Criando Bancos e Tabelas no PostgreSQL

16.5

Aula 8

Introdução a linguagem SQL

Gerenciando Usuários e Permissões no PostgreSQL

Privilégios

Quando um objeto é criado, é atribuído um dono ao mesmo.

Normalmente, o dono do objeto é a função de banco de dados (*role*) que executou o comando de criação.

Para a maioria dos tipos de objeto, o estado inicial é de apenas o dono (ou um superusuário) poder fazer alguma coisa com o objeto.

Para permitir que outros o usem, devem ser concedidos *privilégios*.

Administrando Usuários e Privilégios

Uma política de backup e recuperação de dados adequadamente elaborada e executada protegerá a organização contra a perda de informação devido a falhas de hardware, defeitos de software, erros humanos, intrusos, sabotagem e desastres naturais.

Entretanto, esta não é a única maneira existente de proteção das informações.

Afinal, precisamos mais do que uma solução relativa ao acontecimento de falhas..

Administrando Usuários e Privilégios

Para mantermos a integridade das informações e realizarmos auditoria das mesmas, podemos utilizar os recursos dos gatilhos.

Além disso, podemos fazer uso de uma linguagem especial para controle de acesso e permissão aos dados.

Esta linguagem é a DCL (Data Control Language– Linguagem de Controle de Dados).

Ela é um subconjunto da SQL para o controle de permissões dos usuários aos objetos do banco de dados.

Administrando Usuários e Privilégios

Todo agrupamento de bancos de dados possui um conjunto de usuários de banco de dados.

Estes usuários são distintos dos usuários gerenciados pelo sistema operacional onde o servidor executa.

Eles possuem objetos de banco de dados (por exemplo, tabelas, visões etc.), e podem conceder privilégios nestes objetos para outros usuários controlando, assim, quem pode acessar qual objeto.

Administrando Usuários e Privilégios

A DCL controla os aspectos de autorização de dados e licenças de usuários para controlar quem tem acesso para ver ou manipular dados dentro do banco de dados.

Os nomes dos usuários de banco de dados são globais para todo o agrupamento de bancos de dados (e não apenas próprio de cada banco de dados).

Criação de Usuário de Banco de Dados

CREATE USER

Comando de Criação de um usuário

```
CREATE USER nome [ [ WITH ] opção [ ... ] ]
```

Opção	Significado
Nome	O nome do usuário
Id_do_usuário	A cláusula SYSID pode ser utilizada para escolher o identificador de usuário do PostgreSQL do novo usuário.
CREATEDB | NOCREATEDB	Estas cláusulas definem a permissão para o usuário criar banco de dados. Se for especificado CREATEDB, o usuário sendo definido terá permissão para criar seus próprios bancos de dados. Se for especificado NOCREATEDB, nega-se ao usuário a permissão para criar banco de dados. Se nenhuma destas cláusulas for especificada, o padrão é NOCREATEDB.
CREATEUSER | NOCREATEUSER	Estas cláusulas determinam se o usuário pode ou não criar novos usuários. CREATEUSER também torna o usuário um super-usuário, o qual pode passar por cima de todas as restrições de acesso. Se nenhuma destas cláusulas for especificada, o padrão é NOCREATEUSER.
Nome_do_grupo	O nome de um grupo existente onde o usuário será incluído como um novo membro. Podem ser especificados nomes de vários grupos.
Senha	Define a senha do usuário. Se não se pretende utilizar autenticação por senha, esta opção pode ser omitida, mas o usuário não poderá mais se conectar se for decidido mudar para autenticação por senha. A senha poderá ser definida ou mudada posteriormente através do comando ALTER USER.
ENCRYPTED | UNENCRYPTED	Estas cláusulas controlam se a senha será armazenada criptografada, ou não, nos catálogos do sistema.
Data_e_hora	A cláusula VALID UNTIL define uma data e hora após a qual a senha do usuário não é mais válida. Se esta cláusula for omitida, a conta será válida para sempre.

Manipulando dados de um usuário

Removendo um Usuário no Banco de Dados	• DROP USER nome_usuário
Alterando um Usuário	• ALTER USER nome [[WITH] opção* [...]]
Comando para alterar o nome de um usuário	• ALTER USER nome RENAME TO novo_nome

* As opções desse comando podem ser vistas no Guia de consulta do PostGreSQL

Conceitos de Grupos de Usuários

Grupos de Usuários

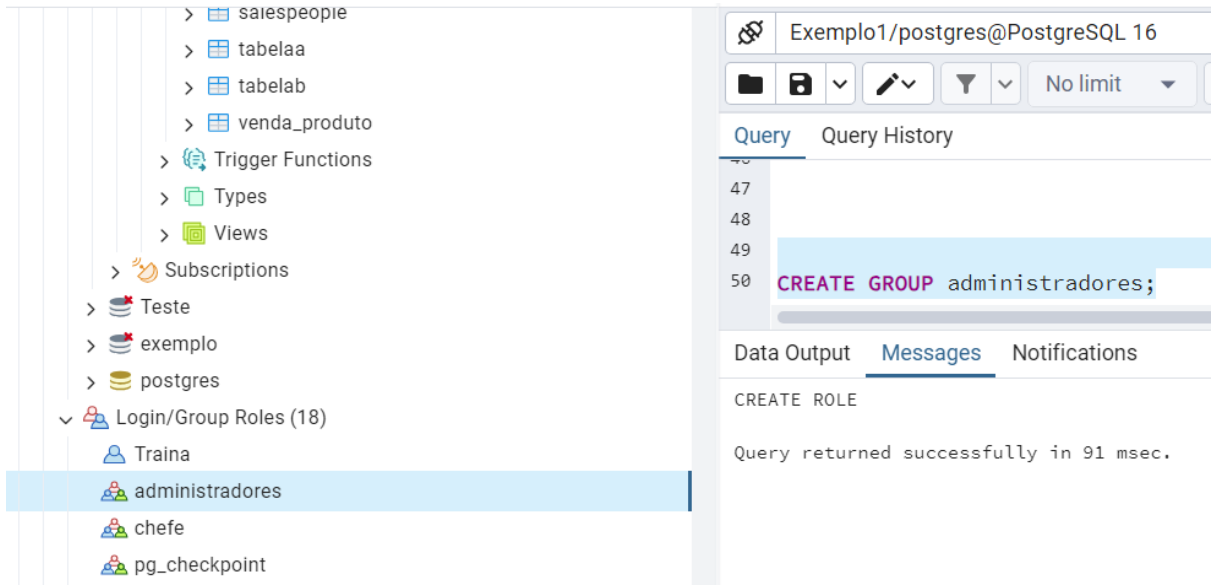
Assim como nos sistemas operacionais baseados em Unix, os grupos são uma forma lógica de juntar usuários para facilitar o gerenciamento de privilégios.

Tais privilégios podem ser concedidos, ou revogados, para o grupo como um todo.

Para criar um grupo, deve ser utilizado o comando CREATE GROUP

```
CREATE GROUP nome_do_grupo;
```

```
CREATE GROUP administradores;
```




```
ALTER GROUP nome_do_grupo ADD USER nome_do_usuário;
ALTER GROUP nome_do_grupo DROP USER nome_do_usuário;
DROP GROUP nome_do_grupo;
ALTER GROUP nome_do_grupo RENAME TO novo_nome;
```

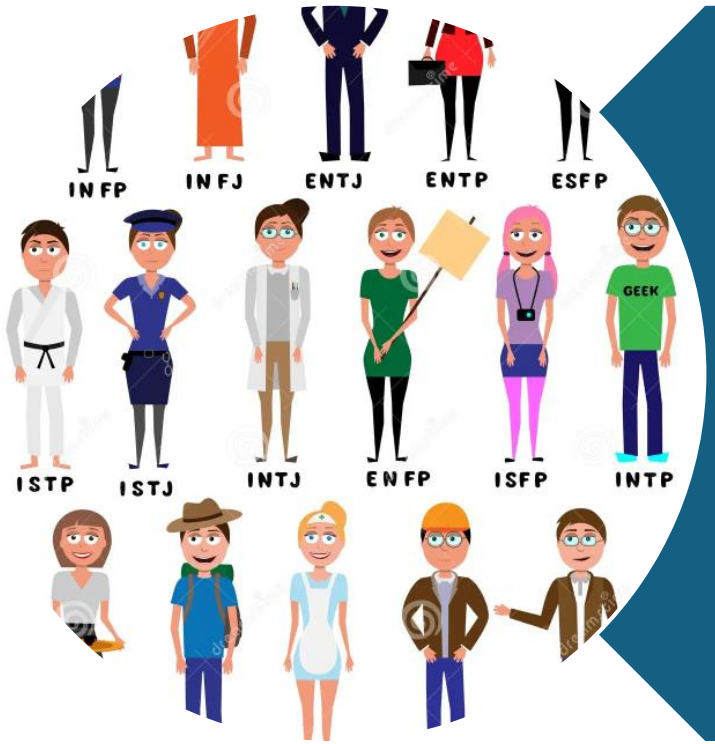
Criando Papéis (roles)
CREATE ROLE

Participação em função de banco de dados

Frequentemente, é conveniente agrupar usuários para facilitar o gerenciamento de privilégios: dessa forma, os privilégios podem ser concedidos ou revogados para um grupo como um todo.

No PostgreSQL isso é feito criando uma função de banco de dados (*role*) que representa o grupo e, em seguida, concedendo a *participação* de usuários individuais nessa função de banco de dados do grupo, tornando-os membros dessa função de banco de dados.

Role



Para definir uma função de banco de dados de grupo, primeiro é criada a função de banco de dados:

CREATE ROLE nome_da_role_de_grupo;

CREATE ROLE

O comando adiciona um novo papel (role) ao agrupamento de bancos de dados do PostgreSQL.

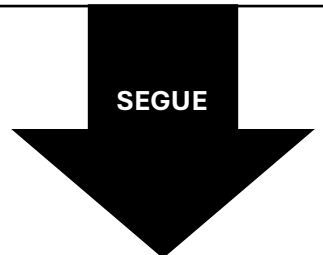
O papel é uma entidade que pode possuir objetos do banco de dados e possuir privilégios do banco de dados.

Ele pode ser considerado como sendo um "usuário", um "grupo", ou ambos, dependendo de como é utilizado.

O comando CREATE ROLE é o substituto dos comandos CREATE USER e CREATE GROUP por possuir mais recursos que os mesmos.

CREATE ROLE nome [[WITH] opção [...]];

Opção	Significado
Nome	**Nome do papel**
SUPERUSER | NOSUPERUSER	Estas cláusulas determinam se o novo papel é um "super-usuário", o qual pode passar por cima de todas as restrições de acesso dos bancos de dados. Se nenhuma destas duas cláusulas for especificada, o padrão é NOSUPERUSER.
CREATEDB | NOCREATEDB	Estas cláusulas definem a permissão para o papel criar bancos de dados. Se nenhuma destas duas cláusulas for especificada, o padrão é NOCREATEDB.
CREATEROLE | NOCREATEROLE	Estas cláusulas determinam se o papel terá permissão para criar novos papéis (ou seja, executar o comando CREATE ROLE).
CREATEUSER | NOCREATEUSER	Estas cláusulas são formas obsoletas, mas ainda aceitas, das cláusulas SUPERUSER e NOSUPERUSER.
INHERIT | NOINHERIT	Estas cláusulas determinam se o papel "herda" os privilégios dos papéis dos quais é membro. Um papel com o atributo INHERIT pode utilizar, automaticamente, todos os privilégios de banco de dados que foram concedidos a todos os papéis dos quais é um membro direto ou indireto.



LOGIN **NOLOGIN**	Estas cláusulas determinam se o papel pode estabelecer uma conexão (log in), ou seja, se o papel pode ser fornecido como nome de autorização inicial da sessão durante a conexão do cliente. Um papel com o atributo LOGIN pode ser considerado como sendo um usuário. Se nenhuma destas duas cláusulas for especificada, o padrão é NOLOGIN, exceto quando CREATE ROLE for chamada através de sua forma alternativa CREATE USER.
CONNECTION LIMIT **limite_de_conexões**	Se o papel puder estabelecer uma conexão, esta cláusula especifica quantas conexões simultâneas este papel pode estabelecer. -1 (o padrão) significa sem limite.
PASSWORD senha	Define a senha do papel (A senha só é útil para os papéis que possuem o atributo LOGIN, mesmo assim pode ser definida uma senha para os papéis sem este atributo).
ENCRYPTED **UNENCRYPTED**	Estas cláusulas controlam se a senha será armazenada criptografada, ou não, nos catálogos do sistema;
VALID UNTIL 'carimbo_do_tempo'	A cláusula VALID UNTIL define uma data e hora após a qual o papel não é mais válido. Se esta cláusula for omitida, a senha será válida para sempre.
IN ROLE nome_do_papel	A cláusula IN ROLE relaciona um ou mais papéis existentes, aos quais o novo papel será adicionado imediatamente como sendo um novo membro.
IN GROUP nome_do_papel	A cláusula IN GROUP é uma forma obsoleta de IN ROLE.
ROLE nome_do_papel	A cláusula ROLE relaciona um ou mais papéis existentes a serem automaticamente adicionados como membros do novo papel (Isto tem como efeito tornar o novo papel um "grupo").
ADMIN nome_do_papel	A cláusula ADMIN é como a cláusula ROLE, mas os papéis especificados são adicionados ao novo papel WITH ADMIN OPTION, dando aos mesmos o direito de permitir que outros papéis se tornem membros deste grupo.
USER nome_do_papel	A cláusula USER é uma forma obsoleta da cláusula ROLE.
SYSID id_usuário	A cláusula SYSID é ignorada, mas é aceita para manter a compatibilidade com as versões anteriores.

