

# Sistema Gerenciador de Banco de Dados Oracle

## O que é um Banco de Dados?

### Banco de dados definido

Um banco de dados é uma coleção organizada de informações - ou dados - estruturadas, normalmente armazenadas eletronicamente em um sistema de computador. Um banco de dados é geralmente controlado por um [sistema de gerenciamento de banco de dados \(DBMS\)](#). Juntos, os dados e o DBMS, juntamente com os aplicativos associados a eles, são chamados de sistema de banco de dados, geralmente abreviados para apenas banco de dados.

Os dados nos tipos mais comuns de bancos de dados em operação atualmente são modelados em linhas e colunas em uma série de tabelas para tornar o processamento e a consulta de dados eficientes. Os dados podem ser facilmente acessados, gerenciados, modificados, atualizados, controlados e organizados. A maioria dos bancos de dados usa a linguagem de consulta estruturada (SQL) para escrever e consultar dados.

## O que é SQL (Structured Query Language, Linguagem de consulta estruturada)?

SQL é uma linguagem de programação usada por quase todos [os bancos de dados relacionais](#) para consultar, manipular e definir dados e fornecer controle de acesso. O SQL foi desenvolvido pela primeira vez na IBM nos anos 1970, com a Oracle como principal contribuinte, o que levou à implementação do padrão SQL ANSI; o SQL estimulou muitas extensões de empresas como IBM, Oracle e Microsoft. Embora o SQL ainda seja amplamente usado hoje em dia, novas linguagens de programação estão começando a aparecer.

## Evolução do banco de dados

Os bancos de dados evoluíram muito desde a sua criação no início dos anos 1960. Bancos de dados de navegação, como o banco de dados hierárquico (que se baseava em um modelo de árvore e permitia apenas um relacionamento um-para-muitos), e o banco de dados de rede (um modelo mais flexível que permitia múltiplos relacionamentos) eram os sistemas originais usados para armazenar e manipular dados. Embora simples, esses primeiros sistemas eram inflexíveis. Nos anos 1980, [bancos de dados relacionais](#) tornaram-se populares, seguidos por [bancos de dados orientados a objetos](#) na década de 1990. Mais recentemente, [bancos de dados NoSQL](#) surgiram como uma resposta ao crescimento da internet e à necessidade de maior velocidade e processamento de dados não estruturados. Hoje, [bancos de dados na nuvem](#) e [bancos de dados autônomos](#) estão

abrindo novos caminhos quando se trata de como os dados são coletados, armazenados, gerenciados e utilizados.

## Qual é a diferença entre um banco de dados e uma planilha?

Bancos de dados e planilhas (como o Microsoft Excel) são modos convenientes de armazenar informações. As principais diferenças entre os dois são:

- Como os dados são armazenados e manipulados

- Quem pode acessar os dados

- Quantos dados podem ser armazenados

As planilhas foram originalmente projetadas para um usuário e suas características refletem isso. São ótimos para um único usuário ou um pequeno número de usuários que não precisam fazer a manipulação de dados muito complicada. Bancos de dados, por outro lado, são projetados para conter coleções muito maiores de informações organizadas - quantidades enormes, às vezes. Os bancos de dados permitem que vários usuários, ao mesmo tempo, acessem e consultem com rapidez e segurança os dados usando lógica e linguagem altamente complexas.

## Introdução Ao Oracle

O Oracle é um dos SGBDs mais utilizados em aplicações corporativas. Robusto, confiável e seguro, a qualidade dessa solução justifica o investimento feito para poder explorar os recursos do produto. Neste guia você encontrará vários artigos, vídeos e cursos que lhe permitirão dominar esse banco de dados, começando pelo curso completo:

O Banco de dados Oracle (Oracle DB) é um sistema de gerenciamento de banco de dados relacional (RDBMS, Relational Database Management System) da Oracle Corporation.

Hoje, mais do que nunca, as empresas exigem bancos de dados de alto desempenho e escaláveis. Muitas aproveitam o Oracle DB para orientar aplicativos corporativos no processamento de transações online (OLTP, Online Transaction Processing), em data warehouse e na análise de negócios. As equipes de TI também precisam de desempenho sob demanda desses bancos de dados para as necessidades de desenvolvimento, teste, análise e continuidade de negócios.

## Oracle Corporation e Banco de dados Oracle

Em 1977, a empresa Software Development Laboratories (SDL) foi formada por Larry Ellison, Bob Miner, Ed Oates e Bruce Scott. Larry e Bob vieram da Ampex, onde estavam trabalhando em um projeto da CIA apelidado de “Oracle”. A CIA foi seu primeiro cliente, embora o produto ainda não tivesse sido

lançado comercialmente. A SDL mudou seu nome para Relational Software Inc. (RSI) no ano de 1978 e, em 1979, colocou a primeira versão comercial do software, que foi vendida à base da Força Aérea em Wright-Patterson. Esse foi o primeiro RDBMS comercial no mercado. Depois disso, em 1982, a RSI mudou seu nome para Oracle Systems Corporation (OSC), que posteriormente foi simplificado para Oracle Corporation. Em poucos anos, a Oracle se tornou líder em bancos de dados relacionais.

Uma grande característica deste SGDBR é a integração no mercado da tecnologia da informação, acompanhando as novidades que vêm surgindo, como a inclusão de multimídia, CAD/CAM, gráficos, textos, desenhos, fotografias, som e imagem. Além disso, contempla desenvolvimentos voltados à programação orientada a objetos e banco de dados semânticos. Implementa a linguagem SQL, inclusive incorporando as modificações realizadas no padrão SQL3, a nova versão aprovada pela ANSI, como também sua linguagem própria, voltada a procedimentos, PL/SQL. Seu teor é constituído de vários componentes, como gerador de aplicação, dicionários de dados, gerador de relatório, planilhas eletrônicas, pré-compiladores para linguagens hospedeiras, utilitários, ferramentas de modelagem, gráficos, cálculos estatísticos e armazenagem de dados de vários tipos (data, valores numéricos com e sem decimais, caracteres variáveis ou fixos, binários, imagens, textos, sons).

