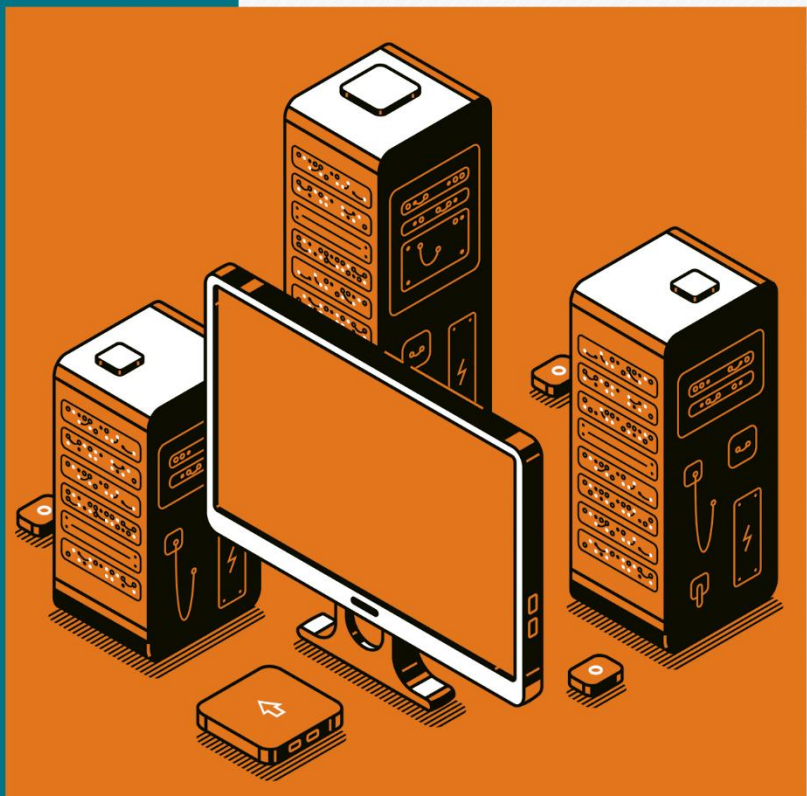

AGENDA 4

VISÕES
CONTROLADAS
E PERMISSÕES



GEEaD - Grupo de Estudos de Educação a Distância
Centro de Educação Tecnológica Paula Souza

GOVERNO DO ESTADO DE SÃO PAULO
EIXO TECNOLÓGICO DE INFORMAÇÃO E COMUNICAÇÃO
CURSO TÉCNICO EM DESENVOLVIMENTO DE SISTEMAS
TECNOLOGIAS DA INFORMAÇÃO III

Expediente

Autor:

José Mendes da Silva Neto

Revisão Técnica:

Eliana Cristina Nogueira Barion

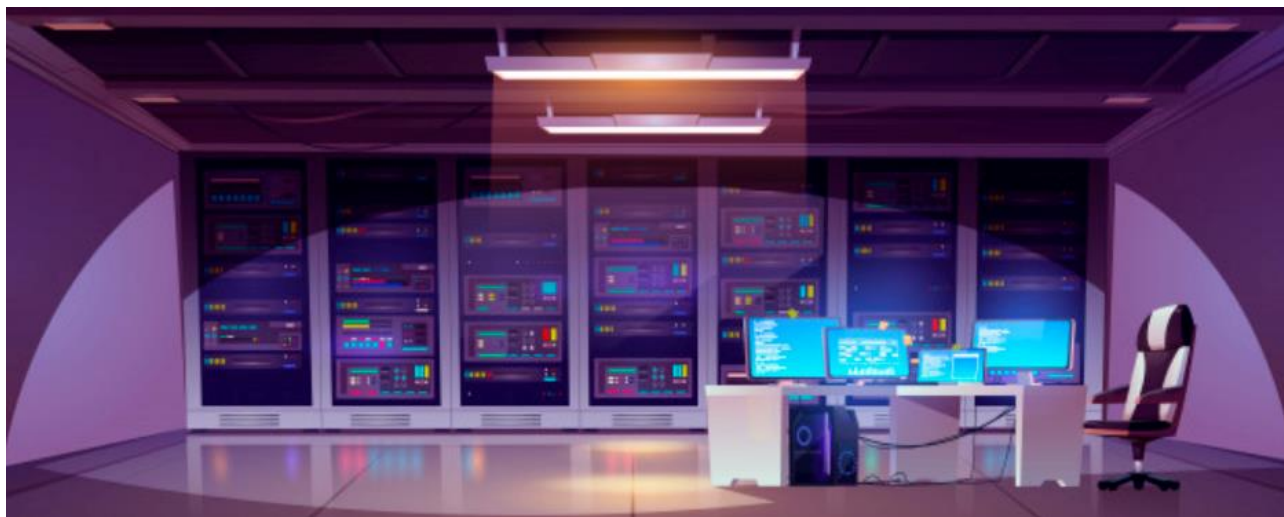
Revisão Gramatical:

Juçara Maria Montenegro Simonsen Santos

Editoração e Diagramação:

Flávio Biazim

São Paulo – SP, 2020



Para qualquer empresa a informação é um ativo bastante precioso!

Com base nessa afirmação imagine que você é responsável pelo **banco de dados** de uma instituição financeira ou fizesse parte da equipe de desenvolvimento. Entre os vários processos existente em um banco, um deles é a utilização do caixa eletrônico.

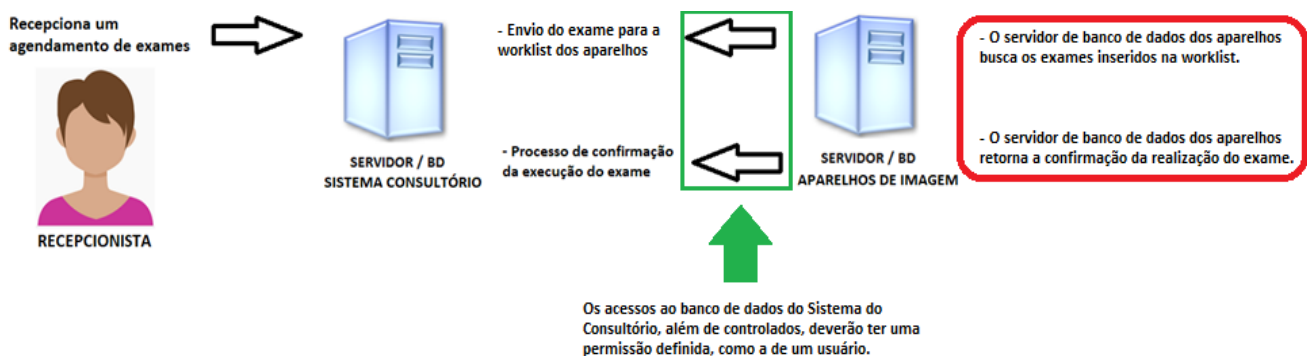
Por exemplo, se uma pessoa vai até o caixa e insere o cartão da conta e suas credencias de acesso, o sistema disponibiliza uma série de transações para ela realizar, porque ele identificou o usuário/cliente, ao contrário, se ela simplesmente tocar na tela do caixa, a quantidade de transações disponíveis será mínima, porque não existe um usuário identificado, não permitindo o acesso aos dados e transações de acordo com as regras de negócio da instituição.



