04

# Multitenant

## Arquitetura Multitenant

Vamos continuar com o nosso curso para certificação Oracle. Iremos ver um novo tipo de arquitetura que Oracle está implementando nesta versão 12c.

Vamos imaginar a seguinte situação: nós somos uma empresa que oferece um Relações Públicas e temos uma cartela de trinta clientes. Como não queremos que as informações se misturem entre si e que o banco de dados seja independente, nós criaremos **30 bases de dados**.

No entanto, cada cliente tem uma necessidade diferente. Um cliente de porte pequeno precisa de um computador mais simples, enquanto um de porte maior, precisa de uma máquina mais potente. Então, as 30 bases de dados estarão em 30 computares diferentes. Qual é o problema deste cenário? Quando trabalhamos com tantas máquinas, quando for preciso configurar um backup, em quantos computadores teremos que realizar o processo? Em trinta.

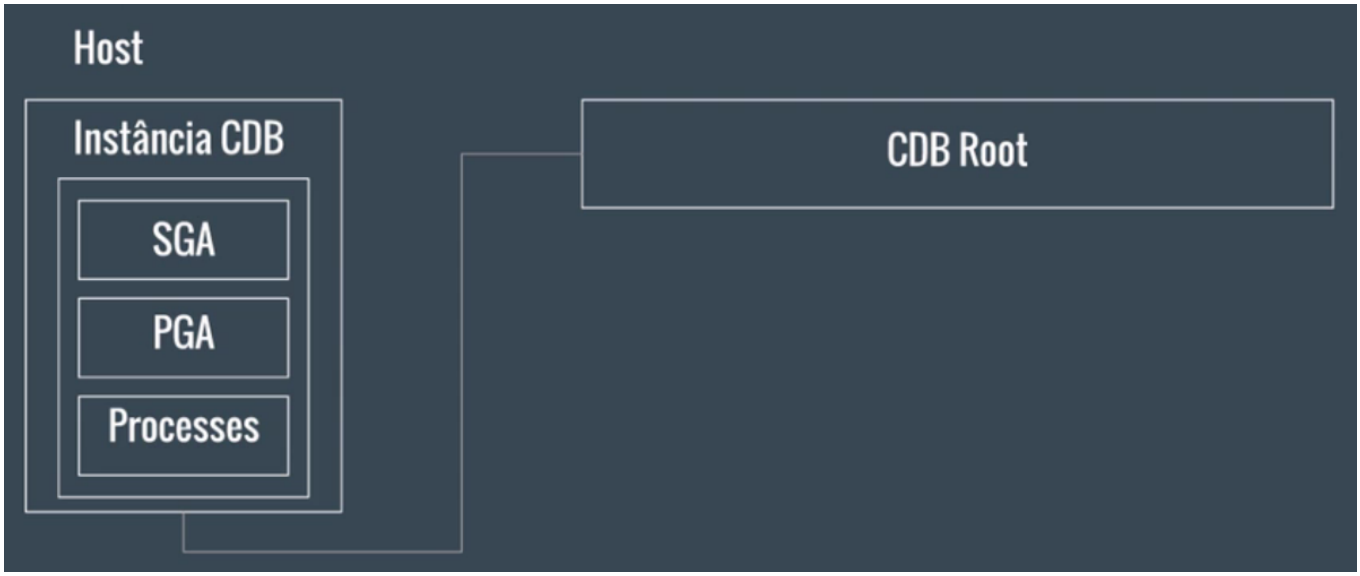
Manter 30 bases de dados não é fácil, quando pensamos em manutenção. Uma estratégia é **juntar tudo!** Em vez de utilizarmos 30 máquinas, iremos usar um servidor potente e vou armazenar nele todas as informações dos clientes.

Mas, mesmo colocando todas as bases em um única máquina, ainda teremos 30 para gerenciar! Unificamos o servidor, porém, cada base precisa de um configuração de backup, precisa das variáveis de ambiente e outros fatores.

Pensando nisto, começaram a criar o **Container Databases**, que é um *container*, onde colocamos várias bases de dados dentro. Vamos ver como isto funciona.

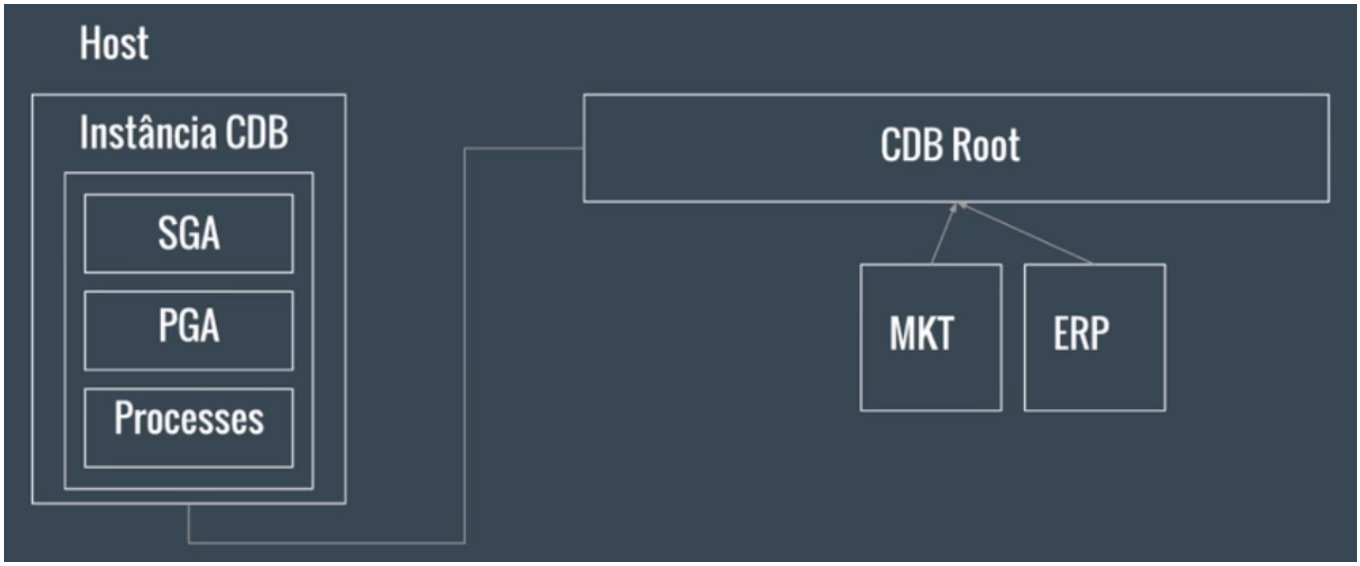
Lembra que a instância é o conjunto da *System Global Area* (SGA), *Process Gloval Area* (PGA) e do Processos (*Processes*). Trata-se do *software* do Oracle rodando. E esta instância está rodando em uma máquina, ou seja, em um *Host*.


A diferença é que, agora, além desta instância estar se comunicando com uma base, ela irá se comunicar com uma *Container Data Base* (CDB), o nosso Root.

