Oracle SQL

MasterTraining – Treinamento e Tecnologia

www.mastertraining.com.br

Márcio Konrath

marcio@mastertraining.com.br

Índice

Capítulo 02

Capítulo 03

Capítulo 09

Capítulo 10

Capítulo 11

Capítulo 01

Introdução

Objetivos

Depois de completar esta lição, você poderá fazer o seguinte:

• Discutir os aspectos teóricos e físicos de um banco de dados relacional

• Descrever a implementação Oracle do RDBMS e ORDBMS

• Descrever os novos recursos do Oracle

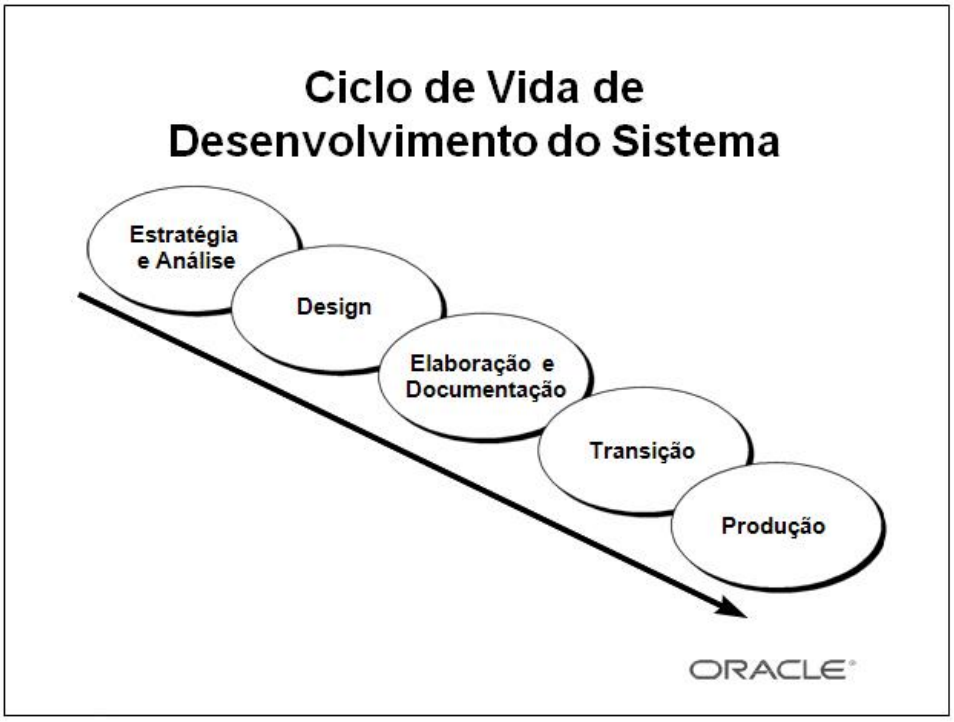
• Descrever como o SQL e o PL/SQL são usados no conjunto de produtos Oracle

• Descrever o uso e os benefícios do PL/SQL

Objetivo da Lição

Nesta lição, você entenderá o RDBMS (relational database management system) e o ORDBMS
(object relational database management system). Você também será apresentado aos seguintes tópicos:

•Instruções SQL específicas do Oracle
•SQL*Plus, usado para executar o SQL e para fins de formatação e elaboração de relatórios
•O PL/SQL, que é a linguagem procedural do Oracle



Ciclo de Vida de Desenvolvimento do Sistema

Do conceito à produção, você pode desenvolver um banco de dados usando o ciclo de vida de desenvolvimento do sistema, que contém vários estágios de desenvolvimento. Essa abordagem completa e sistemática para o desenvolvimento de bancos de dados transforma necessidades de informações comerciais em um banco de dados operacional.

Estratégia e Análise

•Estude e analise as necessidades comerciais. Entreviste usuários e gerentes para identificar as necessidades de informações. Incorpore as declarações de objetivos da aplicação e da empresa, além de qualquer especificação futura do sistema.

•Elabore modelos do sistema. Transfira a narrativa comercial para uma representação gráfica das regras e necessidades de informações comerciais. Confirme e refine o modelo com os analistas e especialistas.

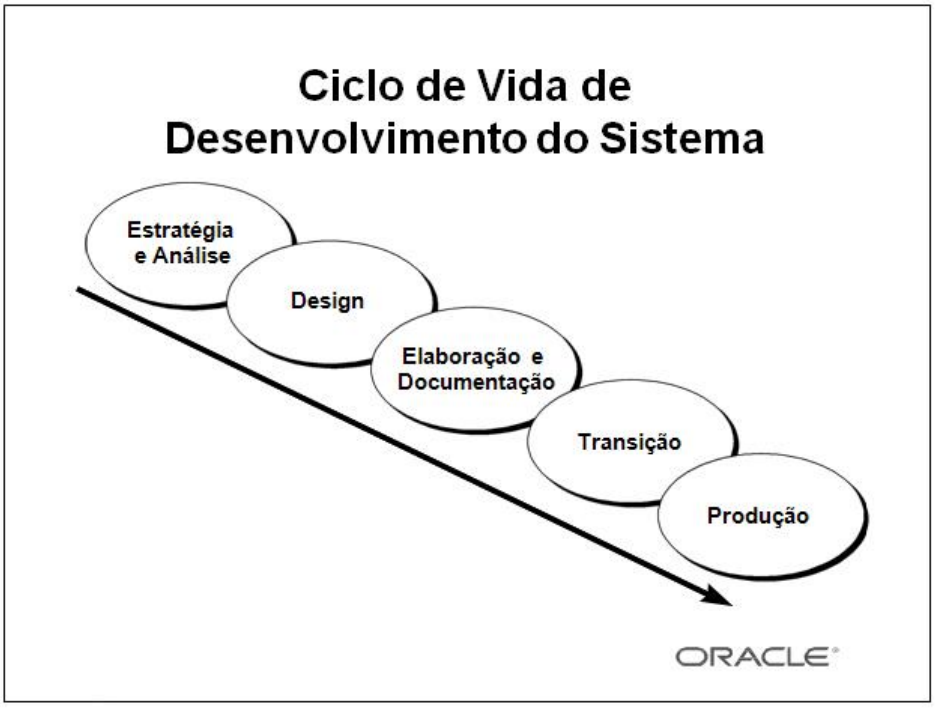
Design

Projete o banco de dados de acordo com o modelo desenvolvido na fase de estratégia e análise.

Elaboração e Documentação

•Elabore o sistema protótipo. Crie e execute os comandos para elaborar tabelas e objetos de suporte para o banco de dados.

•Desenvolva uma documentação para o usuário, textos de ajuda e manuais de operação para suporte ao uso e à operação do sistema.



Ciclo de Vida de Desenvolvimento do Sistema (continuação)

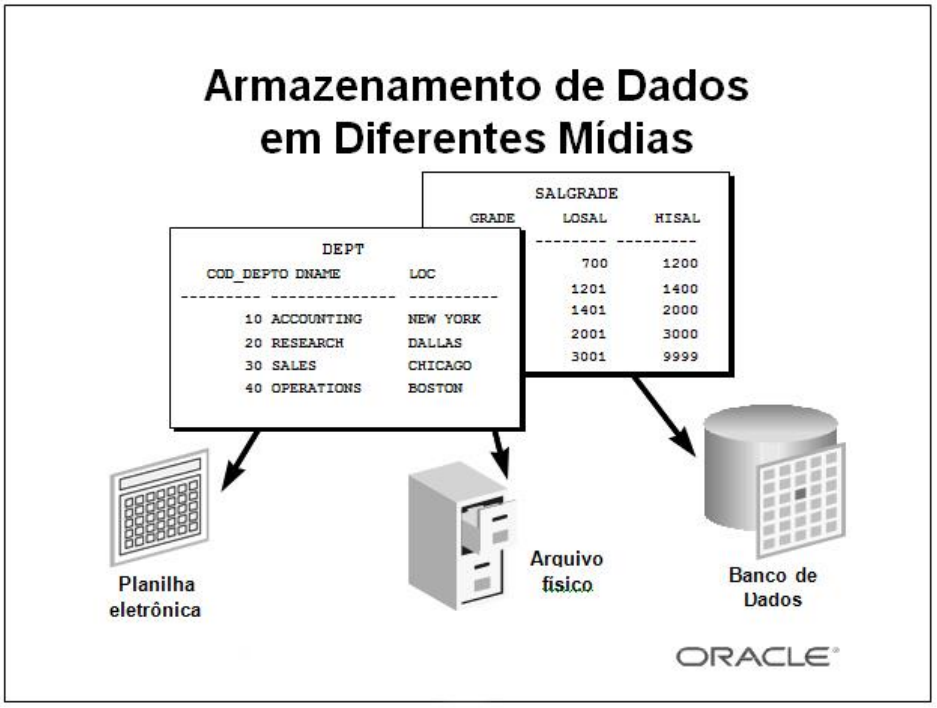
Transição

Refine o protótipo. Mova uma aplicação para a produção com teste de aceitação do usuário, conversão de dados existentes e operações paralelas. Faça as modificações necessárias.

Produção

Forneça o sistema aos usuários. Opere o sistema de produção. Monitore o desempenho, aperfeiçoe e refine o sistema.

Observação: É possível executar as várias fases do ciclo de vida de desenvolvimento do sistema repetidamente. Este curso se concentra na fase de elaboração do ciclo de vida de desenvolvimento do sistema.



Armazenando Informações

Toda organização necessita de informações. Uma biblioteca mantém uma lista de membros, livros, datas de entrega e multas. Uma empresa precisa gravar informações sobre funcionários,

departamentos e salários. Essas informações são chamadas de dados.

As organizações podem armazenar dados em várias mídias e em formatos diferentes, por exemplo, um documento impresso em um arquivo físico ou dados em planilhas eletrônicas ou bancos de dados. Um banco de dados é um conjunto organizado de informações.

Para gerenciar bancos de dados, você precisa de DBMSs (database management systems). Um DBMS é um programa que armazena, recupera e modifica dados do banco de dados a pedido. Há

quatro tipos principais de bancos de dados: hierárquico, de rede, relacional e relacional de objeto, o mais recente.

Conceito de Banco de Dados Relacional

- O Dr. E.F. Codd propôs o modelo relacional de sistemas de bancos de dados em 1970.

- Ele é a base para o RDBMS (relational database management system).

- O modelo relacional consiste nos seguintes itens:

– Conjunto de objetos ou relações

– Conjunto de operadores para agir sobre as relações

– Integridade de dados para precisão e consistência

Modelo Relacional

Os princípios do modelo relacional foram definidos primeiramente pelo Dr. E.F. Codd em junho de 1970 em um estudo chamado "A Relational Model of Data for Large Shared Data Banks". Nesse estudo, o Dr. Codd propôs o modelo relacional de sistemas de bancos de dados.

Os modelos mais populares usados naquele tempo eram hierárquicos, de rede ou mesmo estruturas de dados de arquivos simples. Os RDBMSs (relational database management systems) em breve se tornaram muito populares, especialmente pela facilidade de uso e flexibilidade na estrutura. Além disso, vários fornecedores inovadores, como a Oracle, ofereciam o RDBMS com um conjunto

eficiente de desenvolvimento de aplicações e produtos para usuários, formando uma solução completa.

Componentes do Modelo Relacional

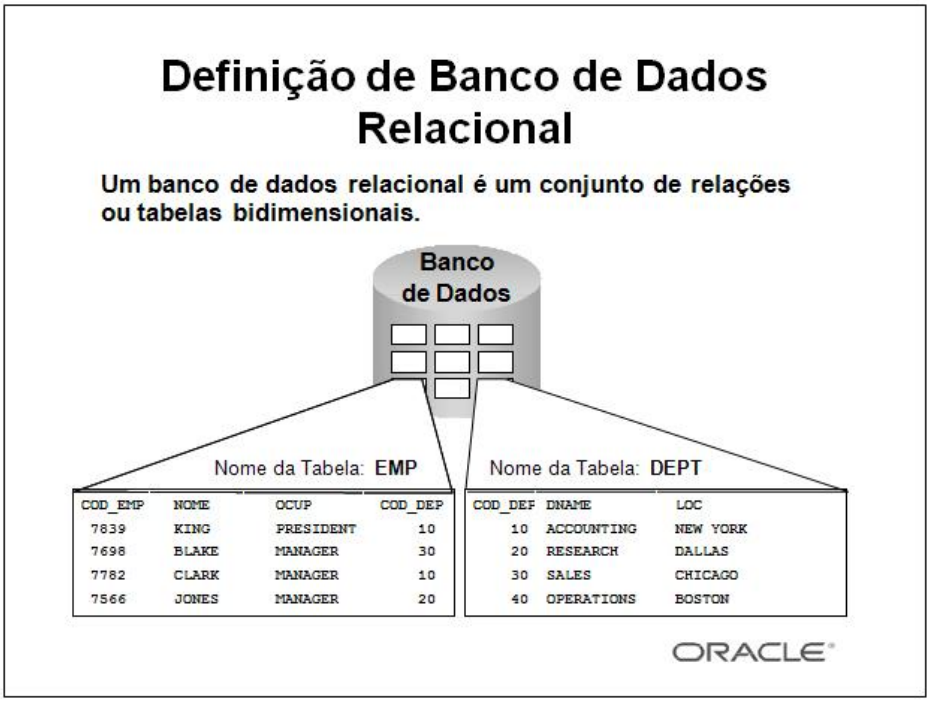
•Conjuntos de objetos ou relações que armazenam os dados

•Conjunto de operadores que podem agir sobre as relações para produzir outras relações

•Integridade de dados para precisão e consistência

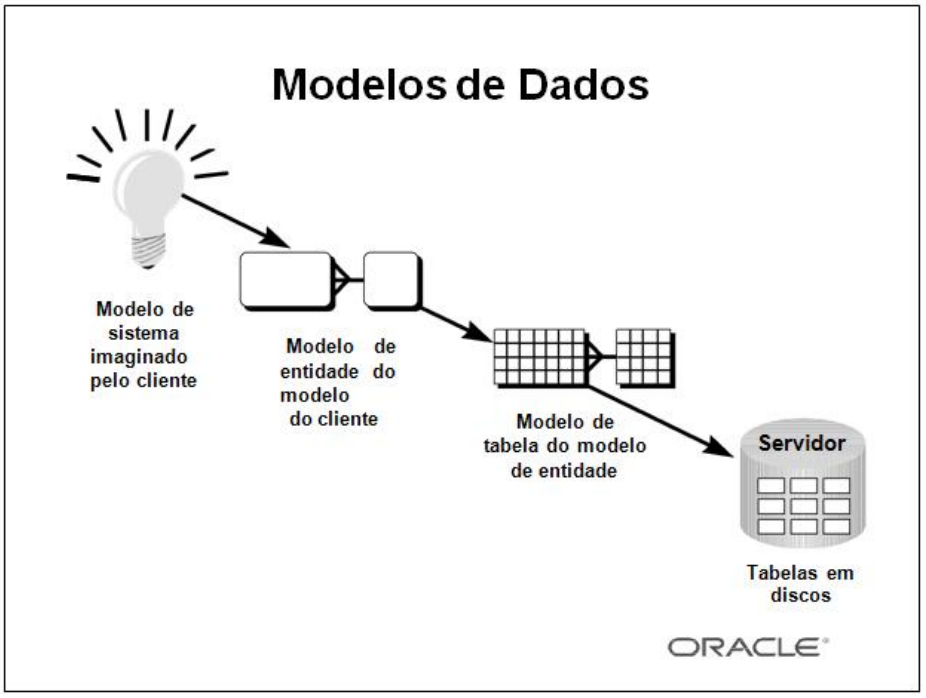
Para obter mais informações, consulte E.F. Codd, The Relational Model for Database Management

Version 2 (Reading, Mass.: Addison-Wesley, 1990).



Definição de Banco de Dados Relacional

Um banco de dados relacional usa relações ou tabelas bidimensionais para armazenar informações. Por exemplo, você pode armazenar informações sobre todos os funcionários de uma empresa. Em um banco de dados relacional, você cria várias tabelas para armazenar informações diferentes sobre funcionários, como tabelas de funcionários, departamentos e salários.



Modelos de Dados

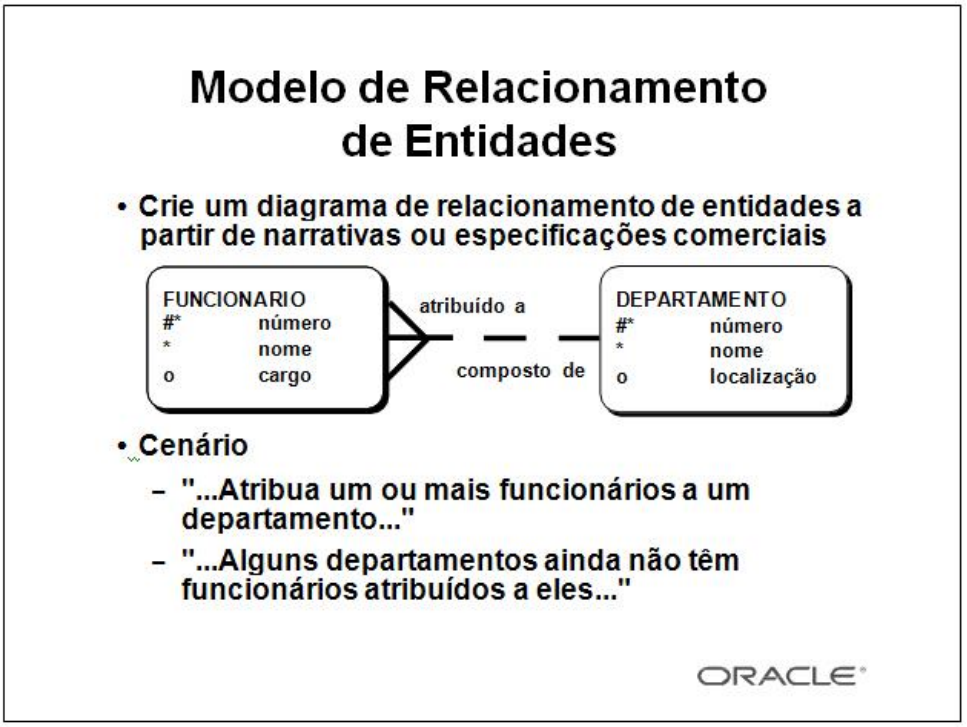
Os modelos são a base do design. Os engenheiros criam um modelo de carro para estudar os detalhes antes de colocá-lo em produção. Da mesma forma, projetistas de sistemas desenvolvem modelos para explorar idéias e compreender melhor o design de um banco de dados.

Objetivo dos Modelos

Os modelos ajudam a comunicar conceitos imaginados pelas pessoas. É possível usá-los com os seguintes objetivos:

- Comunicar
- Categorizar
- Descrever
- Especificar
- Investigar
- Desenvolver
- Analisar
- Imitar

O objetivo é produzir um modelo que se adapte a vários usos, possa ser compreendido por um
usuário final e contenha detalhes suficientes para que um desenvolvedor crie um sistema de banco de dados.



Modelo de Relacionamento de Entidades

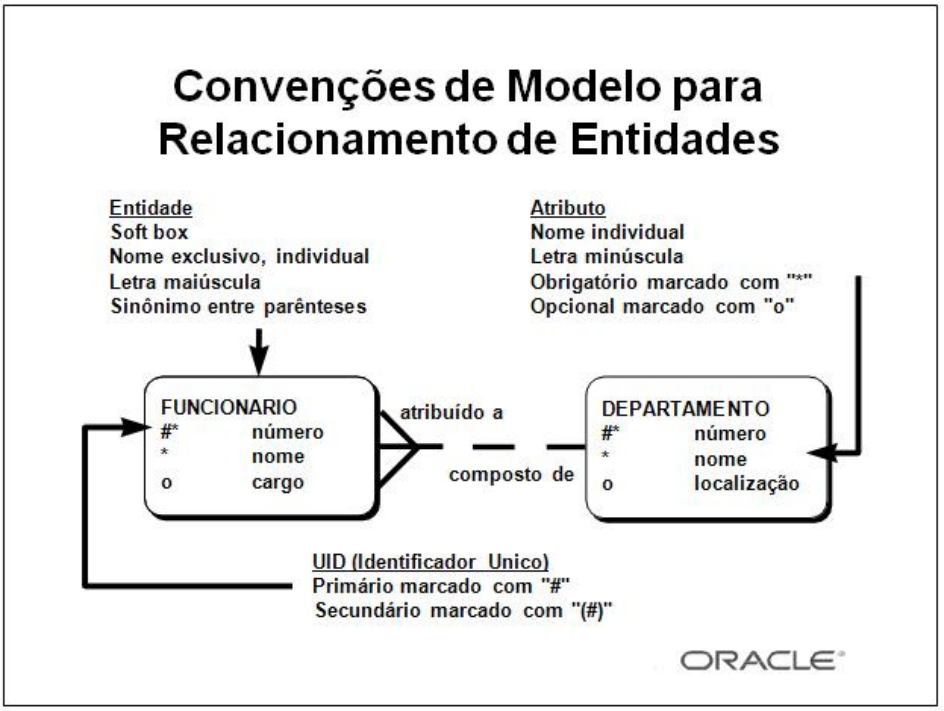
Em um sistema eficiente, os dados são divididos em categorias ou entidades distintas. Um modelo de relacionamento de entidades (ER) é uma ilustração de várias entidades em uma empresa e dos relacionamentos entre elas. Um modelo de relacionamento de entidades é derivado de narrativas ou especificações comerciais e é criado durante a fase de análise do ciclo de vida de desenvolvimento do sistema. Os modelos para relacionamento de entidades separam as informações necessárias para uma empresa das atividades desempenhadas dentro dela. Embora as empresas possam alterar suas atividades, o tipo de informações tende a permanecer constante. Portanto, as estruturas de dados também tendem a ser constantes.

