Relatório de Laboratório: Administração de Usuários Oracle

Gabriel Vasconcelos Ferreira

24 de setembro de 2025

1 Conexão com SYSTEM e Análise Inicial

A primeira etapa consiste em conectar-se com o usuário `SYSTEM`, que possui privilégios administrativos, e verificar informações básicas sobre o ambiente.

1.1 Visão `v$version`

```
SELECT * FROM v$version;
```

Finalidade: A visão dinâmica `v$version` fornece informações sobre a versão do software Oracle Database e seus componentes instalados. Ela é útil para identificar a edição do banco de dados (ex: Enterprise, Standard, Express), a versão principal e de patch, e a compatibilidade do hardware. O resultado mostra, por exemplo:

- Oracle Database 19c Enterprise Edition Release 19.0.0.0.0
- PL/SQL Release 19.0.0.0.0
- TNS for Linux: Version 19.0.0.0.0

1.2 Visão `dba_users`

```
SELECT username FROM dba_users;
```

Finalidade: A visão `dba_users` é parte do dicionário de dados e contém informações sobre todos os usuários definidos no banco de dados. Um administrador (DBA) pode usar esta visão para listar os usuários existentes, verificar seus status (ex: `OPEN`, `LOCKED`), *tablespaces* padrão, data de criação, etc. A consulta retorna uma lista com todos os nomes de usuário cadastrados na instância.

2 Criação e Configuração do Usuário USR_LAB01

2.1 Comando `CREATE USER`

```
CREATE USER USR_LAB01
  IDENTIFIED BY SENHA
  DEFAULT TABLESPACE users
  QUOTA UNLIMITED ON users;
```

Explicação do Comando:

- `CREATE USER USR_LAB01`: Comando que inicia a criação de um novo usuário chamado `USR_LAB01`.
- `IDENTIFIED BY SENHA`: Define a senha inicial do usuário como `SENHA`.
- `DEFAULT TABLESPACE users`: Especifica que todos os objetos criados por este usuário (como tabelas e índices) serão armazenados na *tablespace* chamada `users`, a menos que outra seja explicitamente indicada.
- `QUOTA UNLIMITED ON users`: Concede ao usuário uma cota ilimitada de espaço de armazenamento na *tablespace* `users`. Sem uma cota, o usuário não poderia criar objetos nela.

2.2 Comando GRANT CONNECT, RESOURCE

```
GRANT CONNECT, RESOURCE to USR_LAB01;
```

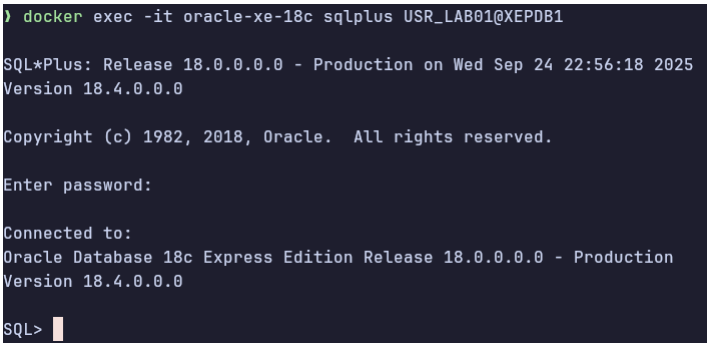
Finalidade das Roles CONNECT e RESOURCE:

- CONNECT: É uma *role* que agrupa privilégios básicos de sistema. O mais fundamental é o CREATE SESSION, que permite ao usuário conectar-se ao banco de dados. Sem este privilégio, a tentativa de login falharia com o erro "ORA-01045: user lacks CREATE SESSION privilege".
- RESOURCE: Esta *role* concede privilégios para criar objetos no próprio *schema* do usuário, como CREATE TABLE, CREATE PROCEDURE, CREATE TRIGGER, entre outros.

Observação: A Oracle atualmente recomenda a criação de *roles* customizadas com os privilégios mínimos necessários, em vez de usar CONNECT e RESOURCE, que podem conceder mais permissões do que o necessário.

2.3 Teste de Conexão

Após a criação e a concessão da *role* CONNECT, abriu-se uma nova janela de terminal/SQL Developer e foi possível conectar com o usuário USR_LAB01 e a senha SENHA.



3 Gerenciamento de Senha e Sessão

3.1 Alteração de Senha

O comando abaixo foi executado na janela do usuário SYSTEM.

```
ALTER USER USR_LAB01 IDENTIFIED BY new_password;
```

Este comando altera a senha do usuário USR_LAB01 para new_password.