Disponibilizar as informações contidas em um **banco de dados** pode parecer uma tarefa fácil, essa é uma das características padrão de qualquer SGBD. Mas, as dúvidas ou problemas começam quando são necessárias as implementações de regras e controles de acesso, ou seja, quando necessitamos restringir parte ou toda informação de determinadas estruturas do **banco de dados**. Por isso, é muito importante que os profissionais de desenvolvimento de sistemas tenham conhecimento do funcionamento dos mecanismos de controles de acessos do MySQL, bem como suas rotinas para criação de usuários e gerenciamento de seus privilégios.

Chegamos a fase final do projeto de integração entre o Sistema do Consultório e os aparelhos de RX e Ultrassom adquiridos pela Dra. Ana Lúcia.

Carlos, Analista responsável pelo projeto desenvolveu todas as rotinas e processos necessários no **banco de dados** do Sistema do Consultório. O que ele precisa agora é dar acesso as estruturas envolvidas na integração para que o Sistema dos aparelhos possa obter sua **worklist** de exames e posteriormente retornar sua realização, para que o fluxo de atendimento dos exames de imagem do Consultório da Dra. Ana Lúcia esteja completo do início ao fim, desde o agendamento até o faturamento. A imagem a seguir demonstra como isso deverá acontecer.



Na prática, o servidor do sistema dos aparelhos seria um usuário, que por sua vez tem que ter suas credenciais de acesso, além das permissões de acordo com as regras de negócio.

E agora, como Carlos pode resolver isso para finalizar essa integração?



Aplicações bem projetadas geralmente expõem uma interface pública enquanto mantêm privados os detalhes de implementação, permitindo, assim, mudanças futuras de projeto sem impacto nos usuários finais. Ao projetar seu **banco de dados**, você pode obter um resultado semelhante mantendo suas tabelas privadas e permitindo que seus usuários acessem dados apenas por meio de um conjunto de **views**. Vamos definir o que são **views**, como são criadas e quando e como você pode querer usá-las.

O que são views?

Uma **view** é simplesmente um mecanismo de consulta de dados. Diferentemente das tabelas, as **views**, não envolvem armazenamento de dados: você não precisa se preocupar com as **views** ocupando seu espaço em disco. Você cria uma **view** atribuindo um nome a uma instrução **select** e, em seguida, armazenando a consulta para que outros a usem. Outros usuários podem, então, usar sua **view**, para acessar dados como se estivessem consultando tabelas diretamente (na verdade, eles podem nem saber que estão usando **views**).

IMPORTANTE: para aplicar os exemplos a seguir utilize o banco de dados minimercado.

Como um exemplo simples, digamos que você queira ocultar, das informações dos clientes do minimercado, o número do CPF (cadastro de pessoa física), da tabela **cliente**. Em vez de permitir o acesso direto à tabela **cliente**, você define uma **view** chamada **cliente_vw** e determina que todos a usem para acessar os dados de cliente. Veja a definição da **view**:

```
create or replace view cliente_vw (nome_cliente, endereco_cliente) as
```

```
select nome, endereco
from cliente;
```

consulta que contém os campos e as regras para seleção dos dados.

nome da view e os apelidos para os campos que serão apresentados quando da sua execução.

A definição de apelidos para as colunas não é obrigatória, você pode criar uma **view**, sem defini-los. Neste caso, o código para sua criação seria:

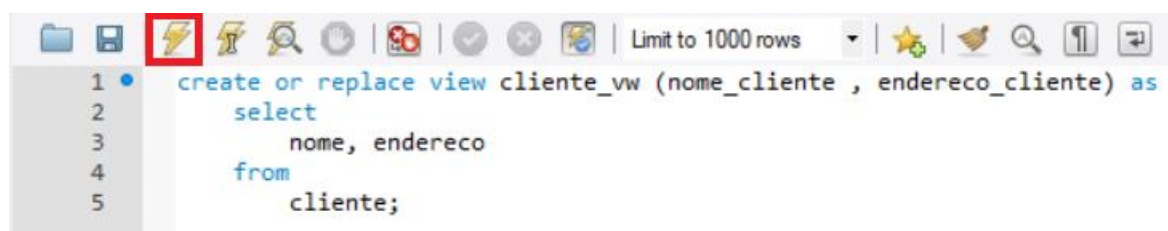
```
create or replace view cliente_vw as
select nome, endereco
from cliente;
```

Note que os apelidos não foram definidos no segundo exemplo, por padrão permanece os nomes definidos na consulta.

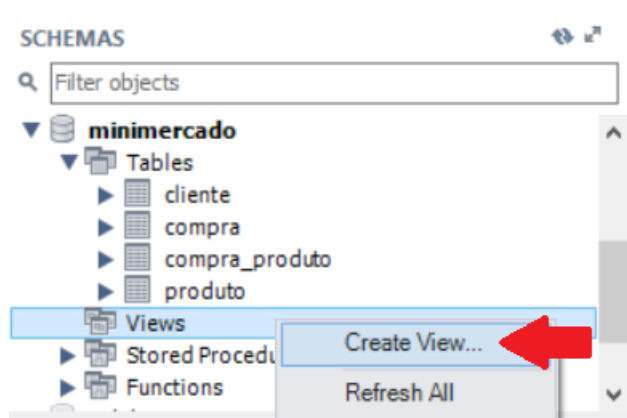
Utilizando a ferramenta WorkBench, assim como nas agendas anteriores você poderá escolher como criar a **view**, pela janela SQL ou utilizando a interface gráfica.

Se preferir, [clique aqui](#) e assista o vídeo que demonstra a criação de view.

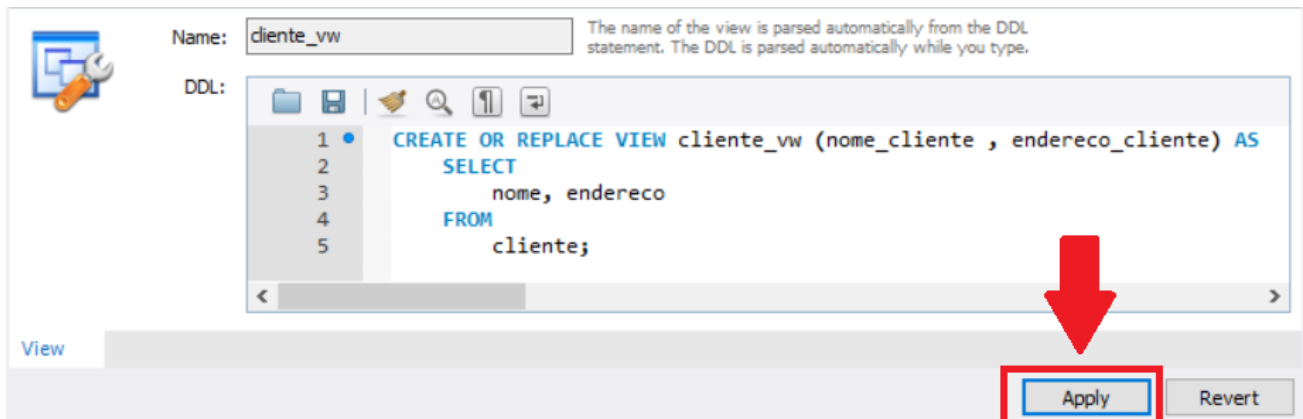
Pela Janela SQL:



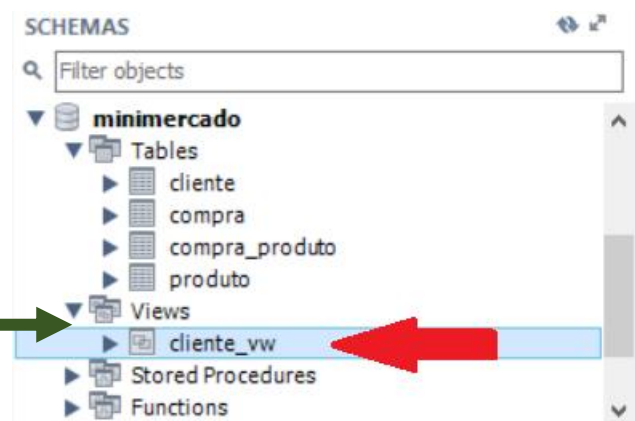
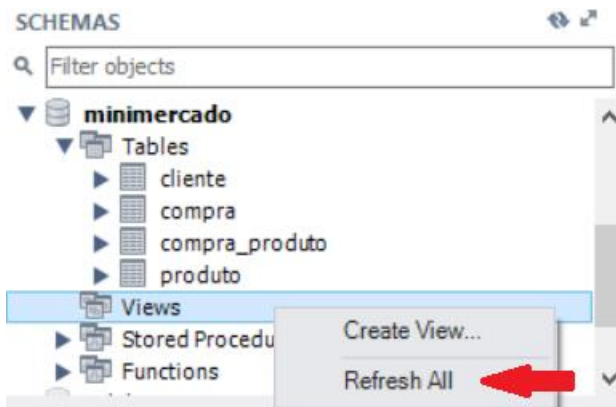
Pela interface gráfica, no quadro **SCHEMAS**, estrutura **Views**:



Será apresentada a interface para criação da **view**. Implemente o código e aplique para que a **view** seja criada.



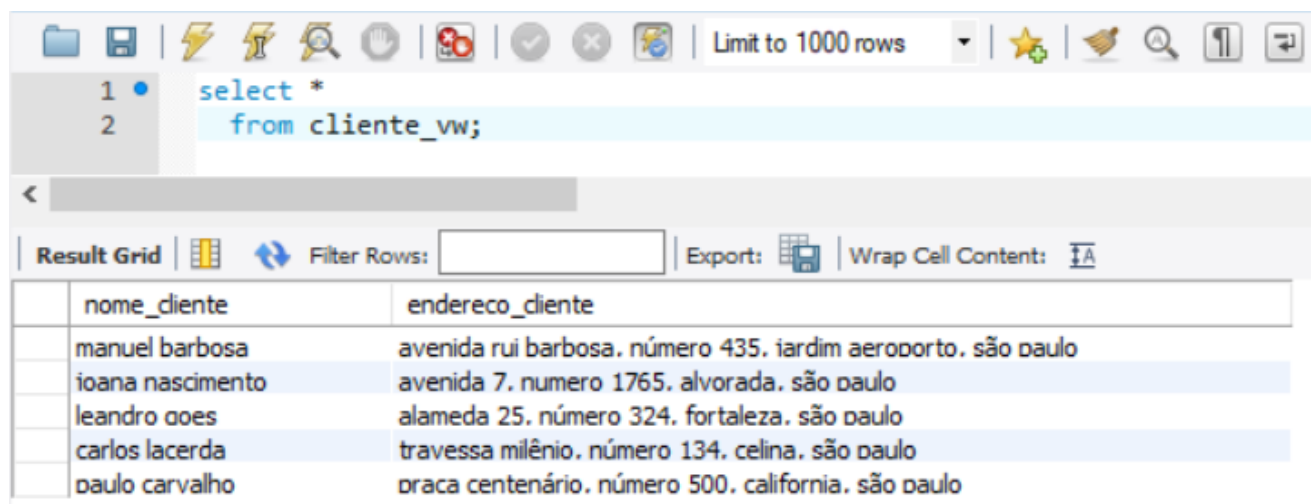
Após confirmar a criação da **view**, atualize a estrutura **Views** do banco de dados **minimercado**, para que ela seja apresentada no quadro **SCHEMAS**.



A primeira parte da instrução lista os nomes das colunas das **views**, que podem ser diferentes dos nomes das colunas da tabela subjacente (por exemplo, a **view** **cliente_vw** tem duas colunas nomeadas **nome_cliente** e **endereco_cliente** que são uma referência às colunas **nome** e **endereco** da tabela **cliente** respectivamente. A segunda parte da instrução é uma instrução **select**, que precisa conter uma expressão para cada coluna da **view**.

Quando a instrução `create view` é executada, o servidor de **banco de dados** simplesmente armazena a definição da **view** para uso futuro. Uma vez que a **view** tenha sido criada, os usuários podem consultá-la como fariam com uma tabela, como em:

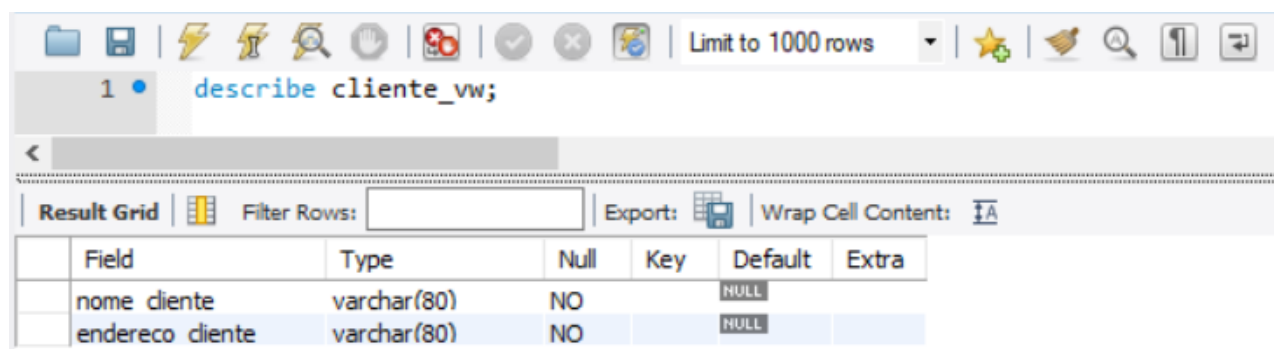
```
select *
  from cliente_vw;
```



nome_cliente	endereco_cliente
manuel barbosa	avenida rui barbosa, número 435, iardim aeroporto, são paulo
ioana nascimento	avenida 7, numero 1765, alvorada, são paulo
leandro ooes	alameda 25, número 324, fortaleza, são paulo
carlos lacerda	travessa milênio, número 134, celina, são paulo
paulo carvalho	praca centenário, número 500, califórnia, são paulo

Do ponto de vista do usuário, uma **view** se parece exatamente com uma tabela. Se quiser saber quais colunas estão disponíveis em uma **view**, você pode usar o comando `describe` para examiná-la.

```
describe cliente_vw;
```



Field	Type	Null	Key	Default	Extra
nome cliente	varchar(80)	NO		NULL	
endereco cliente	varchar(80)	NO		NULL	

IMPORTANTE: Você pode usar qualquer cláusula da instrução `select` quando estiver consultando por meio de uma `view`, incluindo `group by`, `having` e `order by`, que você aprendeu em agendas anteriores.