Benefícios do Modelo de Relacionamento de Entidades

- Documenta as informações da organização em formato claro e preciso
- Fornece uma imagem clara do escopo das necessidades de informações
- Fornece um mapa ilustrado facilmente compreendido para o design do banco de dados
- Oferece uma estrutura eficiente para a integração de várias aplicações

Componentes-chave

- Entidade: Um item importante sobre o qual é necessário obter informações. Os exemplos são departamentos, funcionários e pedidos.
- Atributo: Um item que descreve ou qualifica uma entidade. Por exemplo, para a entidade de funcionários, os atributos são o número, o nome e o cargo do funcionário, além do número do departamento e assim por diante. Cada um desses atributos é necessário ou opcional. Esse
estado é chamado opcionalidade.
- Relacionamento: Uma associação nomeada entre entidades que demonstra opcionalidade e grau. Os exemplos são funcionários e departamentos, além de pedidos e itens.



Entidades

Para representar uma entidade em um modelo, use as seguintes convenções:

- Soft box com qualquer dimensão
- Nome de entidade exclusivo, individual
- Nome de entidade em letras maiúsculas
- Sinônimos opcionais em letras maiúsculas entre parênteses: ( )

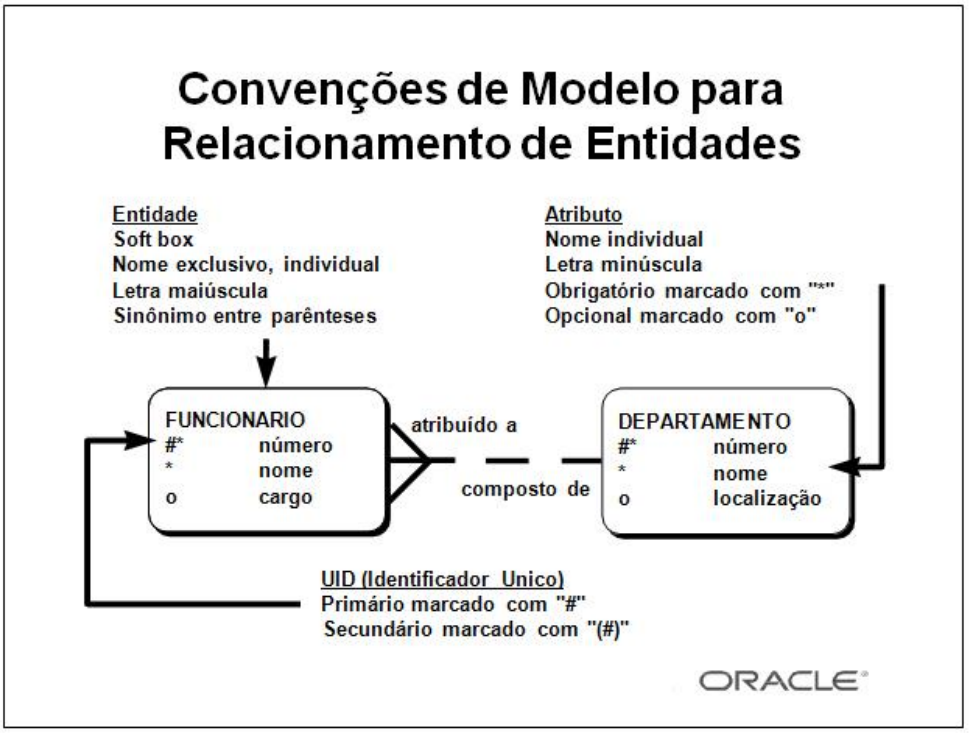
Atributos

Para representar um atributo em um modelo, use as seguintes convenções:

- Use nomes singulares em letras minúsculas
- Marque os atributos obrigatórios ou os valores que devem ser conhecidos com um asterisco: *
- Marque os atributos opcionais ou valores que podem ser conhecidos com a letra o

Relacionamentos

Símbolo	Descrição
Linha tracejada	Elemento opcional que indica algo que "pode ser"
Linha contínua	Elemento obrigatório que indica algo que "deve ser"
Pé-de-galinha	Elemento de classificação que indica "um ou mais"
Linha simples	Elemento de classificação que indica "um único"



Relacionamentos

Cada direção do relacionamento contém:

- Um nome, por exemplo, atribuído
- Uma opcionalidade, que indica algo que deve ser ou pode ser

- Um grau, que indica um único ou um ou mais

Observação: O termo cardinalidade é um sinônimo para o termo grau.

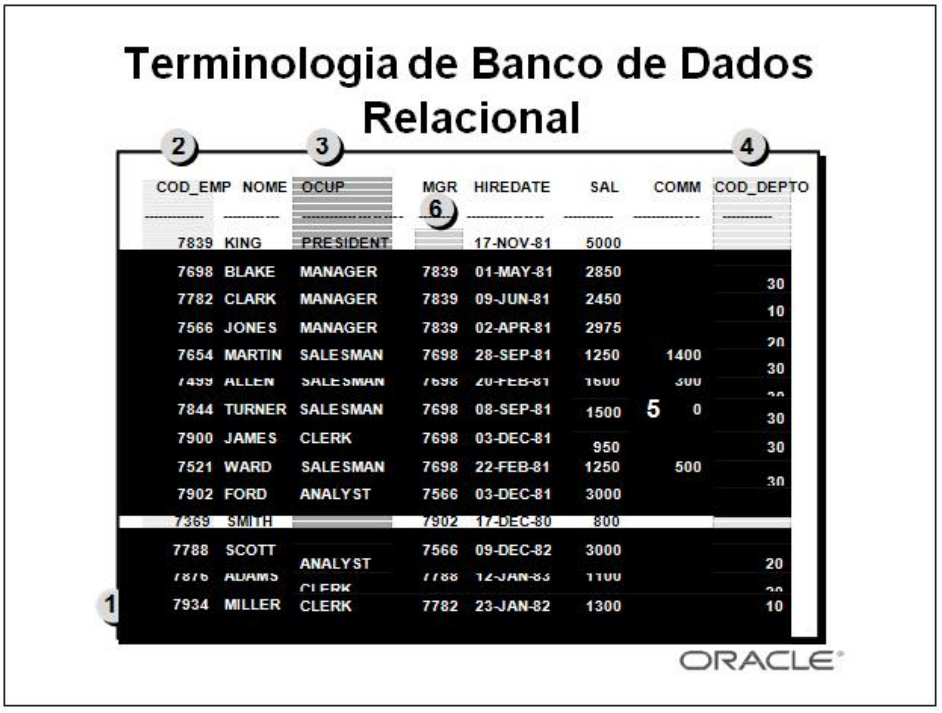
Cada entidade de origem {pode ser | deve ser} um nome de relacionamento {um único | um ou mais} entidade de destino.

Observação: A convenção deve ser lida em sentido horário.

Identificadores Únicos

Um UID (identificador único) corresponde a qualquer combinação de atributos ou relacionamentos (ou os dois) que serve para diferenciar ocorrências em uma entidade. Cada ocorrência de entidade deve ser identificada com exclusividade.

- Marque cada atributo que faz parte do UID com uma tralha: #
- Marque os UIDs secundários com uma tralha entre parênteses: (#)



Terminologia de Banco de Dados Relacional

Um banco de dados relacional pode conter uma ou várias tabelas. Uma tabela é a estrutura de armazenamento básica de um

RDBMS. Ela armazena todos os dados necessários sobre algo do mundo real, por exemplo, funcionários, NFFs ou clientes.

O slide mostra o conteúdo da relação ou tabela EMP. Os números indicam o seguinte:

1.Uma linha simples ou dupla que representa todos os dados necessários para um funcionário específico. Cada linha de uma tabela deve ser identificada por uma chave primária, que não permite linhas duplicadas. A ordem das linhas não é importante; especifique essa ordem quando

os dados forem recuperados.

2.Uma coluna ou atributo que contém o número do funcionário, que é também a chave primária.

O número do funcionário identifica um único funcionário na tabela EMP. Uma chave primária deve conter um valor.

3.Uma coluna que não é um valor de chave. Uma coluna representa um tipo de dados em uma tabela; no exemplo, o cargo de todos os funcionários. A ordem das colunas não é importante durante o armazenamento de dados; especifique essa ordem quando os dados forem recuperados.

4.Uma coluna que contém o número do departamento, que é também uma chave estrangeira. Uma chave estrangeira é uma coluna que define como as tabelas se relacionam umas com as outras.

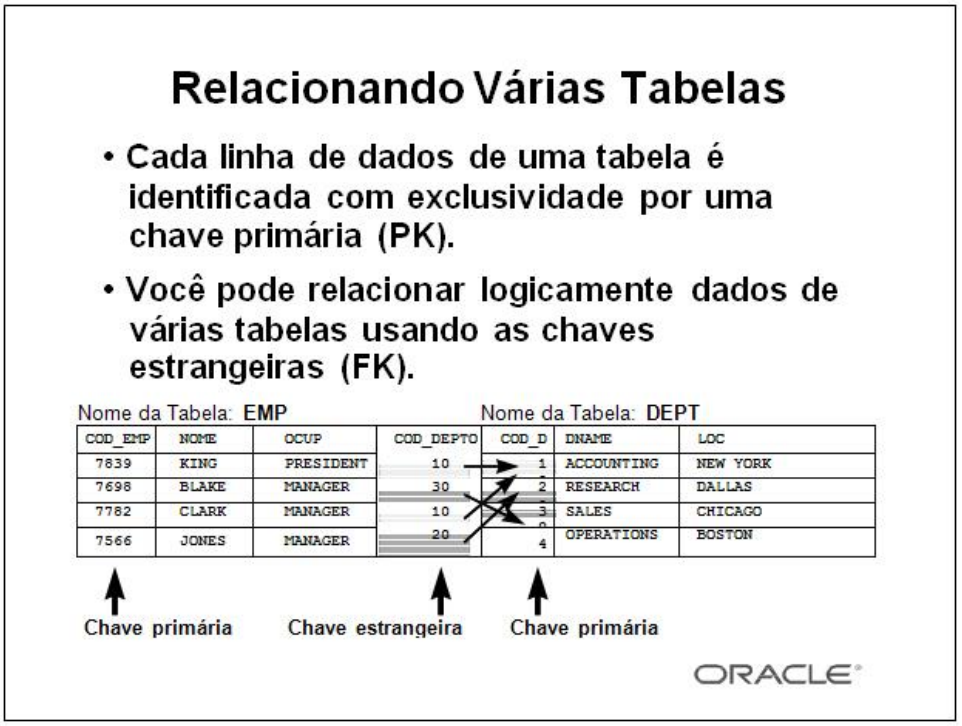
Uma chave estrangeira se refere a uma chave primária ou a uma chave exclusiva em outra tabela. No exemplo, COD_DEPTO identifica com exclusividade um departamento da tabela DEPT.

Terminologia de Banco de Dados Relacional (continuação)

5. É possível encontrar um campo na interseção entre uma linha e uma coluna. Só pode haver um valor nesse campo.

6. Um campo pode não conter nenhum valor. Nesse caso, o valor é nulo. Na tabela EMP, apenas funcionários com cargo de vendedor têm um valor no campo COMM (comissão).

Observação: Os valores nulos são abordados com mais detalhes nas lições posteriores.



Relacionando Várias Tabelas

Cada tabela contém dados que descrevem exatamente uma entidade. Por exemplo, a tabela EMP contém informações sobre funcionários. As categorias de dados são listadas ao longo da parte superior de cada tabela e os casos individuais são listados abaixo da tabela. Usando o formato de tabela, você pode visualizar, entender e usar informações imediatamente.

Como os dados sobre entidades diferentes são armazenados em tabelas diferentes, talvez você precise combinar duas ou mais tabelas para responder a uma pergunta específica. Por exemplo, talvez você queira saber a localização do departamento no qual um funcionário trabalha. Nesse cenário, você precisa de informações da tabela EMP (que contém dados sobre funcionários) e da tabela DEPT (que contém informações sobre departamentos). Um RDBMS permite relacionar os dados de uma tabela aos dados de outra usando as chaves estrangeiras. Uma chave estrangeira é uma coluna ou um conjunto de colunas que se refere a uma chave primária na mesma tabela ou em outra tabela.

O recurso de relacionar dados de uma tabela a dados de outra permite organizar informações em unidades gerenciáveis separadas. É possível manter logicamente os dados dos funcionários separados dos dados dos departamentos armazenando-os em uma tabela separada.

Diretrizes para Chaves Primárias e Estrangeiras

- Não são permitidos valores duplicados em uma chave primária.
- Geralmente, não é possível alterar chaves primárias.
- As chaves estrangeiras são baseadas nos valores dos dados e são ponteiros unicamente lógicos e não físicos.
- Uma chave estrangeira deve corresponder a um valor de chave primária existente, a um valor de chave exclusiva ou ser nula.
- Não é possível definir chaves estrangeiras sem chaves primárias (exclusivas) existentes.

Propriedades de Banco de Dados Relacional

Um banco de dados relacional

- Pode ser acessado e modificado executando instruções SQL (Structured Query Language)

- Contém um conjunto de tabelas sem ponteiros físicos

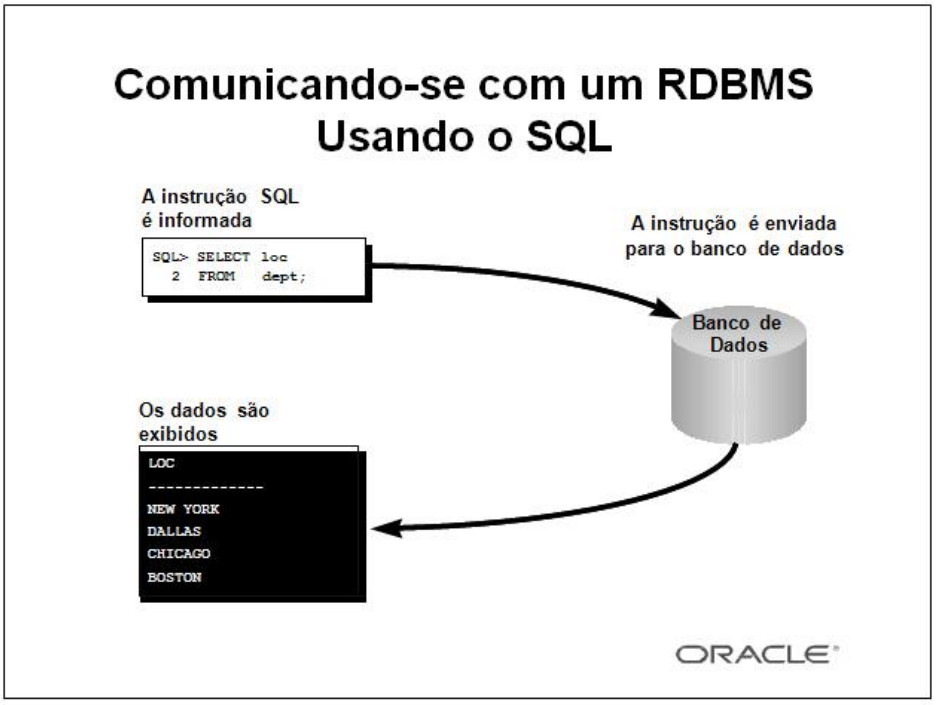
- Usa um conjunto de operadores

Propriedades de Banco de Dados Relacional

Em um banco de dados relacional, você não especifica a rota de acesso às tabelas e não precisa saber como os dados são organizados fisicamente.

Para acessar o banco de dados, execute uma instrução SQL (Structured Query Language), que é a linguagem padrão ANSI (American National Standards Institute) para a operação em bancos de dados relacionais. A linguagem contém um grande conjunto de operadores para dividir e combinar relações.

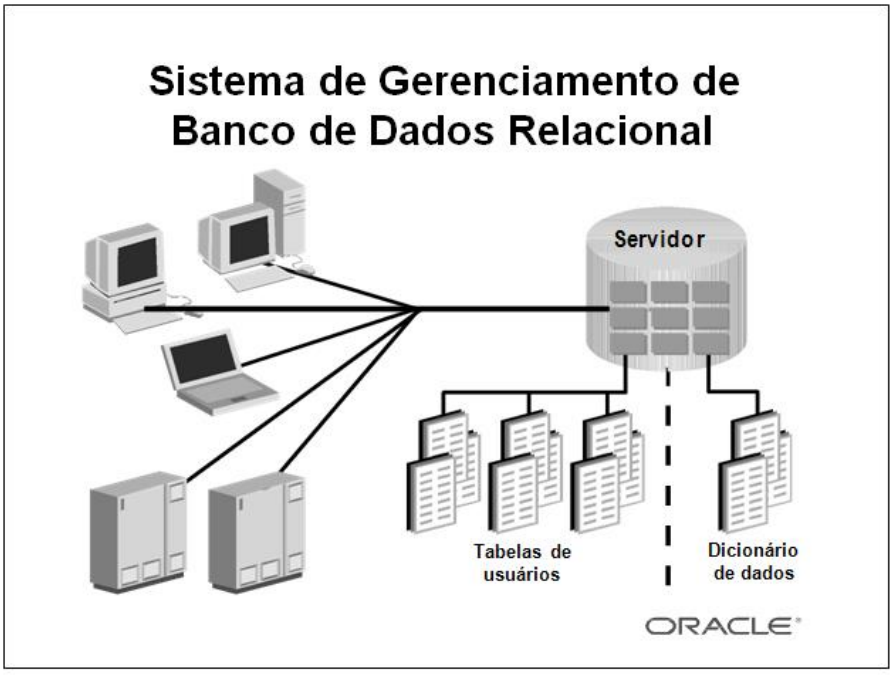
É possível modificar o banco de dados usando as instruções SQL.



SQL (Structured Query Language)

O SQL permite comunicar-se com o servidor e tem as seguintes vantagens:

- Eficiência
- Facilidade de aprendizagem e uso
- Funcionalidade completa (O SQL permite definir, recuperar e manipular dados das tabelas.)



Sistema de Gerenciamento de Banco de Dados Relacional

A Oracle fornece um RDBMS flexível chamado Oracle7. Usando os recursos desse RDBMS, você pode armazenar e gerenciar dados com todas as vantagens de uma estrutura relacional além do PL/SQL, um mecanismo com recurso para armazenar e executar unidades de programas. O servidor oferece as opções de recuperação de dados com base em técnicas de otimização. Ele inclui recursos

de segurança que controlam como um banco de dados é acessado e usado. Outros recursos incluem a consistência e a proteção de dados através de mecanismos de bloqueio.

As aplicações Oracle podem ser executadas no mesmo computador que o Oracle Server. Como opção, você pode executar aplicações em um sistema local de usuário e executar o Oracle Server em outro sistema (arquitetura cliente-servidor). Nesse ambiente cliente-servidor, é possível usar uma grande variedade de recursos de computação. Por exemplo, uma aplicação de reservas de uma linha aérea baseada em forms pode ser executada no PC de um cliente, enquanto o acesso a dados de vôo é
gerenciado convenientemente por um Oracle Server em um computador central.

Para obter mais informações, consulte o Oracle Server Concepts Manual, Release 8.