ROLE

Deve ser tomado cuidado com o privilégio CREATE ROLE.

Não existe o conceito de herança para os privilégios do papel com CREATE ROLE.

Isto significa que mesmo que o papel não possua um determinado privilégio, mas tenha permissão para criar outros papéis, poderá facilmente criar um papel com privilégios diferentes do próprio papel (exceto criar papéis com privilégio de super-usuário).

ROLE

Removendo um Role – DROP ROLE

- DROP ROLE [IF EXISTS] nome [, ...]

Alterando um Role – ALTER ROLE

- ALTER ROLE nome [[WITH] opção [...]]

Concedendo e Revogando Privilégios

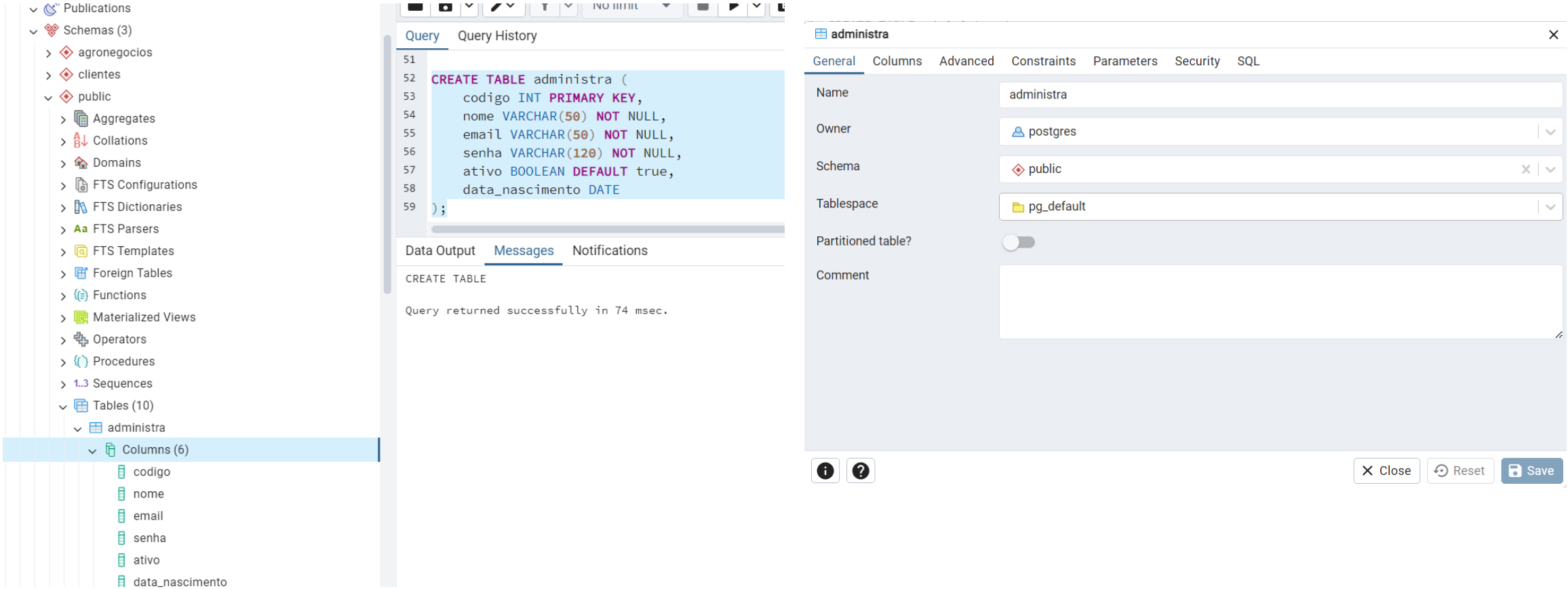
Dono

Quando um objeto do banco de dados é criado, é atribuído um dono ao mesmo.

O dono é o usuário que executou o comando de criação do objeto.

Para mudar o dono de uma tabela, índice, seqüência ou visão deve ser utilizado o comando ALTER TABLE.

ALTER TABLE nome_da_tabela OWNER TO novo_dono



ALTER TABLE nome_da_tabela OWNER TO novo_dono

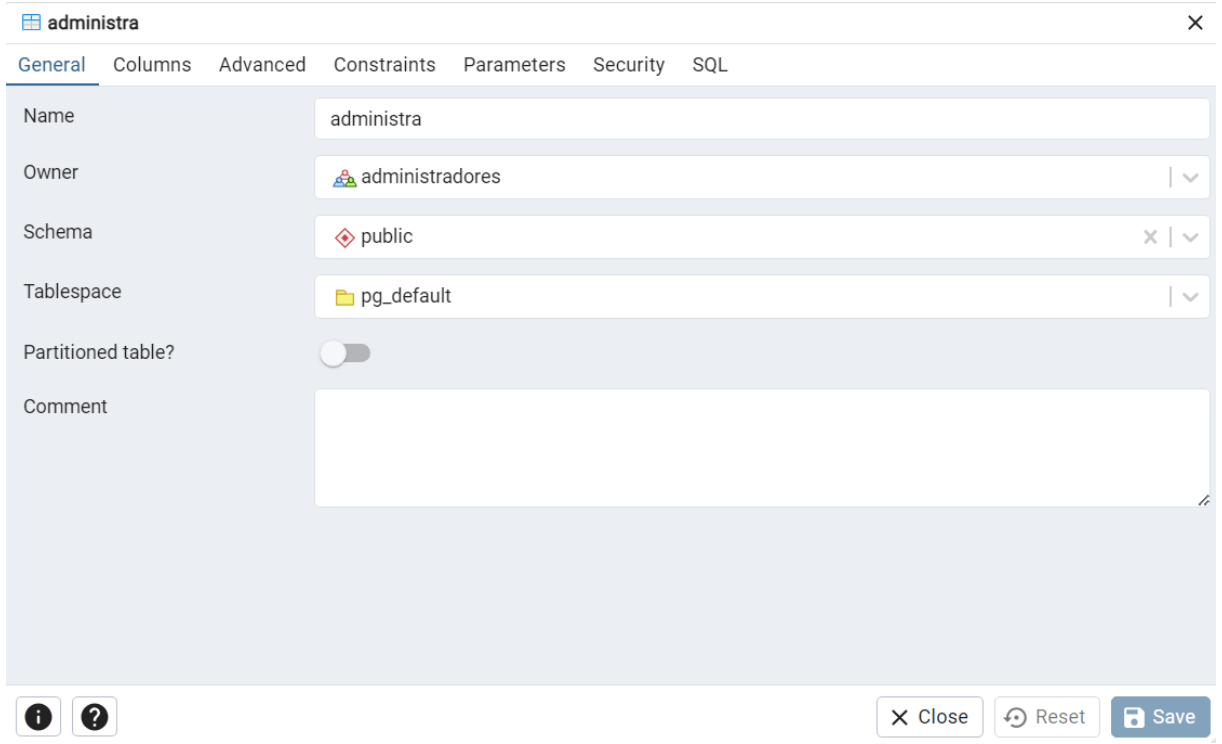
Query | Query History

```
54      nome VARCHAR(50) NOT NULL,
55      email VARCHAR(50) NOT NULL,
56      senha VARCHAR(120) NOT NULL,
57      ativo BOOLEAN DEFAULT true,
58      data_nascimento DATE
59  );
60
61  ALTER TABLE administra OWNER TO administradores;
62
```

Data Output | Messages | Notifications

```
ALTER TABLE

Query returned successfully in 84 msec.
```



Privilégios

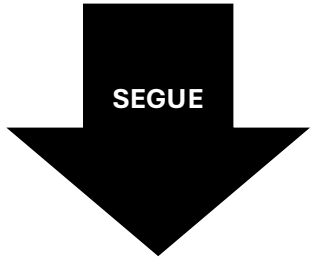
Existem diferentes tipos de privilégios:

SELECT, INSERT, UPDATE, DELETE, TRUNCATE, REFERENCES, TRIGGER, CREATE, CONNECT, TEMPORARY, EXECUTE e USAGE.

Os privilégios aplicáveis a um determinado objeto variam dependendo do tipo do objeto (tabela, função, etc.).

```
CREATE ROLE nome [ [WITH] opção [...]];
```

Privilégio	Significado
SELECT	Permite consultar qualquer coluna da tabela, visão ou seqüência especificada. Também permite utilizar o comando COPY TO.
INSERT	Permite inserir novas linhas na tabela especificada. Também permite utilizar o comando COPY FROM.
UPDATE	Permite modificar os dados de qualquer coluna da tabela especificada.
DELETE	Permite excluir linhas da tabela especificada.
RULE	Permite criar regras para uma tabela ou para uma visão.
REFERENCES	Para criar uma restrição de chave estrangeira é necessário possuir este privilégio, tanto na tabela que faz referência quanto na tabela que é referenciada.
TRIGGER	Permite criar gatilhos na tabela especificada



CREATE	Para bancos de dados, permite a criação de novos esquemas no banco de dados. Para esquemas, permite a criação de novos objetos no esquema. Para mudar o nome de um objeto existente é necessário ser o dono do objetoepossuir este privilégio no esquema que o contém. Para espaços de tabelas (TABLESPACE), permite a criação de tabelas e índices no espaço de tabelas, e permite a criação de bancos de dados possuindo este espaço de tabelas como seu espaço de tabelas padrão (Deve ser observado que revogar este privilégio não altera a colocação dos objetos existentes).
TEMPORARY **TEMP**	Permite a criação de tabelas temporárias ao usar o banco de dados.
EXECUTE	Permite utilizar a função especificada e qualquer operador implementado utilizando a função. Este é o único tipo de privilégio aplicável às funções (Esta sintaxe funciona para as funções de agregação também).
USAGE	Para as linguagens procedurais, permite o uso da linguagem especificada para criar funções nesta linguagem. Este é o único tipo de privilégio aplicável às linguagens procedurais. Para os esquemas, permite acessar os objetos contidos no esquema especificado (assumindo que os privilégios requeridos para os próprios objetos estejam atendidos). Essencialmente, concede a quem recebe o direito de "procurar" por objetos dentro do esquema.
ALL PRIVILEGES	Concede todos os privilégios disponíveis de uma só vez. A palavra chave PRIVILEGES é opcional no PostgreSQL, embora seja requerida pelo SQL estrito.

Após a função de banco de dados do grupo existir, será possível adicionar e remover membros usando os comandos GRANT e REVOKE:

```
GRANT nome_da_role_de_grupo TO role1, ... ;
REVOKE nome_da_role_de_grupo FROM role1, ... ;
```

GRANT

Por padrão, somente o dono (ou um super-usuário) pode fazer qualquer coisa com o objeto.

Para permitir o uso por outros usuários, devem ser concedidos privilégios aos mesmos.

Em SQL, existe o comando GRANT.

Ele possui duas funcionalidades básicas: conceder privilégios para um objeto do banco de dados (tabela, visão, seqüência, banco de dados, função, linguagem procedural, esquema e espaço de tabelas) e conceder o privilégio de ser membro de um papel.

Revoke

Em alguns casos também se torna necessário revogar alguns privilégios de acesso a usuários ou grupos de usuários.