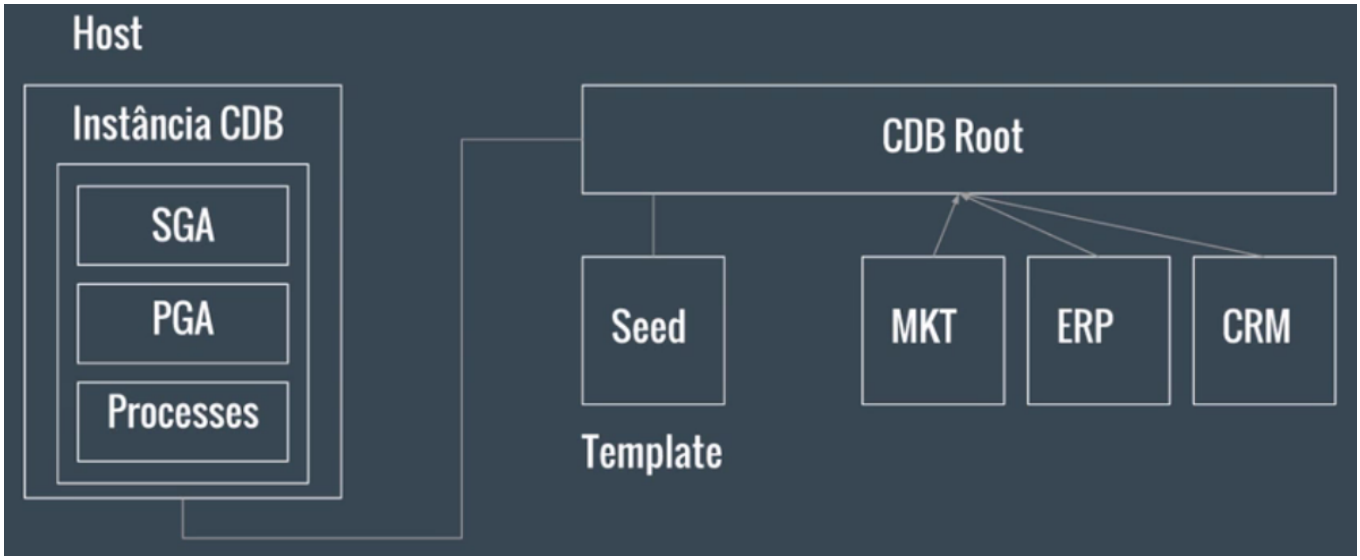



E como ele funciona? O CDB Root funciona de forma parecida a um filtro de linha, aquela régua na qual podemos colocar vários filtros de tomada. É como se tivéssemos uma régua e pudéssemos plugar várias bases de dados.

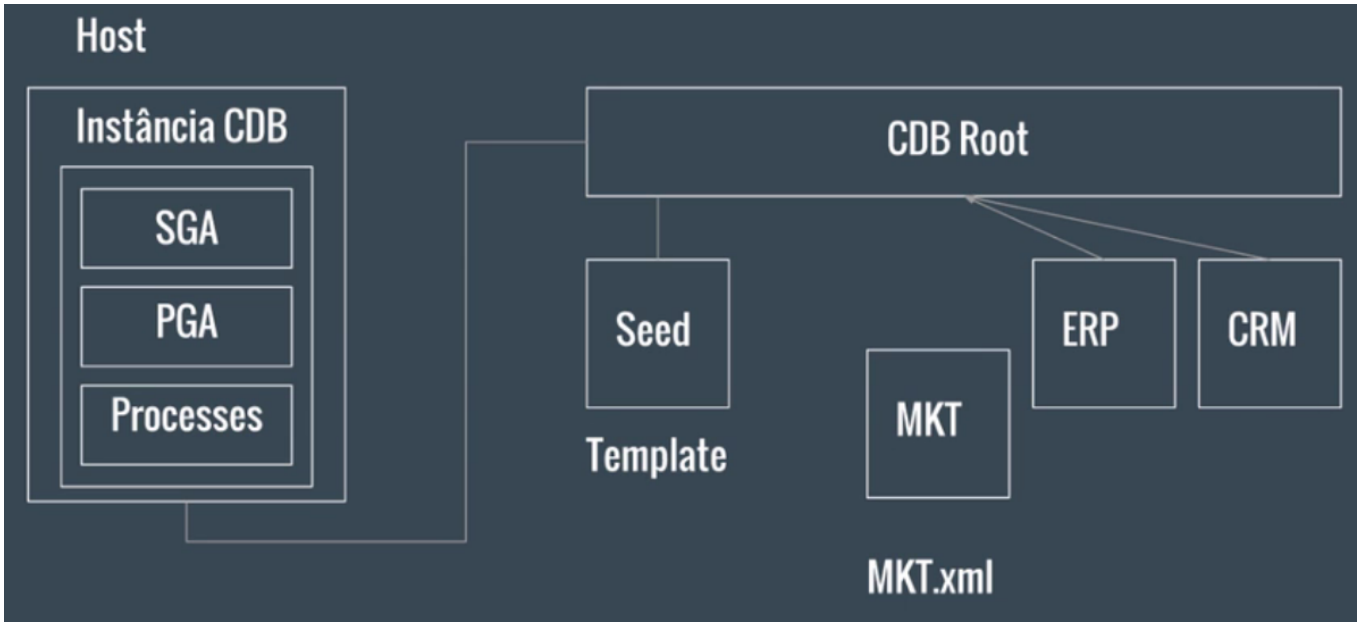
Por exemplo, se precisar de uma base de dados para *Marketing,* eu irei plugar a base específica no *Container Database*. Como plugamos está base de dados, chamamos ela de *Pluggable Databases* (Base de dados plugáveis). Se eu quiser incluir um módulo *ERP,* basta plugá-lo também.




E como faço para criar uma base de dados nova? Todo CDB Root tem **um** *Seed* - uma espécie de *template,* que permite a criação de novas bases de dados. Nele encontramos todas as bases de dados do CDB. Se quisermos criar um módulo de CRM (módulo voltado para o controle do relacionamento com o cliente), basta criar esta base usando como referência o meu *seed*. Após isto, eu posso plugar o CRM também.




Porém, e se precisarmos **trocar por um servidor mais potente?** Vamos imaginar que o meu módulo de *Marketing* está precisando de mais recursos e a máquina já não está mais aguentando. O que fazemos? Nós estamos trabalhando com um *Pluggable Database*, então, podemos simplesmente desplugar o meu módulo de Marketing. Quando eu faço isto, é gerado um arquivo `.xml`




Agora, se eu quiser plugar meu arquivo `Marketing.xml` em outra máquina, só precisarei avisar para minha CDB que ao montar a nova base de dados de Marketing, ela deverá se basear nas configurações do meu arquivo `xml` . Ou seja, o arquivo dirá para o *Container Database* como ele deve montar a base de dados.