Sistema de Gerenciamento de Banco de Dados Relacional de Objeto

- Objetos e tipos de dados definidos pelo usuário
- Compatibilidade total com o banco de dados relacional
- Suporte de objetos grandes e multimídia
- Recursos de servidor de banco de dados de alta qualidade

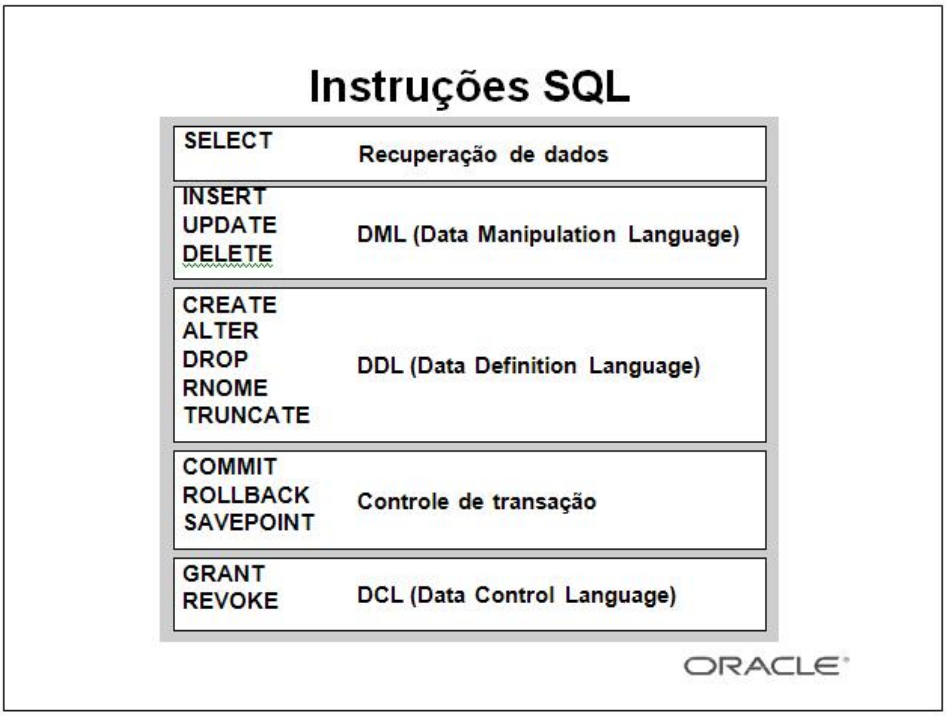
Sobre o Oracle

O Oracle é o primeiro banco de dados com recurso de objeto desenvolvido pela Oracle. Ele estende os recursos de modelo de dados do Oracle7 para suportar um novo modelo de banco de dados relacional de objeto. O Oracle fornece um novo mecanismo que oferece programação orientada a objeto, tipos de dados complexos, objetos comerciais complexos e compatibilidade total com o universo relacional.

O Oracle inclui vários recursos para desempenho e funcionalidade aperfeiçoados de aplicações OLTP (Online Transaction Processing), como um melhor compartilhamento de estruturas de dados no tempo de execução, caches de buffer maiores e restrições diferenciáveis. As aplicações de armazenamento de dados se beneficiarão de aperfeiçoamentos como a execução paralela de operações para inserir, atualizar e deletar, a divisão e a otimização de consultas paralelas. Operando na estrutura NCA (Network Computing Architecture), o Oracle suporta aplicações cliente-servidor e baseadas da Web distribuídas e com várias camadas.

O Oracle pode escalonar dezenas de milhares de usuários simultâneos, suportar 512 petabytes e tratar qualquer tipo de dados, incluindo dados espaciais, de textos, imagens, som, vídeo e séries de tempos, além de dados estruturados tradicionais.

Para obter mais informações, consulte o Oracle Server Concepts Manual, Release 8.



Instruções SQL

O Oracle SQL é compatível com os padrões aceitos pela indústria. A Oracle Corporation garante a compatibilidade futura com padrões em desenvolvimento envolvendo ativamente uma equipe-chave nos comitês de padrões SQL. Os comitês aceitos pela indústria são o ANSI (American National Standards Institute) e o ISO (International Standards Organization). O ANSI e o ISO aceitaram o

SQL como a linguagem padrão para os bancos de dados relacionais.

Instrução	Descrição
SELECT	Recupera dados do banco de dados
INSERT UPDAT DELETE	Informa novas linhas, altera linhas existentes e remove linhas indesejáveis de tabelas do banco de dados, respectivamente. O conjunto dessas instruções é conhecido como DML (*Data Manipulation Language*).
CREAT E ALT ER DROP RNOME TRUNCATE	Configura, altera e remove estruturas de dados de tabelas. O conjunto dessas instruções é conhecido como DDL (*Data Definition Language*).
COMMIT ROLLBACK SAVEPOINT	Gerencia as alterações feitas pelas instruções DML. É possível agrupar as alterações dos dados em transações lógicas.
GRANT REVOKE	Fornece ou remove direitos de acesso ao banco de dados Oracle e às estruturas contidas nele. O conjunto dessas instruções é conhecido

Sobre PL/SQL

• O PL/SQL é uma extensão do SQL com recursos de design de linguagens de programação.

• As instruções de consulta e a manipulação de dados do SQL estão incluídas nas unidades procedurais de código.

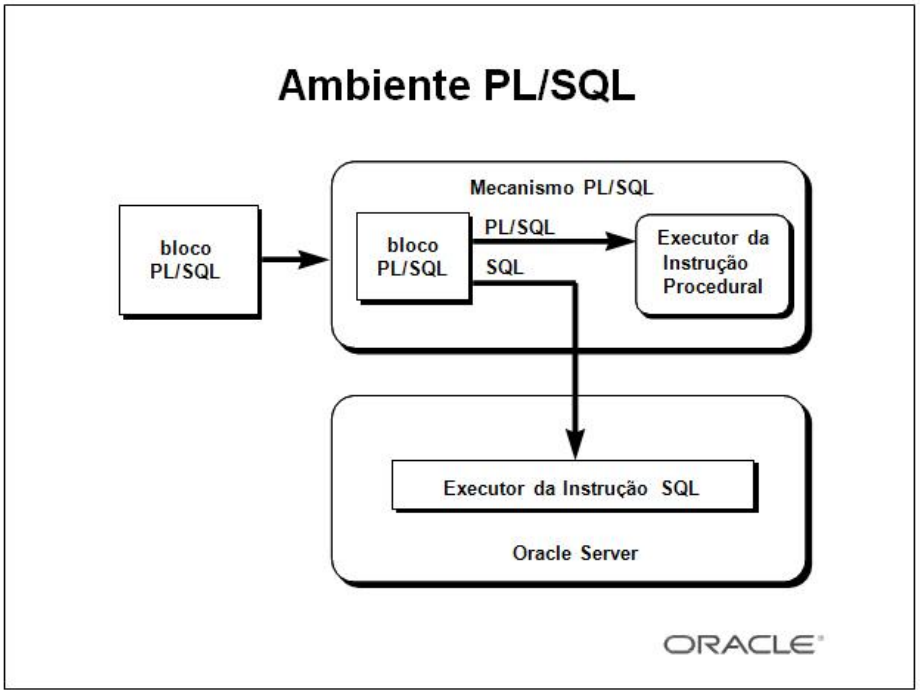
Sobre PL/SQL

O PL/SQL (Procedural Language/SQL) é uma extensão de linguagem procedural da Oracle Corporation para SQL, a linguagem de acesso a dados padrão para bancos de dados relacionais. O PL/SQL oferece recursos de engenharia de software modernos, como, por exemplo, a encapsulação de dados, o tratamento de exceções, a ocultação de informações, a orientação de objeto e assim por diante, trazendo os recursos de programação mais modernos para o Oracle Server e o Toolset.

O PL/SQL incorpora muitos recursos avançados feitos em linguagens de programação projetadas durante as décadas de 70 e

80. Além de aceitar a manipulação de dados, ele também permite que instruções de consulta do SQL sejam incluídas em unidades procedurais de código e estruturadas em blocos, tornando o PL/SQL uma linguagem de processamento de transações poderosa. Com o

PL/SQL, você pode usar as instruções SQL para refinar os dados da Oracle e as instruções de controle do PL/SQL para processar os dados.



Mecanismo PL/SQL e Oracle Server

O PL/SQL não é um produto Oracle em si; ele é uma tecnologia empregada pelo Oracle Server e por algumas ferramentas Oracle. Os blocos PL/SQL são passados e processados por um mecanismo PL/SQL, que pode residir na ferramenta ou no Oracle Server. O mecanismo usado depende do local no qual o PL/SQL é chamado.

O mecanismo PL/SQL no Oracle Server processa os blocos PL/SQL submetidos de um programa de saída de usuário, Pro*, SQL*Plus ou Server Manager. Ele separa as instruções SQL e as envia individualmente para o executor da instrução SQL.

Uma única transferência é necessária para enviar o bloco da aplicação para o Oracle Server, melhorando o desempenho, especialmente em uma rede cliente-servidor. Também é possível armazenar o PL/SQL

no Oracle Server sob a forma de subprogramas que podem ser consultados pelas aplicações conectadas ao banco de dados.

Sumário

- Os bancos de dados relacionais são compostos por relações, gerenciados por operações relacionais e regidos por restrições de integridade de dados.

- O Oracle Server permite armazenar e gerenciar informações usando a linguagem SQL e o mecanismo PL/SQL.

- O Oracle é baseado no ORDBMS (object relational database management system).

- O Oracle Server é o banco de dados de computação na Internet.

- O PL/SQL é uma extensão do SQL com recursos de design de linguagens de programação.

Sumário

Os RDBMSs (relational database management systems) são compostos por objetos ou relações. Eles são gerenciados por operações e regidos por restrições de integridade de dados.

A Oracle Corporation cria produtos e serviços para atender suas necessidades de RDBMS. O produto principal é o Oracle Server, que permite armazenar e gerenciar informações usando o SQL e o mecanismo PL/SQL para construções procedurais.

SQL

O Oracle Server suporta o SQL do padrão ANSI e contém extensões. O SQL é uma linguagem usada para comunicar-se com o servidor e acessar, manipular e controlar dados.

PL/SQL

A linguagem PL/SQL estende a linguagem SQL oferecendo construções procedurais estruturados em blocos combinados com recursos não procedurais do SQL.

Capítulo 02

Criando Instruções SQL Básicas

Objetivos

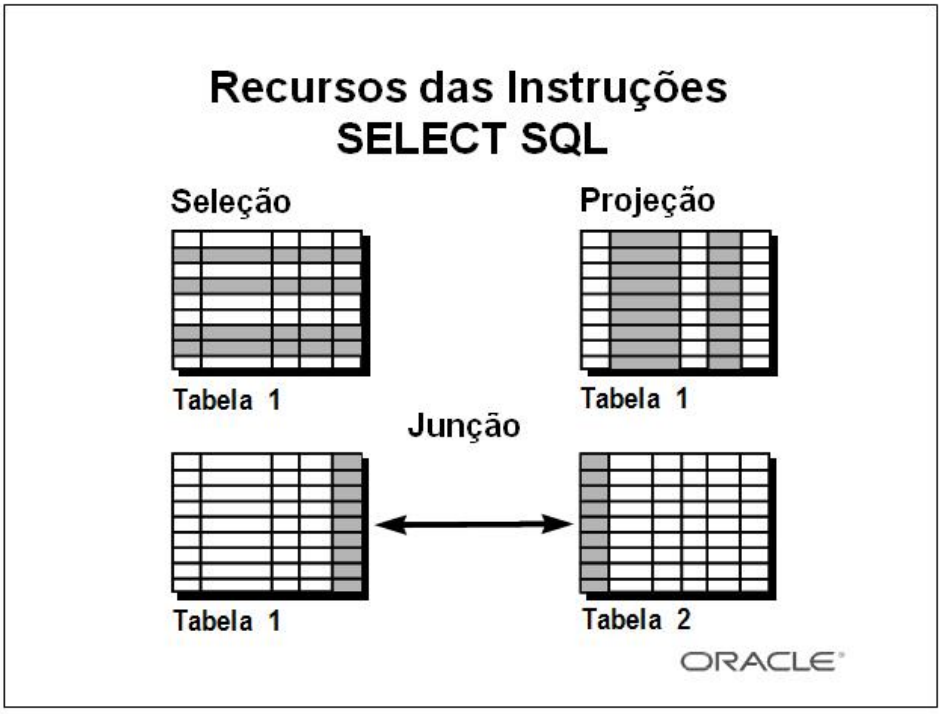
Depois de completar esta lição, você poderá fazer o seguinte:

- Listar os recursos das instruções SELECT SQL
- Executar uma instrução SELECT básica
- Diferenciar instruções SQL e comandos SQL*Plus

Objetivo da Lição

Para extrair dados do banco de dados, é preciso usar a instrução SELECT SQL (Structured Query Language). Talvez você necessite restringir as colunas exibidas. Esta lição descreve todas as instruções SQL necessárias para executar essas ações.

Talvez você deseje criar instruções SELECT que possam ser usadas continuamente. Esta liçãotambém aborda o uso dos comandos SQL*Plus para executar instruções SQL.



Recursos das Instruções SELECT SQL

A instrução SELECT recupera informações do banco de dados. Usando uma instrução SELECT, você pode fazer o seguinte:

• Seleção: Você pode usar o recurso de seleção no código SQL para escolher as linhas de uma tabela que deseja ver retornadas por uma consulta. Pode usar vários critérios para restringir seletivamente as linhas que você vê.

• Projeção: Você pode usar o recurso de projeção no código SQL para escolher as colunas de uma tabela que deseja ver retornadas por uma consulta. É possível escolher mais ou menos colunas da tabela conforme sua necessidade.
• Junção: Você pode usar o recurso de junção no código SQL para reunir dados armazenados em tabelas diferentes, criando um vínculo entre eles. Você aprenderá mais sobre junções em uma lição posterior.

Instrução SELECT Básica

```
SELECT    [DISTINCT] {*, coluna [apelido],...}
FROM tabela;
```

- SELECT identifica que colunas.
- FROM identifica qual tabela.

Instrução SELECT Básica
Da forma mais simples, uma instrução SELECT deve incluir o seguinte:

- Uma cláusula SELECT, que especifica as colunas a serem exibidas
- Uma cláusula FROM, que especifica a tabela que contém as colunas listadas na cláusula SELECT Na sintaxe:

SELECT é uma lista de uma ou mais colunas
DISTINCT suprime os itens duplicados
* seleciona todas as colunas
coluna seleciona a coluna nomeada
apelido fornece cabeçalhos diferentes às colunas selecionadas
FROM *tabela* especifica a tabela contendo as colunas

Observação: Em todo o curso são usados os termos palavra-chave, cláusula e instrução.

- Uma palavra-chave refere-se a um elemento SQL individual. Por exemplo, SELECT e FROM são palavras-chave.
- Uma cláusula é parte de uma instrução SQL.

Por exemplo, SELECT COD_EMP, NOME, ... é uma cláusula.

- Uma instrução é uma combinação de duas ou mais cláusulas. Por exemplo, SELECT * FROM EMPREGADO é uma instrução SQL.

Criando Instruções SQL

• Instruções SQL sem distinção entre maiúsculas e minúsculas.

• Instruções SQL podem estar em uma ou mais linhas.

• Palavras-chave não podem ser abreviadas ou dividas entre as linhas.

• Normalmente, as cláusulas são colocadas em linhas separadas.

• Guias e endentações são usadas para aperfeiçoar a legibilidade.

Criando Instruções SQL
Usando as seguintes diretrizes e regras simples, você pode construir instruções válidas fáceis de ler e editar:

• As instruções SQL não fazem distinção entre maiúsculas de minúsculas, a menos que indicado.
• As instruções SQL podem ser digitadas em uma ou mais linhas.
• As palavras-chave não podem ser divididas entre as linhas nem abreviadas.
• As cláusulas são em geral colocadas em linhas separadas para melhor legibilidade e facilidade de edição.
• As guias e endentações podem ser usadas para tornar o código mais legível.
• Em geral, as palavras-chave são digitadas em letras maiúsculas, todas as outras palavras, como nomes de tabela e colunas são digitadas em minúsculas.
• Dentro do SQL*Plus, uma instrução SQL é digitada no prompt SQL e as linhas subseqüentes são numeradas. Isso chama-se buffer de SQL. Somente uma instrução pode ser a atual a qualquer momento dentro do buffer.

Executando Instruções SQL
• Coloque um ponto-e-vírgula (;) no final da última cláusula.
• Coloque uma barra na última linha do buffer.
• Coloque uma barra no prompt SQL.
• Emita um comando RUN do SQL*Plus no prompt SQL.

Selecionando Todas as Colunas

```
SQL> SELECT *
 2     FROM  EMPREGADO;
```

Selecting All Columns, All Rows

Exiba todas as colunas de dados em uma tabela seguindo a palavra-chave SELECT com um asterisco

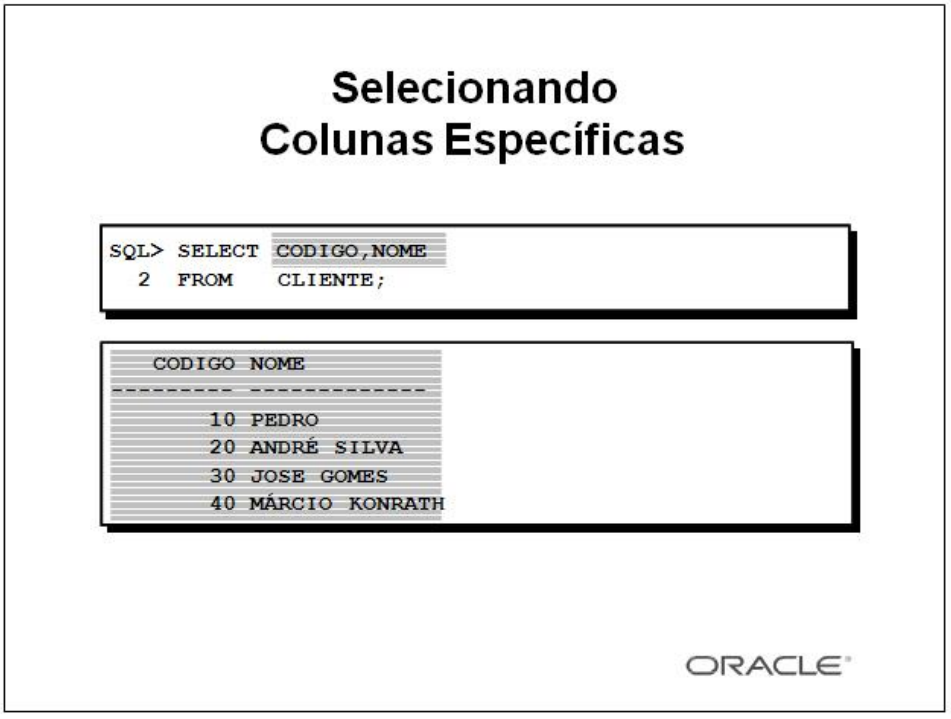
(*). No exemplo do slide, a tabela do departamento contém três colunas: COD_DEPTO, DNAME e LOC.

A tabela contém quatro linhas, uma para cada departamento.

É possível, também, exibir todas as colunas na tabela, listando todas elas após a palavra-chave

SELECT. Por exemplo, a instrução SQL a seguir, como no exemplo do slide, exibe todas as colunas e linhas da tabela DEPT:

```
SQL> SELECT COD_DEPTO, NOME, LOCAL
  2    FROM  DEPARTAMENTO;
```



Selecionando Colunas Específicas, Todas as Linhas

Você pode usar a instrução SELECT para exibir colunas específicas da tabela, ao especificar os nomes da coluna, separado por vírgulas. O exemplo do slide exibe todos os números dos departamentos e locais na tabela CLIENTE.
Na cláusula SELECT, especifique as colunas a serem vistas, na ordem que deseja que apareçam na saída. Por exemplo, para exibir o local antes do número do departamento, use a seguinte instrução:

```
SQL> SELECT  NOME, COD_CIDADE
2     FROM  CLIENTE;


NOME  COD_CIDADE
------------- --------- PEDRO 1
ANDRÉ SILVA 2
JOSE GOMES  3
MÁRCIO KONRATH    4
```

Defaults de Cabeçalho de Coluna

• Justificativa default

– Esquerda: Dados de caractere e data

– Direita: Dados numéricos

• Exibição default: Letra maiúscula

Defaults de Cabeçalho de Coluna

Os dados e o cabeçalho da coluna de caracteres bem como os dados e o cabeçalho da coluna de data são justificados à esquerda na largura da coluna. Os cabeçalhos de número e dados são justificados à direita.

```
SQL> SELECT NOME, DATA, SALARIO
2     FROM  EMPREGADOS;

NOME  DATA  SAL
---------- --------- ---------
JOSE  17-NOV-81   5000
PEDRO 01-MAY-81   2850
ANA   09-JUN-81   2450
MARIA 02-APR-81   2975
```

```
PEDRO 28-SEP-81   1250
PAULA 20-FEB-81   1600
...
14 rows selected.
```

Os cabeçalhos da coluna de caracteres e datas podem ser truncados, mas os cabeçalhos de números, não. Os cabeçalhos de coluna aparecem por default em letra minúscula. Você pode sobrepor a exibição do cabeçalho de coluna com um apelido. Os apelidos de coluna são abordados posteriormente nesta lição.