O SGDBR da Oracle pode ser usado em vários tipos de equipamentos, sendo eles computadores de pequeno e grande porte. Quanto à plataforma, dá suporte a uma gama de ambientes incluindo, UNIX, VMS, MVS, HP, Macintosh, Rede Novell, MS-Windows entre outros. Possui compatibilidade com outros SGDBRs, tais como, DB/2 e SQL/DS da IBM, Sybase, Ingres, Informix, e outros não relacionais como SAS, Lipper, Lotus etc. Outras características relacionais são:

- Controle sofisticado de concorrências;
- Suporte total para visões e junções;
- Consistência garantida na leitura de dados;
- Otimização de consultas avançadas baseadas em custo;
- Transações de multietapas garantidas contra falhas;
- Recuperação automática contra quebras ou falhas de computador;
- Indexação dinâmica;
- Pesquisas distribuídas;
- Junções externas (outer joins inexistentes em outros SGBDRDs);
- Funções lógicas de textos;
- Usuários e SCHEMAS;
- Índices, clusters e hash clusters;
- Sequências;
- Procedimentos, pacotes e triggers;
- Sinônimos, papéis e privilégios;
- Segmentos de Rollback;
- Snapshots e visões materializadas.

No que diz respeito à segurança, o banco de dados Oracle não deixa a desejar, pois o alto grau de desempenho, portabilidade e interoperabilidade o transforma em líder mundial em seu segmento. A integridade e consistência permitem que usuários construam sistemas seguros, confiáveis e com alta dis-

ponibilidade de recursos. Algumas características relacionadas à segurança são:

- Suporte multinível;
- Integridade de dados;
- Disponibilidade e tolerância a falhas;
- Administração de privilégios;
- Administração da segurança;
- Alto desempenho;
- Suporte a migração;
- Controle de concorrência;
- Suporte de segurança multinível a BD distribuído;
- Suporte a padrões internacionais.

Atualmente, a versão do banco de dados encontra-se na 11g, sendo a 9i e 10g as mais difundidas no mercado. A Oracle também disponibiliza outros produtos, que têm como objetivo auxiliar os usuários nas mais diversas áreas de negócio.

Comunicando com o banco de dados Oracle

A ferramenta SQL\*Plus acompanha o banco de dados Oracle quando ele é instalado. Ela tem a responsabilidade de fazer a comunicação entre o usuário e o banco de dados. Ao ser acionada, é apresentada uma tela de identificação onde devem ser informados o nome do usuário, senha e a string de conexão com o banco de dados com o qual deseja se conectar.

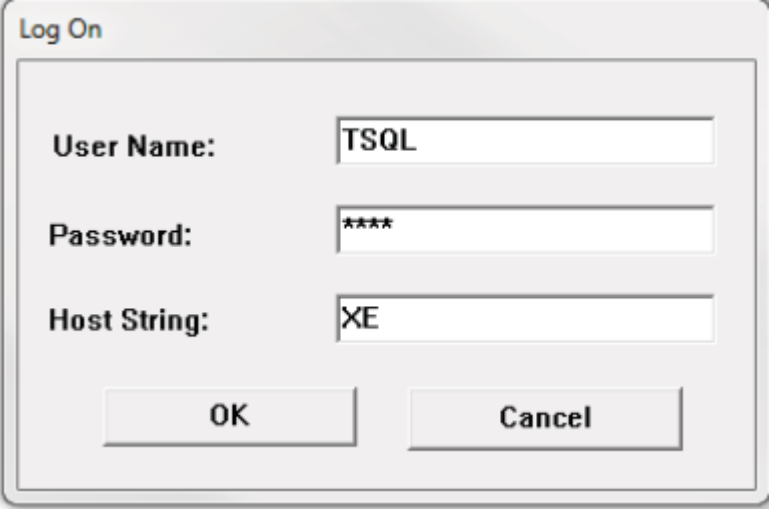


FIG Tela de identificação.

Sua finalidade geral é a execução de comandos SQL, código PL/SQL, procedimentos (procedures), gatilhos (triggers) e funções (functions). A interface desta ferramenta é semelhante ao prompt do MS-DOS, pois disponibiliza um prompt que fica esperando a entrada de uma instrução. Ela também apresenta algumas informações, como, por exemplo, a versão programa e a versão do banco de dados Oracle conectado.

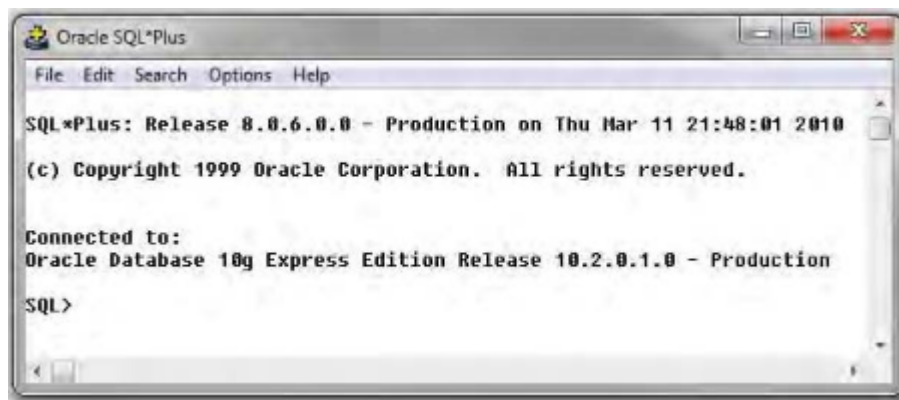


Fig. Exemplo da tela do SQL\*Plus

Como na maioria das ferramentas, existem diversas configurações de visualização de instruções e resultados, como formatação de linhas, colunas, páginas, comandos para mostrar informações sobre a execução do código, entre outras. Além disso, há uma série de comandos que são úteis para a manipulação destas instruções:

- Editar o comando SQL armazenado no buffer;
- Formatar os resultados retornados pelo banco de dados;
- Armazenar os comandos de SQL para disco e recuperá-los para execução;
- Modificar o modo de trabalhar do SQL\*Plus;
- Enviar mensagens e receber respostas de outros usuários;
- Listar a definição de qualquer tabela;
- Fazer acesso e copiar dados entre banco de dados.