**E como os dados não se misturam?**

Se observarmos o nosso esquema, veremos que temos uma instância. Então, eu tenho um software do Oracle rodando e tenho várias bases plugadas nele. Mas os nossos usuários são independentes. Nós temos usuários da nossa *Pluggable Database* (PDB). Logo, cada base de dados (Marketing, ERP e CRM) terá um usuário e nem saberá da existência do outro. Eles são independentes entre si.

Mas e se eu quiser fazer um backup, como faremos para configurá-lo? Lembra-se quando tínhamos 30 bases soltas, nós precisamos fazer 30 vezes a configuração de backup. Agora, como temos o *Container Database* queremos que o backup seja configurado nele. Então, também temos o **usuário de um CDB**.

Agora, conseguiremos separar as seções por responsabilidades. A parte relacionada ao que cada usuário enxerga, será o usuário da *Pluggable Database*, e as tarefas administrativas como atualizar a versão do banco ou configurar um backup, podemos fazer pelo usuário da *Container Database*.

## Vantagens

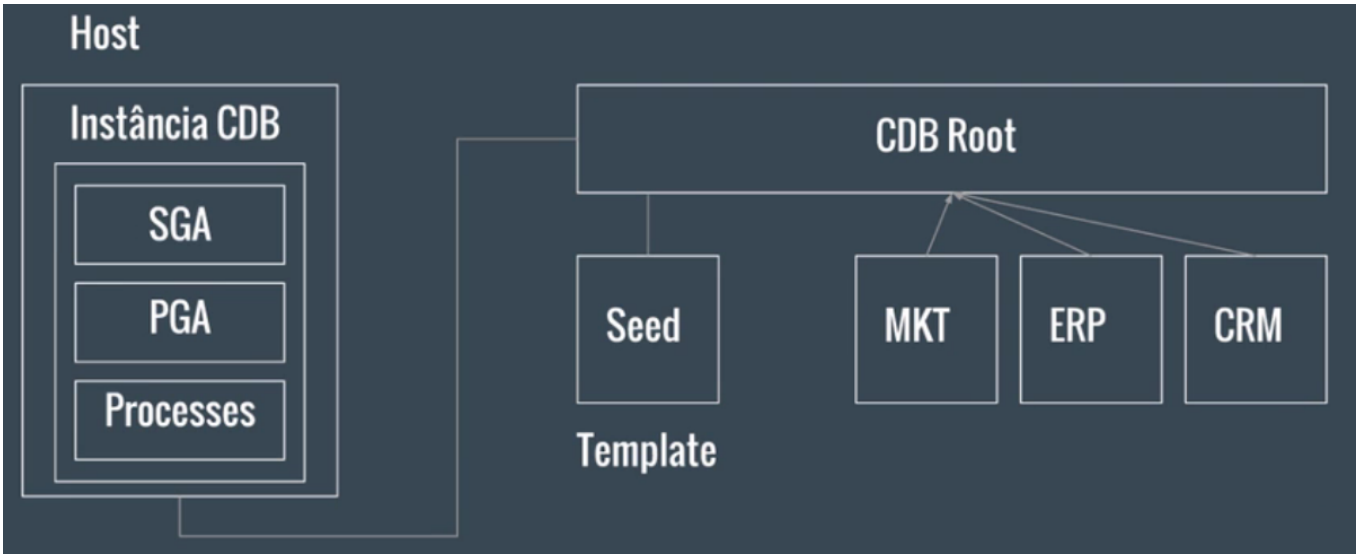
Iremos discutir agora, quais são as vantagens de usar o modelo de CDB. A primeira é que conseguimos **gerenciar diversas bases como se fossem apenas uma**. Por exemplo, se quisermos configurar um backup, iremos configurá-lo apenas uma vez. Se quiser atualizar a versão do Oracle, também só irei realizar o processo uma vez.

Com o modelo, nós também temos um melhor **aproveitamento do servidor**. Porque temos uma instância do Oracle, usando a máquina inteira. Assim, cada *Container Database* vai estar acessando esta instância, que irá usar o máximo possível da potencia do meu servidor. É diferente de termos um computador rodando várias instâncias, em que é preciso configurar o quanto de recurso cada uma poderá usar.

Uma terceira vantagem é que se quisermos migrar para uma máquina maior, vimos que usando o *Pluggable Database* isto é fácil. Nós conseguimos **mover bases facilmente**.

Além disso, nós **conseguimos separar responsabilidades**. Agora, temos um usuário específico para as tarefas administrativas e relacionado as bases de dados, cada uma terá um usuário independente. Elas são como inquilinos do meu Root. Por isso, o Oracle resolveu dizer que se trata de uma arquitetura de **vários inquilinos**, ou usando o termo em inglês, ***Multitenant***. Esta é uma arquitetura que temos no Oracle 12c, usada principalmente quando falamos de *Cloud*.

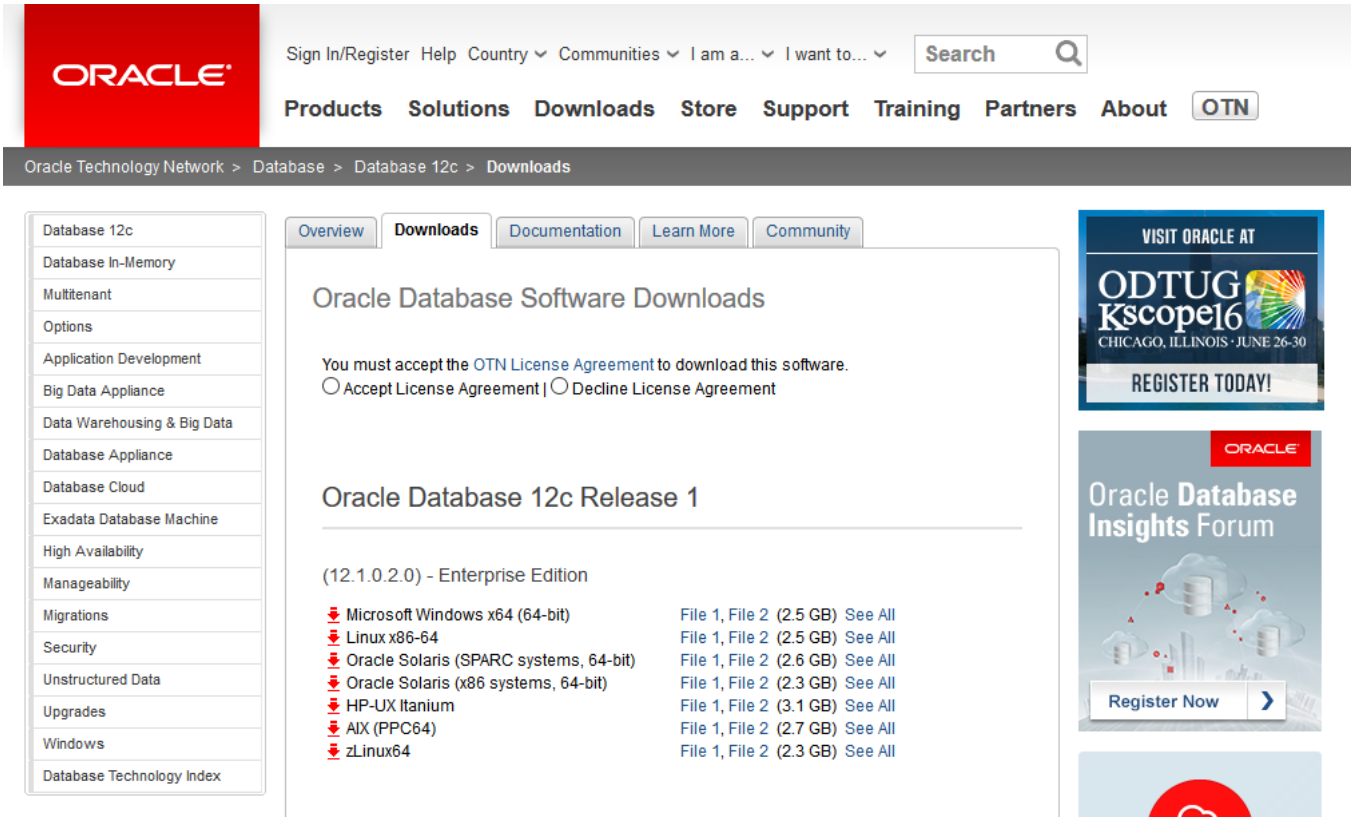
Em seguida, veremos como trabalhar com o Oracle na prática, mostrando o processo de instalação e apresentando alguns comandos.