Expressões Aritméticas

Criar expressões com dados NUMBER e DATE usando operadores aritméticos.

Operador	Descrição
+	Adicionar
-	Subtrair
*	Multiplicar
/	Dividir

Expressões Aritméticas

Talvez você necessite modificar a forma de exibição dos dados, efetuar cálculos ou consultar cenários what-if. Isso é possível usando expressões aritméticas. Uma expressão aritmética possui nomes de coluna, valores numéricos constantes e operadores aritméticos.

Operadores Aritméticos

O slide lista os operadores aritméticos disponíveis no código SQL. Você pode usar operadores aritméticos em qualquer cláusula de uma instrução SQL exceto na cláusula FROM.

Usando Operadores Aritméticos

```
SQL> SELECT NOME, SALARIO, SALARIO+300
2     FROM  EMPREGADO;


NOME   SAL    SAL+300
---------- --------- ---------

JOSE   5000   5300
PEDRO 2850   3150
ANA    2450   2750
MARIA        2975   3275
PEDRO 1250   1550
PAULA 1600   1900
...
14 rows selected.
```

Usando Operadores Aritméticos

O exemplo no slide usa o operador de adição para calcular um aumento de salário de US$300 para todos os funcionários e exibe uma nova coluna SAL+300 na saída.

Observe que a coluna SAL+300 resultante do cálculo não é uma nova coluna na tabela EMP; ela é somente para exibição. Por default, o nome de uma coluna surge do cálculo que a criou — nesse caso, sal+300.

Observação: O SQL*Plus ignora espaços em branco antes e depois do operador aritmético.

Precedência do Operador

*	/	+	_

• A multiplicação e a divisão têm prioridade sobre a adição e a subtração.

• Os operadores com a mesma prioridade são avaliados da esquerda para a direita.

• Os parênteses são usados para forçar a avaliação priorizada e para esclarecer as instruções.

Precedência do Operador

Se uma expressão aritmética tiver mais de um operador, a multiplicação e a divisão serão avaliadas primeiro. Se os operadores dentro uma expressão tiverem a mesma prioridade, então a avaliação será realizada da esquerda para a direita.

Você pode usar os parênteses a fim de forçar a avaliação da expressão entre parênteses primeiro.

Precedência do Operador

```
SQL> SELECT NOME, SALARIO, 12*SALARIO+100
2     FROM  EMPREGADO;


NOME  SAL 12*SAL+100
---------- --------- ----------
JOSE  5000  60100
PEDRO 2850  34300
ANA   2450  29500
MARIA 2975  35800
PEDRO 1250  15100
PAULA 1600  19300
...

14 rows selected.
```

Precedência do Operador (continuação)

O exemplo no slide exibe o nome, o salário e a remuneração anual dos funcionários. Ele calcula a remuneração anual como 12 multiplicado pelo salário mensal, mais um bônus de US$100. Observe que a multiplicação é realizada antes da adição.

Observação: Use os parênteses para reforçar a ordem de procedência padrão e aumentar a compreensão. Por exemplo, a expressão acima pode ser criada como (12*sal)+100 sem que haja alteração no resultado.

Usando Parênteses

```
SQL> SELECT NOME, sal, 12*(sal+100)
2     FROM  EMPREGADO;


NOME  SAL 12*(SAL+100)
---------- --------- -----------
JOSE  5000  61200
PEDRO 2850  35400
ANA   2450  30600
MARIA 2975  36900
PEDRO 1250  16200
...

14 rows selected.
```

Usando Parênteses

Você pode sobrepor as normas de precedência usando parênteses para especificar a ordem de execução dos operadores.

O exemplo no slide exibe o nome, o salário e a remuneração anual dos funcionários. Ele calcula a remuneração anual como o salário mensal mais um bônus mensal de US$100, multiplicado por 12. Por causa dos parâmetros, a adição tem prioridade sobre a multiplicação.

Definindo um Valor Nulo

• Um valor nulo não está disponível, não é atribuído, é desconhecido ou não é aplicável.

• Um valor nulo não é o mesmo que um zero ou um espaço em branco.

```
SQL> SELECT NOME, OCUPACAO, SALARCIO, COMM
2     FROM  EMPREGADO;


NOME  OCUPACAO    SAL   COMM
---------- --------- --------- ---------
PEDRO PRESIDENTE  5000
PAULA SECRETARIA  2850
...
```

```
TURNER      SALESMAN    1500  0
...
14 rows selected.
```

Valores Nulos

Se faltar o valor de dados em uma linha de uma determinada coluna, diz-se que esse valor é nulo ou contém nulo.

Um valor nulo não está disponível, não é atribuído, é desconhecido ou não é aplicável. Um valor nulo não é o mesmo que um zero ou um espaço. O zero é um número e o espaço é um caractere.

As colunas de qualquer tipo de dados podem conter valores nulos, a menos que tenham sido definidas como NOT NULL ou como PRIMARY KEY ao serem criadas.

Na coluna COMM da tabela EMP, note que somente o SALESMAN pode ganhar comissão. Outros funcionários não estão autorizados a ganhar comissão. Um valor nulo representa esse fato. Turner, que é um vendedor, não ganha nenhuma comissão. Observe que sua comissão é zero e não nula.

Valores Nulos nas Expressões Aritméticas

Expressões aritméticas contendo um valor nulo avaliado como nulo.

```
SQL> select NOME, 12*SALARIO+comm
2     from  EMPREGADO
3     WHERE NOME='MARCIO';

NOME  12*SALARIO+COMM
---------- ------------
PEDRO
```

Valores Nulos (continuação)

Se qualquer valor da coluna em uma expressão aritmética for nulo, o resultado será nulo. Por exemplo, se você tentar executar uma divisão com zero, obterá um erro. No entanto, se dividir um número por nulo, o resultado será nulo ou desconhecido.

No exemplo do slide, o funcionário KING não está em SALESMAN e não receberá nenhuma comissão. Como a coluna COMM na expressão aritmética é nula, o resultado será nulo.

Para obter mais informações, consulte o Oracle Server SQL Reference, Release 8, "Elementsof SQL".

Definindo um Apelido de Coluna

• Renomeia um cabeçalho de coluna

• É útil para cálculos

• Segue imediatamente o nome da coluna; palavra-chave AS opcional entre o nome da coluna e o apelido.

• Necessita de aspas duplas caso contenha espaços ou caracteres especiais ou faça distinção entre maiúsculas e minúsculas

Apelidos de Coluna

Ao exibir o resultado de uma consulta, o SQL*Plus normalmente usa o nome da coluna selecionada como o cabeçalho da mesma. Em muitos casos, esse cabeçalho pode não ser descritivo e, desse modo,

de difícil compreensão. É possível alterar um cabeçalho de coluna usando um apelido da coluna.

Especifique o apelido após a coluna na lista SELECT usando um espaço como um separador. Por default, os cabeçalhos de apelidos aparecem em letras maiúsculas. Se o apelido possuir espaços, caracteres especiais (tais como # ou $) ou fizer distinção entre maiúsculas e minúsculas, coloque o apelido entre aspas duplas (" ").

Usando Apelidos de Coluna

```
SQL> SELECT NOME AS EMP, SALARIO SAL
2     FROM  EMPREGADO;


EMP    SAL
--------- -------------
...

SQL> SELECT NOME  "Nome",
2     SALARIO*12 "SALARIO ANUAL"
3     FROM  EMPREGADO;

NOME  SALARIO ANUAL
------------- -------------
...
```

Apelidos de Coluna (continuação)

O primeiro exemplo exibe o nome e o salário mensal de todos os funcionários. Note que a palavra- chave AS opcional foi usada antes do nome do apelido de coluna. O resultado da consulta deverá ser

o mesmo caso a palavra-chave AS seja usada ou não. Note também que a instrução SQL possui os apelidos de coluna, nome e salário, em letra minúscula, enquanto o resultado da consulta exibe os cabeçalhos da coluna em maiúscula. Conforme mencionado no último slide, os cabeçalhos de coluna aparecem, por default, em maiúscula.

O segundo exemplo exibe o nome e o salário anual de todos os funcionários. Como o Annual Salary possui espaços, ele foi incluído entre aspas duplas. Note que o cabeçalho da coluna na saída é exatamente o mesmo do apelido da coluna.

Operador de Concatenação

- Concatena colunas ou strings de caractere a outras colunas
- É representado por duas barras verticais (||)
- Cria uma coluna resultante que é uma expressão de caracteres

Operador de Concatenação

Você pode vincular colunas à outras colunas, expressões aritméticas ou valores constantes usando o operador de concatenação (||). As colunas em cada lado do operador são combinadas para formar uma coluna de saída única.

Usando um Operador de Concatenação

```
SQL> SELECT NOME||OCUPACAO AS "FUNCIONARIOS"
  2   FROM  EMPREGADO;
```

Operador de Concatenação (continuação)

No exemplo, NOME e OCUP estão concatenados e recebem o apelido FUNCIONARIOs. Note que o número
e cargo do funcionário são combinados para formar uma coluna de saída única.

A palavra-chave AS antes do nome do apelido torna a cláusula SELECT mais fácil de ser lida.

Strings Literais de Caracteres

• Uma literal é um caractere, um número ou uma data incluída na lista SELECT.

• Os valores literais de caractere e data devem estar entre aspas simples.

• Cada string de caractere é gerada uma vez para cada linha retornada.

Strings Literais de Caracteres

Uma literal é um caractere, um número ou uma data incluída na lista SELECT que não seja um nome ou apelido de coluna. Ela é impressa para cada linha retornada. Strings literais de formato de texto livre podem ser incluídas no resultado da consulta e são tratadas da mesma forma que uma coluna na lista SELECT.

As literais de caractere e data precisam estar entre aspas simples (' '); as literais de número, não.

Usando Strings Literais de Caracteres

```
SQL> SELECT NOME ||' é '||OCUPACAO
2     AS "DETALHE FUNCIONARIO"
3     FROM  EMPREGADO;


DETALHE FUNCIONARIO
---------------------------
JOSE é PRESIDENTE
PEDRO é GERENTE PAULA é GERENTE
ANA é GERENTE
GOMES é GERENTE
...
14 rows selected.
```

Strings Literais de Caracteres (continuação)

O exemplo do slide exibe os nomes e cargos de todos os funcionários. A coluna possui o cabeçalho FUNCIONARIO Details. Note

os espaços entre as aspas simples na instrução SELECT. Os espaços melhoram a legibilidade da saída.

No exemplo a seguir, o nome e o salário de cada funcionário estão concatenados a uma literal para dar mais sentido às linhas retornadas.

```
SQL> SELECT NOME ||': '||'1'||' SALARIO MENSAL = '||SALARIO
MENSAL
2     FROM  EMPREGADO;


MONTHLY
--------------------------------------------------------------
----
PEDRO: 1 Month salary = 5000
ANA: 1 Month salary = 2850
JOSE: 1 Month salary = 2450
ANDRE: 1 Month salary = 2975
MARIA: 1 Month salary = 1250
...
14 rows selected.
```

Linhas Duplicadas

A exibição default das consultas é de todas as linhas, incluindo linhas duplicadas.

```
SQL> SELECT COD_DEPTO
2     FROM  EMPREGADO;

COD_DEPTO
---------
10
30
10
20
...
14 rows selected.
```

Linhas Duplicadas

Exceto se indicado o contrário, o SQL*Plus exibe os resultados de uma consulta sem eliminar as linhas duplicadas. O exemplo do

slide exibe todos os números de departamento a partir da tabela EMP. Note que os números de departamento estão repetidos.

Eliminando Linhas Duplicadas

Elimine linhas duplicadas usando a palavra- chave DISTINCT na cláusula SELECT.

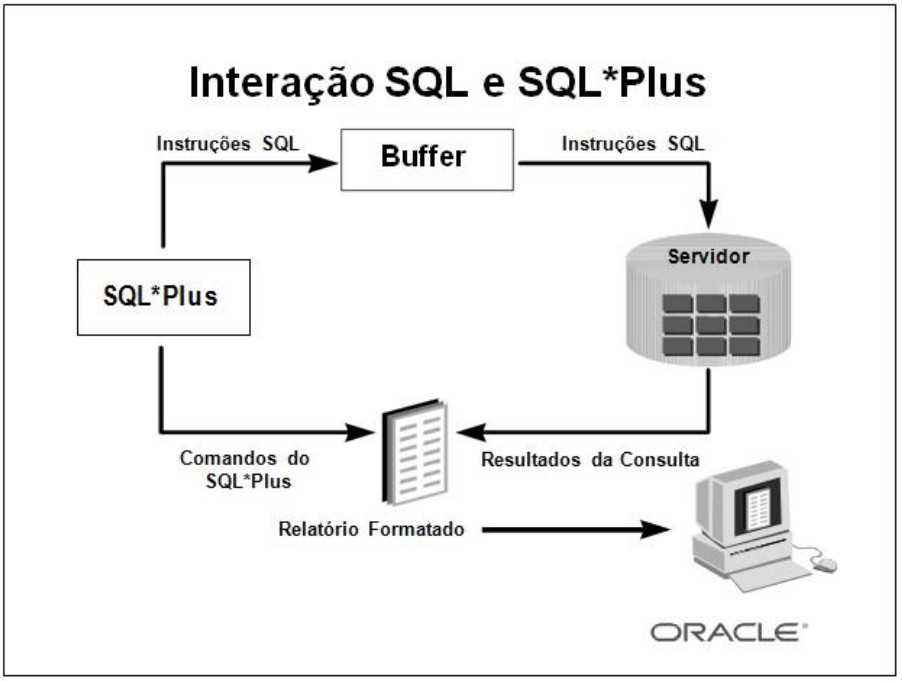
```
SQL> SELECT DISTINCT COD_DEPTO
2     FROM  EMPREGADO;

COD_DEPTO

---------

10

20

30
```

Linhas Duplicadas (continuação)

Para eliminar linhas duplicadas de um resultado, inclua a palavra-chave DISTINCT na cláusula SELECT logo após a palavra-chave SELECT. No exemplo do slide, a tabela EMP contém, na verdade, quatorze linhas, mas há somente três números de departamento exclusivos na tabela.

Você pode especificar várias colunas após o qualificador DISTINCT. O qualificador DISTINCT afeta todas as colunas selecionadas e o resultado representa uma combinação distinta das colunas.

```
SQL> SELECT DISTINCT COD_DEPTO, OCUPACAO
2     FROM  EMPREGADO;


COD_DEPTO   OCUPACAO
------      ---------
10          SECRETARIO
10          GERENTE
10          PRESIDENT
20          ANALISTA
...
9 rows selected.
```



SQL e SQL*Plus

SQL é uma linguagem de comando para comunicação com o Oracle Server a partir de qualquer ferramenta ou aplicação. O Oracle SQL possui muitas extensões. Quando você informa uma instrução SQL, ela é armazenada em uma parte da memória chamada buffer de SQL e permanece lá até que você informe uma nova instrução.

O SQL*Plus é uma ferramenta Oracle que reconhece e submete instruções SQL ao Oracle Server para execução e contém sua própria linguagem de comando.

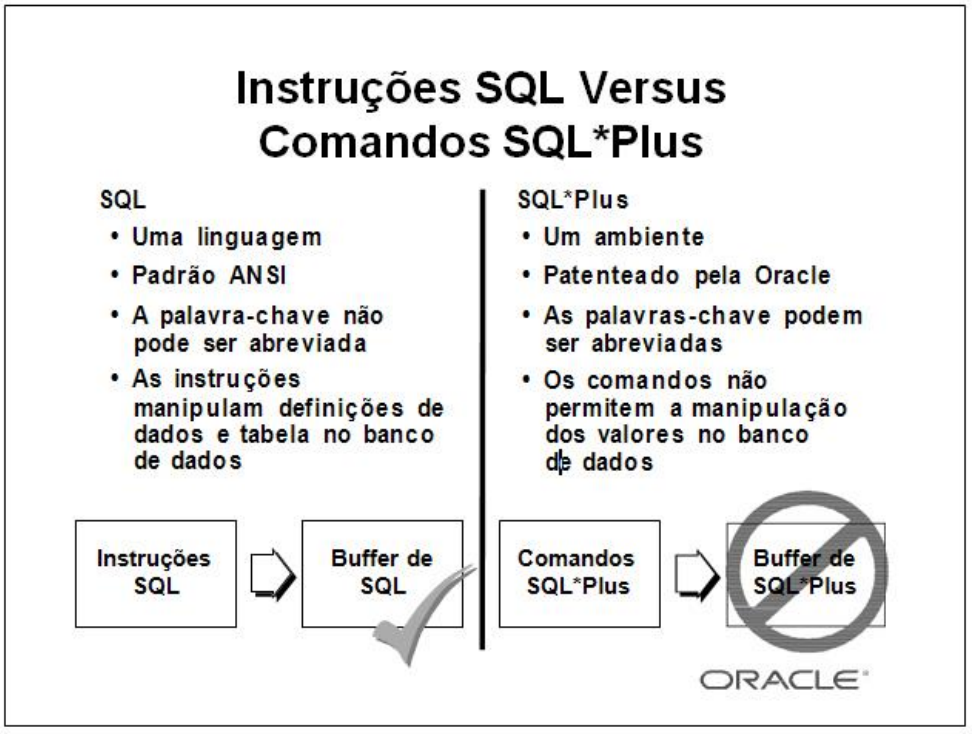
Recursos do Código SQL

- Podem ser utilizados por uma grande faixa de usuários, incluindo aqueles com pouca ou nenhuma experiência em programação
- É uma linguagem não procedural
- Reduz o período de tempo necessário para a criação e manutenção de sistemas
- É uma linguagem similar ao inglês

Recursos do SQL*Plus

- Aceita entrada ad hoc das instruções

- Aceita entrada SQL a partir dos arquivos
- Oferece um editor de linha para modificar instruções SQL
- Controla as configurações ambientais
- Formata os resultados da consulta em um relatório básico
- Acessa bancos de dados remotos e locais



SQL e SQL*Plus (continuação)
A tabela a seguir compara os códigos SQL e SQL*Plus:

SQL	SQL*Plus
É uma linguagem de comunicação com o Oracle Server para acesso aos dados.	Reconhece instruções SQL e as envia ao Servidor.
É baseada no padrão SQL da American National Standards Institute (ANSI).	É a interface patenteada da Oracle para execução de instruções SQL.
Manipula definições de dados e tabela no banco de dados.	Não permite a manipulação dos valores no banco de dados.
Digita-se no buffer de SQL em uma ou mais linhas.	Digita-se uma linha de cada vez; não armazenada no buffer de SQL.
Não possui caractere de continuação.	Possui um hífen (-) como caractere de continuação caso os comandos ultrapassem uma linha.
É possível abreviar.	Não é possível abreviar.
Usa um caracter de finalização para executar o comando imediatamente.	Não necessita de caracteres de finalização; os comandos são executados imediatamente.
Usa funções para executar algumas formatações.	Usa comandos para formatar dados.

Visão Geral do SQL*Plus

• Estabelecer login no SQL*Plus.

• Descrever a estrutura de tabela.

• Editar a instrução SQL.

• Executar o código SQL a partir do SQL*Plus.

• Salvar as instruções SQL em arquivos e anexar as instruções SQL a arquivos.

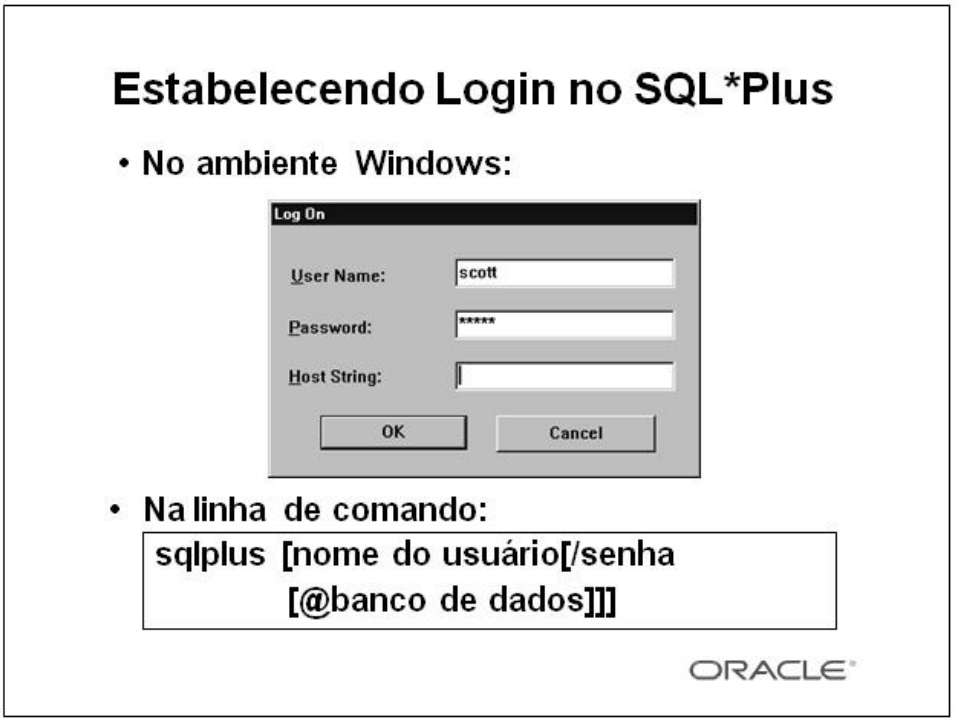
• Executar arquivos salvos.