Na sequência, veja um exemplo de uma instrução digitada e executada do SQL\*Plus:

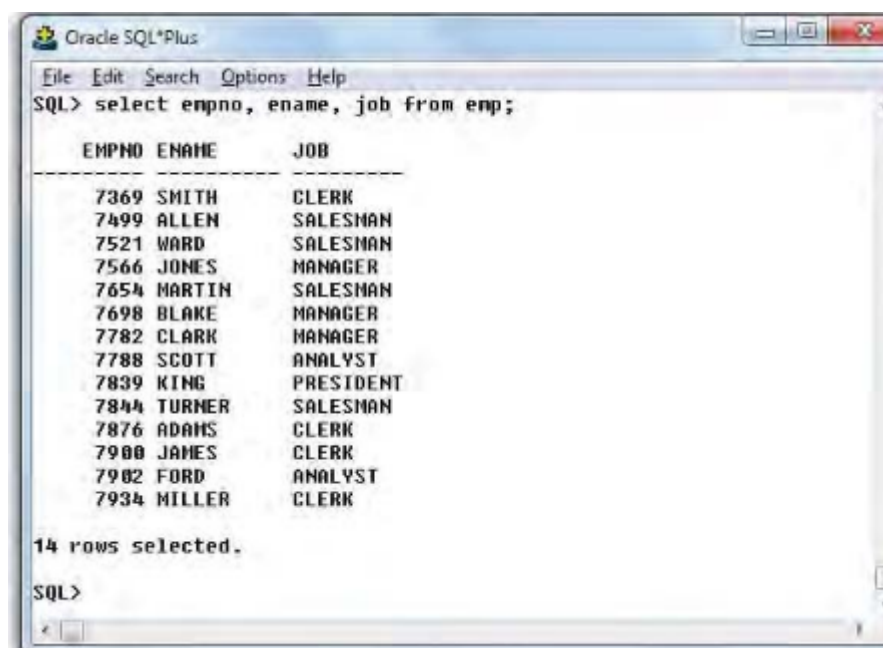


FIG Exemplo de instrução digitada no SQL\*PLUS

Existem outras ferramentas com interfaces mais amigáveis disponibilizadas pela Oracle para fazer este trabalho com mais rapidez utilizando outros recursos, como, por exemplo, a ferramenta SQL Developer. Outras ferramentas de terceiros que também possuem muitos recursos são SQLTools (Free Software), Toad (Free Software), SQL Navigator e PL/SQL Developer (necessitam de licença).

## Escopo do usuário

Para se ter acesso ao banco de dados Oracle são necessários um usuário e uma senha. Se você não estiver acessando o banco através do usuário administrador do banco de dados, que foi configurado no momento da instalação do banco, o cadastro terá obrigatoriamente que ser feito para que você tenha acesso.

Geralmente é o DBA (Database Administrator) quem cria os usuários que poderão ter acesso ao banco de dados, mas nada impede que você os crie apesar de não ser recomendado quando estamos falando de bancos de produção. Deixe esta tarefa para o DBA. Ele é o administrador do banco e não só cria os usuários como também garante o acesso aos objetos e recursos dos quais precisamos para operar o banco. O papel deste profissional é manter o banco em perfeita ordem e funcionando adequadamente.

Tendo o usuário e senha devidamente habilitados e com permissões para acesso ao banco de dados, basta escolher a ferramenta e efetuar o login.

Ao conectarmos a um banco de dados através de uma das ferramentas mencionadas anteriormente, iniciamos uma sessão dentro dele. Quando solicitamos esta conexão, o banco de dados abre uma sessão onde somente este usuário terá acesso a esta área. Agregado ao usuário também existe uma estrutura chamada de SCHEMA, onde todas suas configurações e todos os objetos que foram criados para ele estão disponíveis para acesso, ou seja, esses objetos e configurações têm um dono, que dentro do banco de dados Oracle é chamado de OWNER. É óbvio que, se você acabou de criar um usuário, muito provavelmente ele não deverá possuir nenhum objeto criado.

Se um usuário cria um objeto no banco de dados, este objeto pertence a ele. Desta forma, somente ele terá acesso a este objeto, a menos que ele mesmo dê acesso a outros usuários do banco de dados. Sim, isso pode ser feito, e é muito comum de acontecer.

Quando falamos em objetos, referimo-nos a tabelas, packages, triggers, procedures, functions etc. No caso das tabelas que contêm os dados, os usuários com acesso a estes objetos também poderão acessar os dados contidos nelas.

Em grandes sistemas corporativos vemos muito isto acontecer. Por exemplo, temos várias tabelas correspondentes a diversos módulos dentro do sistema. Temos o módulo de RH, o módulo de contas a pagar, contas a receber e assim por diante. Podemos ter criado no banco de dados um usuário para cada módulo e, através deste usuário, dar acessos a estes objetos aos outros usuários (módulos) do sistema ou, ainda, para outro usuário principal, digamos assim, o qual iniciará nosso sistema. Ou seja, podemos ter um único

usuário responsável por acessar os objetos que estão criados em outros usuários.

O SGDB Oracle é muito flexível e íntegro quando se trata da manipulação de dados e seus SCHEMAS de usuário. Outro ponto interessante é que os dados que disponibilizarmos podem ser restringidos a tal ponto que, se assim quisermos, outros usuários só poderão ter acesso de leitura, não podendo fazer qualquer alteração.

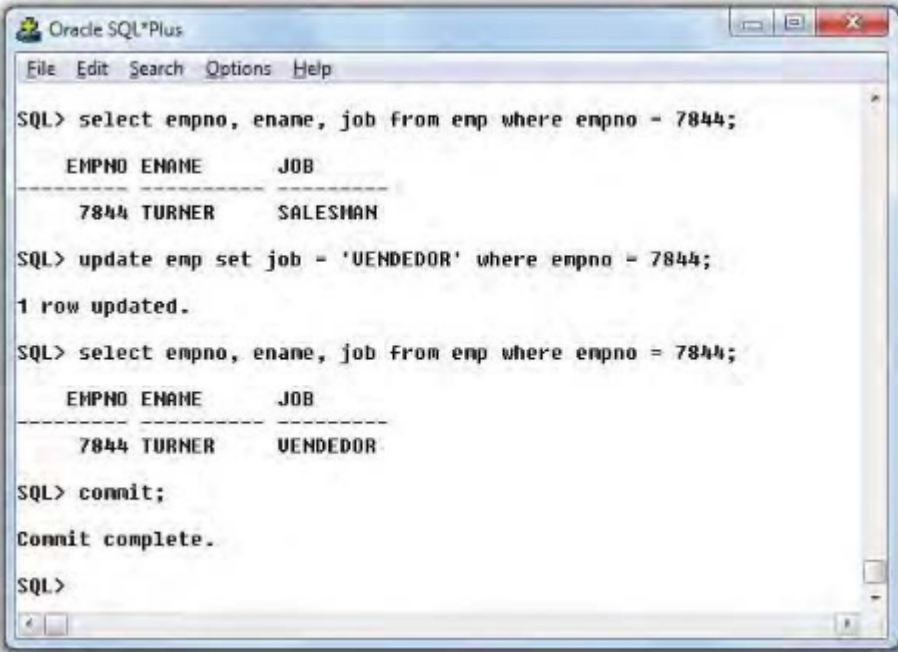
Para que esta manipulação de dados aconteça sem problemas, visando a integridade das informações e garantindo o ótimo funcionamento do banco de dados e programas conectados a ele, o banco Oracle trabalha com transações. No próximo tópico, vamos ver o que significa uma transação.