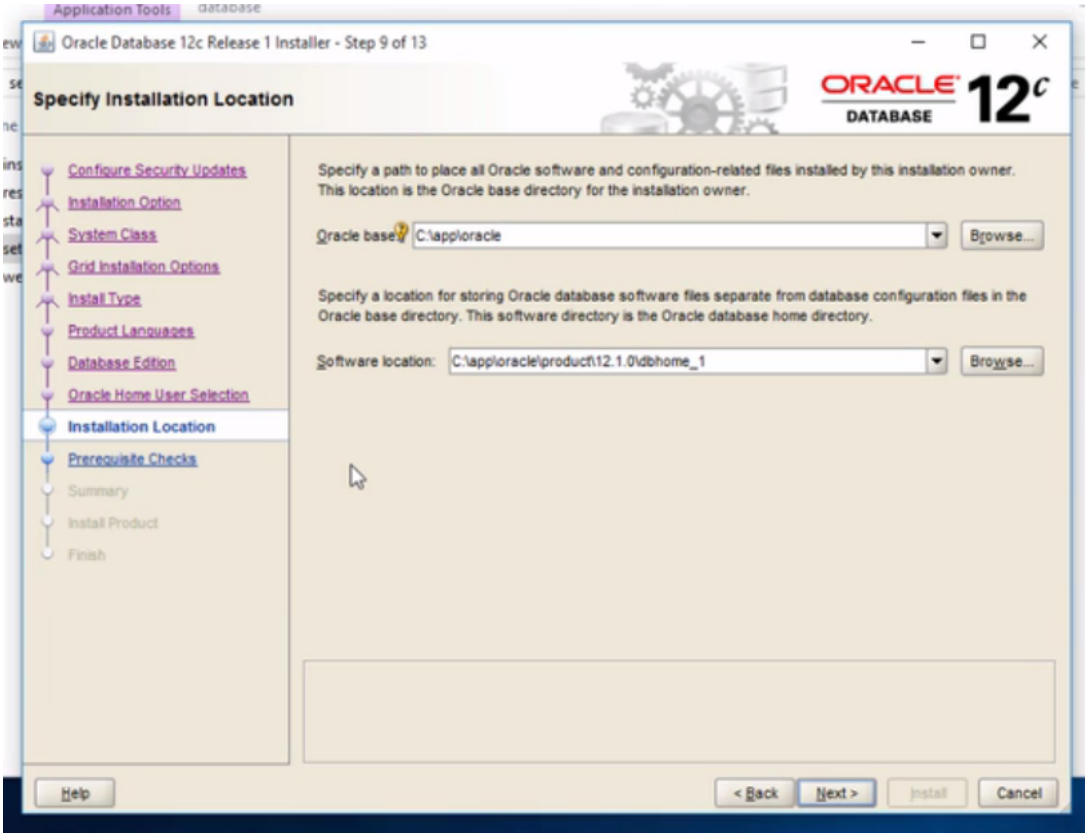

Esta é a nossa estrutura do Oracle 12c. Vamos pensar em como funciona esta estrutura: Temos um Root, no qual plugamos várias bases.

## Setup Oracle

Agora, iremos instalar a nossa base de dados Oracle. Para isto, precisamos fazer o download. Para isto, iremos acessar o site da Oracle, na seção de ***Downloads***, iremos clicar em ***Oracle Database*** (http://www.oracle.com/technetwork/pt/database/enterprise-edition/downloads/index.html). Na página, nós iremos procurar *Oracle Database 12c*.




Iremos usar a "Enterprise Edition*, precisaremos aceitar os termos de licença e teremos dois arquivos para baixar (*File 1 *e* File2*). Após clicarmos no link, ele irá iniciar o download, a página pedirá o seu login e senha da Oracle. Ele irá começar a baixar o arquivo. Como eu já fiz o download, vamos direto para o setup. Vamos extrair os dois arquivos na mesma pasta. Eu extrai na pasta* `setup-oracle` *, que virá na pasta* `database` *. Nós vamos iniciar o* setup*, que terá dez parte. Será preciso preencher um email para envio de novas configurações, no próximo passo, diremos que queremos "criar e configurar uma base de dados". Após clicar em "Next", escolheremos a opção "Server class". No passo seguinte, como queremos praticar alguns comandos, iremos selecionar a opção "Single Instance Database", porque queremos apenas uma instância. Depois, iremos escolher opções avançadas da instalação. Selecionarei o idioma "Inglês", porque a prova será em inglês. Clicaremos em "Next". No próximo passo, ele irá pedir para você colocar um usuário do Windows que estará associado a esta instância. Vou preencher meus dados, e no meu caso, ele dirá que temos privilégios administrativos. Para instalar a base de dados Oracle, eu vou precisar de um usuário que não tenha este privilégio. Então, eu vou criar um usuário novo do Windows, com o nome "Oracle" e vou definir uma senha. Em seguida, ele irá perguntar onde quero salvar os arquivos do meu setup. Deixarei a escolha padrão, que é no Disco `C:` , que é o principal.