• Carregar comandos a partir de arquivo para buffer e editá-los.

SQL*Plus
SQL*Plus é um ambiente no qual você pode realizar o seguinte:

- Executar instruções SQL para recuperar, modificar, adicionar e remover dados do banco de dados.
- Formatar, calcular, armazenar e imprimir resultados de consulta em formulários.
- Criar arquivos de script para armazenar instruções SQL para uso repetitivo no futuro. Os comandos SQL*Plus podem ser divididos nas seguintes categorias principais:

Categoria	**Objetivo**
Ambiente	Afeta o comportamento geral das instruções SQL para a seção.
Formato	Formata o resultado da consulta.
Manipulação de arquivo	Salva, carrega e executa arquivos de script.
Execução	Envia instruções SQL do buffer de SQL para o Oracle Server.
Editar	Modifica as instruções SQL no buffer.
Interação	Permite criar e passar variáveis para instruções SQL, imprimir valores de variáveis e imprimir mensagens na tela.
Diversos	Possui diversos comandos para conectar o banco de dados, manipular o ambiente SQL*Plus e exibir definições de coluna.



Estabelecendo Login no SQL*Plus

A forma de chamar o SQL*Plus depende do tipo de sistema operacional ou ambiente Windows que você está executando.

Para estabelecer login através de um ambiente Windows:

1.Clique em Iniciar —> Programas —> Oracle for Windows NT —> SQL*Plus 8.0.

2.Preencha o nome do usuário, a senha e o banco de dados.

Para estabelecer login através de um ambiente de linha de comando:

1.Estabeleça login na máquina.

2.Digite o comando SQL*Plus conforme mostrado no slide. No comando:

nome do usuário	é o seu nome de usuário do banco de dados
senha	é a sua senha do banco de dados (se digitar sua senha aqui, ela estará visível)
@banco de dados	é a string de conexão do banco de dados

Observação: Para garantir a integridade da senha, não a digite no prompt do sistema operacional. Em vez disso, digite somente o nome de usuário. Digite a senha no prompt Password.

Uma vez estabelecido o login corretamente no SQL*Plus, você verá a seguinte mensagem:

```
SQL*Plus Release 8.0.3.0.0 - Produzido em Ter Jun 22
16:03:43 2008 (c) Copyright 2008 Oracle Corporation.
(Release 8.0.3.0.0 - Production on Tue Jun 22 16:03:43 2008
(c) Copyright 2008 Oracle Corporation. Todos os direitos
reservados. (All rights reserved).
```

Exibindo a Estrutura de Tabela

Use o comando DESCRIBE do SQL*Plus para exibir a estrutura de uma tabela.

```
DESC[RIBE] nome da tabela
```

Exibindo a Estrutura de Tabela

No SQL*Plus, é possível exibir a estrutura de uma tabela usando o comando DESCRIBE. O resultado do comando é para ver os nomes da coluna e tipos de dados, assim como se uma coluna deve conter dados.

Na sintaxe:

nome da tabela é o nome de qualquer tabela, view ou sinônimo existente disponível para o usuário

Exibindo a Estrutura de Tabela

```
SQL> DESCRIBE DEPARTAMENTO


Name         Null?       Type
-----------  --------     ------------
COD_DEPTO    NOT NULL    NUMBER(2)
DNAME                    VARCHAR2(14)
LOC                      VARCHAR2(13)
```

Exibindo a Estrutura de Tabela (continuação)

O exemplo do slide exibe as informações sobre a estrutura da tabela DEPT. No resultado:

Null?	indica se uma coluna deve conter dados; NOT NULL indica que uma coluna deve conter dados
Type	exibe o tipo de dados de uma coluna

Os tipos de dados são descritos na tabela a seguir:

Tipo de dado	**Descrição**
NUMBER(*p*,*s*)	Valor numérico que possui um número máximo de dígitos *p*, o número de dígitos à direita do ponto decimal *s.*
VARCHAR2(*s*)	Valor de caracteres com comprimento variável do tamanho máximo *s.*
DATE	Valor de data e hora entre 1 de janeiro, 4712 A.C. e 31 de dezembro, 9999 D.C.
CHAR(*s*)	Valores de caracteres com comprimento fixo do tamanho *s.*

Comandos de Edição do SQL*Plus

- A[PPEND] texto
- C[HANGE] / antigo / novo
- C[HANGE] / texto /
- CL[EAR] BUFF[ER]
- DEL
- DEL n
- DEL m n

Comandos de Edição do SQL*Plus
Os comandos SQL*Plus são digitados em uma linha de cada vez e não são armazenados no buffer de SQL.

Comando	Descrição
A[PPEND] *texto*	Adiciona texto no final da linha atual.
C[HANGE] / *antigo* / *novo*	Altera o texto *antigo* para o *novo* na linha atual.
C[HANGE] / *texto* /	Deleta o *texto* da linha atual.
CL[EAR] BUFF[ER]	Deleta todas as linhas a partir do buffer de SQL.
DEL	Deleta a linha atual.

Diretrizes

• Ao pressionar [Return] antes de completar o comando, o SQL*Plus informa o número da linha incompleta.
• Finalize o buffer de SQL digitando um dos caracteres finalizadores (ponto-e-vírgula ou barra) ou pressionando duas vezes [Return]. Em seguida, aparecerá o prompt SQL.

Comandos de Edição do SQL*Plus

• I[NPUT]

• I[NPUT] texto

• L[IST]

• L[IST] n

• L[IST] m n

• R[UN]

• n

• n texto

• 0 texto

Comandos de Edição do SQL*Plus (continuação)

Comando	Descrição
I[NPUT]	Insere um número indefinido de linhas.
I[NPUT] *texto*	Insere uma linha consistindo em *texto.*
L[IST]	Lista todas as linhas no buffer de SQL.
L[IST] *n*	Lista uma linha (especificada pelo *n*).
L[IST] *m n*	Lista uma faixa de linhas (de *m* a *n*).
R[UN]	Exibe e executa a instrução SQL atual no buffer.
n	Especifica a linha que deve ser tornar a linha atual.
n texto	Substitui a linha *n* pelo *texto*.
0 *texto*	Insere uma linha antes da linha 1.

Você pode digitar somente um comando SQL*Plus por prompt SQL. Os comandos SQL*Plus não ficam armazenados no buffer. Para continuar um comando SQL*Plus na próxima linha, finalize a linha atual com um hífen (-).

Comandos de Arquivo do SQL*Plus

- SAVE nome de arquivo
- GET nome de arquivo
- START nome de arquivo
- @ nome de arquivo
- EDIT nome de arquivo
- SPOOL nome de arquivo
- EXIT

Comandos de Arquivo do SQL*Plus

As instruções SQL comunicam-se com o Oracle Server. Os comandos SQL*Plus controlam o ambiente, formatam os resultados de consulta e gerenciam arquivos. Você pode usar os comandos identificados na tabela a seguir:

Comando	Descrição
SAV[E] *nome de arquivo* [.ext] [REP[LACE]APP[END]]	Salva o contéudo atual do buffer de SQL para um arquivo. Use APPEND para adicionar um arquivo existente; use REPLACE para substituir um arquivo existente. A extensão default é .sql.
GET *nome de arquivo* [.ext]	Salva o conteúdo de um arquivo salvo anteriormente para o buffer de SQL. A extensão default para o nome de arquivoé .sql.
STA[RT] *nome de arquivo* [.ext]	Executa um arquivo de comando salvo anteriormente.
@ nome de arquivo	Executa um arquivo de comando salvo anteriormente (o mesmo que START).
ED[IT]	Chama o editor e salva o conteúdo do buffer para um arquivo chamado afiedt.buf.
ED[IT] [*nome de arquivo* [.ext]]	Chama o editor para editar o conteúdo de um arquivo salvo.
SPO[OL] [*nome de arquivo* ext]| OFF|OUT]	Armazena os resultados da consulta em um arquivo. OFF fechao arquivo periférico. OUT fecha o arquivo periférico e envia o os resultados do arquivo para a impressora do sistema.
EXIT	Sai do código SQL*Plus.

Sumário

```
SELECT      [DISTINCT] {*,coluna    [apelido],...}
FROM  tabela;
```

Use o SQL*Plus como um ambiente para:

- Executar instruções SQL
- Editar instruções SQL

Instrução SELECT

Nesta lição, você aprendeu sobre a recuperação de dados de uma tabela de banco de dados com a instrução SELECT.

```
SELECT      [DISTINCT] {*,coluna    [apelido],...}
FROM  tabela;
```

onde:	SELECT	é uma lista de pelo menos uma coluna.
	DISTINCT	suprime as duplicatas.
	*	seleciona todas as colunas.
	coluna	seleciona a coluna nomeada.
	apelido	dá um cabeçalho diferente à coluna selecionada.
	FROM *tabela*	especifica a tabela contendo as colunas.

SQL*Plus

SQL*Plus é um ambiente de execução que pode ser usado para enviar instruções SQL ao servidor do banco de dados, editar e salvar as instruções SQL. As instruções podem ser executadas a partir do prompt SQL ou de um arquivo de script.

Visão Geral do Exercício

• Selecionando todos os dados a partir de tabelas diferentes

• Descrevendo a estrutura de tabelas

• Executando cálculos aritméticos e especificando nomes de coluna

• Usando o editor do SQL*Plus

Visão Geral do Exercício

Esse é o primeiro de muitos exercícios. As soluções (caso necessite delas) podem ser encontradas no Anexo A. Os exercícios têm a intenção de apresentar todos os tópicos abordados nesta lição. As questões 2 a 4 são impressas.

Em qualquer exercício, pode haver questões do tipo "se você tiver tempo" ou "se quiser mais desafios". Faça-as somente se tiver concluído todas as outras questões dentro do tempo alocado e desejar mais desafios às suas habilidades.

Realize o exercício devagar e com exatidão. É possível exercitar-se salvando e executando arquivos de comando. Se tiver qualquer pergunta, chame o instrutor.

Questões Impressas

Para as questões de 2 a 4, marque Falso ou Verdadeiro.

Exercício 1

1. Inicie uma sessão SQL*Plus usando um ID e uma senha de usuário fornecidos pelo instrutor.
2. Os comandos SQL*Plus acessam o banco de dados.
Verdadeiro/Falso
3. A instrução SELECT será executada corretamente?
Verdadeiro/Falso

```
SQL> SELECT NOME, OCUPACAO, SALARIO SAL
  2   FROM  EMPREGADO;
```

4. A instrução SELECT será executada corretamente?
Verdadeiro/Falso

```
SQL> SELECT *
  2   FROM  EMPREGADO;
```

5. Há quatro erros de codificação nesta instrução. Você pode identificá-los?

```
SQL> SELECT COD_EMPREGADO, NOME
  2   SALARIO * 12 SALARIO ANUAL
  3   FROM  EMPREGADO;
```

6. Mostre a estrutura da tabela DEPARTAMENTO. Selecione todos os dados da tabela DEPARTAMENTO

```
Name         Null?       Type
------------ --------    ------------
COD_DEPTO    NOT NULL    NUMBER(2)
NOME                     VARCHAR2(14)
LOCAL                    VARCHAR2(13)
COD_DEPTO          NOME             LOCAL
---------          ----------       -------------
10                 ANALISTA         PORTO ALEGRE
20                 PROGRAMADOR      SAPIRANGA
30                 ANALISTA         SÃO LEO
40                 PROGRAMADOR      IVOTI
```

Capítulo 03

Restringindo e Classificando Dados

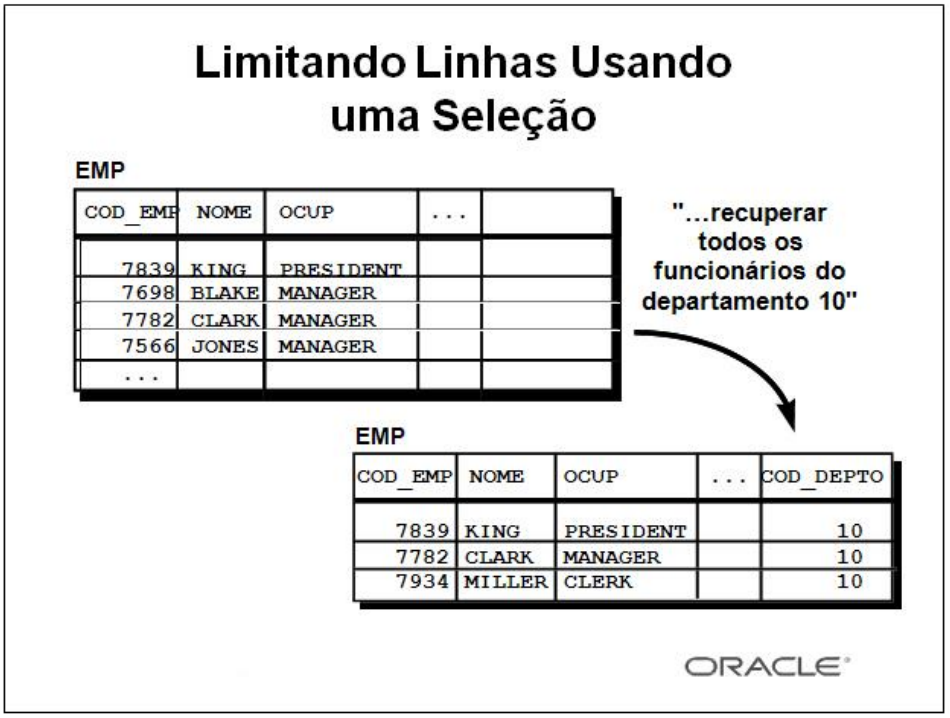
Objetivos

Depois de completar esta lição, você poderá fazer o seguinte:

- Limitar linhas recuperadas por uma consulta
- Classificar linhas recuperadas por uma consulta

Objetivo da Lição

Ao recuperar dados do banco de dados, pode ser preciso restringir as linhas de dados exibidas ou especificar a ordem de exibição das mesmas. Essa lição explica as instruções SQL que você utiliza para executar essas ações.



Limitando Linhas Usando uma Seleção

No exemplo do slide, suponha que você deseje exibir todos os funcionários do departamento 10.

O conjunto de linhas realçadas com um valor 10 na coluna COD_DEPTO são as únicas retornadas. Esse método de restrição é a base da cláusula WHERE na linguagem SQL.

Limitando Linhas Selecionadas

• Restringe as linhas retornadas usando a cláusula WHERE.

```
SELECT      [DISTINCT] {*| coluna [apelido], ...}
FROM        tabela
[WHERE      condição(ões)];
```

• A cláusula WHERE segue a cláusula FROM.

Limitando Linhas Selecionadas

É possível restringir as linhas retornadas da consulta utilizando a cláusula WHERE. Uma cláusula WHERE contém uma condição que deve coincidir e seguir diretamente a cláusula FROM. Na sintaxe:

WHERE restringe a consulta às linhas que atendem uma condição.

condição é composta por nomes de colunas, expressões, constantes e um operador de comparação.

A cláusula WHERE pode comparar valores em colunas, valores literais, expressões aritméticas ou funções. A cláusula WHERE é formada por três elementos:

- Nome de coluna
- Operadores de comparação
- Nome da coluna, constante ou lista de valores

Usando a Cláusula WHERE

```
SQL> SELECT NOME, OCUP, COD_DEPTO
2     FROM  EMPREGADO
3     WHERE OCUP='CLERK';


NOME        OCUP        COD_DEPTO
----------  ---------   ---------

JAMES       CLERK       30
SMITH       CLERK       20
ADAMS       CLERK       20
MILLER      CLERK       10
```

Usando a Cláusula WHERE

No exemplo, a instrução SELECT recupera o nome, o cargo e o número do departamento de todos os funcionários cujo cargo é CLERK.

Observe que o cargo CLERK foi especificado em letras maiúsculas para garantir que a correspondência seja feita com a coluna do cargo na tabela EMPREGADO. As strings de caractere não fazem distinção entre maiúsculas de minúsculas.

Strings de Caractere e Datas

• As strings de caractere e valores de data aparecem entre aspas simples.

• Os valores de caractere fazem distinção entre maiúsculas e minúsculas e os valores de data diferenciam formatos.

• O formato de data default é DD-MON-YY.

```
SQL> SELECT NOME, OCUP, COD_DEPTO
  2   FROM  EMPREGADO

  3   WHERE NOME ='JAMES';
```

Strings de Caractere e Datas

Strings de caractere e datas na cláusula WHERE devem estar entre aspas simples (''). Constantes de número, no entanto, não precisam.

Todas as pesquisas de caractere fazem distinção entre maiúsculas de minúsculas. No exemplo a seguir, nenhuma linha retornou pois a tabela EMPREGADO armazena todas as datas em letras maiúsculas:

```
SQL> SELECT NOME, COD_EMP, OCUP, COD_DEPTO
  2   FROM  EMPREGADO
  3   WHERE OCUP='clerk';
```

Os produtos da Oracle armazenam datas em um formato numérico interno, representando o século, ano, mês, dia, horas,

minutos e segundos. A exibição da data default é DD-MON-YY. Observação: A alteração do formato da data default será abordada na lição subseqüente. Valores de número não aparecem entre aspas.

Operadores de Comparação

Operador	Significado
=	Igual a
>	Maior do que
>=	Maior do que ou igual a
<	Menor do que
<=	Menor ou igual a
<>	Diferente de

Operadores de Comparação

Usa-se os operadores de comparação em condições que comparam uma expressão a outra. Eles são usados na cláusula WHERE no seguinte formato:

Sintaxe

```
… WHERE expr valor operador
```

Exemplos

```
… WHERE hiredate='01-JAN-95'

… WHERE sal>=1500

… WHERE NOME='SMITH'
```

Usando Operadores de Comparação

```
SQL> SELECT NOME, sal, comm
  2   FROM  EMPREGADO
  3   WHERE sal<=comm;


NOME         SAL    COMM
---------- --------- ---------
MARTIN       1250  1400
```

Usando Operadores de Comparação

No exemplo, a instrução SELECT recupera o nome, salário e comissão da tabela EMPREGADO, em que o salário do funcionário é menor ou igual à quantia da comissão. Observe que não há valor explícito fornecido para a cláusula WHERE. Os dois valores que estão sendo comparados são retirados das colunas SAL e COMM na tabela EMPREGADO.

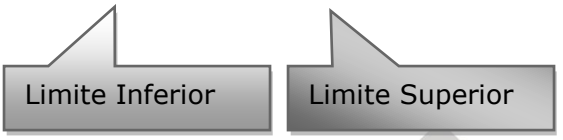
Outros Operadores de Comparação

Operador	Significado
BETWEEN ...AND...	Entre dois valores (inclusive)
IN(list)	Vincula qualquer um de uma lista de valores
LIKE	Vincula um padrão de caractere
IS NULL	É um valor nulo

Usando o Operador BETWEEN

Use o operador BETWEEN para exibir linhas baseadas em uma faixa de valores.

```
SQL> SELECT       NOME, sal
  2   FROM        EMPREGADO
  3   WHERE       sal BETWEEN 1000  AND   1500;

NOME         SAL
---------- ---------
MARTIN       1250
TURNER       1500
WARD         1250
ADAMS        1100
MILLER       1300
```



O Operador BETWEEN

Você pode exibir linhas baseadas em uma faixa de valores usando o operador BETWEEN. A faixa que você especificar possuirá uma faixa inferior e uma superior.

A instrução SELECT no slide retorna as linhas da tabela EMPREGADO para qualquer funcionário cujo salário esteja entre US$1.000 e US$1.500.

Valores especificados com o operador BETWEEN são inclusivos. Você deve especificar primeiro o limite inferior.