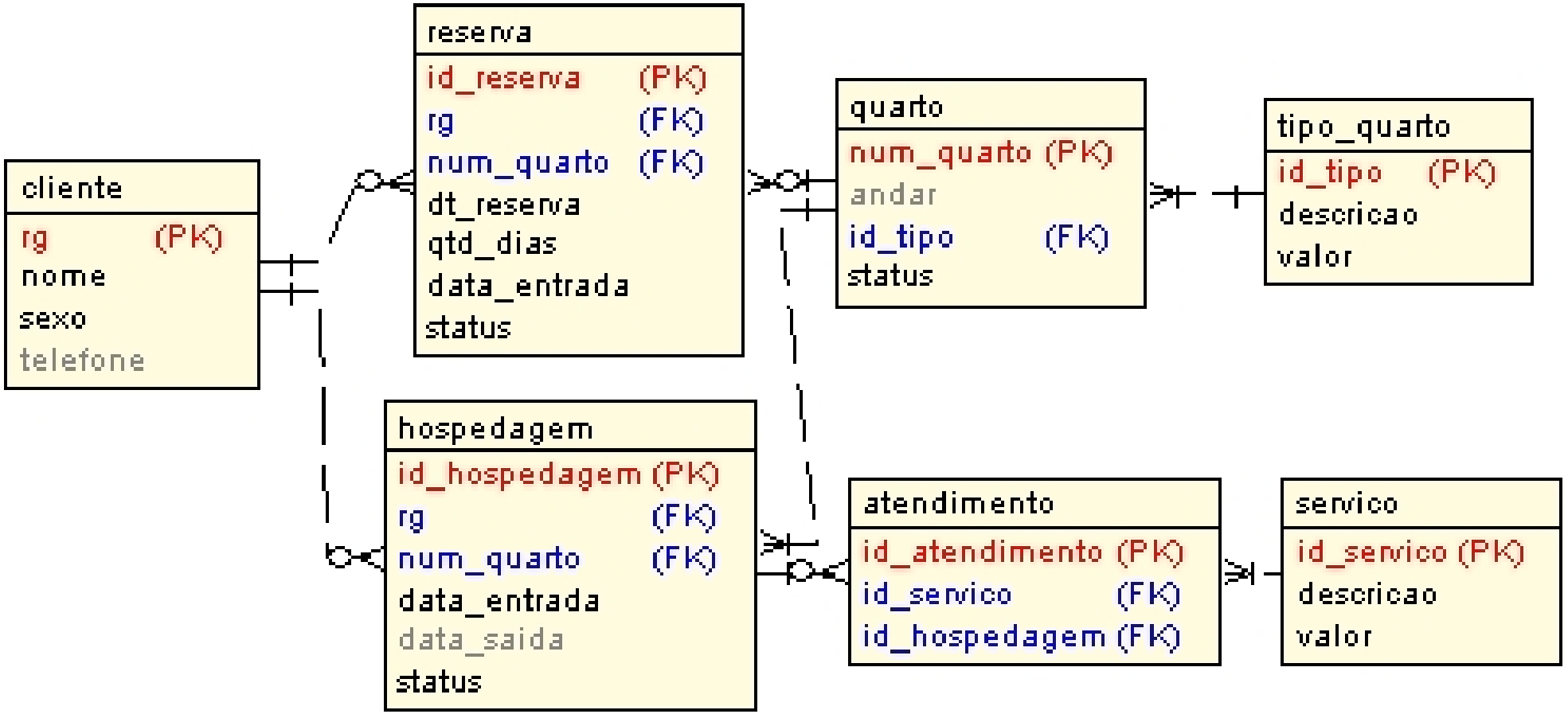
Para que isso seja feito, utiliza-se o comando REVOKE.

Deve ser observado que os super-usuários do banco de dados podem acessar todos os objetos, independentemente dos privilégios definidos para o objeto.

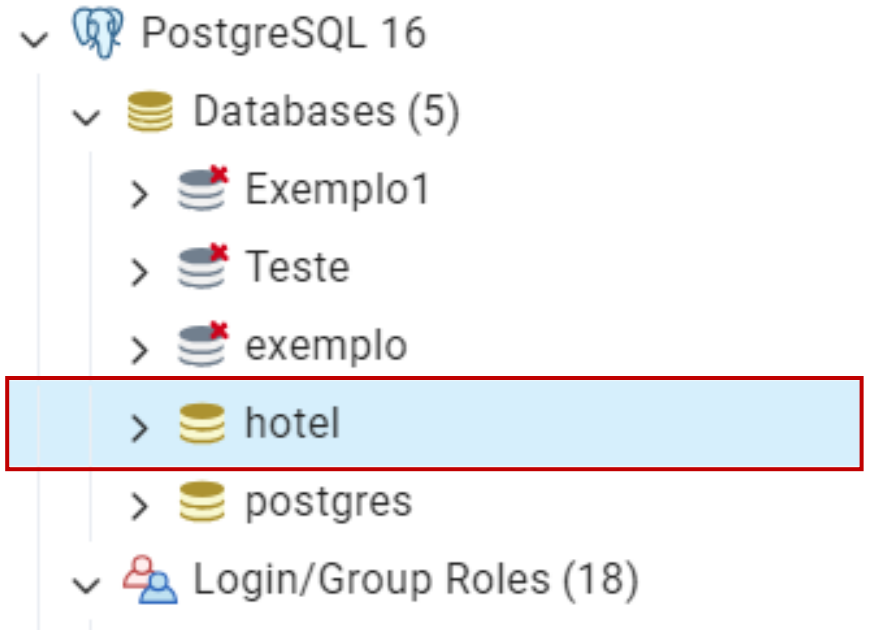
Assim, não é aconselhável operar como um super-usuário a não ser quando for absolutamente necessário.

Exemplo

[1,1]


reserva
id_reserva (PK)
rg (FK)
num_quarto (FK)
dt_reserva
qtd_dias
data_entrada
status
cliente
rg (PK)
nome
sexo
telefone
quarto
num_quarto (PK)
andar
id_tipo (FK)
status
tipo_quarto
id_tipo (PK)
descricao
valor
hospedagem
id_hospedagem (PK)
rg (FK)
num_quarto (FK)
data_entrada
data_saida
status
atendimento
id_atendimento (PK)
id_servico (FK)
id_hospedagem (FK)
servico
id_servico (PK)
descricao
valor

```
CREATE DATABASE hotel;
```



```
CREATE DATABASE hotel;
// Tabela de CLIENTES
 CREATE TABLE cliente
 (
 rg NUMERIC NOT NULL,
 nome VARCHAR(40) NOT NULL,
 sexo CHAR(1) NOT NULL,
 telefone NUMERIC(10,0),
 PRIMARY KEY (rg)
 ) WITHOUT OIDS;

 // Tabela TIPO_QUARTO
 CREATE TABLE tipo_quarto
 (
 id_tipo SERIAL NOT NULL,
 descricao VARCHAR(40) NOT NULL,
 valor NUMERIC(9,2) NOT NULL,
 PRIMARY KEY (id_tipo)
 ) WITHOUT OIDS;
```

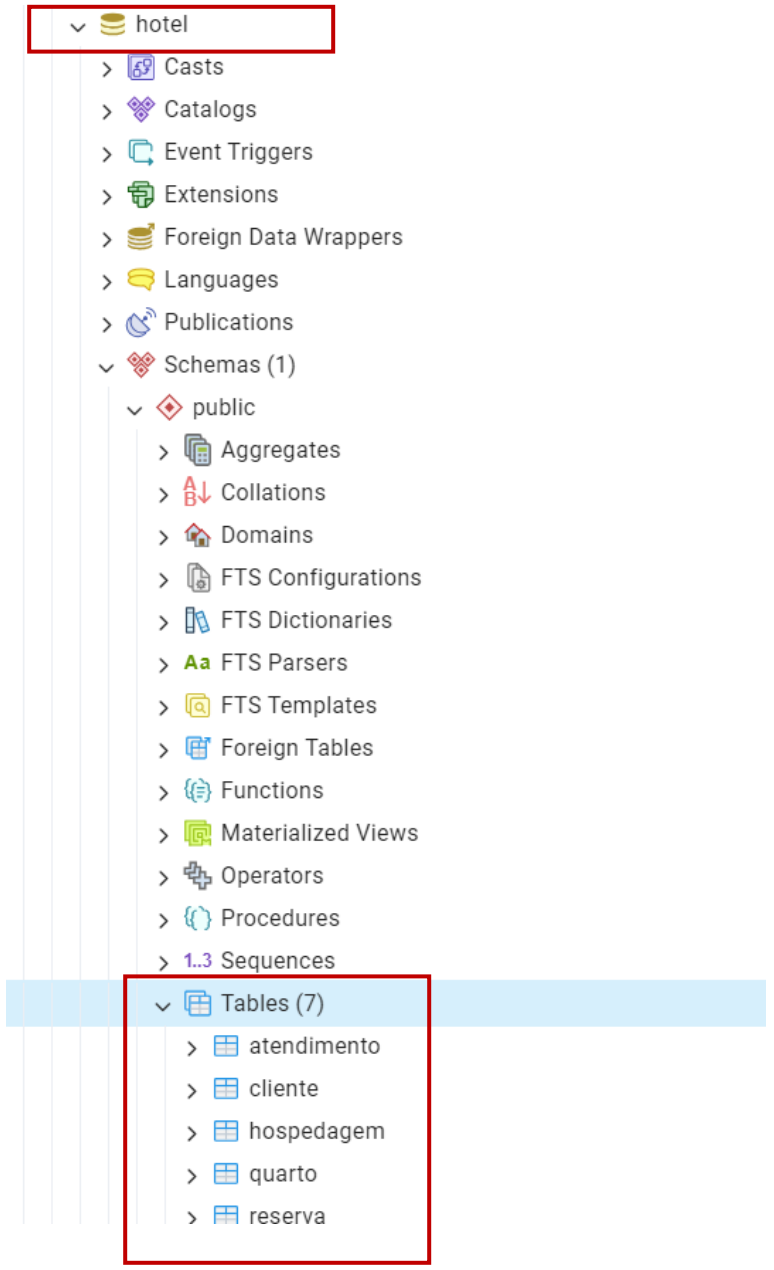
```
// Tabela QUARTO
 CREATE TABLE quarto
 (
 num_quarto INTEGER NOT NULL,
 andar CHAR(10),
 id_tipo INTEGER NOT NULL,
 status CHAR(01) NOT NULL DEFAULT 'D',
 PRIMARY KEY (num_quarto),
 FOREIGN KEY (id_tipo) REFERENCES tipo_quarto (id_tipo)
 ON UPDATE RESTRICT ON DELETE RESTRICT
 ) WITHOUT OIDS;

 // Tabela SERVIÇO
 CREATE TABLE servico
 (
 id_servico SERIAL NOT NULL,
 descricao VARCHAR(60) NOT NULL,
 valor NUMERIC(9,2) NOT NULL,
 PRIMARY KEY (id_servico)
 ) WITHOUT OIDS;
```

```
// Tabela RESERVA
 CREATE TABLE reserva
 (
 id_reserva SERIAL NOT NULL,
 rg NUMERIC NOT NULL,
 num_quarto INTEGER NOT NULL,
 dt_reserva DATE NOT NULL,
 qtd_dias INTEGER NOT NULL,
 data_entrada DATE NOT NULL,
     status CHAR(1) NOT NULL DEFAULT 'A',
  PRIMARY KEY (id_reserva),
  FOREIGN KEY (rg) REFERENCES cliente (rg)
  ON UPDATE RESTRICT ON DELETE RESTRICT,
  FOREIGN KEY (num_quarto) REFERENCES quarto
(num_quarto)
  ON UPDATE RESTRICT ON DELETE RESTRICT
 ) WITHOUT OIDS;
```

```
// Tabela HOSPEDAGEM
 CREATE TABLE hospedagem
 (
     id_hospedagem SERIAL NOT NULL,
     rg NUMERIC NOT NULL,
     num_quarto INTEGER NOT NULL,
     data_entrada DATE NOT NULL,
     data_saida DATE,
     status CHAR(1) NOT NULL,
  PRIMARY KEY (id_hospedagem),
  FOREIGN KEY (rg) REFERENCES cliente (rg)
  ON UPDATE RESTRICT ON DELETE RESTRICT,
  FOREIGN KEY (num_quarto) REFERENCES quarto
(num_quarto)
  ON UPDATE RESTRICT ON DELETE RESTRICT
 ) WITHOUT OIDS;
```

```
// Tabela ATENDIMENTO
 CREATE TABLE atendimento
 (
    id_atendimento SERIAL NOT NULL,
    id_servico INTEGER NOT NULL,
    id_hospedagem INTEGER NOT NULL,
  PRIMARY KEY (id_atendimento),
  FOREIGN KEY (id_servico) REFERENCES servico
(id_servico)
  ON UPDATE RESTRICT ON DELETE RESTRICT,
  FOREIGN KEY (id_hospedagem) REFERENCES
hospedagem (id_hospedagem)
  ON UPDATE RESTRICT ON DELETE RESTRICT
 ) WITHOUT OIDS;
```



```
INSERT INTO tipo_quarto VALUES
	(1, 'LUXO', 100.00),
	(2, 'SALAO DE FESTAS', 5000.00),
	(3, 'SUITE', 400.00),
	(4, 'NORMAL', 100.00) ;
```

```
SELECT * FROM tipo_quarto;
```

