**Modificadores de informação**

Select – resultado montado em uma tabela

**select** (col1,col2.. \*) **from** table **where** condição **order by** col sentido **limit** 2**;**

**select \* from** venda (; daria select em toda tabela) (\* = traga todas colunas)

**where** id>2 (; id maior que 2 apareceria)

**select** ‘Ola’ *ou* 1+1 *ou* nome, idade from table; (log: Ola ; 2; colunas nome e idade) (SEMPRE definir o select em DB grandes)

**order by** nome; (default puxa **ASC**, apareceria “Arnaldo, Beto, Caio,..”. pode: **DESC**)

(pode por mais de uma ordem: nome, valor;)

**limit** 2; (aqui limita só rows – quantos números aparecer)

**limit** 1,2; (1º numero – por onde começar, 2º número quantos números aparecer) (limit 2,2 seria começar no id 3 e por 2 registros)

**insert into** vendedor (nome, comissao, matricula)

**values** ('Olanda',10,'123456739');

-Prática- se usa select dentro da modificação:

**insert into** cidade (nome, estado\_id) **values**

(‘Imbé’, (**select** id **from** estado **where** sigla=’RS’)) (estado é unique, puxa id certo)

**update** venda **set** cliente=2 (; update em toda coluna de uma tabela)

**where** id=1;

**delete from** venda (; - deletaria tudo)

**where** id=4;

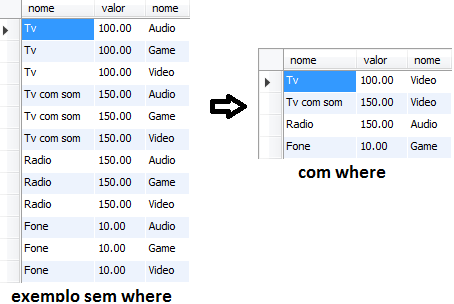
**Joining / joins – juntar tabelas**

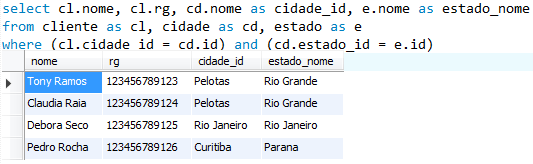
**select** p.nome, p.valor, t.nome

**from** produto **as** p, tipo **as** t

**where** p.tipo\_id = t.id; (isso faria uma tabela sem info falsa) (as é opcional)

(poderia mudar o nome da 3ª coluna usando *t.nome as nome\_tipo*)





**Inner join: (traz dados relacionados)**

**Select** col1, col2... **from** table1 **inner|left|right join** table2 **on** condição;

... from produto as p **Inner join** tipo as t **on** p.tipo\_id = t\_id

(é preferivel usar essa sintexe do que o simples from + where)

**Right join: (prioriza direita mesmo que nao estejam relacionados – aí ambos podendo ser nulos)**

... from produto as p **right join** tipo as t **on** p.tipo\_id = t\_id

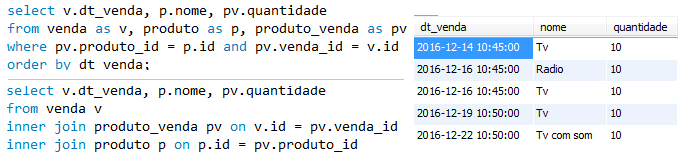
(aqui puxaria produtos E tipos, e ainda os tipos que não possuem produtos)

**Left join: (prioriza esquerda)**

... from produto as p **left join** tipo as t **on** p.tipo\_id = t\_id

(aqui puxaria produtos E tipos, e ainda os produtos que não possuem produtos)

- Obs.: outras linguagens ainda apresentam mais Joins. (<https://blog.codinghorror.com>)



**Operadores/ funções**

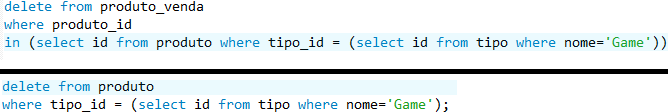
id = 2; >; <; >=; <=; <>;

dt\_venda < ‘2016-01-01’ (todos registros de 2016)

condicao1 AND condição2 (único caso verdadeiro é V AND V)

condição1 OR condição2 (único caso falso é F AND F)

in – dentro de alguma coisa ( **delete from** x **where** y **IN** varias\_respostas)



Comando Like + “%” – puxa informações não completas (thomas – like a virgin)

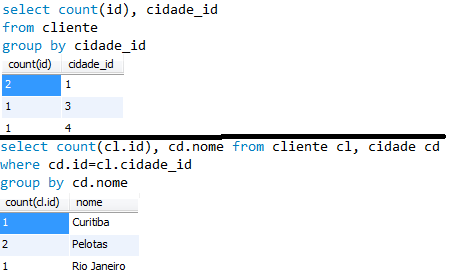


(alguns DBA’s banem os like, pois usam mto processamento)

Função Lower / Upper () –

Select \* from tipo where **lower** (nome) like ‘v%’ – (algumas DB tem diferenciação de maisc. e minus., assim se usa função pra primeiro ‘minuscular’ todos registro)

Função Count () – select **count** (id) from table – (irá contar o número de registros id) (group by – mais usada no count, pois agrupa vários ‘contados’ por grupos)



Função Max/ Min/ Avg () – select max (valor) from table – (traz valor mais alto,baixo)

Outros casos.: Not in (2,4,6) / not like (‘A%’) / Where coli s Not Null

**mysql > mysql console**

**use** loja\_dez;

(por padrão o mysql abre a sua database, que contém as DB salvas, por isso selecionar primeiro qual vai usar.)

---

**begin;** (começa uma transação, assim quando aplicar no console

nao ira dar auto-commit)

**rollback;** (volta o erro apos begin)

se estiver tudo ok> **commit;**

(assim toda transação começa com: begin + codigo + conferencia(rollback ou commit); por padrao no mwsql, é auto-commit; por isso sempre por begin em modificação de informação)

oracle não é auto-commit;

quit; (ao final – opcional)