Na "Configuração de Tipo", eu vou selecionar a opção "General Purpose" (Propósitos Gerais). No passo seguinte, ele irá me perguntar qual é o identificador do sistema, vou manter a seleção padrão. Neste momento, como não queremos trabalhar com a arquitetura *Multitenant*, eu direi que não quero criar um CDB. Mais adiante, nós iremos alterar a nossa opção. Depois de clicarmos em "Next", irei escolher o quanto de memória será alocado, vou manter o valor padrão: 6504 MB. Ainda no mesmo passo, mudaremos de aba para "Character sets", vamos alterar o *charset* para "Use unicode". Depois, vamos para o passo "Database Storage". O sistema de arquivos estará no `C:\app\oracle\oradata`, que se baseou no caminho definido no começo. Vou dar um "Next". Não quero me registrar no *Interpirse Manager*, porque só queremos testar uns comandos . Em "Schema Passwords", ele irá me perguntar quais são as senhas dos usuários padrões do Oracle Database, o SYS, o SYSTEM, e o DBSNMP. Vou selecionar a opção "Use the same password", para indicar que quero usar a mesma senha para todos eles e depois, irei defini-la. É importante que você lembre a senha, porque nós iremos utilizá-la durante todo o processo para administrar a nossa base de dados. No próximo passo, ele irá verificar se temos os pré-requisitos. Após a verificação, ele irá fazer um resumo do que será instalado. Em seguida, vou clicar em "Install". Agora, ele irá começar o processo de instalação, que demora cerca de 1 hora.

Quando ele chegar a 80%, ele irá abrir o *Database Configuration Assistant*. Na janela, aparecerá o `EM Database Express URL`. Devemos anotar a URL fornecida - pode ser em um bloco de notas -, porque ela será muito útil mais adiante. Enquanto isso, estão terminando as últimas configurações. Após terminar a instalação, vou fechar as janelas e teremos o banco de dados rodando.

Para acessá-lo, podemos abrir o prompt de comando. Quando instalamos o Oracle, veio juntamente o programa de console para manipular o banco de dados, o **SQL *Plus**. Ele irá pedir um *username*. No setup, ele pediu alguns *usernames* padrão, eu vou tentar logar com o `sys`.

```
/...

C:\Users\caelum>sqlplus

SQL*Plus: Release 12.1.0.2.0 Production on Mon Feb 29 12:10:35 2016
```

```
Copyright (c) 1982, 2014, Oracle. All rigths reserved.

Enteruser-name: sys
Enter password:
ERROR:
ORA-28009: connection as SYS should be as SYSDBA or SYSOPER

Enter user-name:
```

Ele está me avisando que para conectar com o `SYS`, precisa ser um `SYSDBA`. Então, eu direi que quero conectar como um `sysdba`. Ao digitar a minha senha novamente, vou me conectar com minha instância do Oracle.

```
Enter user-name: sys as sysdba
Enter password:

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 - 64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real Application Testing options

SQL>
```

Mas será que a minha base de dados está funcionando? Por isso, usaremos o comando `STARTUP`, lembrando que os comandos do SQL *Plus são *case insensitive*, então podemos digitá-lo com as letras maiúsculas ou minúsculas. E todos vão acabar com `;` (ponto e vírgula). Ele irá dizer que não conseguimos iniciar uma instância que já está rodando. Ou seja, o Oracle está funcionando.

```
SQL> STARTUP;
ORA-01081: cannot start already-running ORACLE - shut it down first
SQL>
```

Se quiséssemos derrubar a base de dados, usaríamos o `SHOUTDOWN`.

```
SQL> STARTUP;
ORA-01081: cannot start already-running ORACLE - shut it down first
SQL> SHOUTDOWN;
Database closed.
Database dismounted.
ORACLE instance shut down.
```

Ele falou que fechou a base de dados e a instância do Oracle caiu. Mas nós queremos executá-lo novamente, então vamos usar o comando `STARTUP`.

Para **subir** ou **derrubar** o Oracle, usaremos o `STARTUP` e `SHOUTDOWN`.

Ele irá informar quantos bites a *System Global Area* está utilizando, tamanho de variáveis, do buffer, em seguida, mostrará que a base de dados já foi montada ( `Database mounted` ).

Agora ele precisa iniciar a instância.

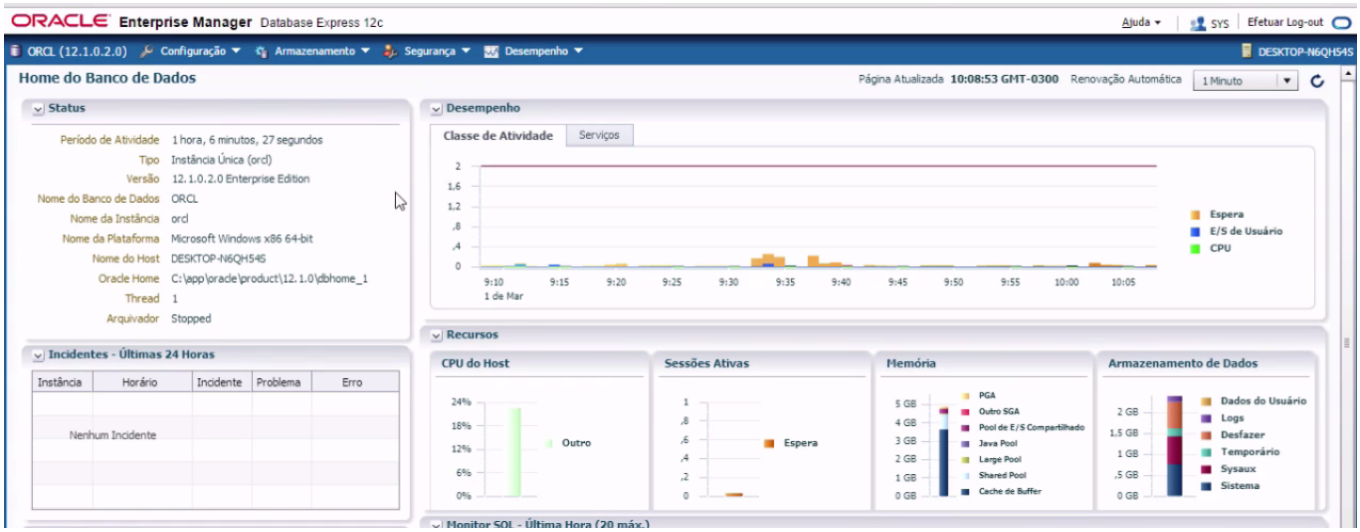
Nós já vimos como derrubar e subir o banco de dados Oracle. Logo adiante, veremos como criar um usuário e como trabalhar melhor com SQL *Plus.

## Final de SQL Buffer

Terminamos a instalação do Oracle, vamos começar a utilizar o nosso banco. Durante a instalação, havíamos copiado uma URL, vamos acessá-la para ver o que acontece.

Ele irá abrir o **Enterprise Manager Database Express 12c**. Esta é uma ferramenta para gerenciarmos a instância do Oracle. A página irá nos pedir um login e senha. Vamos utilizar o usuário `SYS` e a senha que determinamos durante o processo de instalação.