Query | Query History

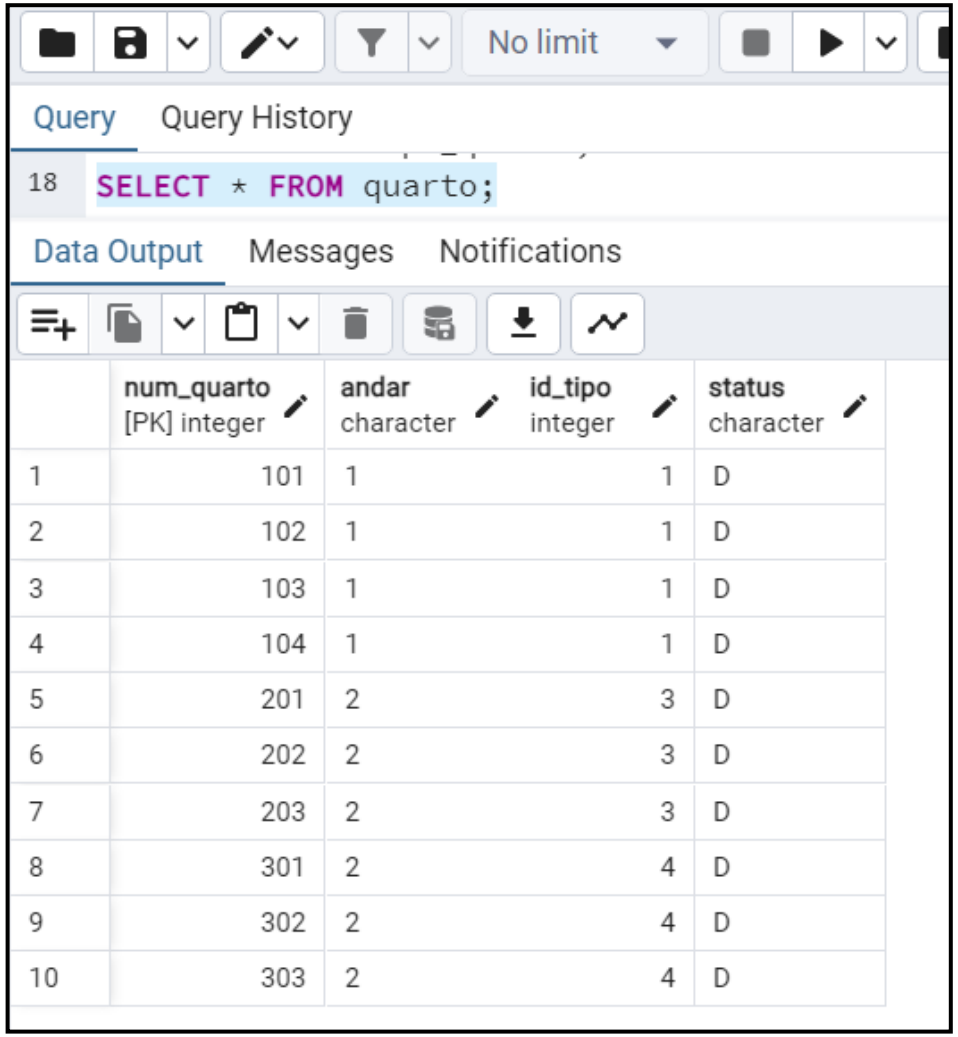
```
1  INSERT INTO tipo_quarto VALUES
2  --(1, 'LUXO', 100.00),
3  (2, 'SALAO DE FESTAS', 5000.00),
4  (3, 'SUITE', 400.00),
5  (4, 'NORMAL', 100.00)
6  ;
```

Data Output | Messages | Notifications

	id_tipo [PK] integer	descricao character varying (40)	valor numeric (9,2)
1	1	LUXO	100.00
2	2	SALAO DE FESTAS	5000.00
3	3	SUITE	400.00
4	4	NORMAL	100.00

```
INSERT INTO quarto VALUES
    (102, 1, 1, 'D'),
    (103, 1, 1, 'D'),
    (104, 1, 1, 'D'),
    (201, 2, 3, 'D'),
    (202, 2, 3, 'D'),
    (203, 2, 3, 'D'),
    (301, 2, 4, 'D'),
    (302, 2, 4, 'D'),
    (303, 2, 4, 'D');
```

```
SELECT * FROM quarto;
```



	num_quarto [PK] integer	andar character	id_tipo integer	status character
1	101	1	1	D
2	102	1	1	D
3	103	1	1	D
4	104	1	1	D
5	201	2	3	D
6	202	2	3	D
7	203	2	3	D
8	301	2	4	D
9	302	2	4	D
10	303	2	4	D

O Comando Create Function

Para funções implementadas em SQL, o comando create function tem a seguinte estrutura:

```
CREATE [OR REPLACE] FUNCTION NomeDaFuncao([parâmetro 1, ...parâmetro n])
    RETURNS RetornoTipoDeDados AS $$
        Corpo da Função;
$$
LANGUAGE 'SQL';
```

onde

CREATE FUNCTION

- Define o nome da função e seus respectivos parâmetros, caso existam. Esses parâmetros são identificados internamente como $1 (Parâmetro 1), $2 (Parâmetro 2), $3 (Parâmetro 3), e assim por diante.

RETURNS RetornoTipoDeDados

- Indica o tipo de dado de retorno da função. Uma função pode retornar um tipo simples como integer, varchar etc. Funções em SQL também podem retornar um conjunto de valores ou uma estrutura composta de várias linhas (resultset).

Corpo da Função

- Contém a implementação da função e deve estar entre aspas simples.

LANGUAGE ‘SQL’

- indica que a linguagem utilizada para implementação da função é SQL (se estivéssemos utilizando a linguagem PL/PgSQL, usaríamos LANGUAGE ‘PLPGSQL’);

Função adicionaHospedagem

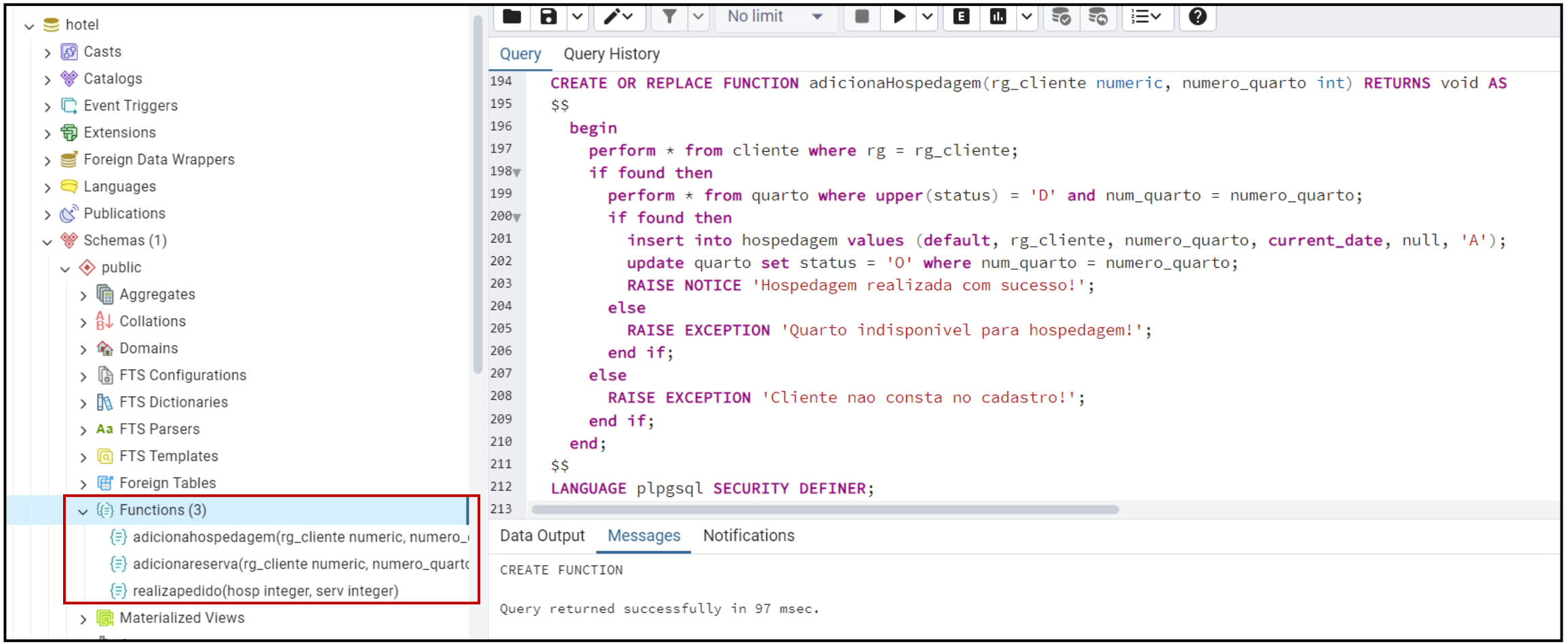
```
CREATE OR REPLACE FUNCTION adicionaHospedagem(rg_cliente  numeric,
numero_quarto int) RETURNS void AS
 $
  begin
   perform * from cliente where rg = rg_cliente;
   if found then
    perform * from quarto where upper(status) = 'D' and num_quarto =
numero_quarto;
    if found then
     insert into hospedagem values (default, rg_cliente, numero_quarto,
current_date, null, 'A');
     update quarto set status = 'O' where num_quarto = numero_quarto;
     RAISE NOTICE 'Hospedagem realizada com sucesso!';
    else
     RAISE EXCEPTION 'Quarto indisponivel para hospedagem!';
    end if;
   else
    RAISE EXCEPTION 'Cliente nao consta no cadastro!';
   end if;
  end;
 $
 LANGUAGE plpgsql SECURITY DEFINER;
```

Função adicionaReserva

```
CREATE OR REPLACE FUNCTION adicionaReserva(rg_cliente numeric,
numero_quarto int, dias int, data_entrada date) RETURNS void AS
 $
  begin
   perform * from cliente where rg = rg_cliente;
   if found then
    perform * from quarto where upper(status) = 'D' and num_quarto =
numero_quarto;
     if found then
      insert into reserva values (default, rg_cliente, numero_quarto,
current_date, dias, data_entrada, 'A');
       update quarto set status = 'R' where num_quarto = numero_quarto;
       RAISE NOTICE 'Reserva realizada com sucesso!';
     else
      RAISE EXCEPTION 'Quarto indisponivel para reserva!';
     end if;
   else
    RAISE EXCEPTION 'Cliente nao consta no cadastro!';
   end if;
  end;
 $
 LANGUAGE plpgsql SECURITY DEFINER;
```

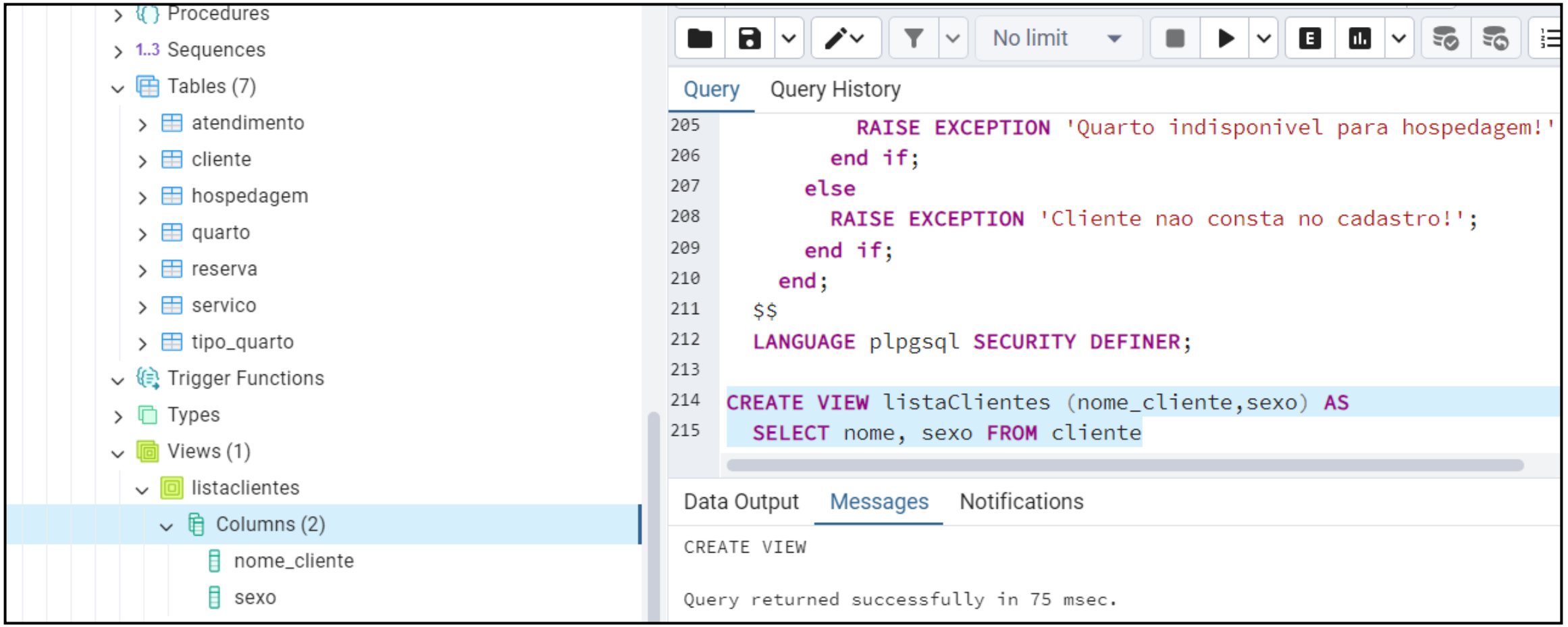
Função realizaPedido

```
CREATE OR REPLACE FUNCTION realizaPedido(hosp int, serv int) RETURNS
void AS
 $$
  begin
   perform * from hospedagem where upper(status) = 'A' and id_hospedagem
= hosp;
    if found then
     perform * from servico where id_servico = serv;
     if found then
      insert into atendimento values (default, serv, hosp);
      RAISE NOTICE 'Pedido realizado com sucesso!';
     else
      RAISE EXCEPTION 'Servico indisponivel!';
     end if;
    else
     RAISE EXCEPTION 'Hospedagem nao consta no cadastro ou ja foi
desativada!';
    end if;
  end;
 $$
 LANGUAGE plpgsql SECURITY DEFINER;
```


hotel
Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wrappers
Languages
Publications
Schemas (1)
public
Aggregates
Collations
Domains
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions (3)
adicionahospedagem(rg_cliente numeric, numero_
adicionareserva(rg_cliente numeric, numero_quart
realizapedido(hosp integer, serv integer)
Materialized Views
No limit
Query
Query History
194 CREATE OR REPLACE FUNCTION adicionaHospedagem(rg_cliente numeric, numero_quarto int) RETURNS void AS
195 $$
196 begin
197 perform * from cliente where rg = rg_cliente;
198 if found then
199 perform * from quarto where upper(status) = 'D' and num_quarto = numero_quarto;
200 if found then
201 insert into hospedagem values (default, rg_cliente, numero_quarto, current_date, null, 'A');
202 update quarto set status = 'O' where num_quarto = numero_quarto;
203 RAISE NOTICE 'Hospedagem realizada com sucesso!';
204 else
205 RAISE EXCEPTION 'Quarto indisponivel para hospedagem!';
206 end if;
207 else
208 RAISE EXCEPTION 'Cliente nao consta no cadastro!';
209 end if;
210 end;
211 $$
212 LANGUAGE plpgsql SECURITY DEFINER;
213
Data Output
Messages
Notifications
CREATE FUNCTION
Query returned successfully in 97 msec.

