

Criando e trabalhando com VIEWS

Uma view é um objeto que pertence a um banco de dados, definida e baseada em declarações selects retornando uma determinada visualização de dados de uma ou mais tabelas. Esses objetos são chamados, por vezes, de virtual tables, formados a partir de outras tabelas que, por sua vez, são denominadas de based tables ou ainda outras Views . Em alguns casos, estas são atualizáveis e podem ser alvos de declaração insert, update e delete, que, na verdade, modificam sua based tables.

Vamos criar uma View onde iremos fazer um select e Join nas tabelas cadastro e produtos

```
create or replace view cadastro_produtos as  
  
select Nome_Cliente,Sobrenome,Nome,Quantidade,Valor  
  
from cadastro  
  
INNER JOIN produtos  
  
on cadastro.id_cad = produtos.id_cad  
  
order by 1
```

Para testarmos a nossa view vamos fazer um select na view criada.

```
select * from cadastro_produtos  
  
select Nome,Quantidade from cadastro_produtos  
  
select Nome,Sobrenome,Nome_Cliente,Quantidade from cadastro_produtos  
  
where Quantidade = 5
```

Inserir atualizar ou deletar registros através de Views. Vamos criar uma view simples para uma tabela do banco de dados

create or replace view view_produtos as

select Nome,Valor,Quantidade,id_cad

from produtos

Inserir Dados

insert into view_produtos values("Jabuticaba",8.70,20,9)

Update Dados

Update view_produtos set Valor = 10.20

where Nome = "Jabuticaba"

Delete Dados

delete from view_produtos where Nome = "Jabuticaba"

Para excluir uma View utilizar DROP

drop view view_produtos

Criando Índices Banco de Dados.

Para nos auxiliar no aprimoramento das consultas, o MySQL nos fornece os chamados **índices**. Quando criamos as nossas tabelas, nós já criamos um índice de chave primária (primary key).

Porém, ele não é utilizado para fazer essa otimização de performance, pois serve apenas para cuidar da integridade dos dados.

A medida inicial que devemos tomar para melhorar o tempo das consultas é a criação de índices para as tabelas. Se elas estão demorando para serem concluídas, as primeiras suspeitas são: ou

eles não foram feitos ou estão mal criados. A adequação ou criação dos índices é necessário pode resolver o problema na maioria das vezes; porém não sempre, pois seu banco de dados pode estar lento por outros motivos.

Entretanto, como seu uso é o mais eficaz na questão de otimização, pode ser um desperdício de tempo tentar esse mesmo resultado por outros meios.

Podemos definir índices basicamente como, uma referência associada a uma chave com o objetivo de otimização, permitindo que uma consulta localize rapidamente um registro. Uma comparação muito frequente é a de um índice remissivo de um livro, onde selecionamos determinado assunto no índice do livro verificamos sua página e depois localizamos essa página para leitura.

O MySQL suporta os seguintes tipos de índices:

PRIMARY KEY
UNIQUE
INDEX
FULLTEXT
BTREE (Default)

Abaixo seguem algumas características e regras para utiliza-los:

Índice não único é um no qual qualquer valor da chave pode ocorrer múltiplas vezes. Este tipo de índice é definido com a palavra chave INDEX ou KEY;

Índice UNIQUE possui valor único, ou seja, cada valor da chave deve ser diferente de todos os outros (a exceção é que valores NULL podem ocorrer múltiplas vezes);

PRIMARY KEY também é um índice de valores únicos. Ela é semelhante a um índice UNIQUE, mas tem restrições adicionais;

Uma tabela pode conter múltiplos índices UNIQUE, mas no máximo uma PRIMARY KEY;

Um índice UNIQUE pode conter valores NULL, enquanto uma PRIMARY KEY não;

Índice FULLTEXT é projetado especialmente para a busca em texto;

O B-tree, ou alguma variação dele, é o mais comum em todos os sistemas de banco de dados. Ele é muito eficiente para quase todos os casos comuns de uso. É uma árvore binária balanceada que permite todos os tipos de acesso (leitura, inserção remoção, em qualquer lugar). O,que é muito rápido sempre, com volume muito grande, ou muito pequeno de dados, mas importa mesmo quando é grande. É razoavelmente eficiente em espaço e mantém os dados ordenados, permitindo a leitura sequencial e de faixas eficientemente.