### Transações

Primeiramente, antes de começarmos a trabalhar com bancos dados Oracle, é preciso conhecer o conceito de transação. Uma das características mais bem-vindas dos bancos de dados Cliente/Servidor, em relação aos bancos de dados desktop, é o conceito de transações. Uma transação é uma unidade lógica de trabalho composta por uma ou mais declarações da Data Manipulation Language (DML) ou Data Definition Language (DDL). Os comandos para o controle da transação são os comandos necessários para que se possa controlar a efetivação ou não das modificações feitas no banco de dados. Para explicar o que é uma transação, pode-se tomar como exemplo uma transferência bancária na qual um determinado valor é transferido para outra conta. O processo de transferência deste valor consiste em vários passos, que são agrupados de modo que, se não forem concluídos em sua totalidade, ou seja, se a transação não chegar até o final, todas as outras alterações já realizadas serão descartadas e a conta voltará ao seu estado anterior, como se nenhuma transferência tivesse sido realizada. Através deste controle, pode-se garantir que os dados permanecerão consistentes. Os comandos COMMIT e ROLLBACK da DCL auxiliam neste trabalho.

## COMMIT

Torna permanentes todas as alterações feitas no banco de dados durante a sessão. Todas as alterações realizadas em uma determinada transação serão confirmadas caso este comando seja aplicado.



```
Oracle SQL*Plus
File Edit Search Options Help

SQL> select empno, ename, job from emp where empno = 7844;

  EMPNO ENAME      JOB
-----
  7844  TURNER      SALESMAN

SQL> update emp set job = 'VENDEDOR' where empno = 7844;

1 row updated.

SQL> select empno, ename, job from emp where empno = 7844;

  EMPNO ENAME      JOB
-----
  7844  TURNER      VENDEDOR

SQL> commit;

Commit complete.

SQL>
```

FIG Exemplo de commit.

Neste exemplo, é mostrada atualização do cargo referente ao empregado TURNER, e logo após, a efetivação da atualização através do comando COMMIT.

## ROLLBACK

Usado para remover todas as alterações feitas desde o último COMMIT durante a sessão. Esse comando vai restaurar os dados ao lugar onde eles estavam no último COMMIT. Alterações serão desfeitas caso a transação seja encerrada pelo comando ROLLBACK.



The screenshot shows the Oracle SQL\*Plus interface. The command prompt shows the following sequence of commands and their outputs:

```
SQL> select empno, ename, deptno from emp where deptno = 20;
```

EMPNO	ENAME	DEPTNO
7369	SMITH	20
7566	JONES	20
7788	SCOTT	20
7876	ADAMS	20
7902	FORD	20

```
SQL> delete from emp where deptno = 20;
```

5 rows deleted.

```
SQL> select empno, ename, deptno from emp where deptno = 20;
```

no rows selected

```
SQL> rollback;
```

Rollback complete.

```
SQL> select empno, ename, deptno from emp where deptno = 20;
```

EMPNO	ENAME	DEPTNO
7369	SMITH	20
7566	JONES	20
7788	SCOTT	20
7876	ADAMS	20
7902	FORD	20

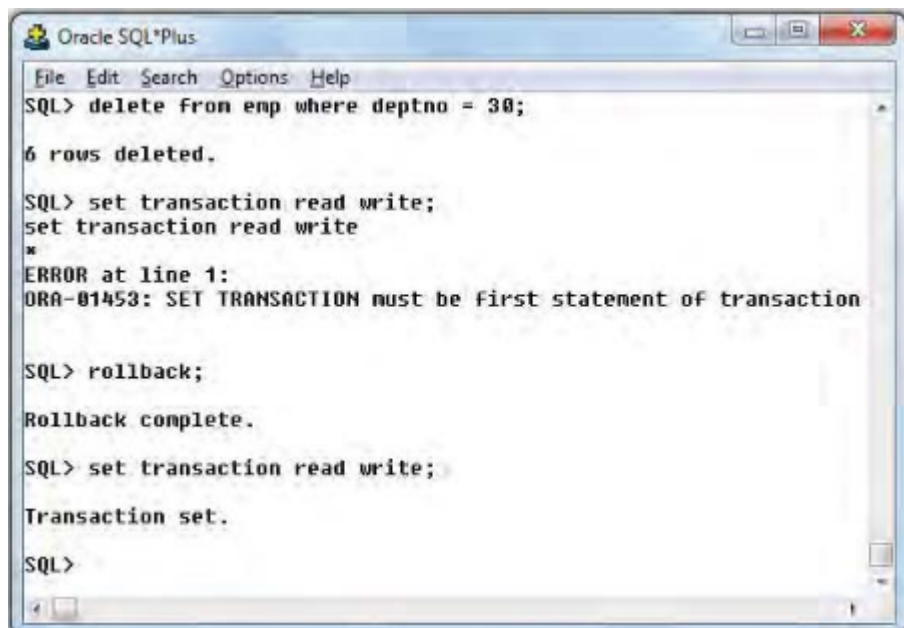
```
SQL>
```

FIG Exemplo ROLLBACK

Já neste exemplo, temos a exclusão de cinco linhas referentes aos empregados que trabalham no departamento 20. Contudo, logo após a exclusão, foi executado um ROLLBACK, anulando a ação.

No Oracle, uma transação se inicia com a execução da primeira instrução SQL e termina quando as alterações são salvas ou descartadas. O comando SET TRANSACTION também inicia uma transação – transação explícita. O uso do comando SET TRANSACTION determina algumas regras, listadas a seguir.