Criação da visão para consultar o nome e o sexo dos clientes

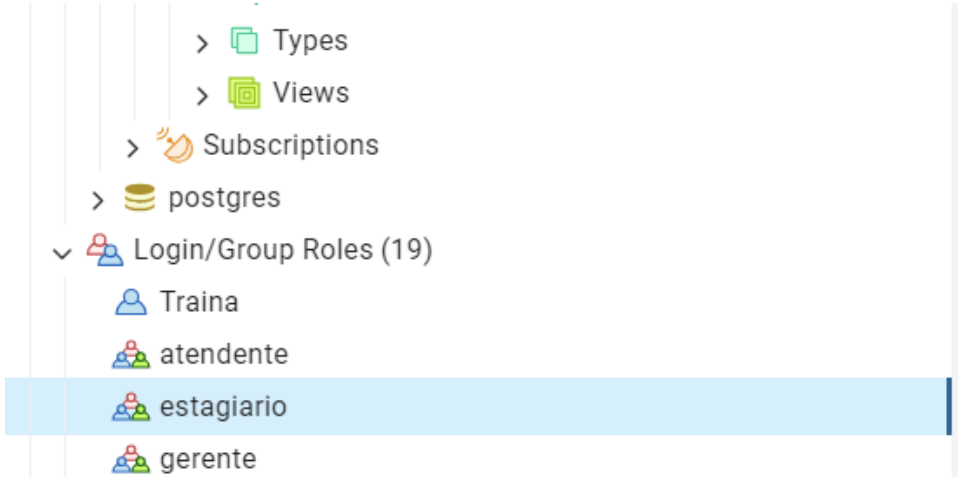
```
CREATE VIEW listaClientes (nome_cliente,sexo) AS
  SELECT nome, sexo FROM cliente
```

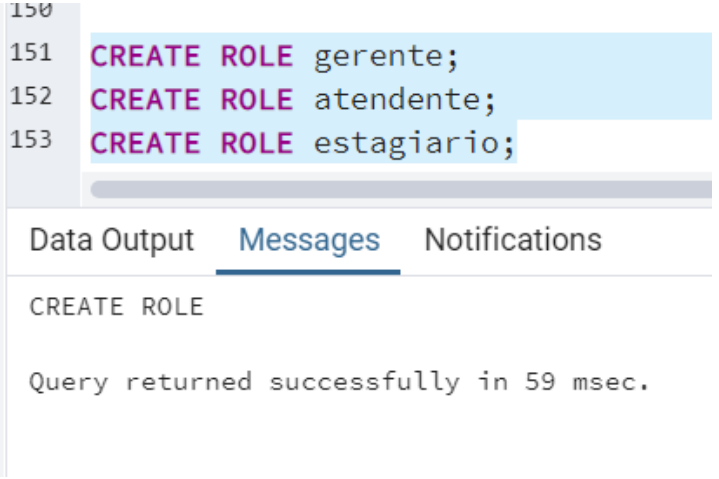


Criação dos papéis (roles) gerente, atendente e estagiário

```
CREATE ROLE gerente;
CREATE ROLE atendente;
CREATE ROLE estagiario;
```

Role	Permissão
Gerente	É o grupo de usuário que pode modificar todos os registros de todas as tabelas e utilizar as funções adicionaReserva, adicionaHospedagem e realizaPedidos,além de poder dar direitos de acesso aos demais usuários.
Atendente	É o grupo de usuários que pode manipular apenas as funções adicionaReserva, adicionaHospedagem e realizaPedido.
Estagiário	É o grupo de usuário que só tem acesso a visão listaClientes.



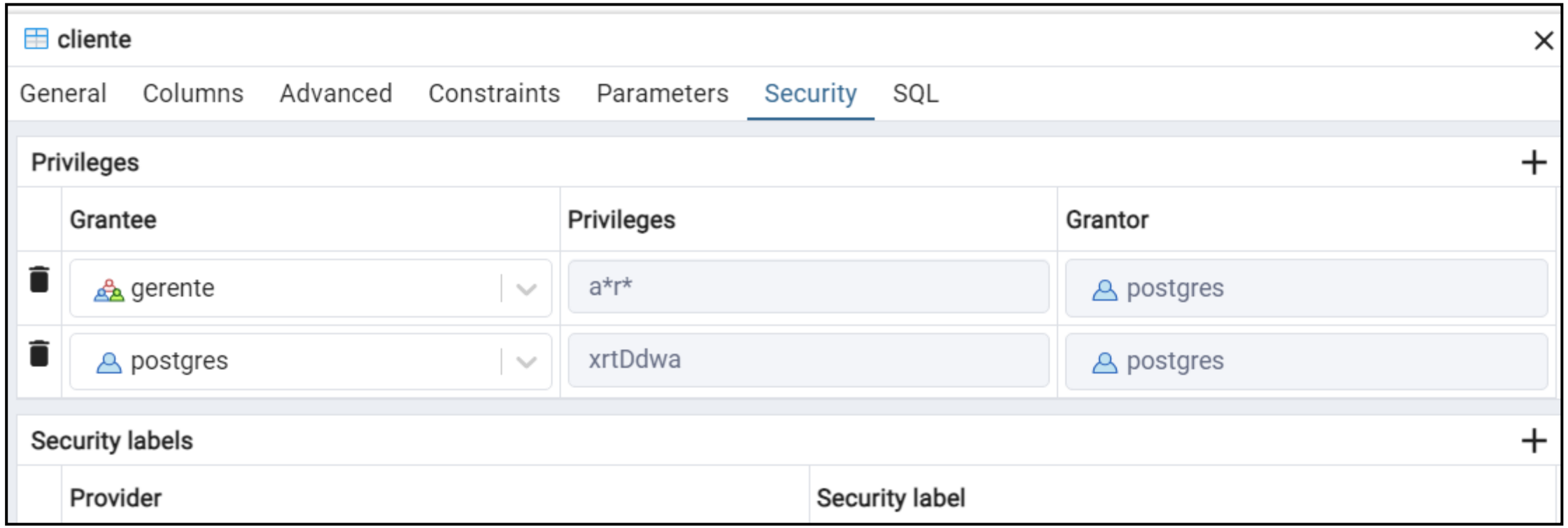


Suspendendo todos acessos públicos das funções

- adicionaReserva
- adicionaHospedagem
- realizaPedido

```
REVOKE ALL ON FUNCTION adicionaReserva(numeric,int,int,date) FROM PUBLIC;
 REVOKE ALL ON FUNCTION adicionaHospedagem(numeric,int) FROM PUBLIC;
 REVOKE ALL ON FUNCTION realizaPedido(int,int) FROM PUBLIC;
```

```
GRANT SELECT, INSERT ON cliente, reserva, hospedagem, quarto, tipo_quarto, atendimento, servico, listaClientes TO gerente WITH GRANT OPTION;
```



Concedendo permissão para o role gerente para acessar a função adicionaHospedagem

```
GRANT EXECUTE ON FUNCTION adicionaHospedagem(numeric,int)
TO gerente;
```

Concedendo permissão para o role gerente para acessar a função adicionaReserva

```
GRANT EXECUTE ON FUNCTION adicionaReserva
(numeric,int,int,date) TO gerente;
```

Concedendo permissão para o role gerente para acessar a função realizarPedido

```
GRANT EXECUTE ON FUNCTION realizaPedido(int,int) TO gerente;
```

Concedendo permissão para o role gerente para acessar a view listaClientes

```
GRANT SELECT ON listaClientes TO gerente
```

Concedendo permissão para o role atendente para acessar a função adicionaHospedagem

```
GRANT EXECUTE ON FUNCTION adicionaHospedagem(numeric,int)
TO atendente;
```

Concedendo permissão para o role atendente para acessar a função adicionaReserva

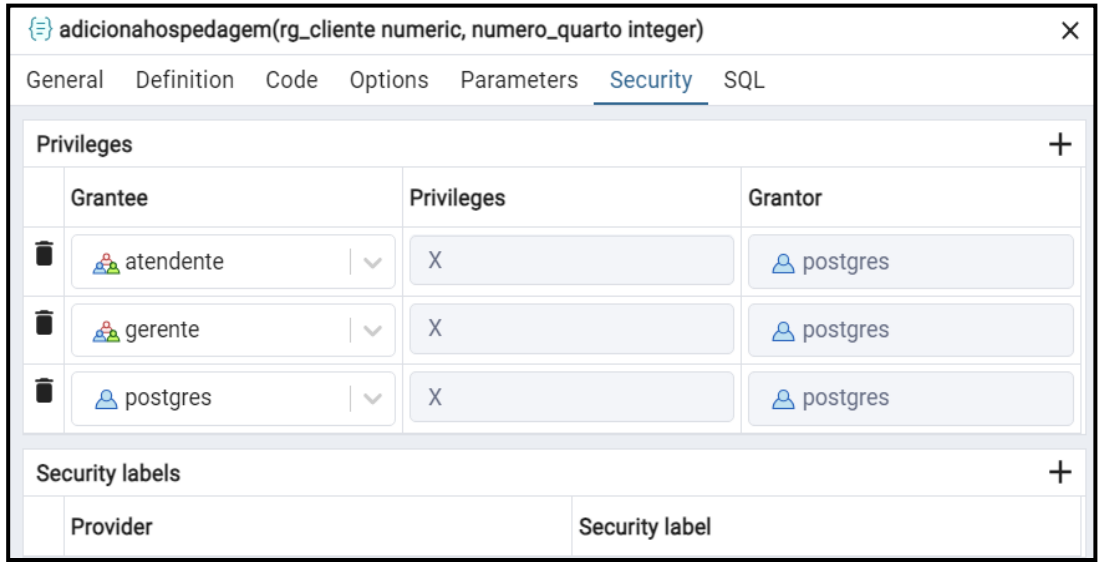
```
GRANT EXECUTE ON FUNCTION adicionaReserva
(numeric,int,int,date) TO atendente;
```

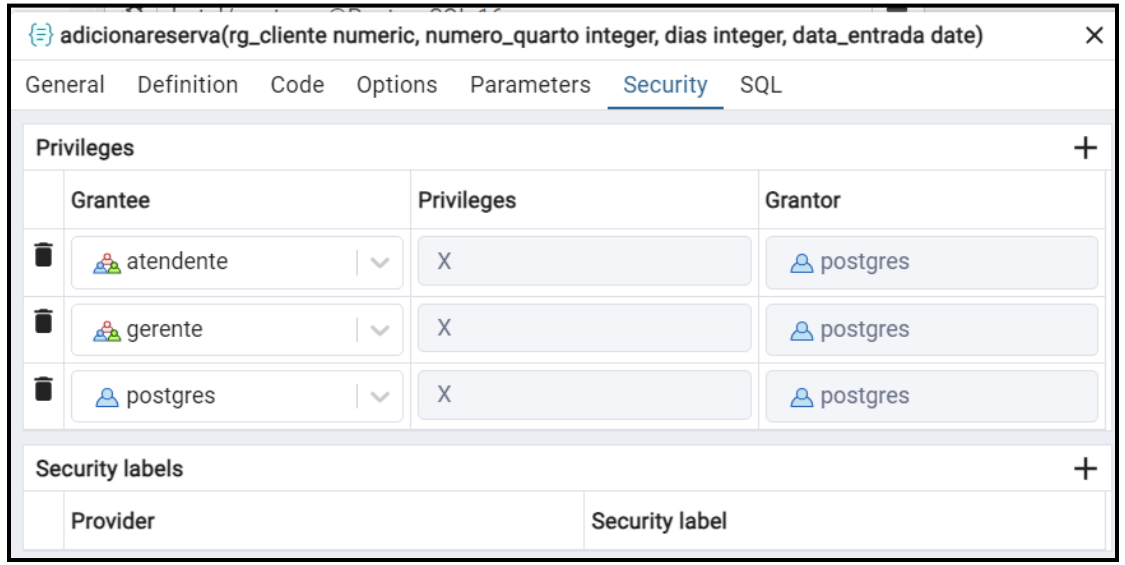
Concedendo permissão para o role atendente para acessar a função realizaPedidos