3.2 Verificação da Sessão Ativa

Na janela onde o usuário USR_LAB01 já estava conectado, o seguinte comando foi executado:

```
SELECT table_name FROM all_tables;
```

Análise: O comando funcionou normalmente. A alteração de senha por um DBA **não invalida as sessões já existentes** daquele usuário. A nova senha só será exigida para novas tentativas de conexão.

3.3 Tentativa de Nova Conexão

Após encerrar a sessão de `USR_LAB01`, tentou-se conectar novamente.

- **Usando a senha antiga (`SENHA`):** A conexão falhou com o erro "ORA-01017: invalid username/password".
- **Usando a nova senha (`new_password`):** A conexão foi bem-sucedida.

O que aconteceu? O comando `ALTER USER` efetivou a troca da senha no dicionário de dados. Qualquer nova autenticação deve, obrigatoriamente, usar a credencial atualizada.

4 Propriedade e Criação de Objetos

Esta seção explora como a propriedade de objetos (*schema*) funciona no Oracle.

4.1 Criação de Tabelas pelo `SYSTEM`

Os comandos foram executados na janela do usuário `SYSTEM`.

```
SHOW USER; -- Retorna: USER is "SYSTEM"
CREATE TABLE xyz (name VARCHAR2(30));
```

Em qual usuário a tabela foi criada? A tabela `xyz` foi criada no *schema* do usuário que executou o comando, ou seja, no *schema* `SYSTEM`.

```
CREATE TABLE USR_LAB01.xyz (name VARCHAR2(30));
```

Em qual usuário a tabela foi criada? A tabela `xyz` foi criada no *schema* do usuário `USR_LAB01`.
Que nível de privilégio foi necessário? Para criar um objeto no *schema* de outro usuário, é necessário o privilégio de sistema `CREATE ANY TABLE`. O usuário `SYSTEM` possui este privilégio por padrão.

4.2 Verificação de Propriedade pelo `USR_LAB01`

Na janela do `USR_LAB01`, os comandos abaixo foram executados.

```
DESC xyz
```

O comando funcionou, exibindo a estrutura da tabela `xyz`, confirmando que ela pertence ao *schema* `USR_LAB01`.

```
DESC system.xyz
```

Esse comando funcionou? Não. O comando resultou no erro "ORA-04043: object system.xyz does not exist".

```
SQL> show user;
USER is "USR_LAB01"
SQL> desc sys.xyz;
ERROR:
ORA-04043: object sys.xyz does not exist
```

```
SQL> show user;
USER is "SYS"
SQL> desc sys.xyz;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 NAME                                               VARCHAR2(30)
```

O que falta ao usuário `USR_LAB01`? Falta o privilégio de acessar objetos do *schema* `SYSTEM`. Para que o comando `DESC` (e outros como `SELECT`) funcione, o usuário `USR_LAB01` precisaria receber o privilégio `SELECT` sobre a tabela `system.xyz` ou um privilégio mais amplo como `SELECT ANY TABLE`.

5 Concessão de Privilégios de Objeto

Nesta etapa, criamos um segundo usuário e concedemos a ele acesso a um objeto específico do USR_LAB01.

5.1 Criação do USR_LAB02 e Concessão de Privilégios

Na janela do SYSTEM:

```
CREATE USER USR_LAB02 IDENTIFIED BY SENHA DEFAULT TABLESPACE users;

GRANT INSERT, DELETE, SELECT ON USR_LAB01.XYZ TO USR_LAB02;

GRANT CONNECT TO USR_lab02;
```

Que operação está acontecendo? O comando GRANT INSERT, DELETE, SELECT ON USR_LAB01.XYZ TO USR_LAB02; está concedendo privilégios de objeto. Especificamente, o usuário USR_LAB02 está recebendo permissão para executar as operações de INSERT, DELETE e SELECT na tabela XYZ que pertence ao *schema* USR_LAB01.

5.2 Verificação de Privilégios Concedidos

```
select * from dba_tab_privs where grantee = 'USR_LAB02';
```

Qual o significado do resultado? Esta consulta à visão dba_tab_privs (privilégios de tabelas) mostra todos os privilégios de objeto concedidos a USR_LAB02. O resultado exibirá três linhas, indicando que o GRANTEE (beneficiário) USR_LAB02 recebeu os privilégios (PRIVILEGE) INSERT, DELETE e SELECT na tabela (TABLE_NAME) XYZ do proprietário (OWNER) USR_LAB01.