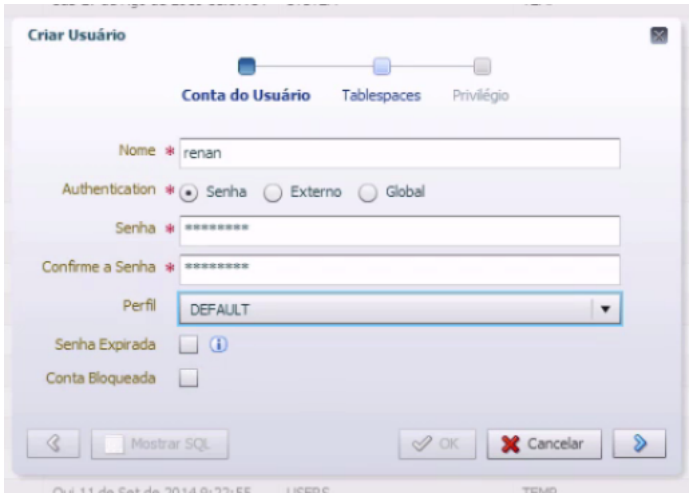
Um painel será aberto, onde poderemos ver diversas informações referentes ao nosso banco de dados.



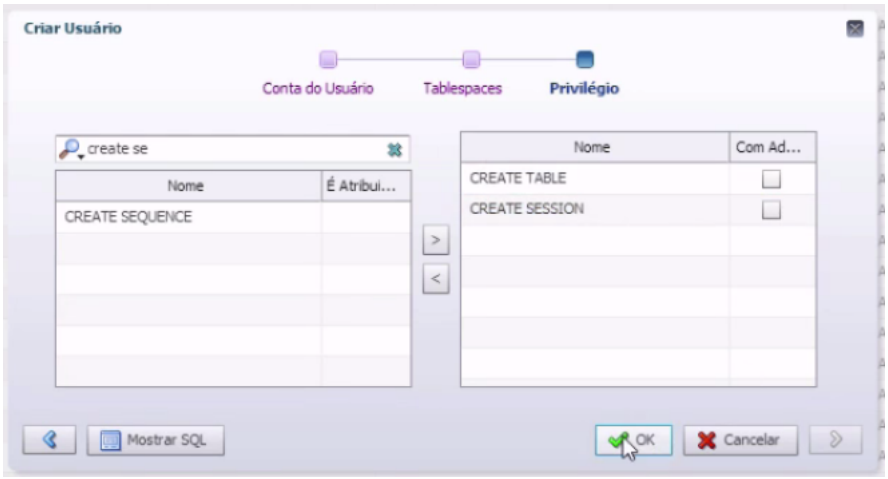

Ele consegue mostrar o desempenho, quantas sessões temos ativas, como está distribuído o armazenamento, a memória, o uso da CPU e outros. Será neste painel que nós criaremos usuários novos para o nosso banco.

Na aba "Segurança", clicaremos em "Usuários". Irá surgir uma tela com diversos usuários e se formos observar quais estão ativos, só encontraremos o `SYS` e o `SYSTEM` - criados durante a instalação. Vamos criar um usuário novo.

Criarei um usuário com o meu nome, "Renan", em seguida, vou definir uma senha, o perfil vou deixar como `DEFAULT`. Como queremos começar a utilizar imediatamente, não vou selecionar nenhuma opção abaixo dos campos preenchidos.



Iremos avançar para o próximo passo e ele irá perguntar o que é o "Tablespace". Trata-se de uma divisão lógica do nosso banco de dados. Então, ele se refere a quais tabelas eu quero enxergar. Neste caso, vou deixar as seleções padrões: "Tablespace Padrão" como "USERS" e "Tablespace Temporario" como "TEMP". Seguiremos para o passo "Privilégios". Quando logarmos com o usuário, queremos poder criar elementos. Por isso, iremos adicionar "CREATE TABLE", para poder criar uma tabela. Além disso, precisaremos logar, por isso, vamos adicionar também "CREATE SESSION" (criar uma sessão).




Ele vai criar o meu usuário. Como iremos acessá-lo? Abriremos o CMD e iremos executar o SQL *\Plus, o nosso programa da linha de comando para conversar com o banco. Ele irá pedir o usuário e senha, após preenchê-los, ele irá conectar.

```
Enter user-name: renan
Enter password:

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 . 64bit Production With the Partitioning,

SQL>
```

Observe que o cursor está marcando `SQL` . Quais tabelas será que temos dentro? Para saber quais tabelas temos, podemos usar o comando `SELECT` para selecionar o nome das tabelas do usuário.

```
SQL> select table_name from user_tables;
```

Mas, ao executarmos o comando, ele dirá que não encontrou nenhuma linha.

```
SQL> select table_name from user_tables;

no rows selected

SQL>
```

Então, iremos começar a criar uma tabela, com o comando `create table` , que se chamará `usuario` .

```
SQL> create table usuario
  2
```

No entanto, ele não irá executar o meu comando. Por quê? Repare, como não tinha um `;` no fim da linha, quando eu dei um "Enter", ele permitiu que continuássemos digitando. Isto acontece, porque o SQL Buffer salva tudo que foi digitado. Com ele, conseguimos ir digitando aos poucos e ir manipulando os dados que eu tenho dentro deste *buffer*. Por exemplo, para ver o que está dentro do *buffer*, eu posso executar o comando `list`.

```
SQL> list
1* create table usario
SQL>
```

Ele irá responder: você tem uma linha no *buffer*, que está escrito `create table usario`. No nosso caso, temos um erro de digitação. Eu quero corrigir o texto e escrever corretamente `usuario`. O que posso fazer? Podemos usar o comando `change` e fazer um *replace* em coisas que tenho no *buffer*. Primeiro, vou escrever entre barras o que texto que quero alterar e depois escrever a alteração.

```
SQL> list
1* create table usario
SQL> change \usario\usuario
```

Quando eu der "Enter", ele já irá mostrar o *buffer* atualizado.

```
SQL> change \usario\usuario
  1* create table usuario
SQL>
```

Porém, gostaríamos de acrescentar um parênteses na primeira linha. Para isto, vou usar o comando `append`.

```
SQL> append (
  1* create table usuario(
SQL>
```

Estes dois comandos têm abreviações. O comando `change` pode ser substituído por `c`.

```
SQL> c \usario\usuario
```

Será o mesmo comando utilizado anteriormente. Agora, eu quero adicionar uma linha. Para isto, usaremos outro comando, o `input`. Podemos usar também a abreviação: `i`.

```
SQL> input id int not null primary key,
SQL>
```

Para verificarmos se ele adicionou, podemos usar o comando `list`, que pode ser abreviado com `l`. E ele irá mostrar o *buffer* com as alterações.

```
SQL> input id int not null primary key,
SQL> list
  1  create table usuario(
```

```
  2* id int not null primary key,
SQL> l
  1  create table usuario(
  2* id int not null primary key,
SQL>
```

Mas eu não quero precisar dar `input` e depois, digitar tudo novamente. Se executamos um `input` sem um parâmetro, ele irá voltar para o *buffer* para continuarmos digitando. Como eu quero que seja texto, vou digitar `varchar2` , de tamanho `30` e não quero que seja `null` .

```
SQL> input
  3 varchar2(30) not null
```