- Deve ser o primeiro comando da transação (caso contrário, ocorrerá um erro);
- Somente consultas são permitidas na transação;
- Um COMMIT, ROLLBACK ou qualquer outro comando de DDL (pos-suem COMMITS implícitos) encerram o efeito do comando SET TRANSACTION.



```
Oracle SQL*Plus
File Edit Search Options Help
SQL> delete from emp where deptno = 30;

6 rows deleted.

SQL> set transaction read write;
set transaction read write
*
ERROR at line 1:
ORA-01453: SET TRANSACTION must be first statement of transaction

SQL> rollback;

Rollback complete.

SQL> set transaction read write;

Transaction set.

SQL>
```

### Dicionário de dados do Oracle

O banco de dados Oracle é muito robusto e sua gama de parametrizações, estruturas e arquiteturas requer uma organização para que tudo possa ocorrer de forma harmônica, e para que tanto o banco de dados quanto nós que trabalhamos com ele possamos achar as informações quando necessitamos. Para que isso aconteça, todas as informações sobre os objetos estão gravadas em tabelas que formam o chamado dicionário de dados. Este dicionário é fundamental para que o banco de dados possa operar de forma adequada. A estrutura do dicionário é bastante complexa, por isso existem visões do dicionário com nomes e estruturas mais legíveis para que possamos utilizá-las para recuperar informações pertinentes ao banco de dados. Existem centenas dessas visões. Com toda certeza, trata-se de um SGBD muito completo e estruturado de forma a se adequar aos mais diferentes tipos de negócio.

### Cuidado!

As informações no dicionário de dados são mantidas por comandos SQL. Por exemplo, CREATE TABLE EMPLOYEES atualiza as tabelas do dicionário, de modo que a visão USER\_TABLES apresente as informações sobre a nova tabela. Nunca insira ou altere diretamente as tabelas do dicionário: isto invalida o banco de dados, corrompendo totalmente a base de dados. Obviamente não é qualquer usuário que pode alterar o dicionário mas, se você for um DBA, terá acesso total.

Veja algumas das visões mais usadas pelo usuário:

- DICTONARY: mostra todas as visões que compõem o dicionário de dados;
- USER\_TABLES: mostra as tabelas das quais o usuário é dono;
- ALL\_TABLES: mostra as tabelas às quais o usuário tem acesso;
- USER\_TAB\_COLUMNS: mostra todas as colunas definidas nas tabelas das quais o usuário é dono;

- ALL\_TAB\_COLUMNS: mostra todas as colunas definidas nas tabelas às quais o usuário tem acesso;
- ALL\_OBJECTS / ALL\_SOURCE: mostra todos os objetos aos quais o usuário tem acesso. Alguns deles são: TABLES, VIEWS, PACKAGES, PROCEDURES, FUNCTIONS, CLUSTERS etc. Os tipos de objeto podem mudar conforme a versão do banco.

Além destas, ainda temos visões com informações sobre índices (tabelas e colunas), VIEWS, CONSTRAINTS, TABLESPACES etc.



FIG Neste exemplo, temos a execução da view USER\_TABLES, que mostra todas as tabelas pertencentes ao usuário logado.

## Como o Oracle executa comandos SQL

Existem várias formas de se executar declarações DML. O Oracle, através do Optimizer, escolhe a forma mais eficaz para fazer isto. O Optimizer é o programa otimizador do banco de dados Oracle, que é responsável pela melhor escolha possível para executar um comando SQL. Mas não vamos nos ater aqui a explorar suas funcionalidades e capacidades (que são muitas, diga-se de passagem). O que é interessante entender são os passos da execução de um comando SQL.

Quando se executa um comando SQL em um banco de dados, três etapas são executadas, sendo elas: PARSE, EXECUTE e FETCH. Em cada etapa, o Oracle define tarefas que devem ser seguidas para que o resultado seja obtido.

### PARSE

A primeira tarefa desta etapa é verificar a sintaxe do comando e validar a semântica.

Depois, é a vez de pesquisar se este comando já foi executado anteriormente, encontrando-se na memória. Se existir, o plano de execução já estará

traçado, tendo em vista que o comando já foi analisado e executado antes. Se for constatada a existência, será passado direto para a fase EXECUTE, não repetindo as outras ações.

Caso não seja encontrado na memória, o Oracle verifica no dicionário de dados a existência das tabelas e colunas envolvidas no comando, bem como as permissões de acesso a estes objetos.

Após a pesquisa, o otimizador poderá traçar o caminho de acesso para cada tabela presente no comando, montando um plano de execução (também chamado de PARSE TREE) para obter e/ou atualizar os dados referentes à requisição feita através deste comando.

Tendo o plano montado, o Oracle o inclui em memória para que a execução seja realizada e para que este plano seja utilizado por outro comando idêntico ao atual.

## EXECUTE

A primeira tarefa do EXECUTE é aplicar o plano de execução. Com isto, a leitura dos dados será feita de forma física ou lógica, dependendo da sua localização em disco ou em memória. Se estiver em memória, não será necessário fazer qualquer acesso a disco. Caso contrário, a leitura é feita, e o retorno é armazenado na área de memória do banco de dados.

Já com os dados em memória, será realizada a análise das restrições que consiste em verificar as regras de integridade ( CONSTRAINTS). Esta tarefa é aplicada a processos de atualização.

Como a anterior, esta tarefa também só é aplicada em processos de atualização, sendo que é preciso bloquear as linhas de dados ( LOCKED), alocar o segmento de ROLLBACK, para que a atualização das linhas seja realizada.

## FETCH

Se for o caso em que o comando SQL executado for um SELECT, o Oracle retorna as informações.

Executando comandos SQL com

SQL\*Plus

### 4.1

O que é SQL\*Plus?

É uma ferramenta disponibilizada pela Oracle que funciona como um prompt de comando para execução de comandos SQL e PL/SQL. Suas principais funcionalidades são:

- Processamento online de comandos SQL que:
  - Definem a estrutura de base de dados;
  - Realizam consultas ad-hoc à base de dados;
  - Manipulam as informações armazenadas na base de dados.

Formatação de linhas e colunas recuperadas por uma consulta SQL.

- Edição de comandos SQL através de um editor de linhas built-in ou de seu editor de textos favorito.
- Armazenamento, recuperação e execução de comandos SQL em/de arquivos.

Na sequência, veremos comandos do SQL\*Plus que são utilizados para a manipulação de comandos SQL nesta ferramenta.