Existem várias razões para se utilizar **view**, tais como:

- **Segurança de dados:** restrição de colunas e linhas presentes em uma tabela.
- **Agregação de dados:** aplicações em relatórios, que geralmente requerem dados agregados.
- **Escondendo a complexidade:** restringir aos usuários finais a visualização de regras complexas e instruções complexas para obtenção de dados apresentados no resultado de uma consulta.

Views atualizáveis

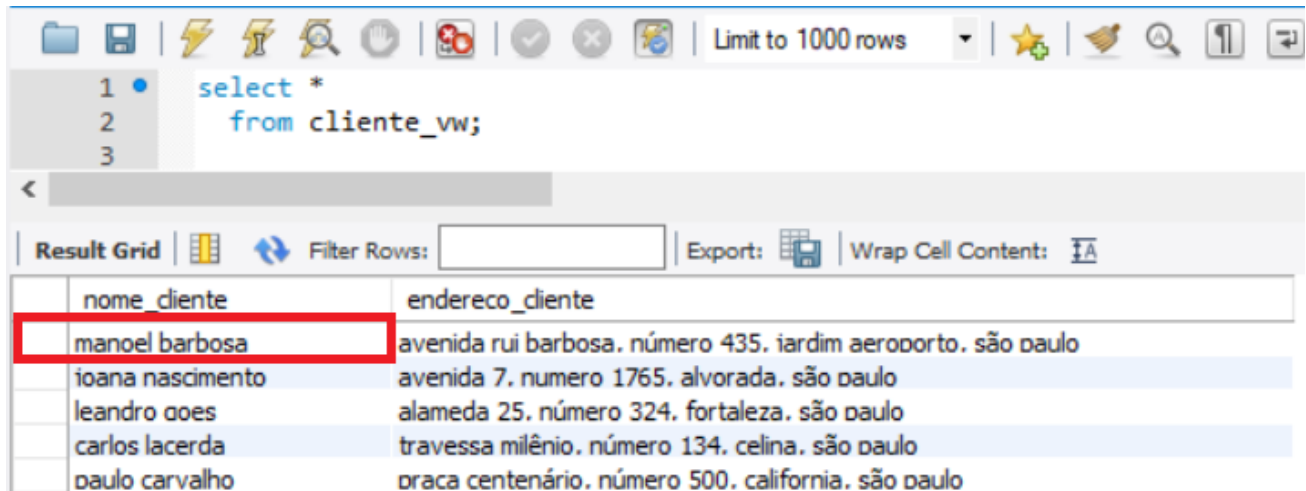
Alguns SGBD permitem modificar dados por meio de uma **view**, e o MySQL é um deles, desde que obedeça a algumas regras:

- Não são usadas funções de agregação (`max()`, `min()`, `avg()` etc).
- A **view** emprega cláusulas `group by` ou `having`.
- Não existem subconsultas na cláusula `select` ou `from`, e nenhuma subconsulta na cláusula `where` não se refere a tabelas na cláusula `from`.
- A **view** não utiliza `union`, `union all` ou `distinct`.
- A cláusula `from` inclui pelo menos uma tabela ou **view** atualizável.
- A cláusula `from` usa junções internas se houver mais de uma tabela ou **view**.

Retirado e adaptado do Livro Aprendendo SQL. Dominando os Fundamentos do SQL. Beaulieu, Alan. Tradução Edgard Batista Damiani. São Paulo. Novatec Editora, 2010.

Para demonstrar a atualização de **views**, vamos atualizar a **view** `cliente_vw`. Alterando o nome do primeiro cliente de “manuel barbosa” para “manoel barbosa”, para isso utilize a seguinte codificação:

```
update cliente_vw
  set nome_cliente = 'manoel barbosa'
 where nome_cliente = 'manuel barbosa';
```



The screenshot shows a database client window with a toolbar at the top. The SQL editor contains the query: `select * from cliente_vw;`. Below the editor, the 'Result Grid' tab is active, displaying a table with two columns: `nome_cliente` and `endereco_cliente`. The first row, containing 'manoel barbosa' and 'avenida rui barbosa, número 435, iardim aeroporto, são paulo', is highlighted with a red rectangle. The other rows are: 'ioana nascimento' (avenida 7, numero 1765, alvorada, são paulo), 'leandro ooes' (alameda 25, número 324, fortaleza, são paulo), 'carlos lacerda' (travessa milênio, número 134, celina, são paulo), and 'paulo carvalho' (praca centenário, número 500, califórnia, são paulo).

nome_cliente	endereco_cliente
manoel barbosa	avenida rui barbosa, número 435, iardim aeroporto, são paulo
ioana nascimento	avenida 7, numero 1765, alvorada, são paulo
leandro ooes	alameda 25, número 324, fortaleza, são paulo
carlos lacerda	travessa milênio, número 134, celina, são paulo
paulo carvalho	praca centenário, número 500, califórnia, são paulo

Alterar a View

Para alterar um **view** utilize a seguinte sintaxe:

```
alter view nome_da_view as
  select * from nome_outra_tabela;
```

Exemplo:

```
alter view cliente_vw as
  select nome_cliente, endereco_cliente
         from cliente
         order by nome_cliente;
```

Excluindo a View

Para excluir um **view** utilize a seguinte sintaxe:

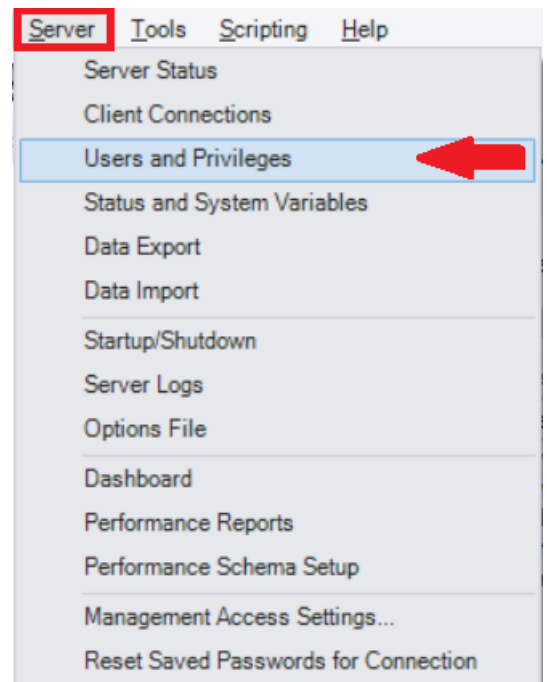
```
drop view nome_da_view;
```

Exemplo:

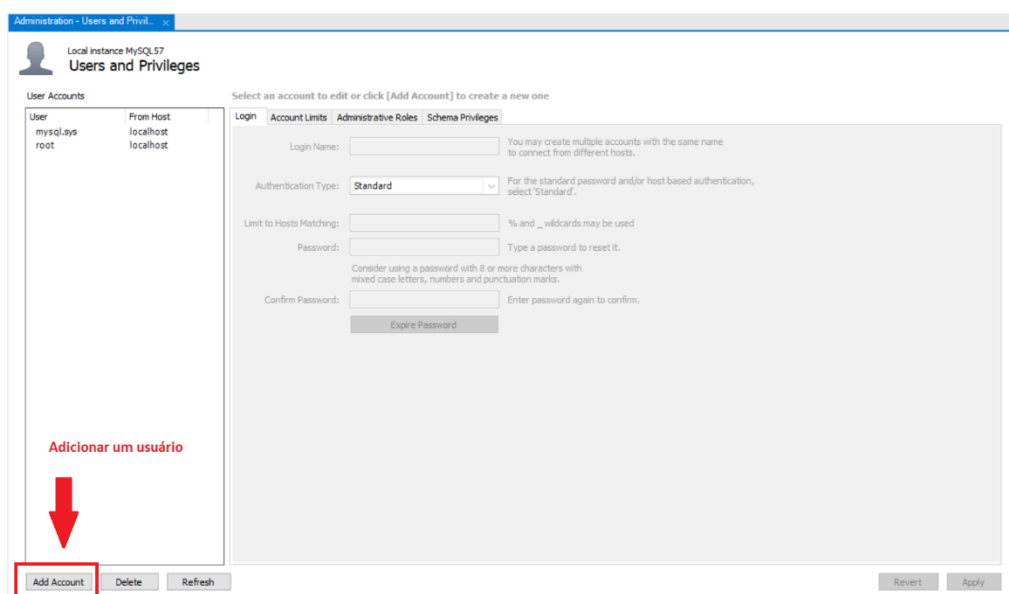
```
drop view cliente_vw;
```

Concluimos então que com uma **view** além de definirmos o que será disponibilizado para o usuário final, dependendo das regras utilizadas para sua criação, ela poderá ser atualizável. E falando em usuário final, depois de definirmos os dados que serão disponibilizados, nosso próximo passo é dar acesso somente aos usuários que preenchem as regras estabelecidas pelo negócio para visualização desses dados.

A Ferramenta WorkBench oferece o recurso para a criação de usuários pela interface gráfica, por meio do Menu **Server**, opção **Users e Privileges**, conforme apresentado na imagem. Se preferir, [clique aqui](#) para assistir um vídeo demonstrando o que será apresentado nas imagens a seguir.



Após a seleção dessa opção o WorkBench irá apresentar uma interface para administração de usuários e seus privilégios.



Nessa interface além de criar o usuário, você poderá gerenciar os privilégios dele que estão divididos em 3 (três) categorias:

- **Account Limits:** nessa guia você definirá alguns limites para conta que você está criando. Como por exemplo a quantidade máxima de queries (consultas) que o usuário poderá executar em uma hora entre outras.

Setting	Value	Description
Max. Queries:	0	Number of queries the account can execute within one hour.
Max. Updates:	0	Number of updates the account can execute within one hour.
Max. Connections:	0	The number of times the account can connect to the server per hour.
Concurrent Connections:	0	The number of simultaneous connections to the server the account can have.

- **Administrative Roles:** nessa guia você poderá atribuir alguns privilégios administrativos para o usuário, como por exemplo de **DBA** (*Database Administrator, Administrador do Banco de Dados*), onde ele terá todos os privilégios para executar todas as ações possíveis em um **banco de dados**.

Role	Description
<input type="checkbox"/> DBA	grants the rights to perform all tasks
<input type="checkbox"/> MaintenanceAdmin	grants rights needed to maintain server
<input type="checkbox"/> ProcessAdmin	rights needed to assess, monitor, and kill any user process
<input type="checkbox"/> UserAdmin	grants rights to create users logins and reset passwords
<input type="checkbox"/> SecurityAdmin	rights to manage logins and grant and revoke server privileges
<input type="checkbox"/> MonitorAdmin	minimum set of rights needed to monitor server
<input type="checkbox"/> DBManager	grants full rights on all databases
<input type="checkbox"/> DBDesigner	rights to create and reverse engineer any database schema
<input type="checkbox"/> ReplicationAdmin	rights needed to setup and manage replication
<input type="checkbox"/> BackupAdmin	minimal rights needed to backup any database

Global Privileges
<input type="checkbox"/> ALTER
<input type="checkbox"/> ALTER ROUTINE
<input type="checkbox"/> CREATE
<input type="checkbox"/> CREATE ROUTINE
<input type="checkbox"/> CREATE TABLESPACE
<input type="checkbox"/> CREATE TEMPORARY TABLES
<input type="checkbox"/> CREATE USER
<input type="checkbox"/> CREATE VIEW
<input type="checkbox"/> DELETE
<input type="checkbox"/> DROP
<input type="checkbox"/> EVENT
<input type="checkbox"/> EXECUTE
<input type="checkbox"/> FILE
<input type="checkbox"/> GRANT OPTION
<input type="checkbox"/> INDEX
<input type="checkbox"/> INSERT
<input type="checkbox"/> LOCK TABLES
<input type="checkbox"/> PROCESS
<input type="checkbox"/> REFERENCES
<input type="checkbox"/> RELOAD
<input type="checkbox"/> REPLICATION CLIENT
<input type="checkbox"/> REPLICATION SLAVE
<input type="checkbox"/> SELECT
<input type="checkbox"/> SHOW DATABASES
<input type="checkbox"/> SHOW VIEW
<input type="checkbox"/> SHUTDOWN
<input type="checkbox"/> SUPER
<input type="checkbox"/> TRIGGER

Revoke All Privileges

- **Schema Privileges:** nessa guia você poderá atribuir ao usuário um ou vários privilégios a determinados **Schemas** disponíveis no SGBD.

Login Account Limits Administrative Roles Schema Privileges

Schema Privileges

Schema and Host fields may use % and _ wildcards.
The server will match specific entries before wildcarded ones.

Revoke All Privileges Delete Entry Add Entry...

Object Rights

☐ SELECT
☐ INSERT
☐ UPDATE
☐ DELETE
☐ EXECUTE
☐ SHOW VIEW

DDL Rights

☐ CREATE
☐ ALTER
☐ REFERENCES
☐ INDEX
☐ CREATE VIEW
☐ CREATE ROUTINE
☐ ALTER ROUTINE
☐ EVENT
☐ DROP
☐ TRIGGER

Other Rights

☐ GRANT OPTION
☐ CREATE TEMPORARY TABLES
☐ LOCK TABLES

Unselect All Select "ALL"

Segue abaixo uma lista com o significado de alguns privilégios:

Privilégio	Descrição
ALL [PRIVILEGES]	Todos os privilégios exceto GRANT OPTION
ALTER	Permite executar ALTER TABLE
CREATE	Permite executar CREATE TABLE
CREATE TEMPORARY TABLES	Permite executar CREATE TEMPORARY TABLE
DELETE	Permite executar DELETE
DROP	Permite executar DROP TABLE
EXECUTE	Permite executar stored procedures (MySQL 5.0)
FILE	Permite executar SELECT ... INTO OUTFILE e LOAD DATA INFILE
INDEX	Permite executar CREATE INDEX e DROP INDEX

INSERT	Permite executar INSERT
LOCK TABLES	Permite executar LOCK TABLES em tabelas que você tenha o privilégio SELECT
PROCESS	Permite executar SHOW FULL PROCESSLIST
REFERENCES	Ainda não está implementado
RELOAD	Permite executar FLUSH
REPLICATION CLIENT	Permite ao usuário obter a localização do Master ou Slave
REPLICATION SLAVE	Necessário para a replicação Slave (leitura dos eventos do log binário do Master)
SELECT	Permite executar SELECT
SHOW DATABASES	exibe todos os bancos de dados
SHUTDOWN	Permite executar mysqladmin shutdown
SUPER	Permite executar CHANGE MASTER , KILL , PURGE MASTER LOGS e SET GLOBAL . Permite conectar-se ao servidor uma vez, mesmo que o max_connections tenha sido atingido
UPDATE	Permite executar UPDATE
USAGE	Sinônimo para "no privileges"
GRANT OPTION	Permite ao usuário repassar os seus privilégios

Retirado de <https://www.devmedia.com.br/gerenciamento-de-usuarios-e-controle-de-acessos-do-mysql/1898>.