Usando o Operador IN

```
Use o operador  IN para testar os valores de uma lista.

SQL> SELECT       COD_EMP, NOME, sal, mgr
  2   FROM        EMPREGADO
  3   WHERE       mgr IN      (7902,7566, 7788);

COD_EMP     NOME         SAL         MGR
--------- ---------- --------- ---------

7902        FORD        3000        7566
7369        SMITH       800         7902
7788        SCOTT       3000        7566
7876        ADAMS       1100        7788
```

O Operador IN

Para testar os valores em uma determinada lista, use o operador IN.

O exemplo do slide exibe o número do funcionário, o nome, o salário e o número de funcionário do gerente de todos os funcionários cujo número de funcionário do gerente for 7902, 7566 ou 7788.

O operador IN pode ser usado com qualquer tipo de dados. O exemplo seguinte retorna uma linha da tabela EMPREGADO para qualquer funcionário cujo nome estiver incluído na lista de nomes na cláusula WHERE:

```
SQL> SELECT COD_EMP, NOME, mgr, COD_DEPTO
  2  FROM   EMPREGADO
  3  WHERE  NOME IN     ('FORD'     , 'ALLEN');
```

Se forem utilizados caracteres ou datas na lista, eles devem estar entre aspas simples ('').

Usando o Operador LIKE

• Use o operador LIKE para executar pesquisas curinga de valores de string de pesquisa válidos.

• As condições de pesquisa podem conter caracteres literais ou números.

– % denota zero ou muitos caracteres.

– _ denota um caractere.

```
SQL> SELECT NOME
  2   FROM  EMPREGADO
  3   WHERE NOME LIKE   'S%';
```

O Operador LIKE

Talvez nem sempre você saiba o valor exato pelo qual procurar. É possível selecionar linhas que vinculem um padrão de caractere usando o operador LIKE. A operação de vinculação de um padrão

de caractere refere-se a uma pesquisa de curinga. Dois símbolos podem ser utilizados para construir a string de pesquisa.

Símbolo	Descrição
%	Representa qualquer sequência de zero ou mais caracteres.
_	Representa qualquer caractere único.

A instrução SELECT acima retorna o nome do funcionário da tabela EMPREGADO para qualquer funcionário cujo nome começa com "S". Note o "S". maiúsculo. Os nomes iniciados com "s" não retornarão.

O operador LIKE pode ser usado como um atalho para algumas comparações BETWEEN. O exemplo a seguir exibe os nomes as datas de admissão de todos os funcionários admitidos entre janeiro e dezembro de 1981:

```
SQL>  SELECT      NOME, hiredate
  2   FROM        EMPREGADO
  3   WHERE       hiredate LIKE     '%1981';
```

Usando o Operador LIKE

• Você pode combinar caracteres de vinculação de padrão.

```
SQL> SELECT NOME
  2   FROM  EMPREGADO
  3   WHERE NOME LIKE   '_A%';

NOME
----------
MARTIN
JAMES
WARD
```

• É possível usar o identificador ESCAPE para procurar por "%" ou "_".

Combinando Caracteres Curinga

Os símbolos % e _ podem ser usados em qualquer combinação com caracteres literais. O exemplo do slide exibe os nomes de todos os funcionários cujos nomes possuem a letra "A" como segundo caractere.

A Opção ESCAPE

Quando for necessário ter uma correspondência exata para os caracteres '%' e '_' reais, use a opção ESCAPE. Essa opção especifica qual é o caractere ESCAPE. Caso possua HEAD_QUARTERS como um nome de departamento, deve procurar por ele utilizando a seguinte instrução SQL:

```
SQL> SELECT * FROM dept
  2   WHERE dname LIKE '%\_%'  ESCAPE '\';

COD_DEPTO   DNAME LOC
--------- -------------- -------------
    50      HEAD_QUARTERS     ATLANTA
```

A opção ESCAPE identifica a barra invertida (\) como o caractere de escape. No padrão, o caractere de escape vem antes do sublinhado (_). Isso faz com que o Oracle Server interprete o sublinhado literalmente.

Usando o Operador IS NULL

Teste para valores nulos com o operador IS NULL.

```
SQL> SELECT NOME, mgr
  2   FROM  EMPREGADO
  3   WHERE mgr IS      NULL;

NOME        MGR
---------- ---------
KING
```

O Operador IS NULL

O operador IS NULL testa valores que são nulos. Um valor nulo significa que o valor não está disponível, não-atribuído, desconhecido ou não-aplicável. Assim, não é possível testar com (=) porque um valor nulo não pode ser igual ou desigual a qualquer valor. O exemplo do slide recupera o nome e o gerente de todos os funcionários que não possuem um gerente.

Por exemplo, para exibir o nome, o cargo e a comissão de todos os funcionários que não estão nomeados para obter uma comissão, use a seguinte instrução SQL:

```
SQL> SELECT NOME, OCUP, comm
  2   FROM  EMPREGADO
```

```
3   WHERE comm  IS    NULL;
```

```
NOME         OCUP         COMM
-------- ----------- ------
KING         PRESIDENT
BLAKE        MANAGER
CLARK        MANAGER
...
```

Operadores Lógicos

Operador	Significado
AND	**Retorna TRUE se as condições de componentes forem TRUE**
OR	**Retorna TRUE se cada condição de componente for TRUE**
NOT	**Retorna TRUE se a condição seguinte for FALSE**

Operadores Lógicos

Um operador lógico combina o resultado de duas condições de componente para produzir um único resultado com base neles ou inverter o resultado para uma condição única. Três operadores lógicos estão disponíveis no SQL:

- AND
- OR
- NOT

Todos os exemplos até aqui especificaram somente uma condição na cláusula WHERE. Você pode usar várias condições em uma cláusula WHERE usando operadores AND e OR.

Usando o Operador AND

AND exige que ambas as condições sejam TRUE.

```
SQL> SELECT COD_EMP, NOME, OCUP, sal
  2   FROM  EMPREGADO
  3   WHERE sal>=1100
  4   AND   OCUP='CLERK';

COD_EMP     NOME        OCUP        SAL
--------- ---------- --------- ---------
7876        ADAMS       CLERK       1100
7934        MILLER      CLERK       1300
```

O Operador AND

No exemplo, as duas condições devem ser verdadeiras para cada registro a ser selecionado. Assim, um funcionário que possua o cargo CLERK e receba mais de US$1.100 será selecionado.

Todas as pesquisas de caractere fazem distinção entre maiúsculas de minúsculas. Nenhuma linha retornará se CLERK não estiver em letra maiúscula. As strings de caractere devem estar entre aspas.

Tabela de Verdade AND

A tabela a seguir mostra os resultados da combinação de duas expressões com AND:

AND	**TRUE**	**FALSE**	**NULL**
TRUE	TRUE	FALSE	NULL
FALSE	FALSE	FALSE	FALSE
NULL	NULL	FALSE	NULL

Usando o Operador OR

```
OR exige que cada condição seja TRUE.
SQL> SELECT COD_EMP, NOME, OCUP, sal
2     FROM  EMPREGADO
3     WHERE sal>=1100
4     OR    OCUP='CLERK';


COD_EMP     NOME        OCUP        SAL
---------   ----------  ---------   ---------
7839        KING        PRESIDENT   5000
7698        BLAKE       MANAGER     2850
7782        CLARK       MANAGER     2450
7566        JONES       MANAGER     2975
7654        MARTIN      SALESMAN    1250
...
7900 JAMES  CLERK 950
...
14 rows selected.
```

O Operador OR

No exemplo, cada condição pode ser verdadeira para qualquer registro a ser selecionado. Assim, um funcionário que possua o cargo CLERK ou que receba mais de US$1.100 será selecionado.

A Tabela de Verdade OR

A tabela a seguir mostra os resultados da combinação de duas expressões com OR:

OR	**TRUE**	**FALSE**	**NULL**
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	NULL
NULL	TRUE	NULL	NULL

Usando o Operador NOT

```
SQL> SELECT NOME, OCUP
2     FROM  EMPREGADO
3     WHERE OCUP NOT    IN
      ('CLERK','MANAGER','ANALYST');


NOME        OCUP
---------- ---------
KING        PRESIDENT
MARTIN      SALESMAN
ALLEN       SALESMAN
TURNER      SALESMAN
WARD        SALESMAN
```

O Operador NOT

O exemplo do slide exibe o nome e o cargo de todos os funcionários que não possuem os cargos CLERK, MANAGER ou ANALYST.

A Tabela de Verdade NOT

A tabela a seguir mostra o resultado da aplicação do operador NOT para uma condição:

NOT	TRUE	FALSE	NULL
TRUE	FALSE	TRUE	NULL

Observação: O operador NOT pode ser utilizado também com outros operadores SQL, como BETWEEN, LIKE e NULL.

```
... WHERE   OCUP  NOT   IN ('CLERK', 'ANALYST')
... WHERE   sal   NOT   BETWEEN     1000 AND    1500
... WHERE   NOME  NOT   LIKE '%A%'
... WHERE   comm  IS    NOT   NULL
```

Regras de Precedência

Ordem de Avaliação	Operador
1	Todos os operadores de comparação
2	NOT
3	AND
4	OR

Sobreponha regras de precedência usando parênteses.

Regras de Precedência

```
SQL> SELECT NOME, OCUP, sal
  2   FROM  EMPREGADO
  3   WHERE OCUP='SALESMAN'
  4   OR    OCUP='PRESIDENT'
  5   AND   sal>1500;


NOME        OCUP        SAL
----------  ---------   ---------
KING        PRESIDENT   5000
MARTIN      SALESMAN    1250
ALLEN       SALESMAN    1600
TURNER      SALESMAN    1500
WARD        SALESMAN    1250
```

Exemplo de Precedência do Operador AND

No exemplo do slide, há duas condições:

- A primeira condição é que o cargo seja PRESIDENT e o salário maior que 1500.
- A segunda condição é que o cargo seja SALESMAN. Portanto, a instrução SELECT faz a leitura do seguinte modo:

"Selecione a linha se o funcionário for um PRESIDENT e receber mais que US$1.500 ou se o funcionário for um SALESMAN".

Regras de Precedência

Use parênteses para forçar a prioridade.

```
SQL> SELECT NOME, OCUP, sal
  2   FROM  EMPREGADO
  3   WHERE (OCUP='SALESMAN'
  4   OR    OCUP='PRESIDENT')
  5   AND   sal>1500;


NOME        OCUP        SAL
----------  ---------   ---------
KING        RESIDENT    5000
ALLEN       SALESMAN    1600
```

Usando Parênteses
No exemplo, há duas condições:
- A primeira condição é que o cargo seja PRESIDENT ou SALESMAN.
- A segunda é que o salário sejam maior que 1500. Portanto, a instrução SELECT faz a leitura do seguinte modo:

"Selecione a linha se um funcionário for um PRESIDENT ou um SALESMAN e se o funcionário receber mais de US$1500".

Cláusula ORDER BY

• Classificar as linhas com a cláusula ORDER BY
– ASC: ordem crescente, default
– DESC: ordem decrescente
• A cláusula ORDER BY vem depois na instrução SELECT.

```
SQL> SELECT NOME, OCUP, COD_DEPTO, hiredate
  2   FROM  EMPREGADO
  3   ORDER BY hiredate;

NOME        OCUP        COD_DEPTO HIREDATE
----------  ---------   --------- ---------
SMITH       CLERK       20 17-DEC-80
ALLEN       SALESMAN    30 20-FEB-81
...
14 rows selected.
```

A Cláusula ORDER BY

A ordem das linhas retornadas em um resultado de consulta é indefinida. A cláusula ORDER BY pode ser utilizada para classificar as linhas. Se você usar a cláusula ORDER BY, deve colocá-la por último. É possível especificar uma expressão ou um apelido para classificação.

Sintaxe

```
SELECT      expr
FROM        tabela
[WHERE      condição(ões)]
[ORDER BY   {coluna, expr} [ASC|DESC]];
```

onde: ORDER BY especifica a ordem em que as linhas recuperadas são exibidas

ASC ordena as linhas na ordem crescente (essa é a ordem default)

DESC ordena as linhas na ordem decrescente

Se a cláusula ORDER BY não for usada, a ordem de classificação será indefinida e o Oracle Server talvez não extraia as linhas na mesma ordem ao realizar a mesma consulta duas vezes. Use a cláusula

ORDER BY para exibir as linhas em uma ordem específica.

Classificando em Ordem Decrescente

```
SQL> SELECT NOME, OCUP, COD_DEPTO, hiredate
  2   FROM  EMPREGADO
  3   ORDER BY hiredate DESC;


NOME        OCUP        COD_DEPTO   HIREDATE
----------  ---------   ---------   ---------
ADAMS       CLERK       20          12-JAN-83
SCOTT       ANALYST     20          09-DEC-82
MILLER      CLERK       10          23-JAN-82
JAMES       CLERK       30          03-DEC-81
FORD        ANALYST     20          03-DEC-81
KING        PRESIDENT   10          17-NOV-81
MARTIN      SALESMAN    30          28-SEP-81
...
```

```
14 rows selected.
```

Ordenação de Dados Default

A ordem de classificação default é crescente:

•Valores numéricos são exibidos primeiro com os valores mais baixos — por exemplo, 1–999.

•Valores de datas são exibidos primeiro com os valores mais recentes — por exemplo,

01-JAN-92 antes de 01-JAN-95.

•Valores de caracteres são exibidos em ordem alfabética — por exemplo, o A primeiro e o Z por último.

•Os valores nulos são exibidos por último em sequências ascendentes e primeiro em sequências descendentes.

Invertendo a Ordem Default

Para inverter a ordem de exibição das linhas, especifique a palavra-chave DESC após o nome da coluna na cláusula ORDER BY. O exemplo do slide classifica o resultado pelo funcionário contratado mais recentemente.

Classificando por Apelido de Coluna

```
SQL> SELECT COD_EMP, NOME, sal*12 annsal

COD_EMP     NOME       ANNSAL
---------   ---------- ---------
7369        SMITH      9600
7900        JAMES      11400
7876        ADAMS      13200
7654        MARTIN     15000
7521        WARD       15000
7934        MILLER     15600
7844        TURNER     18000
...

14 rows selected.
```

Classificando por Apelidos de Coluna

Você pode usar um apelido de coluna na cláusula ORDER BY. O exemplo do slide classifica os dados por salário anual.

Classificando por Várias Colunas

• A ordem da lista ORDER BY é a ordem de classificação.

```
SQL> SELECT NOME, COD_DEPTO, sal
  2   FROM  EMPREGADO
  3   ORDER BY COD_DEPTO, sal DESC;


NOME        COD_DEPTO   SAL
----------  ---------   ---------
KING        10          5000
CLARK       10          2450
MILLER      10          1300
FORD        20          3000
...
14 rows selected.
```

• Você pode classificar por uma coluna que não esteja na lista SELECT.

Classificando por Várias Colunas

Você pode classificar os resultados da consulta por mais de uma coluna. O limite de classificação é o número de colunas de uma determinada tabela.

Na cláusula ORDER BY, especifique as colunas e separe seus nomes usando vírgulas. Se deseja inverter a ordem de uma coluna, especifique DESC após seu nome. É possível ordenar por colunas que não estão incluídas na cláusula SELECT.

Exemplo

Exiba o nome e o salário de todos os funcionários. Ordene o resultado por número de departamento e, em seguida, em ordem decrescente de salário.

```
SQL> SELECT NOME, sal
  2   FROM  EMPREGADO
  3   ORDER BY    COD_DEPTO, sal    DESC;
```

Sumário

```
SELECT      [DISTINCT] {*| coluna [apelido], ...}
FROM        tabela
[WHERE      condição(ões)]
```

```
[ORDER BY   {coluna, expr, apelido} [ASC|DESC]];
```

Sumário

Nesta lição, você aprendeu sobre a restrição e classificação de colunas retornadas pela instrução
SELECT. Também aprendeu como implementar vários operadores.

Capítulo 04

Funções de Uma Única Linha

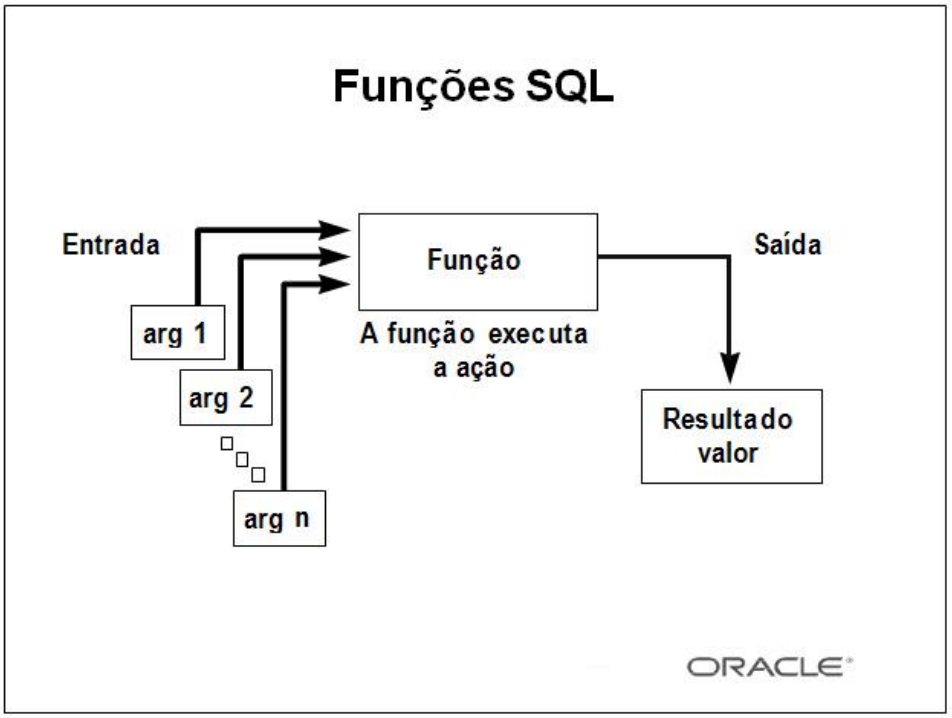
Objetivos

Depois de completar esta lição, você poderá fazer o seguinte:

- Descrever vários tipos de funções disponíveis no SQL
- Usar funções de data, número e caractere nas instruções SELECT
- Descrever o uso das funções de conversão

Objetivo da Lição

As funções formam o bloco de consulta básico mais avançado e são usadas para manipular valores de dados. Esta é a primeira de duas lições a explorar funções. Serão abordadas as funções de caractere de uma única linha, de número e de datas, bem como aquelas funções que convertem dados de um tipo para outro — por exemplo, dados de caractere para numérico.



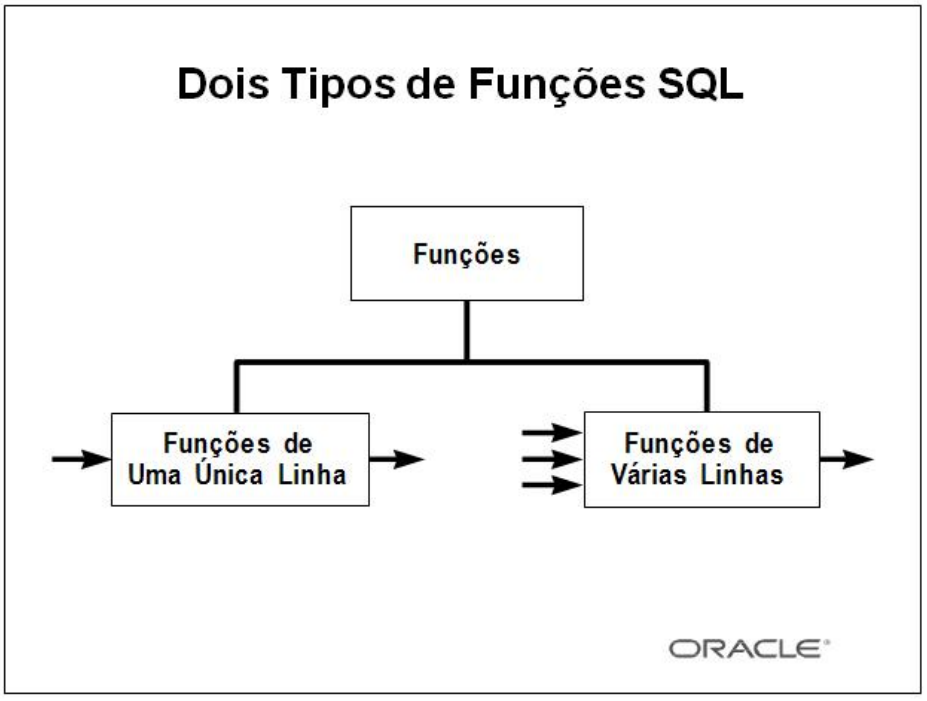
Funções SQL

As funções são um recurso avançado do SQL e podem ser usadas para realizar o seguinte:

- Executar cálculos usando dados
- Modificar itens de dados individuais
- Manipular saída para grupos de linhas
- Formatar datas e números para exibição
- Converter tipos de dados de coluna

As funções SQL podem aceitar argumentos e sempre retornar um valor.