# Comandos de edição do SQL\*Plus

Os comandos de edição têm o objetivo de modificar a informação armazenada no SQL Buffer. Estes comandos afetam apenas uma linha de cada vez, isto é, a edição é feita linha a linha. Fica guardado no buffer sempre o último comando SQL executado. Os comandos de edição não ficam no Buffer. Quando listamos o conteúdo do SQL Buffer, o SQL\*Plus mostra do lado esquerdo do texto (ao lado da numeração) um asterisco em uma das linhas, indicando que se trata da linha corrente, ou seja, da linha apta para modificação. Desta forma, qualquer comando de edição que viermos a fazer afetará apenas a linha corrente.

## L-LIST

Este comando tem a finalidade de listar uma ou mais linhas do SQL Buffer. A última linha listada pelo comando se tornará a linha corrente.

```
SQL> list
select * from DEPT
SQL> l
select * from DEPT
SQL>
```

## A-PPEND

Com este comando, podemos adicionar um trecho de texto ao fim da linha corrente.

```
SQL> a where rownum < 10
select * from DEPT where rownum < 10
```

## C-HANGE

Este comando de edição tem o objetivo de substituir parte do texto (ou todo) por outro.

Observe que os separadores apresentados são iguais. Podemos utilizar qualquer caractere especial não presente nos textos, porém, dentro de um comando, só podemos usar um deles.

```
SQL> l
select * from DEPT where rownum < 10
SQL> c/10/20
select * from DEPT where rownum < 20
```

## DEL

Exclui uma determinada linha da área do SQL Buffer. Caso não sejam informados parâmetros que indiquem a linha a ser removida, será excluída a linha corrente.

```
SQL> l
select *
from DEPT
where rownum < 20
SQL> del 3
SQL> l
select *
from DEPT
SQL>
```

## I-INPUT

Este comando adiciona uma ou mais linhas após a linha corrente (no SQL Buffer). Este comando difere do comando Append uma vez que pode voltar ao estado de digitação, abrindo uma nova linha para digitação imediatamente após a linha corrente.

```
SQL> I
select *
SQL>i from DEPT
SQL> I
select *
from DEPT
SQL>
```

## ED-IT

Este comando aciona um editor registrado no Windows e passa como parâmetro o nome de um arquivo ou o texto presente no SQL Buffer, de acordo com o comando executado.

Quando omitimos o nome do arquivo a ser editado, o SQL\*Plus aciona o editor do sistema passando como parâmetro o texto do SQL Buffer.

Quando desejamos editar um determinado arquivo, seu nome pode ser informado com ou sem a extensão. A extensão default é SQL.

## R-UN

Este comando envia o conteúdo do SQL Buffer para o banco de dados e, ao mesmo tempo, apresenta no vídeo as linhas enviadas (lista o SQL Buffer).

```
SQL> r
select ENAME from EMP where rownum < 3
ENAME
```

```
-----
SMITH
ALLEN
/ (BARRA)
```

Quando digitamos uma barra na linha de prompt e em seguida tecamos Enter, o SQL\*Plus envia o conteúdo do SQL Buffer para o banco de dados, porém não apresenta o texto enviado, isto é, não lista o SQL Buffer. Esta é a diferença entre o comando Run e a barra.

```
SQL> /
ENAME
```

```
-----
SMITH
ALLEN
EXIT / QUIT
```

Os comandos EXIT e QUIT são equivalentes e têm a finalidade de encerrar a sessão do SQL\*Plus. Isto significa que a conexão com o banco de dados e, simultaneamente, o próprio programa serão encerrados.

Quando digitamos apenas QUIT (ou apenas EXIT), o término do programa é considerado normal e a conexão se encerra também normalmente com COMMIT.

## DESC-RIBE

Este comando tem a finalidade de apresentar a definição de um objeto

criado na base de dados Oracle. O objeto pode ser uma TABLE, VIEW, SYNONYM, FUNCTION, PROCEDURE ou PACKAGE.

```
SQL> DESC BONUS
```

Name

Null?

Type

-----

ENAME

VARCHAR2(10)

JOB

VARCHAR2(9)

SAL

NUMBER

COMM

NUMBER

```
SQL>
```

SAVE

Este comando salva o conteúdo do SQL Buffer em um arquivo do sistema operacional.

```
SQL> !
```

```
select ENAME from EMP where rownum < 3
```

```
SQL> save EMPREGADOS.sql
```

Created file EMPREGADOS.sql

```
SQL>
```

GET

A recuperação de um texto para o SQL Buffer é feita por meio do comando GET. Todo o conteúdo do arquivo é copiado para o SQL Buffer, portanto, este arquivo deve conter apenas um comando.

```
SQL> get EMPREGADOS.sql
```

```
select ENAME from EMP where rownum < 3
```

```
SQL>
```

STAR-T / @

Este comando executa o conteúdo de um arquivo de comandos existente no sistema operacional. Cada comando de SQL ou de SQL\*Plus é lido e tratado individualmente. Em um arquivo executado por Start, podemos incluir diversos comandos de SQL.

O comando @ é sinônimo de Start e o @@ é semelhante, com a diferença de que, se este comando for incluído em um arquivo de comandos, ele pesquisará o arquivo associado no diretório do arquivo executado e não no diretório local, como seria o caso do @ e do Start.

```
SQL> star EMPREGADOS.sql
```

ENAME

-----

SMITH

ALLEN

```
SQL>
```

HOST

Abre um prompt no DOS, ou SHELL no UNIX, sem fechar o SQL\*Plus.

Para retornar, basta digitar EXIT.

## SPOOL

SPOOL {NOME\_ARQUIVO|OFF|OUT}

Quando queremos armazenar o resultado de uma consulta utilizamos este comando. O resultado é armazenado em um arquivo do sistema operacional que pode, opcionalmente, ser enviado para a impressora padrão, configurada neste sistema operacional. NOME\_ARQUIVO indica o nome do lugar onde os resultados serão gravados. Ele é gravado com a extensão LST, entretanto, pode ser gravado com outra extensão, como por exemplo, TXT. Esta mudança pode ocorrer de acordo com o sistema operacional. OFF Interrompe a gravação (geração) do arquivo. OUT interrompe a geração do arquivo e o envia para impressora.

```
SQL> spool c:\temp\arquivo1.txt
```

```
SQL> select ename, job, sal from emp where comm is not null;
```

```
ENAME
```

```
JOB
```

```
SAL
```

```
-----
```

```
ALLEN
```

```
SALESMAN
```

```
1600
```

```
WARD
```

```
SALESMAN
```

```
1250
```

```
MARTIN
```

```
SALESMAN
```

```
1250
```

```
TURNER
```

```
VENDEDOR
```

```
1500
```

```
SQL> spool off;
```

```
SQL> edit c:\temp\arquivo1.txt
```

```
SQL>
```

Limites do SGDB Oracle

A seguir, os limites definidos para os diversos componentes que constituem ou que são utilizados dentro da plataforma do banco de dados ou nas suas linguagens.