Se usarmos o comando `l` (de `list` ), ele irá mostrar o `create table` .

```
SQL> input
  3 varchar2(30) not null
  4
SQL> 1
  1 create table usuario(
  2 id int not null primary key,
  3* nome varchar2(30) not null)
SQL>
```

Caso eu tenha errado,e por exemplo, queira alterar o tamanho do texto para 50? Como faremos para deletar uma linha? Usaremos o comando `del` e especificaremos qual linha.

```
SQL> del 3
SQL>
```

Se der um `list` novamente, ele terá deletado a linha 3.

```
SQL> 1ist
  1 create table usuario(
  2 id int not null primary key,
SQL>
```

Iremos escrever de novo a linha 3. Vamos usar novamente o `input` .

```
SQL> input
  3 nome varchar2(50) not null)
  4
SQL>
```

Se dermos um `l` , ele irá atualizar o meu *buffer*.

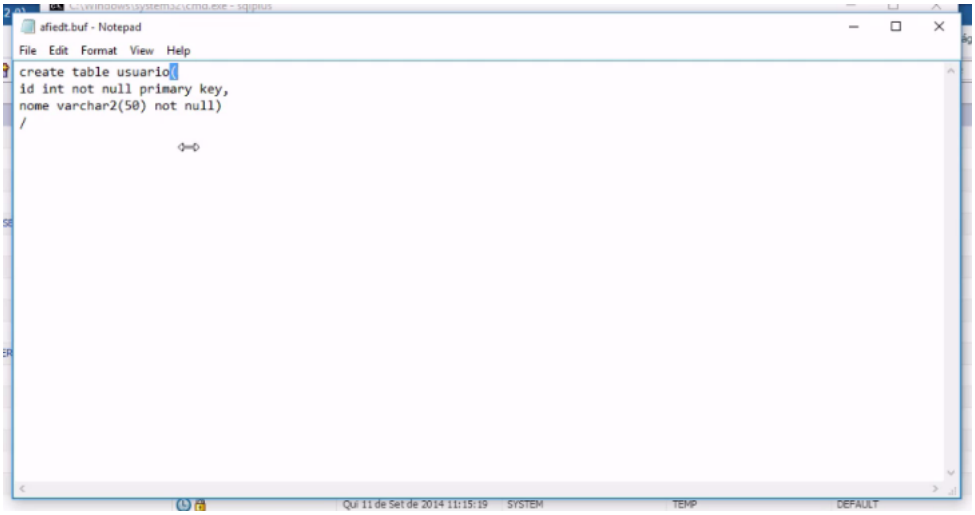
```
SQL> input
  3 nome varchar2(50) not null)
```

```
  4
SQL> l
  1 create table usuario(
  2 id int not null primary key,
  3* nome varchar2(50) not null)
SQL>
```

Agora, que já escrevemos o `create table`, não quero precisar escrevê-lo todas as vezes. Eu quero salvar esta consulta no arquivo. Então, com o comando `save`, iremos adicionar o nome do arquivo `cria-usuario`. Se executarmos, ele irá avisar que o arquivo foi criado.

```
SQL> save cria-usuario
Created file cria-usuario.sql
SQL>
```

Se quiser editar o arquivo, posso usar o comando `edit`. Ele irá abrir uma nova janela com com o arquivo `notepad`.




Nós poderíamos editar o conteúdo. Quero que o meu texto seja `varchet2(30)` e meu `id int` seja tamanho `11`.

```
create table usuario(
id int(11) not null primary key,
nome varchet2(30) not null)
/
```

Após salvar o arquivo, vou salvá-lo. De volta ao arquivo `.exe`, se dermos um `list`, veremos que ele não atualizou, porque nós salvamos, mas não carregamos o conteúdo do arquivo. Para isto, precisaremos limpar o *buffer* antes com o comando `clear buffer`.

```
SQL> clear buffer
buffer cleared
```

Ele irá limpá-lo. Podemos conferir isto com o comando `l`.

```
SQL> l
SP2-0223: NO lines in SQL buffer
```

```
SQL>
```

Ele irá responder que o *buffer* não tem nenhuma linha. Para carregar o arquivo, executaremos o comando `get`, juntamente com o arquivo criado anteriormente `cria-usuario`. Se dermos um "Enter", ele terá carregado no *buffer*.

Se quiser editar de novo o arquivo, usarei o comando `edit`. Ele irá abrir novamente o arquivo `afiedit.buf`. Nós iremos alterar o tamanho do `varcher2` para 40.

```
create table usuario(
id int not null primary key,
nome varchet2(40) not null
```

No arquivo CDM, vamos usar o comando `save` para o arquivo `cria-usuario`. E ele irá concluir o salvamento, porque já existe um arquivo com este nome. Para que ele salve a nova versão, eu preciso usar o `replace`.

```
SQL> save cria-usuario replace
```

Ao executarmos, ele dirá que escreveu o arquivo.

```
SQL> save cria-usuario replace
Wrote file cria-usuario.sql
SQL>
```

E se quisermos executar um arquivo? Eu quero limpar o meu *buffer*, poderemos usar o comando simplificado de `clear buffer` é `cl buff`.

```
SQL> cl buff
buffer cleared
SQL>
```

O buffer está limpo. Agora queremos executar o arquivo `cria-usuario`.

```
SQL> start cria-usuario

Table created.

SQL>
```

A tabela foi criada. Se verificarmos o *buffer* com `list`, veremos o conteúdo que foi executado.

```
SQL> l
  1  create table usuario(
  2  id int not null primary key,
  3* nome varchar2(40) not null)
SQL>
```

Para ver se a tabela foi realmente criada, usaremos um `select`.

```
SQL> select table_name from user_tables;

TABLE_NAME
------------------------------------------------------------
USUARIO

SQL>
```

Se quisermos ver a estrutura da tabela, usaremos o comando `desc`, ou `describe`, porque nós queremos descrever a tabela `usuario`.

```
SQL> describe usuario
 Name                                      Null?    Type
 ----------------------------------------------------------------
 ID                                        NOT NULL NUMBER(38)
 NOME                                      NOT NULL VARCHAR2(40)

SQL>
```