Observação: A maioria das funções descritas nesta lição são específicas para a versão SQL da Oracle.



Funções SQL (continuação)

Há dois tipos distintos de funções:

- Funções de Uma Única Linha
- Funções de Várias Linhas

Funções de Uma Única Linha

Essas funções operam somente linhas únicas e retornam um resultado por linha. Há dois tipos diferentes de funções de uma única linha. Esta lição aborda as seguintes:

- Caractere
- Número
- Data
- Conversão

Funções de Várias Linhas

Essas funções manipulam grupos de linha a fim de obter um resultado por grupo de linhas.

Para obter mais informações, consulte o Oracle Server SQL Reference, Release 8, para uma lista completa de funções e sintaxes disponíveis.

Funções de Uma Única Linha

- Manipulam itens de dados
- Aceitam argumentos e retornam um valor
- Agem em cada linha retornada
- Retornam um resultado por linha
- Podem modificar o tipo de dados
- Podem ser aninhadas

```
function_name (coluna|expressão, [arg1, arg2,...])
```

Funções de Uma Única Linha

Funções de uma única linha são usadas para manipular itens de dados. Elas aceitam um ou mais argumentos e retornam um valor para cada linha retornada pela consulta. Um argumento pode ser um dos seguintes:

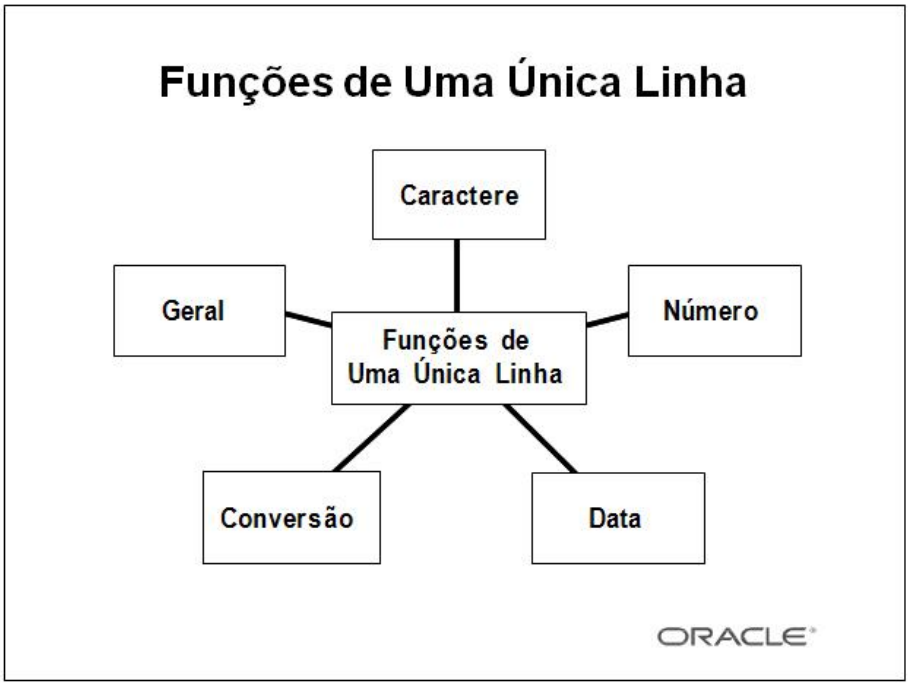
- Constante fornecida pelo usuário
- Valor variável
- Nome da coluna
- Expressão

Os recursos de funções de uma única linha:

- Atuam em cada linha retornada na consulta
- Retornam um resultado por linha
- Podem retornar um valor de dados de um tipo diferente do mencionado
- Podem esperar um ou mais argumentos
- Podem ser usados em cláusulas SELECT, WHERE e ORDER BY; podem ser aninhados

Na sintaxe:

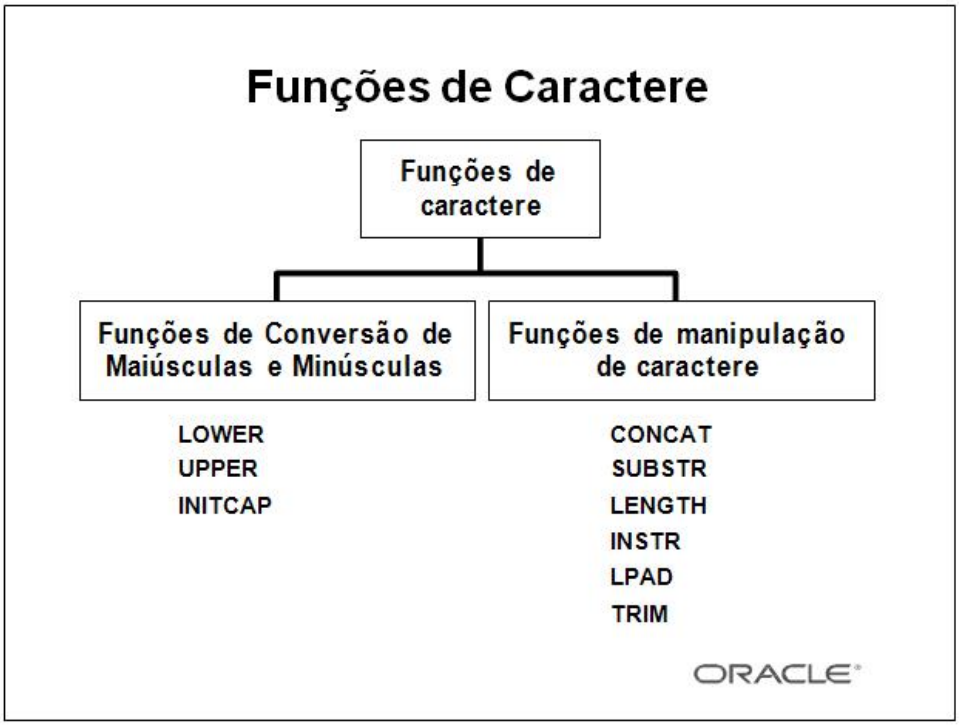
function_name	é o nome da função
coluna	é qualquer coluna de banco de dados nomeada
expressão	é qualquer string de caractere ou expressão calculada
arg1, arg2	é qualquer argumento a ser utilizado pela função



Funções de Uma Única Linha (continuação)
Esta lição aborda as seguintes funções de uma única linha:

- Funções de caractere: Aceitam entrada de caractere e podem retornar valores de número e caractere
- Funções numéricas: Aceitam entrada numérica e retornam valores numéricos
- Funções de data: Operam sobre valores de tipo de dados da data (Todas as funções de data retornam data um valor de tipo de dados de data exceto a função MONTHS_BETWEEN, que retorna um número.)
- Funções de conversão: Convertem um valor de um tipo de dados para outro
- Funções gerais:

– Função NVL
– Função DECODE



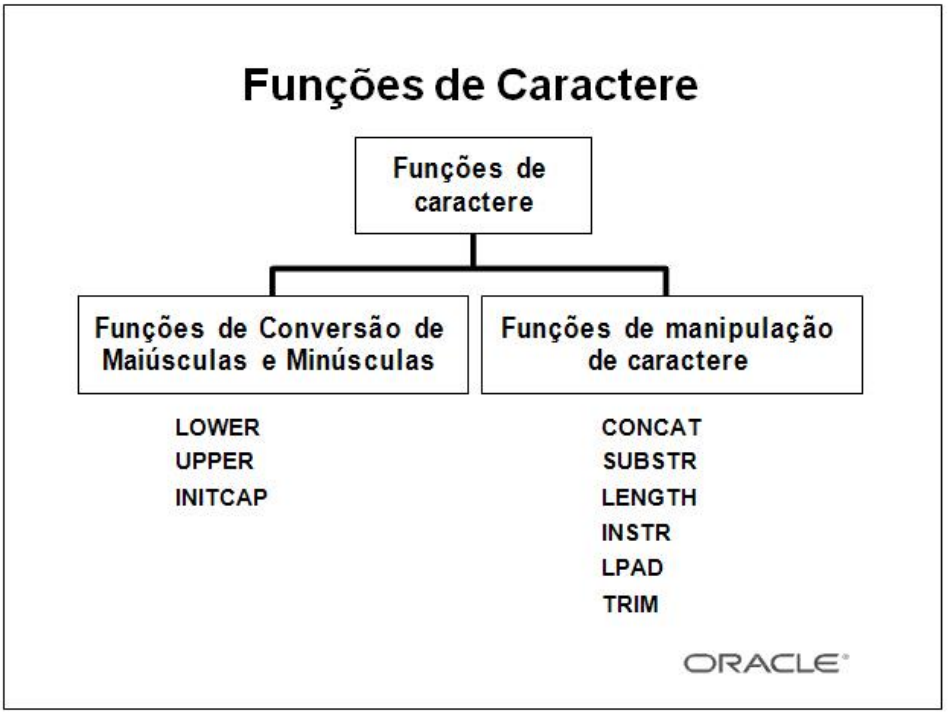
Funções de Caractere

As funções de caractere de uma única linha aceitam dados de caractere como entrada e podem retornar valores de número e de caractere. As funções de caractere podem ser divididas nas seguintes:

- Funções de Conversão de Maiúsculas e Minúsculas
- Funções de manipulação de caractere

Função	Objetivo
LOWER(*coluna|expressão*)	Converte valores de caractere alfabético para letras minúsculas
UPPER(*coluna|expressão*)	Converte valores de caractere alfabético para letras maiúsculas
INITCAP(*coluna|expressão*)	Converte valores de caractere alfabético para usar maiúscula na primeira letra de cada palavra e todas as outras letras em minúsculas
CONCAT(*coluna1|expressão1, coluna2|expressão2*)	Concatena o primeiro valor do caractere ao segundo valor do caractere, equivalente ao operador de concatenação (||)
SUBSTR(*coluna|expressão, m[,n]*)	Retorna caracteres específicos a partir do valor de caractere começando na posição *m,* até *n* caracteres depois (Se *m* for negativo, a conta inicia no final do valor de caractere. Se *n* for omitido, são retornados todos os caracteres até o final da string.)

Observação: Essas funções abordadas nesta lição são um subconjunto de funções disponíveis.



Funções de Caractere (continuação)

Função	**Objetivo**
LENGTH(*coluna|expressão*)	Retorna o número de caracteres do valor
INSTR(*coluna|expressão,m*)	Retorna a posição numérica do caractere nomeado
LPAD(*coluna|expressão*, *n*, '*string*')	Preenche o valor de caracter justificado à direita a uma largura total de *n* posições de caractere
TRIM(*anterior|posterior|ambos, trim_character FROM trim_source*)	Permitem a você organizar cabeçalhos ou caracteres de fim de linha (ou os dois) a partir de uma string de caractere. Se trim_character ou trim_source for um caractere literal, você deve incluí-los entre aspas simples. Este é um recurso disponível a partir Oracle em diante.

Funções de Conversão de Maiúsculas e Minúsculas

Converter maiúsculas em minúsculas para strings de caractere

Função	Resultado
LOWER('SQL Course')	sql course
UPPER('SQL Course')	SQL COURSE
INITCAP('SQL Course')	Sql Course

Funções de Conversão de Maiúsculas e Minúsculas

LOWER, UPPER e INITCAP são as três funções de conversão de maiúscula em minúscula.

- LOWER: Converte string de caracteres com letras maiúsculas e minúsculas ou só maiúsculas para letras minúsculas
- UPPER: Converte string de caracteres com letras maiúsculas e minúsculas ou só minúsculas para letras maiúsculas
- INITCAP: Converte a primeira letra de cada palavra para maiúscula e mantém as outras letras em minúscula

```
SQL>  SELECT 'The OCUP title for '||INITCAP(NOME)||' is '
  2   ||LOWER(OCUP) AS "FUNCIONARIO DETAILS"
  3   FROM EMPREGADO;


FUNCIONARIO DETAILS
----------------------------------------
The OCUP title for King is president
The job     title for   Blake is    manager
The job     title for   Clark is    manager
…
14 rows selected.
```

Usando Funções de Conversão de Maiúsculas e Minúsculas

Exibir o número de funcionário, nome e número de departamento do funcionário Blake.

```
SQL> SELECT COD_EMP, NOME, COD_DEPTO
  2   FROM  EMPREGADO
  3   WHERE NOME =      'blake';
no rows selected


SQL> SELECT COD_EMP, NOME, COD_DEPTO
```

```
  2   FROM  EMPREGADO
  3   WHERE NOME =      UPPER('blake');

COD_EMP NOME      COD_DEPTO
--------- ---------- ---------
7698 BLAKE        30
```

Funções de Conversão de Maiúsculas e Minúsculas (continuação)

O exemplo do slide exibe o número de funcionário, nome e número de departamento do funcionário BLAKE.

A cláusula WHERE da primeira instrução SQL especifica o nome do funcionário como 'blake'. Visto que todos os dados da tabela EMPREGADO são armazenados em letra maiúscula, o nome 'blake' não localiza uma correspondência na tabela EMPREGADO e como resultado nenhuma linha será selecionada.

A cláusula WHERE da segunda instrução SQL especifica que o nome do funcionário na tabela EMPREGADO

seja comparado a 'blake', convertido para letras maiúsculas. Agora que os dois nomes estão em maiúscula, é localizada uma correspondência e uma linha é selecionada. A cláusula WHERE pode ser escrita novamente da seguinte forma para produzir o mesmo resultado:

```
… WHERE     NOME = 'BLAKE'
```

O nome na entrada aparece como se estivesse armazenado no banco de dados. Para exibir o nome com a
primeira letra maiúscula, use a função INITCAP na instrução SELECT.

```
SQL> SELECT COD_EMP, INITCAP(NOME), COD_DEPTO
  2   FROM  EMPREGADO
  3   WHERE NOME =      UPPER('blake');
```

Funções de Manipulação de Caractere

Manipular strings de caractere

Função	Resultado
CONCAT('Good', 'String')	GoodString
SUBSTR('String',1,3)	Str

LENGTH('String')	6
INSTR('String', 'r')	3
LPAD(sal,10,'*')	******5000
TRIM('S' FROM 'SSMITH')	MITH

Funções de Manipulação de Caractere
CONCAT, SUBSTR, LENGTH, INSTR, LPAD e TRIM são as seis funções de manipulação de caractere abordadas nesta lição.

- CONCAT: Une valores de junção (Você está limitado a usar dois parâmetros com CONCAT.)
- SUBSTR: Extrai uma string de determinado tamanho
- LENGTH: Exibe o tamanho de uma string como um valor numérico
- INSTR: Localiza a posição numérica do caractere nomeado
- LPAD: Preenche o valor do caractere justificado à direita

Observação: A função de manipulação de caractere RPAD preenche o valor de caractere justificado à esquerda

TRIM: Organiza cabeçalho ou caracteres de fim de linha (ou os dois) a partir de uma string de caractere. Se trim_character ou trim_source forem um caractere literal, você deve incluí-los entre aspas simples.

Usando as Funções de Manipulação de Caractere

```
SQL> SELECT NOME, CONCAT (NOME, OCUP), LENGTH(NOME),
  2   INSTR(NOME, 'A')
  3 FROM    EMPREGADO
  4 WHERE   SUBSTR(OCUP,1,5) =      'SALES';

NOME       CONCAT(NOME,OCUP) LENGTH(NOME)  INSTR(NOME,'A')
---------- ------------------ ------------- --------------
--
MARTIN     MARTINS            ALESMAN            6      2
ALLEN      ALLENSALESMAN                         5      1
TURNER     TURNERSALESMAN                        6      0
WARD       WARDSALESMAN                          4      2
```

Funções de Manipulação de Caractere (continuação)
O exemplo do slide exibe o nome do funcionário e o cargo juntos, o tamanho do nome e a posição numérica da letra A no nome do funcionário, para todos os funcionários que estão em vendas.
Exemplo

Modifique a instrução SQL no slide a fim de exibir os dados daqueles funcionários cujos nomes terminam com N.

```
SQL> SELECT NOME, CONCAT(NOME, OCUP), LENGTH(NOME),
  2   INSTR(NOME, 'A')
  3   FROM  EMPREGADO
  4   WHERE SUBSTR(NOME, -1,  1)    = 'N';


NOME  CONCAT(NOME,OCUP) LENGTH(NOME) INSTR(NOME,'A')
-------- ------------------- ------------ ----------------
MARTIN      MARTINSALESMAN          6           2
ALLEN       ALLENSALESMAN           5           1
```

Funções Numéricas

As funções numéricas aceitam entrada numérica e retornam valores numéricos. Esta seção descreve algumas das funções numéricas.

Função	Objetivo
ROUND(*coluna*|*expressão*,*n*)	Arredonda a coluna, expressão ou valor para *n* casas decimais ou se *n* for omitido, nenhuma casa decimal (Se *n* for negativo, os números à esquerda do ponto decimal serão arredondados.)
TRUNC(*coluna*|*expressão*,*n*)	Trunca a coluna, expressão ou valor para *n* casas decimais ou se *n* for omitido, nenhuma casa decimal (Se *n* for negativo, os números à esquerda do ponto decimal serão truncados para zero.)
MOD(*m*,*n*)	Retorna o resto de *m* dividido por *n*.

Observação: Esta lista é um subconjunto das funções numéricas disponíveis.

Para obter mais informações, consulte o Oracle Server SQL Reference, Release 8, "Number Functions".

Usando a Função ROUND

```
SQL> SELECT ROUND(45.923,2), ROUND(45.923,0),
  2         ROUND(45.923,-1)
  3   FROM  DUAL;


ROUND(45.923,2) ROUND(45.923,0) ROUND(45.923,-1)
--------------- --------------- ----------------
45.92                        46               50
```

Função ROUND

A função ROUND arredonda a coluna, expressão ou valor para n casas decimais. Se o segundo argumento for 0 ou estiver ausente, o valor será arredondado para nenhuma casa decimal. Se o segundo argumento for 2, o valor será arredondado para duas casas decimais. Da mesma forma, se o segundo argumento for -2, o valor será arredondado para duas casas decimais para a esquerda.

A função ROUND pode também ser utilizada com funções de data. Você verá exemplos mais adiante nesta seção.

A tabela DUAL é fictícia. Mais informações será fornecidas mais tarde.

Usando a Função TRUNC

```
SQL> SELECT TRUNC(45.923,2), TRUNC(45.923),
  2         TRUNC(45.923,-1)
  3   FROM  DUAL;


TRUNC(45.923,2) TRUNC(45.923) TRUNC(45.923,-1)
--------------- ------------- ---------------
45.92                     45              40
```

Função TRUNC

A função TRUNC trunca a coluna, expressão ou valor para n casas decimais.

A função TRUNC trabalha com argumentos similares aos da função ROUND. Se o segundo argumento for 0 ou estiver ausente, o valor será truncado para nenhuma casa decimal. Se o segundo argumento for 2, o valor será truncado para duas casas decimais. Da mesma forma, se o segundo argumento for -2, o valor será truncado para duas casas decimais para esquerda.

Semelhante à função ROUND, a função TRUNC pode ser usada com funções de data.

Usando a Função MOD

Calcular o restante da proporção do salário para comissão de todos os funcionários cujo cargo é salesman.

```
SQL> SELECT NOME, sal, comm, MOD(sal, comm)


NOME       SAL   COMM MOD(SAL,COMM)
---------- ---------- --------- -------------
MARTIN      1250  1400  1250
ALLEN       1600  300   100
TURNER      1500  0     1500
WARD        1250  500   250
```

Função MOD

A função MOD localiza o restante do valor 1 dividido pelo valor 2. O exemplo do slide calcula o restante da proporção de salário para comissão de todos os funcionários cujo cargo é salesman.

Trabalhando com Datas

• O Oracle armazena datas em um formato numérico interno: século, ano, mês, dia, horas, minutos, segundos.

• O formato de data default é DD-MON-YY.

• SYSDATE é uma função de retorno de data e hora.

• DUAL é uma tabela fictícia usada para visualizar SYSDATE.

Formato de Data Oracle
Os produtos da Oracle armazenam datas em um formato numérico interno, representando o século, ano, mês, dia, horas, minutos e segundos.
O formato de entrada e exibição default para qualquer data é DD-MON-YY. Datas válidas para a
Oracle estão entre 1 de janeiro, 4712 A.C. e 31 de dezembro, 9999 D.C.