Acessado em 05/12/2019.

Por se tratar de um privilégio muito específico, para exemplificar, vamos utilizar a criação do usuário por meio de codificação, o que na maioria das vezes é o melhor a se fazer.

IMPORTANTE: [clique aqui](#) caso prefira assistir um vídeo que demonstra os próximos passos para criação de concessão de privilégios.

Sintaxe para criação de usuário:

```
create user 'novousuario'@'localhost' identified by 'password';
```

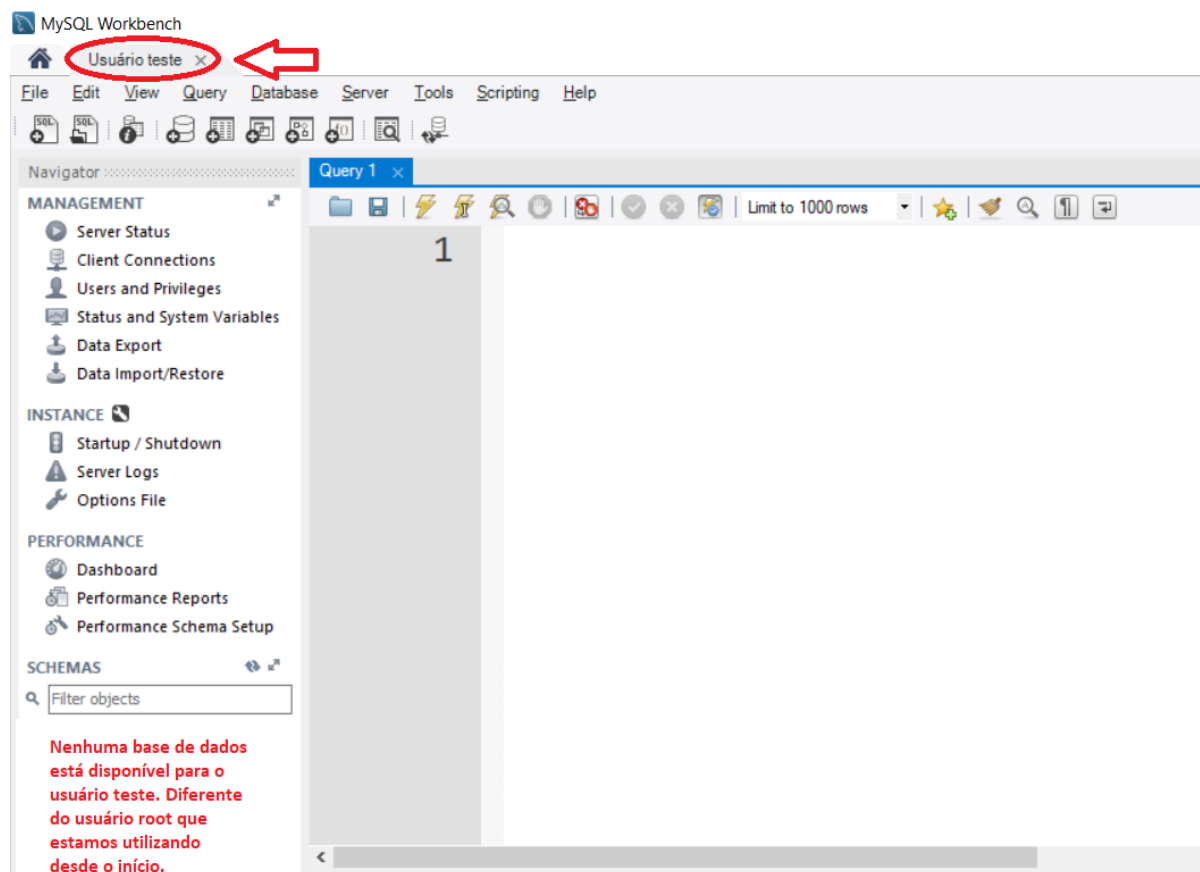
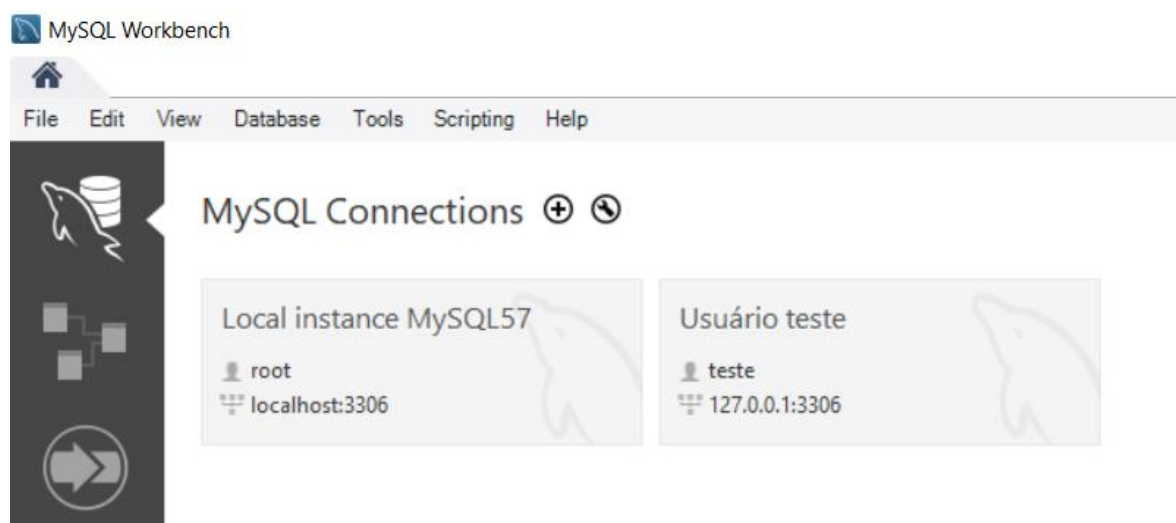
Sintaxe para excluir um usuário:

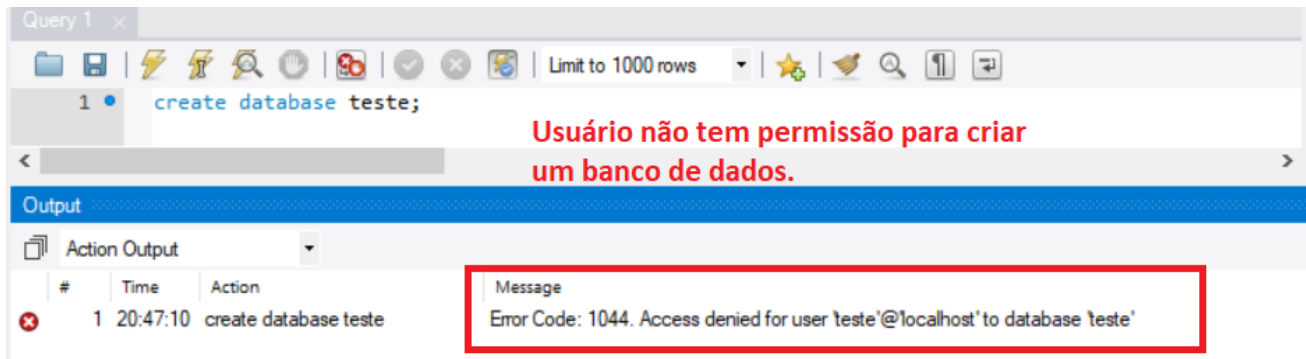
```
drop user 'novousuario'@'localhost';
```