	GRANTEE	OWNER	TABLE_NAME	GRANTOR	PRIVILEGE	GRANTABLE	HIERARCHY	COMMON	TYPE	INHERITED
1	USR_LAB02	USR_LAB01	XYZ	USR_LAB01	DELETE	NO	NO	NO	TABLE	NO
2	USR_LAB02	USR_LAB01	XYZ	USR_LAB01	INSERT	NO	NO	NO	TABLE	NO
3	USR_LAB02	USR_LAB01	XYZ	USR_LAB01	SELECT	NO	NO	NO	TABLE	NO

5.3 Operações do USR_LAB02

Conectado como USR_LAB02:

```
-- Insercao de dados na tabela de outro schema
insert into usr_lab01.xyz values ('teste de nome');
commit;

-- Consulta aos dados
select * from usr_lab01.xyz;
```

Resultado e Explicação: O comando SELECT funcionou e retornou a linha inserida: ''teste de nome''. Ele funcionou porque o usuário USR_LAB02 recebeu explicitamente o privilégio SELECT sobre a tabela usr_lab01.xyz. O INSERT também funcionou pelo mesmo motivo.

```
select * from system.xyz;
select * from xyz;
```

Explicação do porquê NÃO funcionaram:

- `select * from system.xyz;`: Falhou porque `USR_LAB02` não tem permissão de acesso à tabela `xyz` do *schema* `SYSTEM`.
- `select * from xyz;`: Falhou porque, ao omitir o *schema*, o Oracle procura por uma tabela chamada `xyz` no *schema* do próprio usuário (`USR_LAB02`). Como essa tabela não existe, um erro é retornado.

6 Gerenciamento de Privilégios com Roles

Esta seção demonstra como usar *roles* para gerenciar o acesso a visões do dicionário de dados.

6.1 Acesso Restrito ao Dicionário de Dados

Na janela do `USR_LAB01`:

```
select * from dba_sys_privs;
```

O comando funcionou? Não. O acesso a visões `DBA_` é restrito a usuários com altos privilégios, como `SYSTEM` ou aqueles com o privilégio `SELECT ANY DICTIONARY` ou a *role* `SELECT_CATALOG_ROLE`.

6.2 Criação e Concessão da Role `new_dba`

Na janela do `SYSTEM`:

```
CREATE ROLE new_dba;
GRANT CONNECT TO new_dba;
GRANT SELECT ANY TABLE TO new_dba;
GRANT SELECT_CATALOG_ROLE TO new_dba;

GRANT new_dba TO USR_LAB01;
```

6.3 Verificação do Acesso

Na janela do `USR_LAB01`, após reconectar para que os novos privilégios da *role* sejam carregados na sessão:

```
select * from dba_sys_privs;
```

Explicação do processo:

1. Foi criada uma *role* customizada chamada `new_dba`.
2. A esta *role* foram concedidos privilégios de sistema importantes: `CONNECT`, `SELECT ANY TABLE` (permite ler dados de qualquer tabela em qualquer *schema*, exceto `SYS`) e `SELECT_CATALOG_ROLE` (uma *role* pré-definida que concede privilégio de leitura sobre as visões do dicionário de dados).
3. A *role* `new_dba`, contendo todos esses privilégios, foi então concedida ao usuário `USR_LAB01`.
4. Ao herdar os privilégios da *role*, `USR_LAB01` passou a ter permissão para consultar a visão `dba_sys_privs` e muitas outras, tornando o comando bem-sucedido.

7 Análise Final de Privilégios

As seguintes visões permitem auditar como os privilégios e *roles* foram distribuídos.

- `select * from dba_sys_privs`: Mostra os privilégios de sistema concedidos diretamente a usuários ou *roles*.
- `SELECT * FROM DBA_ROLE_PRIVS`: Lista quais *roles* foram concedidas a quais usuários (ex: mostra que `USR_LAB01` possui a *role* `NEW_DBA`).
- `SELECT * FROM ROLE_ROLE_PRIVS`: Mostra *roles* que foram concedidas a outras *roles*.
- `SELECT * FROM ROLE_SYS_PRIVS`: Detalha quais privilégios de sistema uma *role* possui (ex: mostra que `NEW_DBA` tem o privilégio `SELECT ANY TABLE`).
- `SELECT * FROM ROLE_TAB_PRIVS`: Detalha quais privilégios de objeto (em tabelas, views, etc.) uma *role* possui.