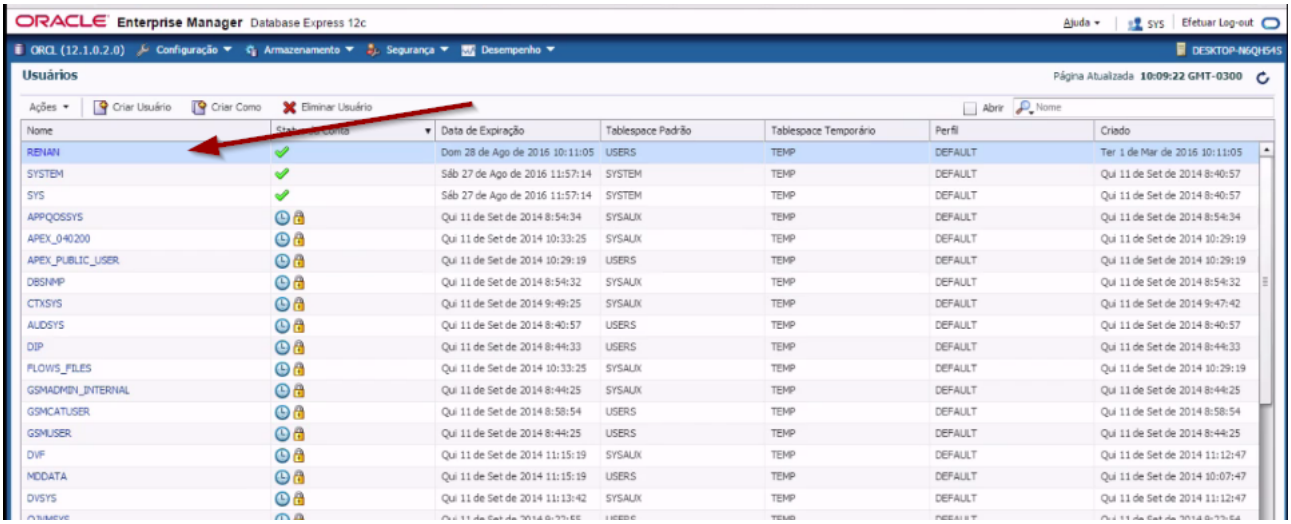
No Oracle as tabelas são *case sensitives*, assim como os comandos e os nomes de colunas. Podemos escrever em maiúscula ou minúscula.

E se precisarmos inserir dados no usuário? Usaremos o comando `insert into usuario`, e nas coluna `id` e o `nome`, queremos inserir o valores (`values`) `1` e `renan`.

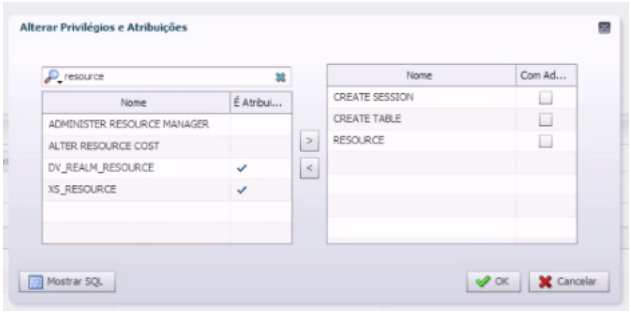
```
SQL> insert into usuario (id,nome) values (1, 'renan');
```

Ao executarmos o comando, ele irá dizer que eu não tenho privilégio na tablespace 'USERS'. Por quê? Porque quando criamos o nosso usuário, nós não especificamos que ele poderia inserir dados. Dentro do Oracle, temos um sistema de cotas, e o meu usuário não tem cota para inserir dados dentro do meu banco. Então, teremos que adicionar isto.

No painel de usuário, vou clicar em `RENAN`.



Uma janela com detalhes do usuário será aberta e iremos editar a seção "Privilégio". Nós iremos adicionar o privilégio `RESOURCE`. Agora, poderemos inserir dados no `table space`.




Vamos executar o comando novamente. Ele precisará carregar novamente os meus privilégios.

Vou usar o comando `select`.

```
SQL> select table_name from user_tables;
```

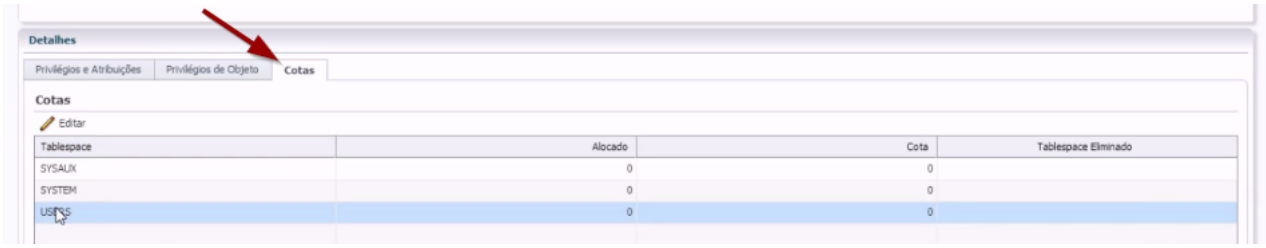
Ele irá me mostrar a tabela usuários.

```
SQL> select table_name from user_tables;

TABLE_NAME
------------------------------------------------------------------------
USUARIO

SQL>
```

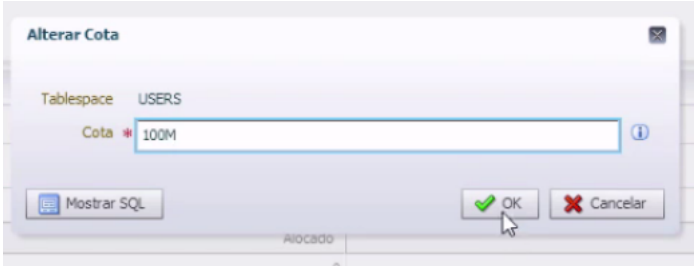
E se quisermos inserir dados na tabela, o que precisaremos fazer? Usaremos o comando `insert`, quero inserir o `id` e o `nome`, com os valores `1` e `renan`.

```
SQL> insert into usuario (id, nome) values(1, 'renan')
```

Mas para este comando funcionar, teremos que voltar ao painel de usuário e na seção "Detalhes", definir uma cota.




Vou especificar que no *table espace* "USERS", este usuário poderá inserir até 100M de dados.




De volta ao SQL *\Plus, ao executarmos o comando `insert`, uma nova linha será criada.

```
SQL> insert into usuario (id,nome) values(1, 'renan');

1 row created.

SQL>
```

Com o comando `select` podemos verificar se o usuário `renan` foi adicionado.

```
SQL> select nome from usuario;

NOME
-----------------
renan

SQL>
```

Então, nós conhecemos um pouco da ferramenta para trabalharmos com o banco de dados Oracle. Tanto o Enterprise Manager Express, usado para gerenciar os nossos usuários, quanto o SQL *\Plus, utilizado para verificar, inserir e manipular os dados das nossas tabelas.

No próximo curso, nos focaremos ainda mais em SQL e veremos como selecionar e inserir dados.

Até aqui, já começamos a estudar o que é uma entidade, um relacionamento, uma chave-primária, uma chave-estrangeira. Nós aprendemos a modelar os dados, conhecemos as novidades do Oracle 12c, incluindo arquiteturas que ele implementa. Vimos que podemos ter uma ou mais instâncias rodando em uma máquina, assim como vários bancos de dados rodando em uma instância. E também, conhecemos um pouco das ferramentas para trabalharmos com bancos de dados.

Aguardo vocês no próximo curso da certificação. Até logo!