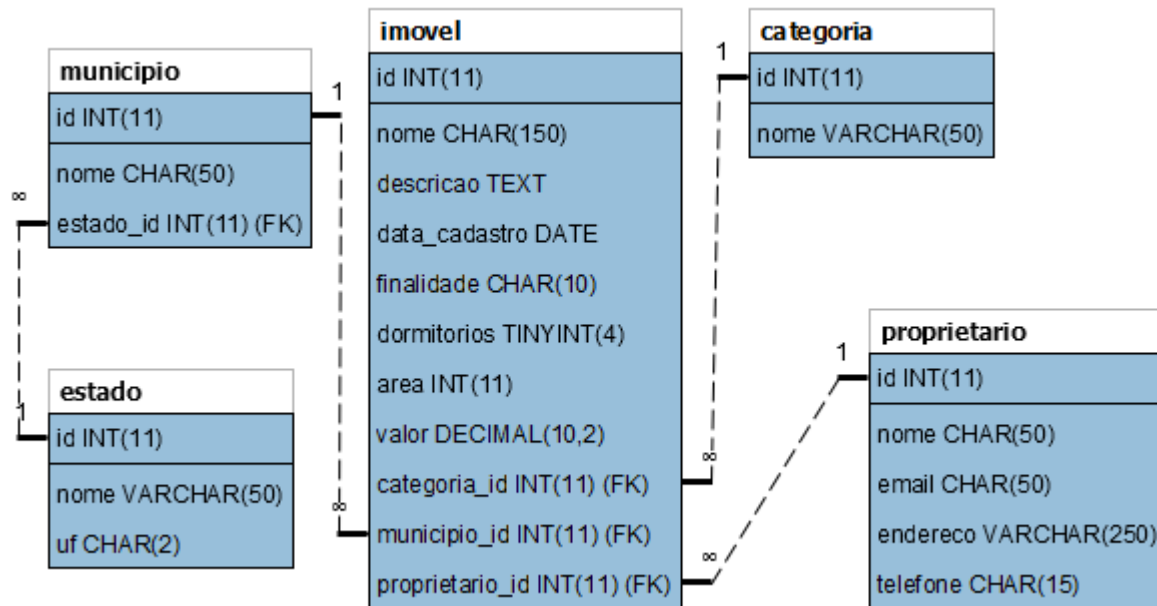
**INSTITUTO
FEDERAL**
Santa Catarina

Banco de Dados 2



Materiais desta aula

Para esta aula precisaremos do BD “Imobiliaria”



- 1) Visões
- 2) Procedimentos
- 3) Gatilhos



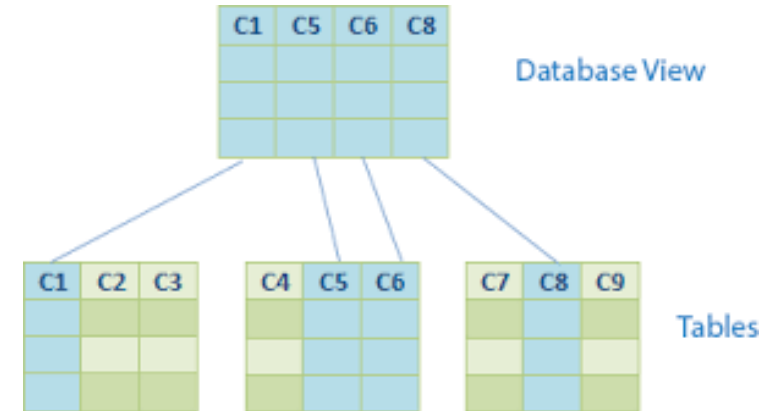
1/3

Visões



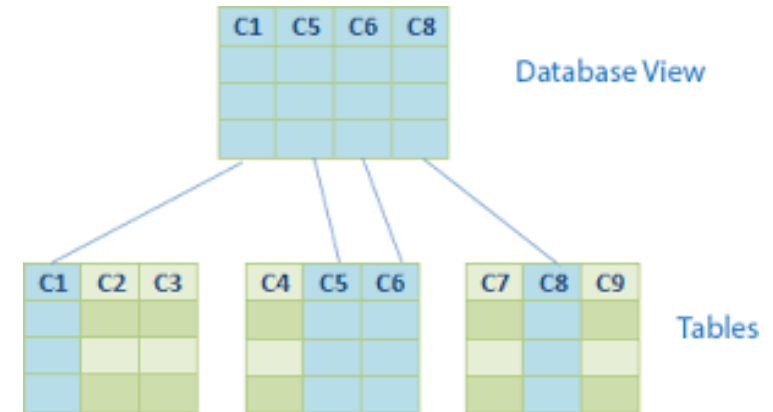
O que são visões?

- Na prática, visões funcionam como uma tabela originária de uma consulta.
- Em vez de sempre utilizar uma consulta complicada e otimizada para uma tarefa rotineira, você pode transformá-la em uma visão para que possa ser consultada sempre que necessário.
- Também são chamadas de “virtual tables”



Vantagens das views

- Podem servir para promover restrições de acesso em nível de tabela e coluna;
- Podem ser utilizadas com um conjunto de tabelas que podem ser unidas com JOINS;
- Podem retornar um valor baseado em um identificador de registro.