Podemos criar os índices já quando criamos uma nova tabela.

```
create table cliente (  
id_cliente integer not null auto_increment,  
cod_cliente varchar(10),  
Nome varchar(50),  
Sobrenome varchar(50),  
CPF integer,  
D_cadastro date,  
primary key (id_cliente),  
index ind_cod_cliente (cod_cliente)  
);
```

Podemos criar um índice alterando uma tabela existente com o **alter table**.

Criando outro índice na tabela cliente

```
alter table cliente add index index_nome_cliente(Nome);
```

Criando índice na tabela cadastro

```
alter table cadastro add index index_nome_cliente(Nome_Cliente);
```

Criando índice do tipo UNIQUE na tabela Produtos***

```
alter table produtos add unique index index_nome_produto(Nome);
```

Para exibir os índices das tabelas cliente e cadastro e também o tipo dos índices criados.

```
show indexes from cliente
```

```
show indexes from cadastro
```

O índice pode ser excluído de uma tabela também com o comando DROP.

Excluindo um índice da tabela cliente.

```
alter table cliente drop index index_nome_cliente ;
```

Importando e Exportando dados em arquivos CSV e TXT.

Para incluir registros no banco podemos usar os arquivos **.sql** porém podemos usar também os arquivos csv ou txt.

Exportar registro de uma tabela para TXT

```
select * from produtos into outfile 'c:/temp/lista_clientes.txt'
```

```
fields terminated by ',' enclosed by ''';
```

Quando executamos esse comando o MySQL vai retornar um erro:

Error Code: 1290. The MySQL server is running with the --secure-file-priv option so it cannot execute this statement

O mysql retorna o erro de uma variável responsável pelo outfile não executou.

Precisamos entender as variáveis de sistema do MySQL e funções de execução do SGBD. O outfile utiliza de uma variável de sistema.

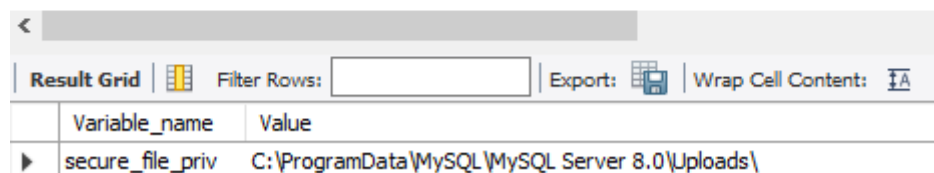
Para vermos as variáveis de ambiente do MySQL executamos o comando.

show variables

Para vermos a variável --secure-file-priv executamos o comando.

Show variables like "secure_file_priv"

```
1 • SHOW VARIABLES like "secure_file_priv"
```



Variable_name	Value
secure_file_priv	C:\ProgramData\MySQL\MySQL Server 8.0\Uploads\

Ela retorna o caminho descrito na imagem. O comando deverá ser executado nesse caminho da variável

Exportando dados para um arquivo txt.

```
select *      from produtos  into outfile  
'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/lista_clientes.txt'  
fields terminated by ',' enclosed by '''';
```

Acessamos o local exportado e o arquivo estará com dado da tabela em TXT.

Importar Arquivos TXT ou CSV para uma tabela do Banco de dados

Criar uma tabela para importar os dados

```
create table ImportVendas (  
CodigoPedido int(10),  
EmailCliente varchar(100),  
CodigoCliente integer,  
Qtd integer,  
CodigoProduto integer,  
CategoriaProduto varchar(50),  
primary key (CodigoPedido)  
);
```

Importar o arquivo CSV gerado com os campos e valores iguais da tabela criada no banco de dados.

```
load data infile 'C:/ProgramData/MySQL/MySQL Server  
8.0/Uploads/clientes.csv' into table importvendas fields terminated by ';'   
enclosed by '"' LINES TERMINATED BY '\n' IGNORE 1 ROWS;
```

Todos os registros do arquivo CSV são importados na tabela ImportVendas.

Para validar podemos executar um select

```
Select * from importvendas
```