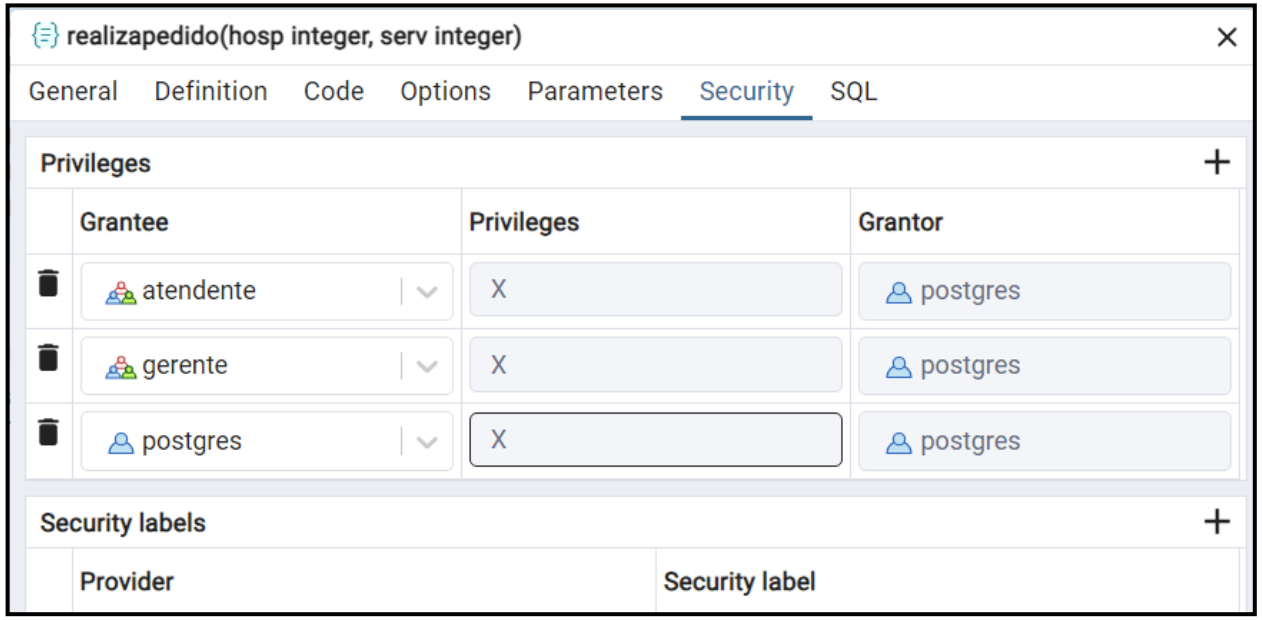
```
GRANT EXECUTE ON FUNCTION realizaPedido (int,int) TO atendente;
```

Concedendo permissão para o role estagiário acessar a visãolistaCliente

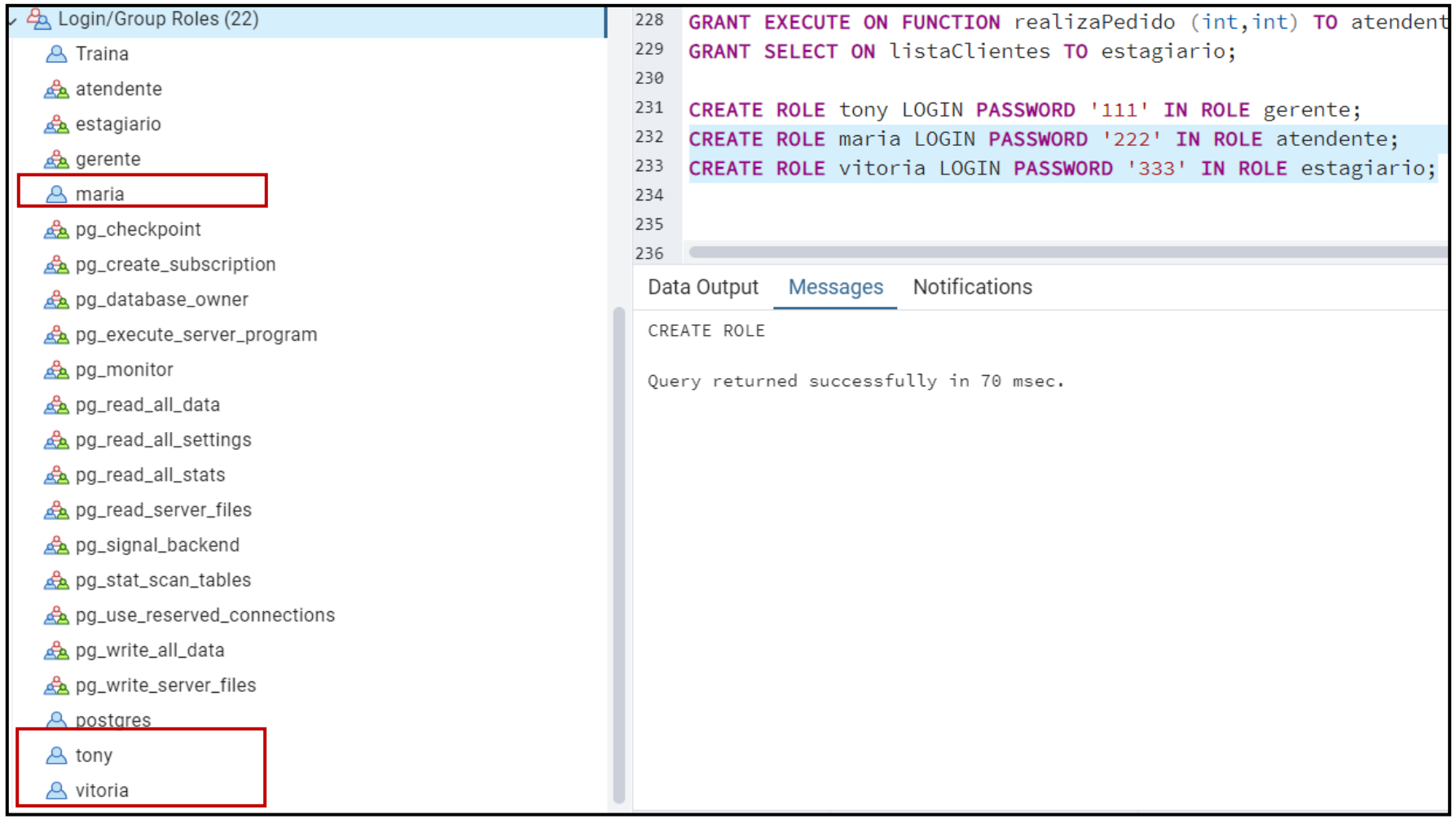
```
GRANT SELECT ON listaClientes TO estagiario;
```


adicionahospedagem(rg_cliente numeric, numero_quarto integer)
General
Definition
Code
Options
Parameters
Security
SQL
Privileges
Grantee
Privileges
Grantor
atendente
X
postgres
gerente
X
postgres
postgres
X
postgres
Security labels
Provider
Security label


adicionareserva(rg_cliente numeric, numero_quarto integer, dias integer, data_entrada date)
General
Definition
Code
Options
Parameters
Security
SQL
Privileges
Grantee
Privileges
Grantor
atendente
X
postgres
gerente
X
postgres
postgres
X
postgres
Security labels
Provider
Security label

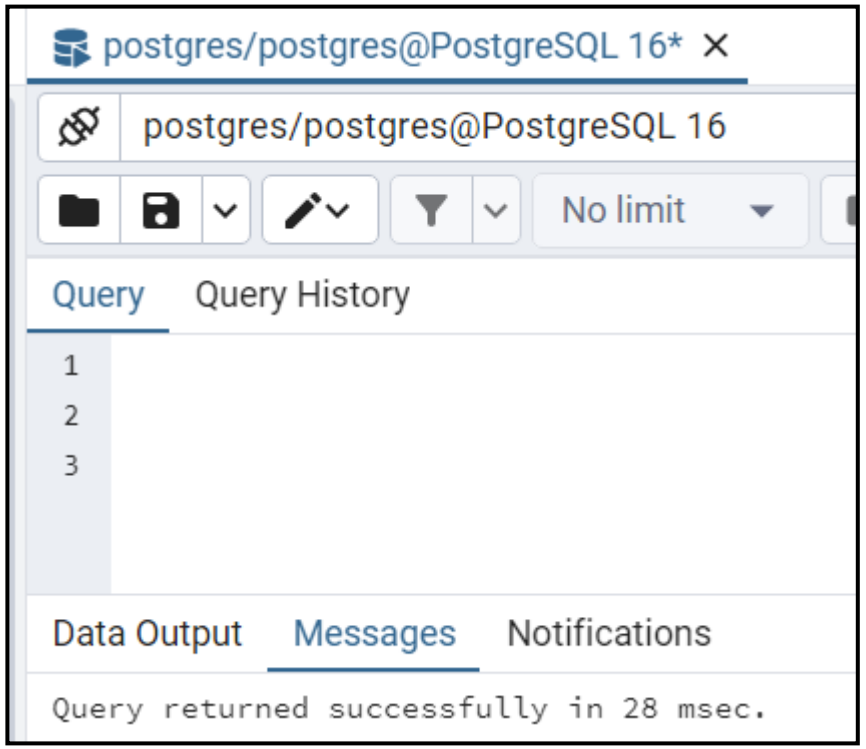

realizapedido(hosp integer, serv integer)
General
Definition
Code
Options
Parameters
Security
SQL
Privileges
Grantee
Privileges
Grantor
atendente
X
postgres
gerente
X
postgres
postgres
X
postgres
Security labels
Provider
Security label

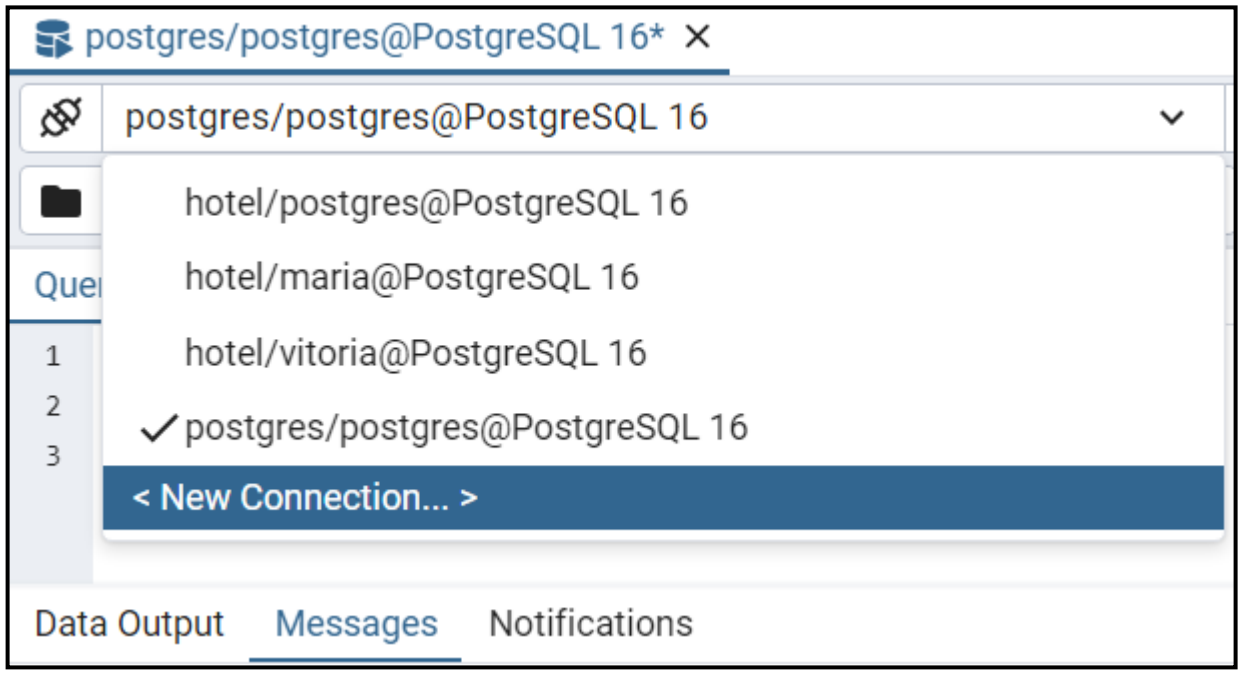
```
CREATE ROLE tony LOGIN PASSWORD '111' IN ROLE gerente;
CREATE ROLE maria LOGIN PASSWORD '222' IN ROLE atendente;
CREATE ROLE vitoria LOGIN PASSWORD '333' IN ROLE estagiario;
```