- Tabelas: o número máximo de tabelas clusterizadas é 32. E é ilimitado por banco de dados.
- Colunas: o número máximo por tabela é de 1000 colunas. Por índice (ou índice clusterizado), é de 32 colunas. Por índice bitmap, é de 30 colunas.
- Linhas: o número de linhas por tabela é ilimitado.
- Índices: o número de índices por tabela é ilimitado. Por tamanho de colunas indexadas, é de 75% do tamanho total do bloco do banco de



dados (minus some overhead).

- Caracteres: a quantidade para nomes de objetos é limitada a 30 caracteres.
- Cláusula GROUP BY: a expressão GROUP BY e todas as expressões de agregação (por exemplo, SUM, AVG) devem estar dentro de um único bloco na base de dados.
- CONSTRAINTS: o número de constraints por coluna é ilimitado.
- SUBQUERIES: o nível de subqueries, em um comando SQL, é ilimitado na cláusula FROM quando no primeiro nível de uma query, e é limitado a 255 na cláusula WHERE.
- Grupo de valores: o campo data pertence ao intervalo de 1-Jan-4712BC a 31-Dec-4712AD.
- Tabelas ou VIEWS “joined” em uma QUERY: ilimitado.
- PACKAGES armazenadas: em PL/SQL e Developer/2000 pode haver limites no tamanho de PROCEDURES armazenadas que elas podem chamar. O limite em geral está entre 2000 a 3000 linhas de código.
- TRIGGERS em cascata: depende do sistema operacional; geralmente 32.
- Usuários e regras: é limitado a 2, 147, 483, 638.
- Partições: comprimento máximo de linha da chave particionada é de 4KB – overhead. O número máximo de colunas na chave particionada é de 16. O número máximo de partições disponíveis por tabela ou índice é de 64KB – 1 partição.
- Definição para CREATE MATERIALIZED VIEW: tem tamanho máximo de 64K bytes.

O limite de até quanto uma instrução SQL pode retornar (resposta) depende de alguns fatores, incluindo configurações, espaço em disco e memória.

Quando um objeto está instanciado na memória, não existe um limite fixo para o número de atributos neste objeto. Mas o total máximo do tamanho da memória para um objeto instanciado é de 4GB. Quando um objeto instanciado é inserido dentro da tabela, os atributos são alocados em colunas separadas, e o limite de 1000 colunas é aplicado.

## Tipos de dados do SGDB Oracle

Na sequência, veja os tipos de dados utilizados pelas linguagens SQL e PLSQL para a construção de programas ou parametrizações do banco de dados.

### VARCHAR2

O tamanho máximo para campos de tabela é de 4000 bytes. O tamanho máximo para PL/SQL é de 32.767 bytes. O VARCHAR2 é variável e somente usa o espaço que está ocupado. Diferentemente do CHAR.

Comentário: VARCHAR é um subtipo (assim como STRING) que existe por questões de compatibilidade com outras marcas de banco de dados e também com o padrão SQL. Entretanto, a Oracle, no momento, não recomenda o uso do tipo VARCHAR porque sua definição deve mudar à medida que o

padrão SQL evoluir. Deve-se usar VARCHAR2.

#### CHAR

O tamanho máximo é de 2000 bytes.

Comentário: O tipo CHAR é usado para conter dados de string de comprimento fixo. Ao contrário das strings de VARCHAR2, uma string CHAR sempre contém o número máximo de caracteres.

Outros tipos

- NCHAR: tamanho máximo de 2000 bytes
- NCHAR VARYING: tamanho máximo de 4000 bytes
- CHAR VARYING: tamanho máximo de 4000 bytes

#### NUMBER(p,s)

É um numero com sinal e ponto decimal, sendo: (p)precisão de 1 a 38 dígitos (e)scala -84 a 127.

Este tipo também possui subtipos como:

- DECIMAL: igual a NUMBER;
- DEC: igual a DECIMAL;
- DOUBLE PRECISION: igual a NUMBER;
- NUMERIC: igual a NUMBER;
- INTEGER: equivalente a NUMBER(38);
- INT: igual a INTEGER;
- SMALLINT: igual a NUMBER(38);
- FLOAT: igual a NUMBER;
- FLOAT(prec): igual a NUMBER(prec), mas a precisão é expressa em termos de bits binários, não de dígitos decimais. A precisão binária pode variar de 1 até 126.

#### DATE

O tipo DATE é usado para armazenar os valores de data e hora. Um nome melhor seria DATETIME porque o componente de hora está sempre lá, independente de você usá-lo ou não. Se não for especificada a hora ao atribuir um valor para uma variável do tipo DATE, o padrão de meia-noite (12:00:00 a.m.) será usado.

Ele é do tipo: 1 JAN 4712 BC até 31 DEC 4712 AD (DATA com hora, minuto e segundo).

#### TIMESTAMP

Data com segundos fracionais. Permite o armazenamento de dados de tempo, como o ano, o mês, a hora, minuto e o segundo do tipo de dados DATE, além dos valores de segundos fracionais. Existem muitas variações desse tipo de dados, como WITH TIMEZONE, WITH LOCALTIMEZONE.

#### INTERVAL YEAR TO MONTH

Armazenados com um intervalo de anos e meses. Permite que os dados de tempo sejam armazenados como um intervalo de anos e meses. Usado para representar a diferença entre dois valores de data/hora em que apenas o ano e o mês são significativos.

#### INTERVAL DAY TO SECOND

Armazenados como um intervalo de dias, horas, minutos e segundos.

Permite que os dados de tempo sejam armazenados como um intervalo de dias, horas, minutos e segundos. Usado para representar a diferença precisa entre dois valores de data/hora.

## BOOLEAN

É do tipo True e False.