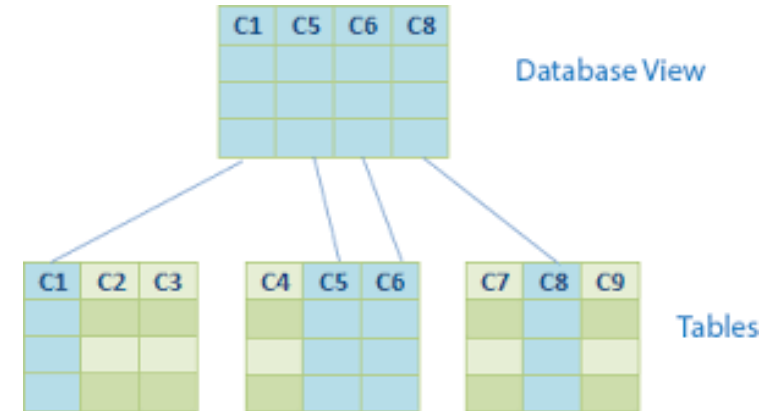
Comando de criação de uma view

- **view_name:** é o nome que se dá ao objeto criado;
- **column_list:** recurso que permite sobrescrever os nomes das colunas que serão recuperadas pela declaração SELECT;
- **select_statement:** é a declaração SELECT que indica a forma na qual se deseja que os dados sejam retornados. Podem ser utilizadas as funções JOINS e UNION, caso necessário. Também podem ser utilizadas quaisquer tabelas ou views contidas no servidor de bancos de dados MySQL;
- **OR REPLACE:** pode ser utilizada para substituir uma view de mesmo nome existente no banco de dados ao qual pertence;
- **ALGORITHM:** essa cláusula define qual algoritmo interno utilizar para processar a view quando esta for invocada. Ele podem ser UNDEFINED (cláusula em branco), MERGE ou TEMPTABLE;
- **WITH CHECK OPTION:** todas as declarações que tentarem modificar dados de uma view definida com essa cláusula serão revisadas para checar se as modificações respeitarão a condição WHERE, definida no SELECT da view. Preserva as regras do where no dado a ser inserido ou atualizado. Se não respeitar estas regras, retorna um erro.

```
CREATE [OR REPLACE] [ALGORITHM = algorithm_type] VIEW view_name [(column_list)]  
AS select_statement [WITH [CASCADED | LOCAL] CHECK OPTION]
```

Atualizações em Views

- Informações de views criadas com comandos selects com funções agregadoras não são atualizáveis ou podem ser excluídas.
- O mesmo acontece para views com group by, having ou distinct.



Exemplos

-- cria uma visão que busca as casas do banco

```
CREATE VIEW casas_garopaba AS
```

```
SELECT nome, valor
```

```
FROM imovel
```

```
WHERE categoria_id=2;
```

Exemplos

-- cria uma visão que busca as cidades de SC com CHECK OPTION

```
CREATE VIEW cidades_sc AS  
SELECT *  
FROM municipio  
WHERE estado_id=24 WITH CHECK OPTION;
```

-- este comando funciona:

```
INSERT INTO cidades_sc (nome, estado_id) VALUES ('teste', 24);
```

-- este não funciona

```
INSERT INTO cidades_sc (nome, estado_id) VALUES ('teste', 20);
```

Desafio de código SQL



- Implemente uma VIEW que exiba o nome, valor e nome da categoria dos imóveis que estão para aluguel em Imbituba.

```
-- exemplo de view  
CREATE VIEW cidades_sc AS  
SELECT *  
FROM municipio  
WHERE estado_id=24 WITH CHECK OPTION;
```

2/3

Procedimentos



Procedimentos

- Um procedimento (Store Procedure) é um conjunto de instruções que realiza determinada tarefa.
- São coleções de comandos SQL que são armazenados e executados no servidor MySQL
- Vantagens:
 - Mais velocidade
 - Menos redundância, melhor manutenção
 - Aumento da segurança
- Desvantagem
 - Difícil portabilidade entre SGBDs



Exemplo 1

- Objetivo: fazer um procedimento que receba um nome e retorne “Olá ”+ nome

```
CREATE PROCEDURE ola (nome char(100))  
SELECT concat("Olá ", nome, "!");
```

Para chamar...

```
CALL ola("Miguel");
```

DELIMITER

- O delimitador de instruções do MySQL é o “;”
- Isto pode trazer problemas na criação de procedimentos mais complexos, portanto usaremos outra definição de delimitador
- o DELIMITER, é um comando do mysql que informa onde seu script tem início, e onde encerra

O uso do Delimiter deve ser acompanhado de algum símbolo que seja reconhecido pelo mysql. O mais comum é utilizar dois cifrões (\$).

```
DELIMITER $$
```

```
-- código aqui
```

```
$$
```