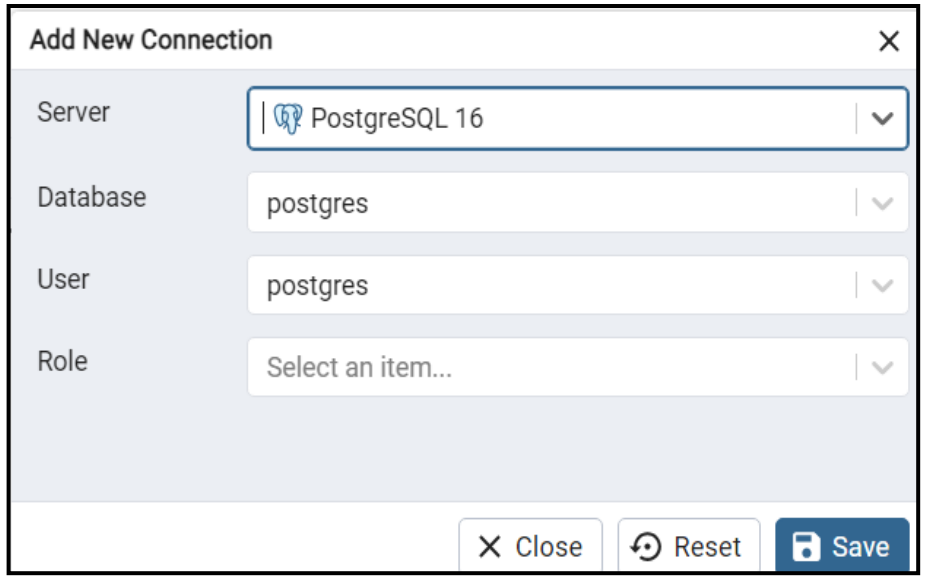
Validando as Permissões

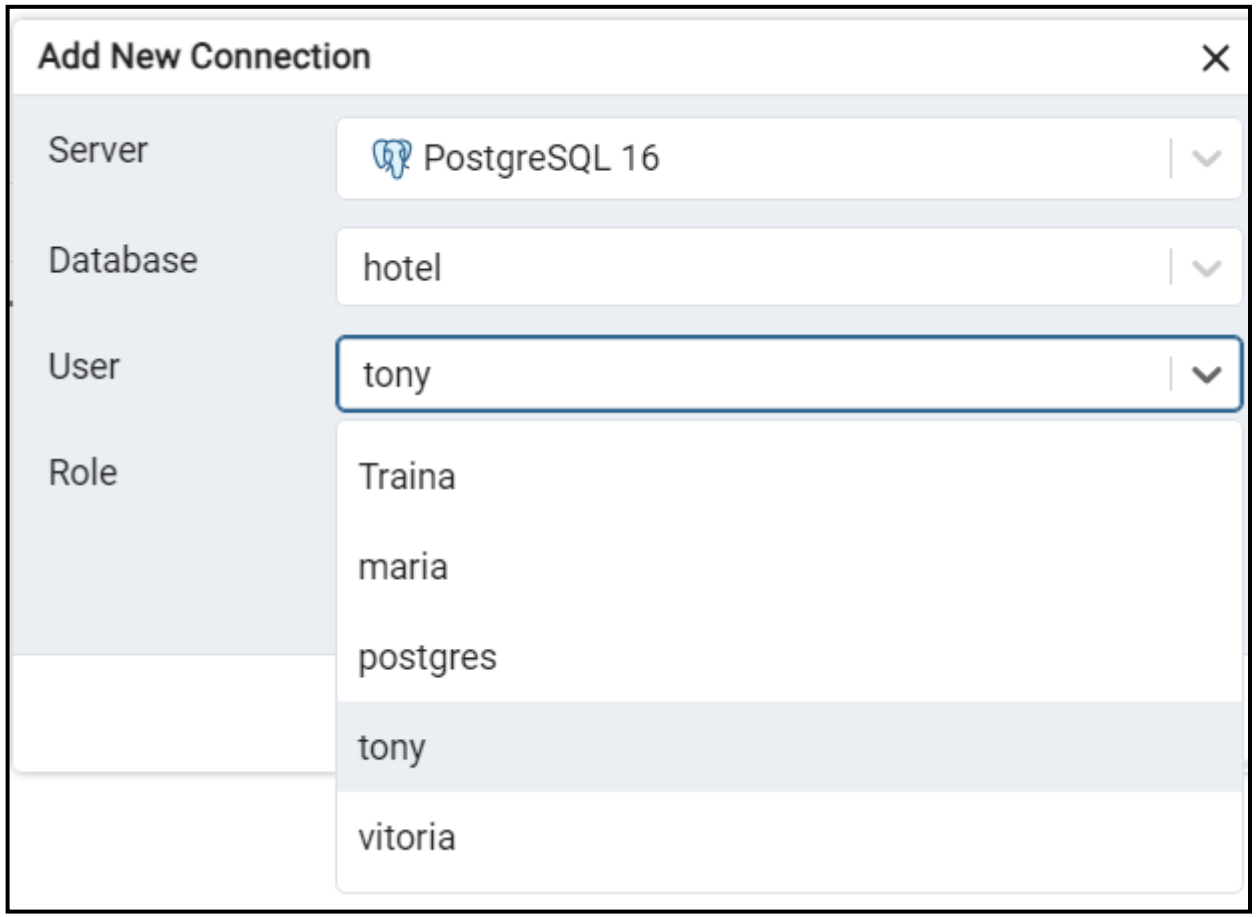
Trocando usuário no PostGreSQL



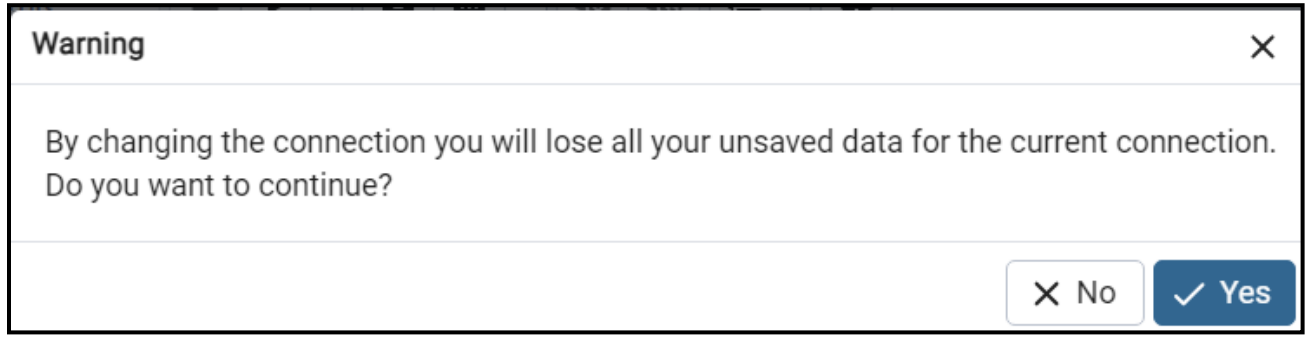


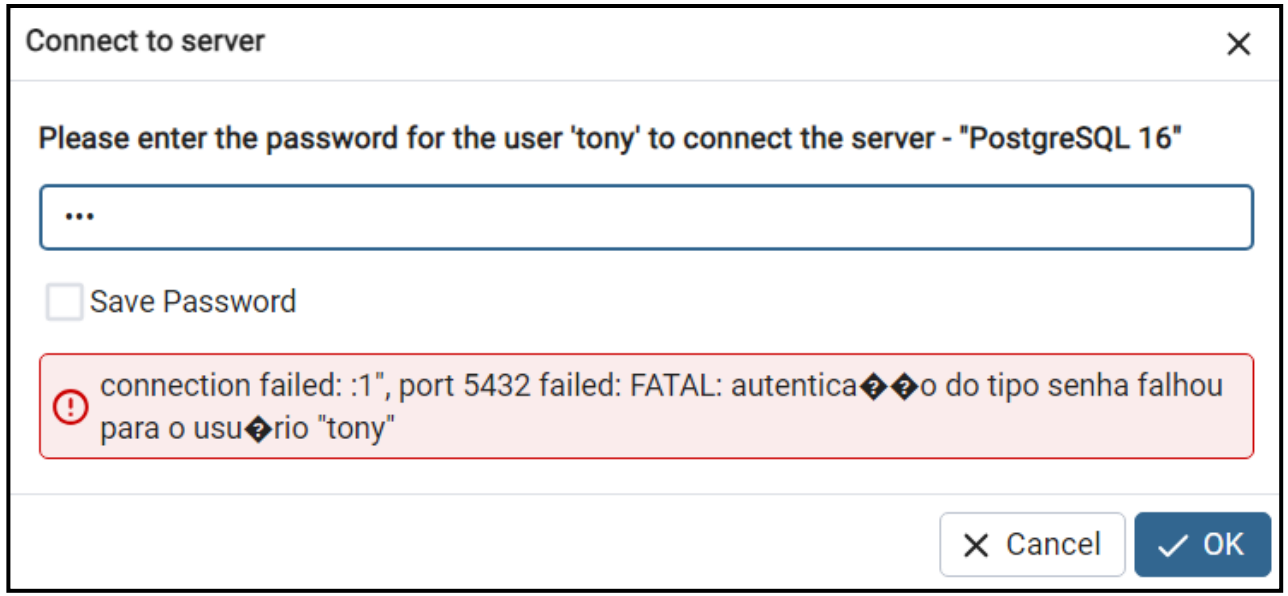
Trocando usuário no PostGreSQL

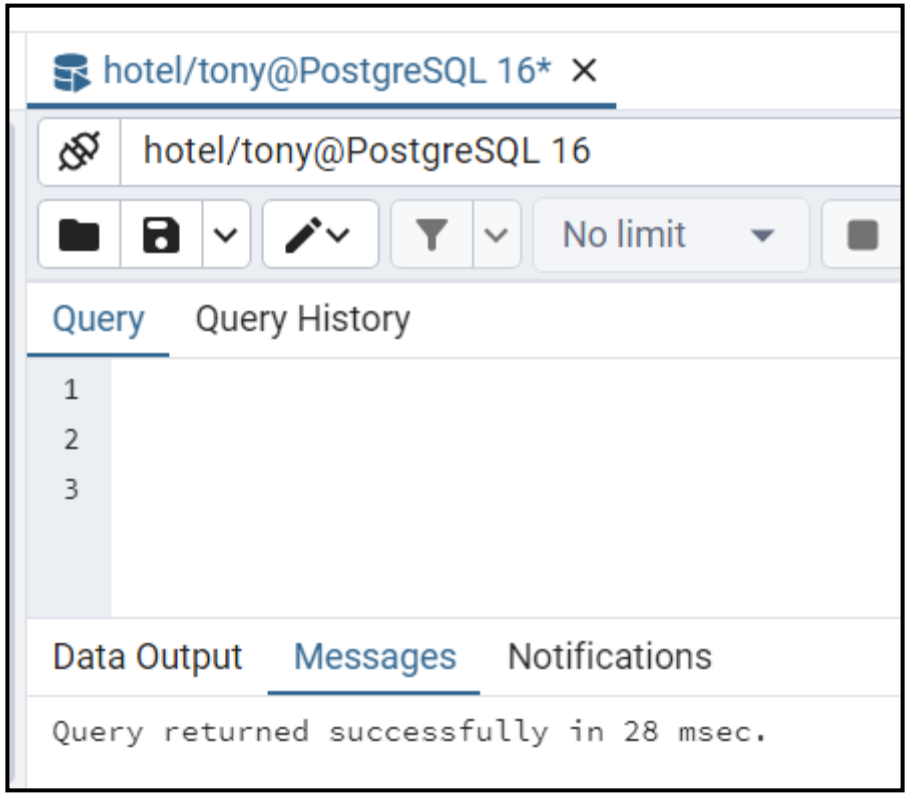




Trocando usuário no PostGreSQL

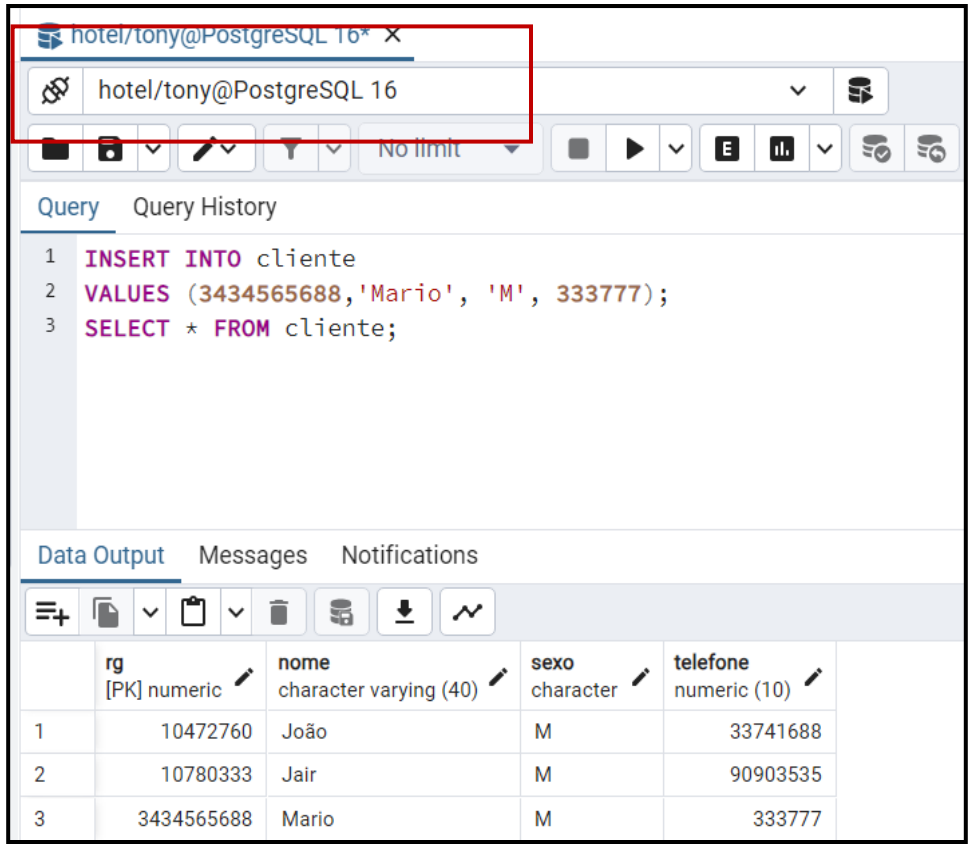






Usuário tony incluindo um novo cliente:

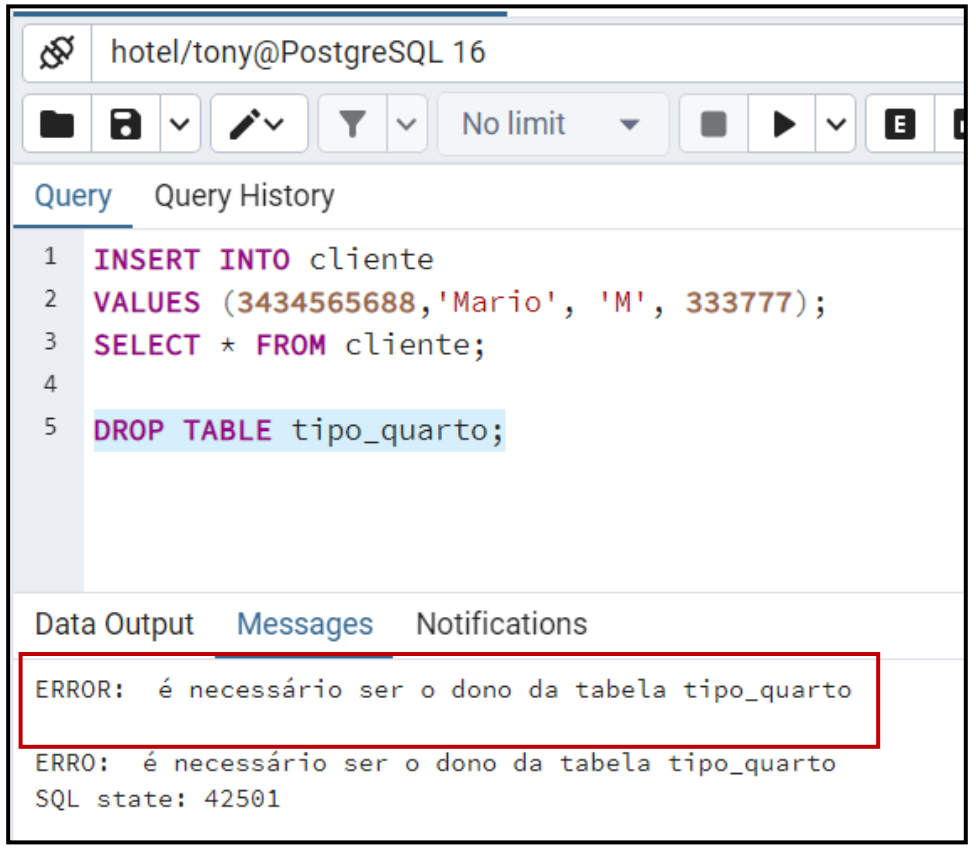
```
INSERT INTO cliente
VALUES (3434565688,'Mario', 'M', 333777);
SELECT * FROM cliente;
```



	rg [PK] numeric	nome character varying (40)	sexo character	telefone numeric (10)
1	10472760	João	M	33741688
2	10780333	Jair	M	90903535
3	3434565688	Mario	M	333777

Usuário tony tentando apagar do banco de dados a tabela tipo_quarto

```
DROP TABLE tipo_quarto;
```

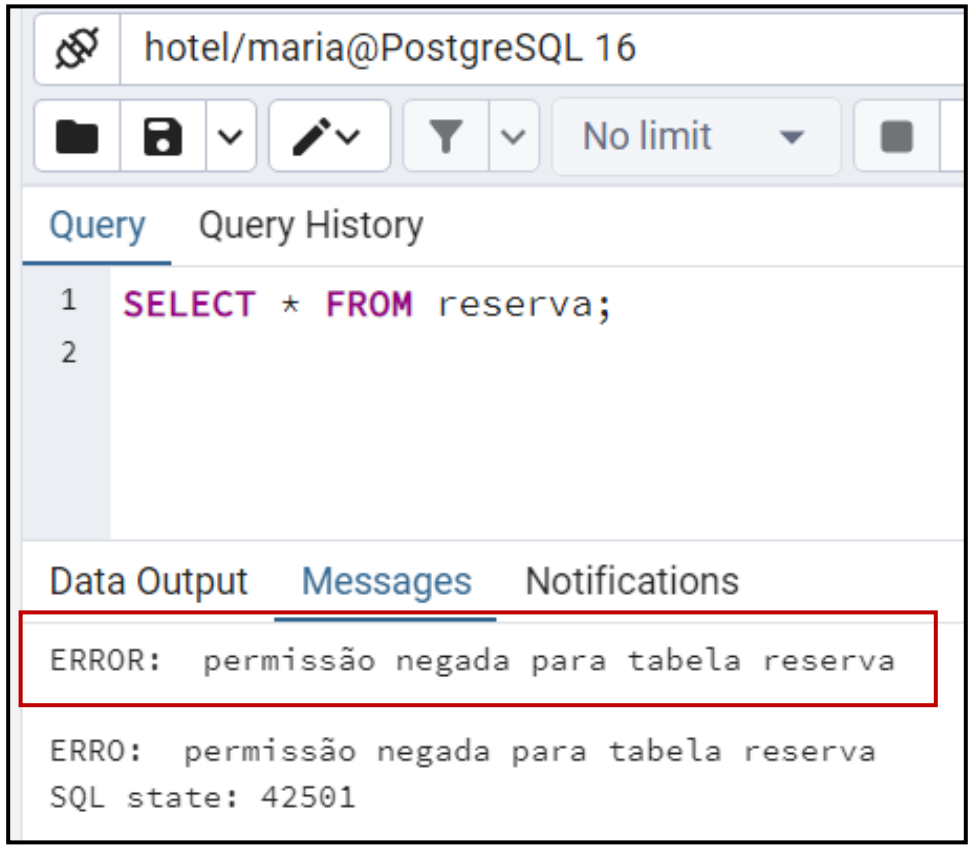


Usuário maria acessando a função *adicionaReserva*

```
SELECT adicionaReserva (3434565688, 301, 2, '2024-12-07');
```

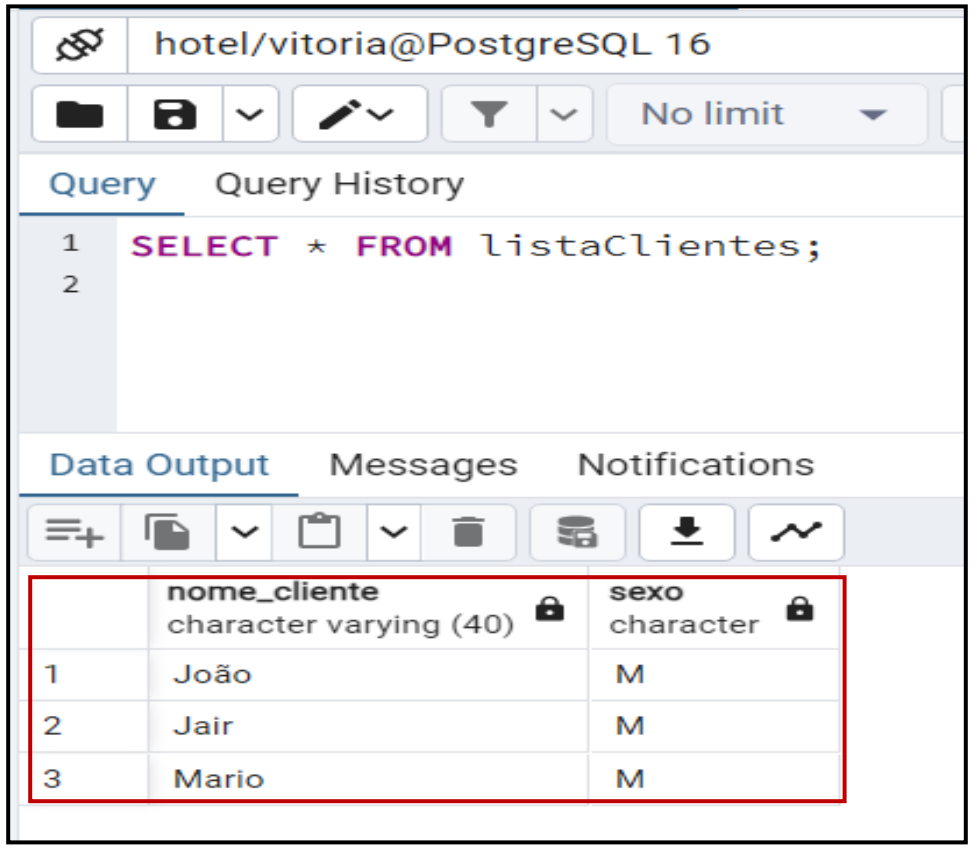
Usuário maria tentando selecionar os registros da tabela reserva

```
SELECT adicionaReserva (3434565688, 301, 2, '2024-12-07');
```



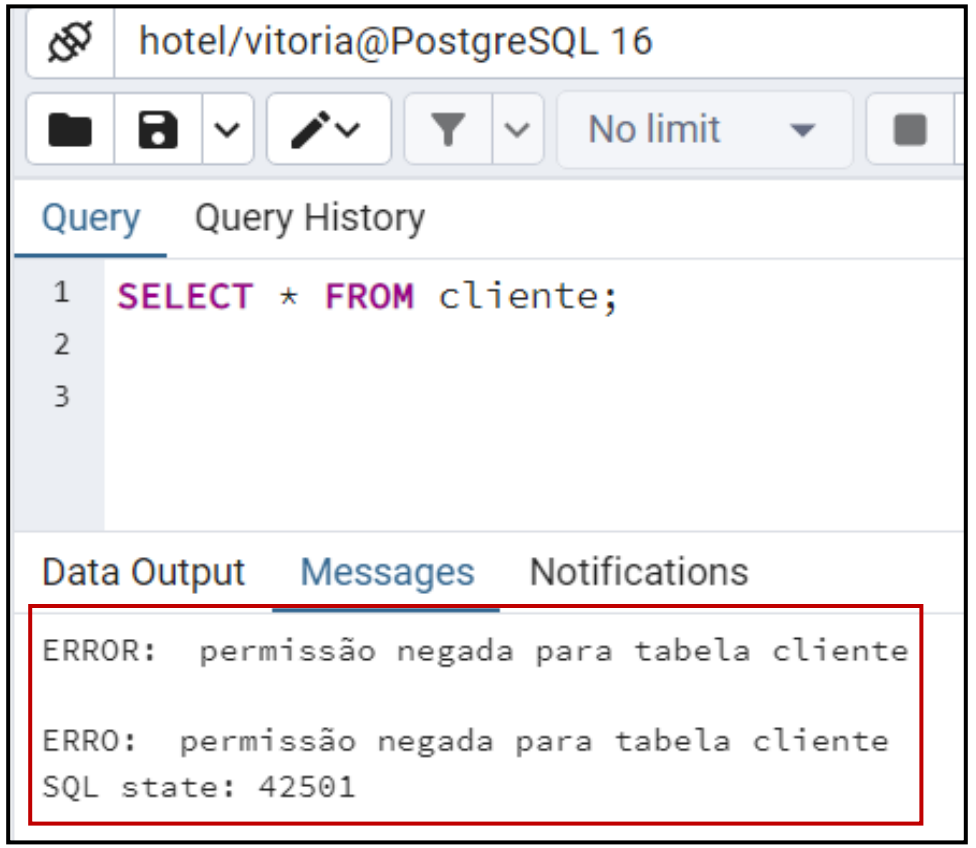
Usuário vitoria, consulta a visão *listaClientes*

```
SELECT * FROM listaClientes;
```



Usuário vitória tentando consultar diretamente a tabela cliente.

```
SELECT * FROM cliente;
```



Segurança

Segurança é importante em todos os níveis, principalmente a nível de informação.

Hoje o maior patrimônio de uma empresa são seus dados e por isso é tão importante preserva-los íntegros.

Referencias

https://www.postgresql.org/docs/current/index.html

https://www.devmedia.com.br/gerenciando-usuarios-e-permissoes-no-postgresql/14301

ELMASRI, Ramez E.; NAVATHE, Shamkant. Sistemas de Banco de Dados. 4 ed. São Paulo: Pearson Addsison Wesley, 2005. p. 544-552.

DAMAS, Luis. SQL – Structered Query Language. 6 ed. São Paulo: LTC, 2007. 396p.