Vamos criar um usuário com o nome de *teste*, definindo como senha os números *12345*:

```
create user teste@localhost identified by '12345';
```

Após a criação do usuário *teste*, ele ainda não terá permissão para fazer alguma ação no SGBD. Ao criar uma conexão na Ferramenta Workbench, no quadro **Schemas** nenhum banco de dados estará disponível, além de não poder criar nenhum, conforme demonstrado nas imagens a seguir:





Para que um usuário possa executar alguma ação no SGBD, ele precisar ter permissão ou privilégios. Sintaxe para dar privilégios:

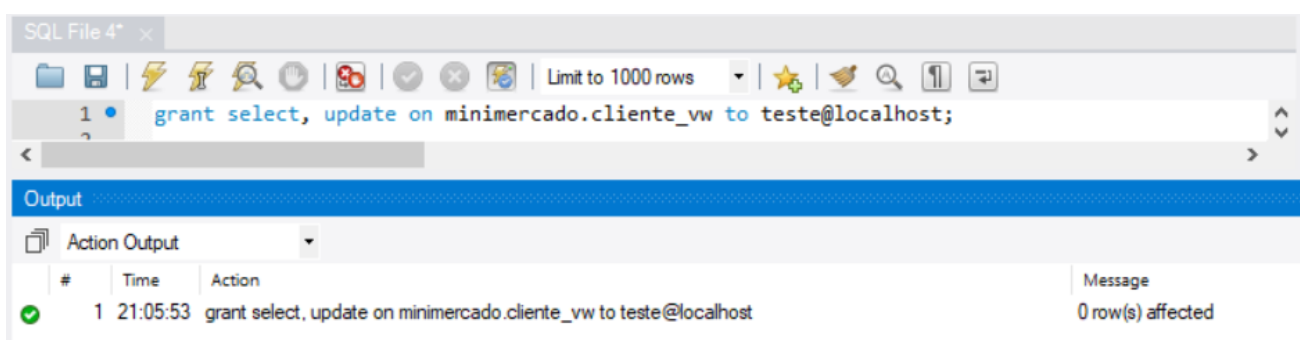
```
grant [tipo_de_permissão]
on [nome_base_dados].[nome_tabela]
to '[nome_usuario]'@'localhost';
```

Sintaxe para revogar privilégios:

```
revoke [tipo_de_permissão]
on [nome_base_dados].[nome_tabela]
to '[nome_usuario]'@'localhost';
```

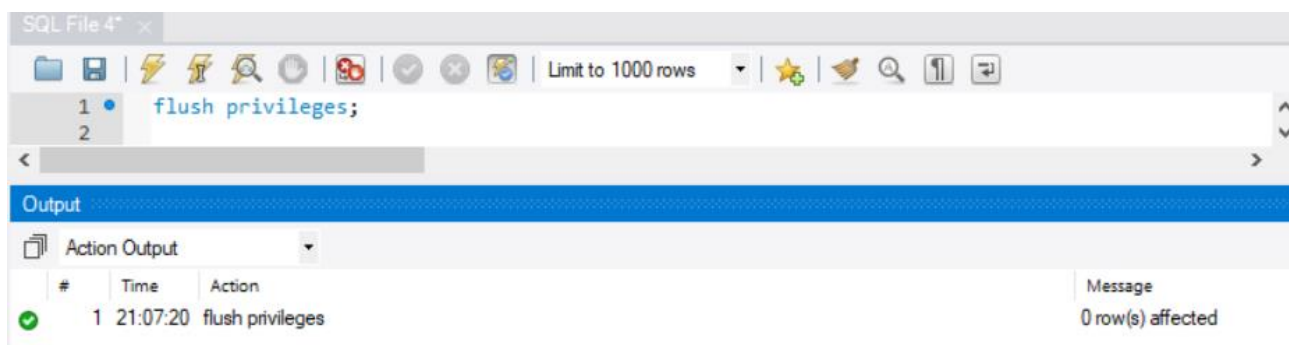
Agora vamos atribuir os privilégios de `select` e `update` na **view** `cliente_vw`:

```
grant select, update on minimercado.cliente_vw to teste@localhost;
```

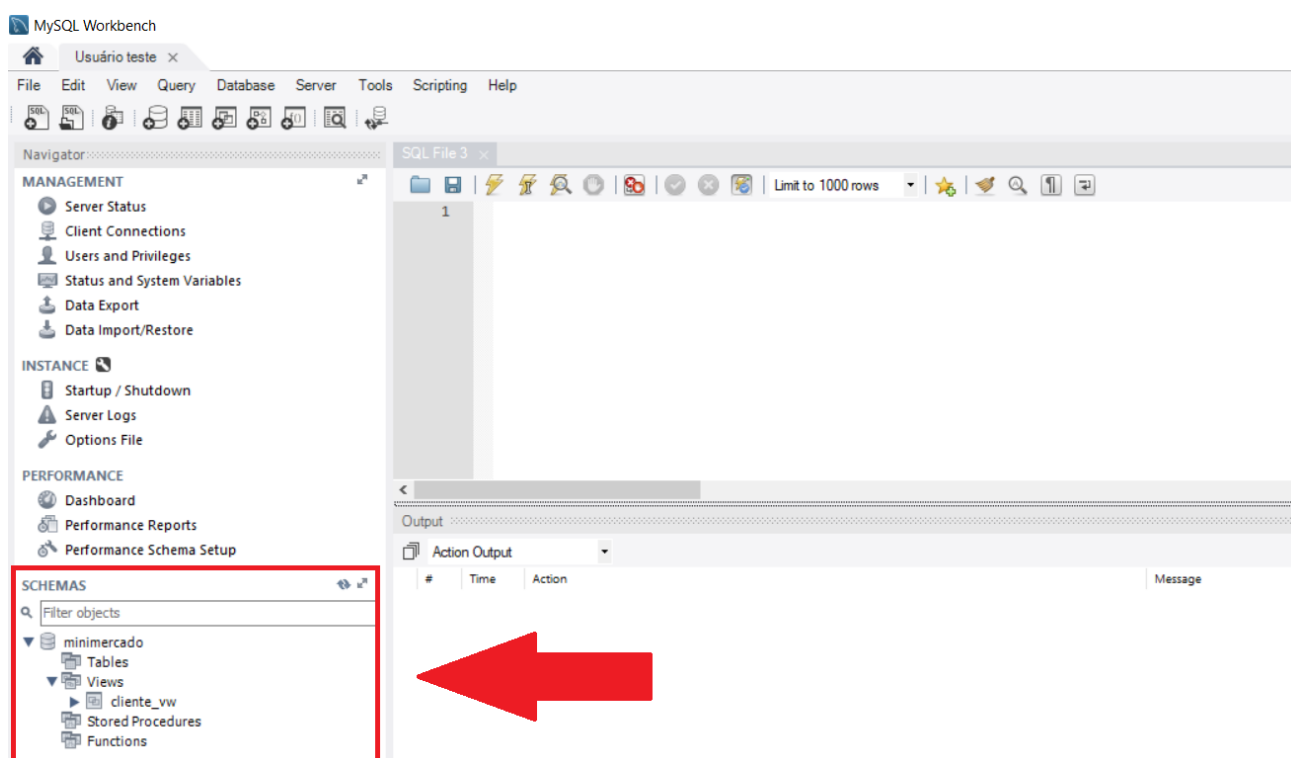


IMPORTANTE: Após aplicar algum tipo de privilégio execute o comando `flush privileges` para que o MySQL possa atualizar os privilégios que estão em memória.