## LONG

Somente pode existir uma coluna por tabela. Seu tamanho máximo para tabela é de 2 GB. Seu tamanho máximo para PL/SQL é de 32.760.

## RAW

O tamanho máximo para campos de tabela é de 2000 bytes. Seu tamanho máximo para PL/SQL é de 32.767.

LONG RAW é outro tipo parecido com RAW, a diferença é que ele possui 7 bytes a menos quando utilizado em PL/SQL.

## Campos LONG

Existem algumas restrições para campos LONG e LONG RAW.

- Não se pode criar um OBJECT TYPE com o atributo de LONG.
- Uma coluna LONG não pode estar dentro da cláusula WHERE ou com referência integral dos dados, exceto NULL ou NOT NULL.
- Uma função não pode retornar um campo LONG.
- Uma tabela poderá ter somente um campo LONG.
- LONG não pode ser indexada.
- LONG não pode usar cláusulas além do WHERE, já mencionado, GROUP BY, ORDER BY e CONNECT BY.

Uma dica para você usar um campo LONG na cláusula WHERE é criar uma tabela temporária com os campos da tabela original, mas alterando um tipo LONG para CLOB. Também é possível alterar diretamente na tabela o campo LONG para CLOB, caso não tenha problema de alterar a estrutura da tabela original. Veja o exemplo:

```
create global temporary table table_temp (campo_lob clob) on
commit preserve rows
```

```
/
```

```
insert into table_temp select to_lob(coluna_lob) from
table_original
```

```
/
```

Outra alternativa, caso você tenha conhecimentos em PLSQL, é a criação de um programa para realizar a pesquisa. Com recursos da PLSQL você consegue utilizar funções que não são permitidas, pelo menos nestes casos, em SQL. O código pode ser parecido com o mostrado a seguir.

```
declare
cursor c1 is
select view_name, text
from user_views;
begin
for r1 in c1 loop
if r1.text like '%emp%' then
dbms_output.put_line(r1.view_name);
end if;
end loop;
end;
```

Por existirem estas limitações, a Oracle criou a família LOB ( CLOB, BLOB, NLOB, BFILE), que visa resolver alguns destes problemas enfren-

tados com o tipo LONG.

#### CLOB

O tamanho máximo é  $(4 \text{ GB} - 1) * \text{DB\_BLOCK\_SIZE}$ , parâmetro de inicialização (8TB a 128TB). O número de colunas CLOB por tabela é limitado somente pelo número máximo de colunas por tabela. Ele armazena textos. Estes textos são validados conforme o character set, ou seja, armazena acentuação etc. Tipos LOB surgiram em substituição aos tipos LONG e LONG RAW, pois estes só permitiam uma coluna por tabela. Os tipos LOB permitem mais de uma coluna.

#### NCLOB

O tamanho máximo é de  $(4 \text{ GB} - 1) * \text{DB\_BLOCK\_SIZE}$ , parâmetro de inicialização (8TB a 128TB). O número de colunas NCLOB por tabela é limitado somente pelo número máximo de colunas por tabela. Objeto de grande capacidade de caracteres nacionais – contém até 4GB de caracteres de bytes simples ou caracteres multibyte que atendem o conjunto de caracteres nacional definido pelo banco de dados Oracle.

#### BLOB

O tamanho máximo é de  $(4 \text{ GB} - 1) * \text{DB\_BLOCK\_SIZE}$ , parâmetro de inicialização (8TB a 128TB).

O número de colunas BLOB por tabela é limitado somente pelo número máximo de colunas por tabela. Armazenam dados não estruturados como: som, imagem, dados binários.

#### BFILE

O tamanho máximo é de 4GB. O tamanho máximo para o nome do arquivo é de 255 caracteres. O tamanho máximo para o nome do diretório é de 30 caracteres. O número máximo de BFILEs abertos é limitado pelo valor do parâmetro de inicialização SESSION\_MAX\_OPEN\_FILES, o qual é limitado pelo número máximo de arquivos abertos que o sistema operacional suporta.

#### ROWID

É um tipo especial usado para armazenar os ROWIDs (endereços físicos) das linhas armazenadas em uma tabela.

#### Resumo

Em resumo, os tipos comumente utilizados são: CHAR, VARCHAR2, NUMBER, DATE, BOOLEAN e os da família LOB.

## Conclusão

Um banco de dados é uma coleção organizada de informações, geralmente armazenada eletronicamente, que permite fácil acesso, gerenciamento e consulta dos dados. Ele utiliza linguagens como SQL para realizar essas operações. A evolução dos bancos de dados passou por diversas fases, desde modelos hierárquicos e em rede até os atuais modelos relacionais, orientados a objetos e NoSQL, cada um com suas próprias

A empresa Oracle surgiu nos Estados Unidos para atender uma demanda governamental e se tornou, um banco de dados relacional popular em ambientes corporativos, conhecido por sua robustez e alto desempenho. Ele é amplamente utilizado para gerenciar grandes volumes de dados e suportar aplicações críticas. Este guia oferece recursos completos para você aprender a dominar o Oracle, desde cursos básicos até conteúdos avançados, permitindo que você explore todo o potencial deste poderoso sistema de gerenciamento de banco de dados, características e aplicações. Atualmente, a tecnologia de bancos de dados continua a evoluir com o surgimento de soluções em nuvem e bancos de dados autônomos. O banco de dados da Oracle trouxe algumas ferramentas e funcionalidades que até então não havia visto em outros bancos como o sistema de Push e rollback, que traz uma segurança maior para o caso de falhas do gerenciador do banco de dados. A forma como ele cria um histórico dos comandos já utilizados para evitar retrabalho, definitivamente é uma solução inteligente que ainda não conhecia.

## Referência

FERNANDES, Lúcia. Oracle 9i Para Desenvolvedores Curso Completo. Rio de Janeiro: Axcel Books, 2002.

GENNICK, Jonathan, LUERS, Tom. Aprenda em 21 dias PL/SQL. 2. ed. Rio de Janeiro: Campus, 2000.

HEUSER, Carlos Alberto. Projeto de Banco de Dados. 4. ed. Porto Alegre: Sagra Luzzatto, 2001.

LIMA, Adilson da Silva. Erwin 4.0 Modelagem de Dados. 2. ed. São Paulo: Érica, 2002.

Sítio da Oracle Disponível em: <<https://www.oracle.com/br/database/what-is-database/>>