Exemplo 2

- Ver uma quantidade limitada de cidades de SC

```
DELIMITER $$  
CREATE PROCEDURE ver_cidades_sc(quantidade INT)  
BEGIN  
    SELECT * FROM municipio WHERE estado_id=24 LIMIT quantidade;  
END $$  
DELIMITER ;
```


Para chamar

```
CALL ver_cidades_sc(50);
```

Exemplo 3

- Fazer um procedimento que salva em uma variável a quantidade de filmes do banco de dados

```
DELIMITER $$  
CREATE PROCEDURE valores_garopaba(OUT valores INT)  
BEGIN  
    SELECT SUM(valor) INTO valores FROM casas_garopaba;  
END $$  
DELIMITER ;
```



Para chamar

-- chama o procedimento que registrará a variável @qnt

CALL total(@qnt);

-- exhibe o valor de @qnt

SELECT @qnt;

Excluindo um procedimento

-- Para excluir o procedimento, utilize o comando de acordo com o exemplo abaixo:

- `DROP PROCEDURE valores_garopaba;`

Exibindo procedimentos salvos

SHOW PROCEDURE STATUS;

Desafio de código SQL



- Implemente uma STORE PROCEDURE que recebe dois valores (mínimo e máximo) e exiba os imóveis para **venda** nesta faixa.
- A lista de imóveis a venda deve vir de uma view específica de imóveis sendo vendidos.

```
-- exemplo de store procedure
DELIMITER $$
CREATE PROCEDURE ver_cidades_sc(quantidade INT)
BEGIN
    SELECT * FROM municipio WHERE estado_id=24 LIMIT quantidade;
END $$
DELIMITER ;
```


3/3

Gatilhos



- Uma trigger, ou gatilho, é um objeto de banco de dados associado a uma tabela, definido para ser disparado, respondendo a um evento em particular.
- Tais eventos são os comandos da DML: INSERT, REPLACE, DELETE ou UPDATE. Podem-se definir inúmeros triggers em uma base de dados, baseados diretamente a um dos comandos acima para dispará-lo, sendo que; para cada um, pode-se definir apenas um gatilho.
 - Não se pode chamar diretamente um TRIGGER com CALL, como se faz com uma stored procedure;
 - Não é permitido iniciar ou finalizar transações em meio à trigger;
 - Não se pode criar uma trigger para uma tabela temporária;

- Dentro das triggers existem duas tabelas virtuais que possibilitam acessar as colunas da tabela-alvo do comando DML: é possível acessar os valores que serão enviados para a tabela antes (BEFORE) ou depois (AFTER) de um comand. Tais tabelas recebem os nomes de OLD e NEW.
- A tabela OLD contém os dados que serão removidos da tabela. A NEW contém os dados que serão inseridos.
 - INSERT possui NEW
 - UPDATE possui NEW e OLD
 - DELETE possui OLD

Gatilhos (triggers)

Gatilhos permitem a execução de um conjunto de instruções SQL ou de um procedimento automaticamente, antes ou depois de comandos INSERT, UPDATE ou DELETE

-- Sintaxe:

DELIMITER \$\$

CREATE TRIGGER nome_da_trigger AFTER/BEFORE INSERT/UPDATE/DELETE ON
nome_da_tabela

FOR EACH ROW

BEGIN

-- sua query aqui terminando com ponto-e-vírgula

END \$\$

DELIMITER ;

Exemplo

```
DELIMITER $$
```

```
CREATE TRIGGER minha_trigger AFTER INSERT ON minha_tabela1  
FOR EACH ROW  
BEGIN  
    DELETE FROM minha_tabela2;  
END $$
```

```
DELIMITER ;
```

Ver os gatilhos

SHOW TRIGGERS;

Exemplo:

- Considere as duas tabelas abaixo:
 - departamento(cod, nome, empregados);
 - empregado (matricula, nome, cod_depto);
- A coluna empregados da tabela departamento deve armazenar a quantidade de empregados que pertencem a seu departamento. Como manter esse número atualizado?

Exemplo

```
DELIMITER $$  
CREATE TRIGGER adiciona AFTER INSERT ON empregado  
FOR EACH ROW BEGIN  
    UPDATE departamento SET empregados =empregados+1 WHERE cod =NEW.cod_depto;  
END $$  
DELIMITER ;
```


Exemplo

```
DELIMITER $$  
CREATE TRIGGER remove AFTER DELETE ON empregado  
FOR EACH ROW BEGIN  
    UPDATE departamento SET empregados = empregados - 1 WHERE cod = OLD.cod_depto;  
END $$  
DELIMITER ;
```

Desafio de código SQL



- 1) Implemente uma TRIGGER que faça backup em outra tabela dos imóveis que forem excluídos.
- 2) Modifique o BD de forma que cada proprietário tenha uma coluna com o valor total de seu patrimônio. Cada vez que um imóvel é adicionado, alterado ou excluído, o respectivo valor de patrimônio do proprietário deve ser atualizado.

```
-- exemplo de trigger
DELIMITER $$
CREATE TRIGGER remove AFTER DELETE ON empregado
FOR EACH ROW BEGIN
    UPDATE departamento SET empregados = empregados-1 WHERE cod = OLD.cod_depto;
END $$
DELIMITER ;
```