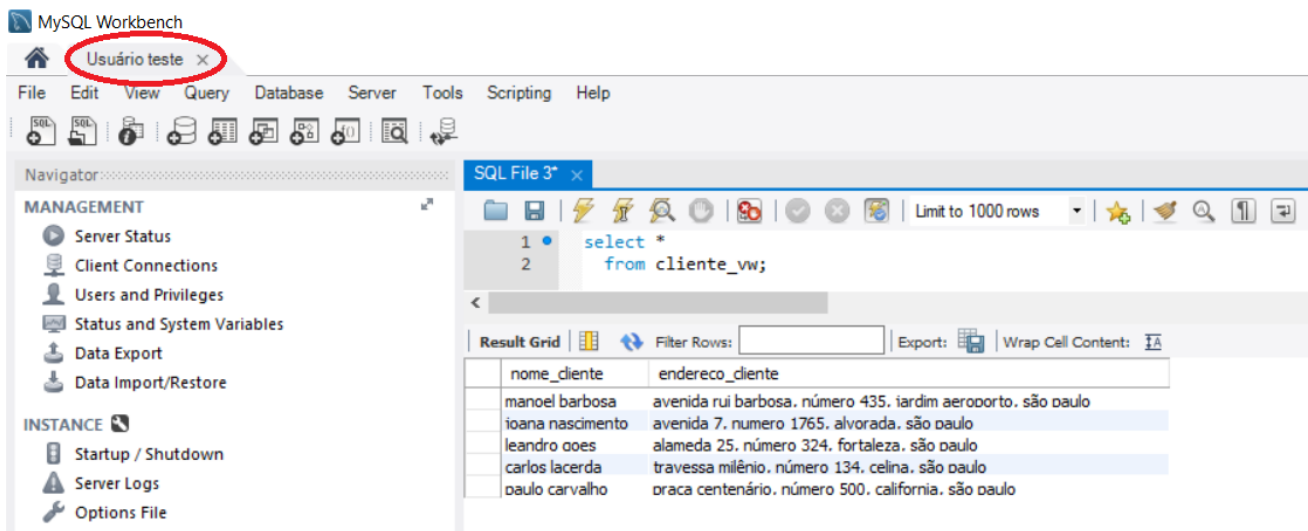
`flush privileges;`



Após conceder os privilégios conecte com o usuário **teste** novamente.



Após conceder os privilégios o usuário **teste** passou a ter acesso ao banco de dados **minimercado** e a **view cliente_vw**.



Agora é com você!



IMPORTANTE: Utilize o banco de dados do consultório para desenvolver os exemplos.

Para finalizar o projeto de integração dos Sistemas no Consultório da Dra. Ana Lúcia, Carlos, analista responsável, precisa disponibilizar o acesso dos aparelhos às estruturas `worklist_aparelhos` e `integracao_aparelhos`, nelas estão as informações necessárias para que o processo de integração seja executado por completo, trafegando as informações entre os Sistemas.

Como Carlos fará isso? Será que você consegue auxiliá-lo? Vamos ver!

Como foi? Difícil? Tenho certeza que não!

Lembrando mais uma vez, que o mais importante nesses casos é o desenvolvedor entender como o processo deverá ser executado, ou seja, conhecer o seu fluxo, suas etapas e particularidades.

Ao analisarmos as estruturas envolvidas podemos chegar as seguintes conclusões:

Nome da Tabela: **WORKLIST_APARELHOS**

CAMPO	TIPO (Tamanho)	Obrigatório	Padrão	PK	Observação
worklist_id	int(6)	sim		sim	auto incremento
agendamento_id	int(6)	sim		não	
dt_exame	date	sim		não	
paciente	varchar(80)	sim		não	
dt_nascimento	date	sim		não	
convenio	varchar(80)	sim		não	
procedimento	varchar(100)	sim		não	
aparelho	varchar(2)	sim		não	RX ou US
integrado	varchar(1)	sim	N	não	

worklist_aparelhos: contém os registros dos agendamentos de exames de imagem recepcionados que deverão ser integrados aos aparelhos.

Neste caso, o Sistema dos aparelhos deverá ter privilégios de **select** e **update**, o primeiro para obter os dados do agendamento e carregá-los, e o segundo para poder alterar o conteúdo do campo **integrado** de 'N' para 'S', não considerando o registro nas próximas execuções da integração.

Nome da Tabela: **INTEGRACAO_APARELHOS**

CAMPO	TIPO (Tamanho)	Obrigatório	Padrão	PK	Observação
integracao_id	int(6)	sim		sim	auto incremento
worklist_id	int(6)	sim		não	
dt_realizacao	date	não		não	

integracao_aparelhos: utilizada para armazenar os registros dos exames que já foram integrados, e posteriormente, sua realização.

Neste outro caso, o Sistema dos aparelhos deverá ter privilégios de **insert** e **update**, o primeiro para incluir um registro, confirmando assim que as informações foram carregadas aos aparelhos, e o segundo para confirmar a realização do exame, preenchendo o conteúdo do campo **dt_realizacao**.

Antes de concedermos os privilégios identificados e necessários para a execução do processo, precisamos criar um usuário, que será utilizado pelo Sistema dos aparelhos para conectar na base

de dados do Sistema do Consultório da Dra. Ana Lúcia. Para isso, vamos utilizar um nome bem sugestivo, **aparelho**.

```
create user aparelho@localhost identified by '12345';
```

Agora sim, vamos dar os privilégios as estruturas `worklist_aparelhos` e `integracao_aparelhos`:

```
grant select, update on consultorio.worklist_aparelhos to
aparelho@localhost;

grant insert, update on consultorio.integracao_aparelhos to
aparelho@localhost;
```

Você pode estar pensando, mas e a **view**, como posso utilizá-la?

No projeto de integração dos Sistemas do consultório da Dra. Ana Lúcia, ela poderia ser utilizada para restringir o acesso as colunas `agendamento_id` e `convenio` da Tabela `worklist_aparelhos`, esses campos não precisam ser integrados, dentro do processo de aquisição das imagens eles não tem nenhuma utilidade.

Neste caso, você poderia criar uma **view** sem esses campos e dar os mesmos privilégios de `select` e `update` ao usuário **aparelho**.

```
create or replace view worklist_ap as
select worklist_id, dt_exame, paciente, dt_nascimento,
       procedimento, aparelho, integrado
from worklist_aparelhos;

grant select, update on consultorio.worklist_ap to aparelho@localhost;
```

Tudo pronto, integração finalizada, projeto concluído, parabéns!!!!