SYSDATE
SYSDATE é a função de data que retorna a data e a hora atual. Você pode usar o SYSDATE da mesma forma como usaria qualquer outro nome de coluna. Por exemplo, é possível exibir a data atual selecionando SYSDATE a partir da tabela. Costuma-se selecionar SYSDATE em uma tabela fictícia chamada DUAL.

DUAL
A tabela DUAL pertence ao usuário SYS e pode ser acessada por todos os usuários. Ela contém uma coluna, DUMMY, e uma linha com o valor X. A tabela DUAL é útil quando deseja retornar um valor somente uma vez — por exemplo, o valor de uma constante, pseudocoluna ou expressão que não é derivada de uma tabela com dados do usuário. A tabela DUAL, em geral, é utilizada pela totalidade de sintaxe da cláusula SELECT, pois as duas cláusulas SELECT e FROM são obrigatórias e diversos cálculos não precisam selecionar das tabelas reais.

Exemplo
Exibir a data atual usando a tabela DUAL.

```
SQL> SELECT SYSDATE
  2   FROM  DUAL;
```

Aritmética com Datas

• Adicionar ou subtrair um número de, ou para, uma data para um valor de data resultante.

• Subtrair duas datas a fim de localizar o número de dias entre estas datas.

• Adicionar horas para uma data dividindo o número de horas por 24.

Aritmética com Datas

Já que os bancos de dados armazenam datas como números, você pode executar cálculos usando operadores aritméticos como adição e subtração. É possível adicionar e subtrair constantes de número bem como datas.
Você pode executar as seguintes operações:

Operação	Resultado	Descrição
data + número	Data	Adiciona um número de dias para uma data
data - número	Data	Subtrai um número de dias de uma data
data - data	Número de dias	Subtrai uma data de outra
data + número/24	Data	Adiciona um número de horas para uma data

Usando Operadores Aritméticos com Datas

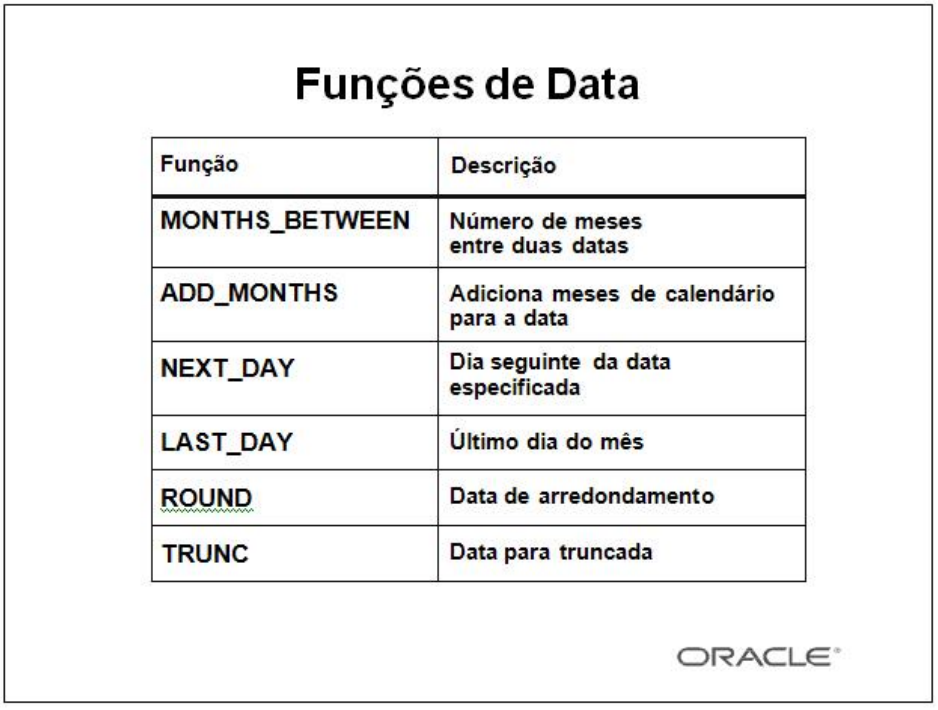
```
SQL> SELECT NOME, (SYSDATE-hiredate)/7 WEEKS

NOME         WEEKS
----------  ---------
KING         830.93709
CLARK        853.93709
MILLER       821.36566
```

Aritmética com Datas (continuação)

O exemplo no slide exibe o nome e o número de semanas empregadas por todos os funcionários do departamento 10. Ele subtrai a data atual (SYSDATE) a partir da data na qual o funcionário

foi admitido e divide o resultado por 7 a fim de calcular o número de semanas que o trabalhador está empregado.

Observação: SYSDATE é uma função SQL que retorna a data e a hora atual. Seus resultados podem ser diferentes dos obtidos no exemplo.



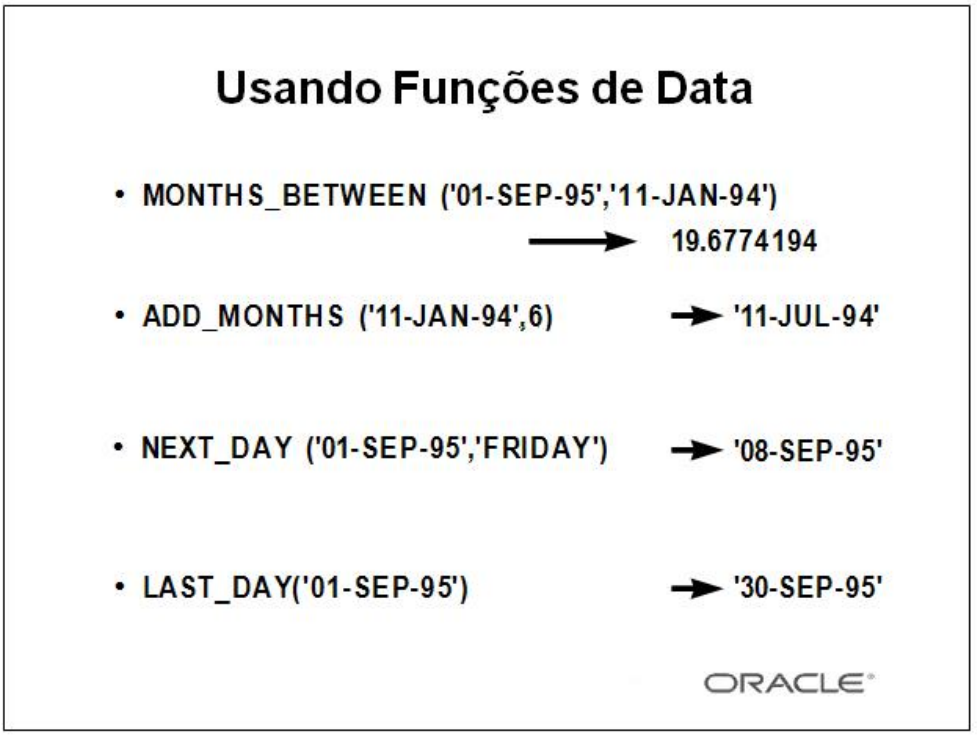
Funções de Data

Funções de data operam em datas Oracle. Todas as funções de data retornam um valor de tipo de dados

DATE exceto MONTHS_BETWEEN, que retorna um valor numérico.

- *MONTHS_BETWEEN(data1, data2):* Localiza o número de meses entre a data 1 e a data 2. O resultado pode ser positivo ou negativo. Se data1 for posterior a data2, o resultado será positivo; se data1 for anterior a data2, o resultado será negativo. A parte não-inteira do resultado representa uma parte do mês.
- *ADD_MONTHS(data, n):* Adiciona um número n de meses de calendário à data. O valor de n deve ser inteiro ou pode ser negativo.
- *NEXT_DAY(data, 'char'):* Localiza a data do próximo dia especificado da data seguinte da semana ('char'). O valor de char pode ser um número representando um dia ou uma string de caractere.
- *LAST_DAY(data):* Localiza a data do último dia do mês que contém a data.
- *ROUND(data[,'fmt']):* Retorna a data arredondada para a unidade especificada pelo modelo de formato fmt. Se o modelo de formato fmt for omitido, a data será arredondada para o dia mais próximo.
- *TRUNC(data[, 'fmt']):* Retorna a data com a parte da hora do dia truncada para a unidade especificada pelo modelo de formato fmt. Se o modelo de formato fmt for omitido, a data será truncada para o dia mais próximo.

Esta lista é um subconjunto de funções de data disponíveis. Os modelos de formato são abordados mais tarde nesta lição. Exemplos de modelos de formato são mês e ano.



Funções de Data (continuação)

Para todos os funcionários contratados por menos de 200 meses, exibir o número do funcionário, a data de admissão, o número de meses contratado, a data de revisão semestral, a primeira sexta-feira após a data da admissão e o último dia do mês em que foi admitido.

```
SQL> SELECT COD_EMP, hiredate,
  2   MONTHS_BETWEEN(SYSDATE, hiredate) TENURE,
  3   ADD_MONTHS(hiredate, 6) REVIEW,
  4   NEXT_DAY(hiredate, 'FRIDAY'), LAST_DAY(hiredate)
  5   FROM  EMPREGADO
  6   WHERE MONTHS_BETWEEN (SYSDATE,     hiredate)<200;


COD_EMP HIREDATE  TENURE REVIEW     NEXT_DAY( LAST_DAY(
      -----
- --------- --------- --------- --------- ---------
7839 17-NOV-81    192.24794   17-MAY-82   20-NOV-81   30-
NOV-81
7698 01-MAY-81    198.76407   01-NOV-81   08-MAY-81   31-
MAY-81
...
11 rows selected.
```

Usando Funções de Data

- ROUND('25-JUL-95','MONTH') → 01-AUG-95
- ROUND('25-JUL-95','YEAR') → 01-JAN-96
- TRUNC('25-JUL-95','MONTH') → 01-JUL-95
- TRUNC('25-JUL-95','YEAR') → 01-JAN-95

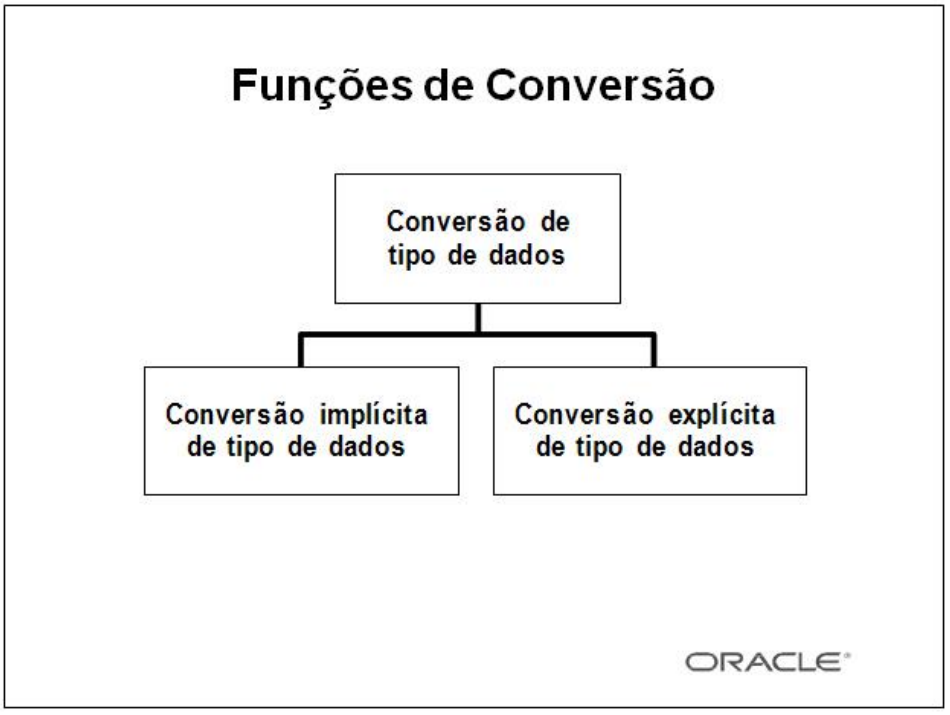
Funções de Data (continuação)

As funções ROUND e TRUNC podem ser usadas para os valores de data e número. Quando usadas com datas, estas funções arredondam ou truncam o modelo de formato especificado. Assim, é possível arredondar datas para o ano ou mês mais próximo.

Exemplo

Compare as datas de admissão de todos os funcionários contratados em 1982. Exiba o número do funcionário, a data de admissão e o mês de início usando as funções ROUND e TRUNC.

```
SQL> SELECT COD_EMP, hiredate,
  2   ROUND(hiredate, 'MONTH'), TRUNC(hiredate, 'MONTH')
  3   FROM  EMPREGADO
  4   WHERE hiredate like     '%1982';


COD_EMP HIREDATE  ROUND(HIR TRUNC(HIR
--------- --------- --------- ---------
7788 09-DEC-82 01-DEC-82 01-DEC-82
7934 23-JAN-82 01-FEB-82 01-JAN-82
```



Funções de Conversão

Além dos tipos de dados Oracle, as colunas de tabela em um banco de dados Oracle podem ser definidas usando os tipos de dados ANSI, DB2 e SQL/DS. No entanto, o Oracle Server converte internamente tais tipos de dados para tipos de dados Oracle.

Em alguns casos, o Oracle Server aceita dados de um tipo de dados em que espera dados de um tipo de dados diferente. Isso é permitido quando o Oracle Server pode converter dados automaticamente para o tipo de dados esperado. Essa conversão de

tipo de dados pode ser realizada de forma implícita pelo Oracle Server ou explícita pelo usuário.

As conversões implícitas de tipo de dados funcionam de acordo com as regras explicadas nos próximos dois slides.

As conversões explícitas de tipo de dados podem ser realizadas usando-se as funções de conversão.

As funções de conversão convertem um valor de um tipo de dados para outro. Geralmente, o formato dos nomes de função seguem a convenção tipo de dados TO tipo de dados. O primeiro tipo de dados é um tipo de dados de entrada; o último tipo de dados é de saída.

Observação: Embora a conversão implícita de tipos de dados esteja disponível, é recomendável que você realize a conversão explícita de tipo de dados a fim de garantir a confiabilidade das instruções SQL.

Conversão Implícita de Tipo de Dados

Para atribuições, o Oracle Server pode converter automaticamente o seguinte:

De	Para
VARCHAR2 ou CHAR	NUMBER
VARCHAR2 ou CHAR	DATE
NUMBER	VARCHAR2
DATE	VARCHAR2

Conversão Implícita de Tipo de Dados

A atribuição ocorre se o Oracle Server puder converter o tipo de dados do valor nela usado para a atribuição destino.

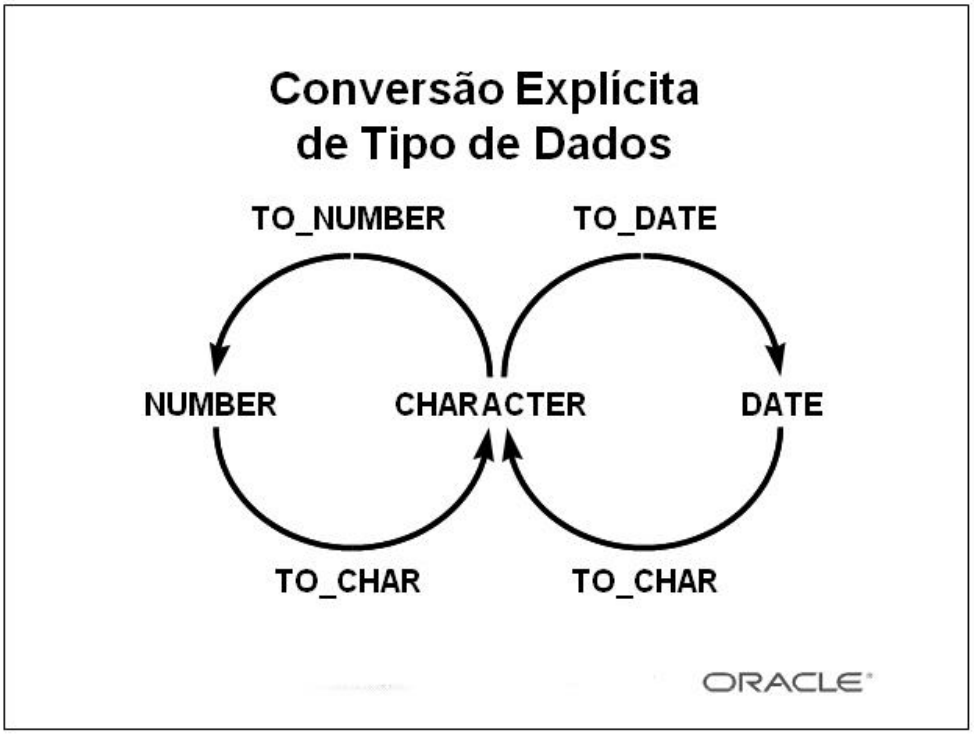
Conversão Implícita de Tipo de Dados

Para avaliação da expressão, o Oracle Server pode converter automaticamente o seguinte:

De	Para
VARCHAR2 ou CHAR	NUMBER
VARCHAR2 ou CHAR	DATE

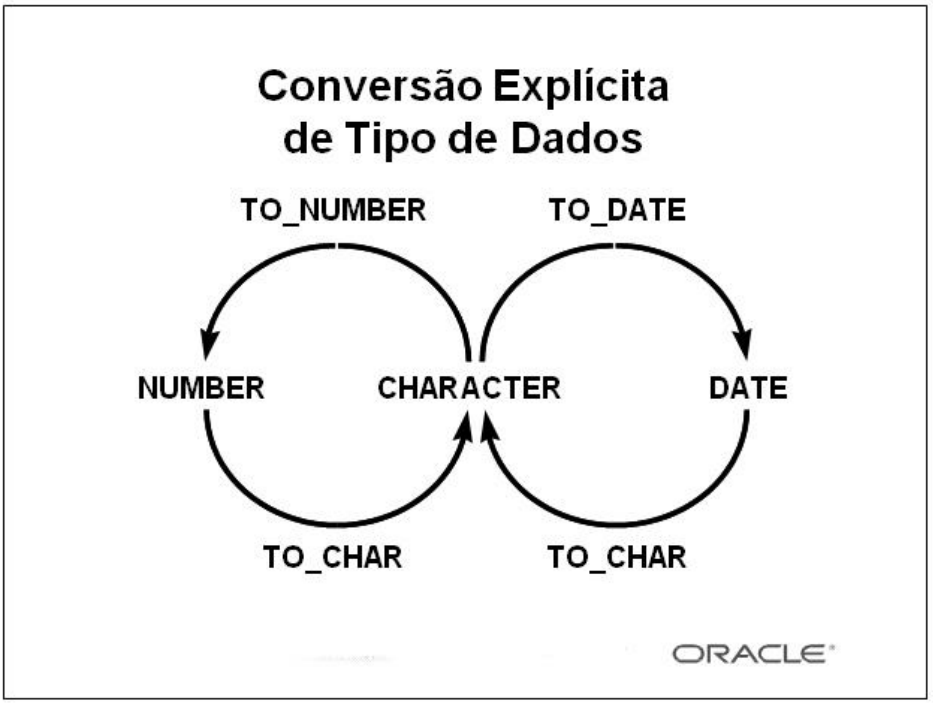
Conversão Implícita de Tipo de Dados

Em geral, o Oracle Server usa a regra para expressão quando a conversão de tipo de dados é necessária em lugares não cobertos por uma regra para as conversões de atribuição.
Observação: Conversões de CHAR para NUMBER ocorrem somente se a string de caractere representar um número válido. Conversões de CHAR para DATE ocorrem somente se a string de caractere possuir o formato default DD-MON-YY.



Conversão Explícita de Tipo de Dados
O SQL oferece três funções para converter um valor de um tipo de dados para outro:

Função	Objetivo
TO_CHAR(*número*|*data*,[*fmt*], *[nlsparams*])	Converte um valor de número ou data para uma string de caractere VARCHAR2 com modelo de formato *fmt*. **Conversão de Número**: O parâmetro nlsparams especifica os seguintes caracteres, que são retornados por elementos de formato de número: ☐ Caractere decimal ☐ Separador de grupo ☐ Símbolo da moeda local ☐ Símbolo da moeda internacional Se nlsparams ou qualquer outro parâmetro for omitido, essa função utilizará os valores de parâmetro default para a seção.



Conversão Explícita de Tipo de Dados (continuação)

Função	Objetivo
TO_CHAR(*número*|*data*,[*fmt*], [*nlsparams*])	**Conversão de data**: O parâmetro nlsparams especifica o idioma no qual os nomes de dias, meses e abreviação retornam. Se este parâmetro for omitido, essa função utilizará os idiomas de data default para a seção.
TO_NUMBER(*carac,[fmt], [nlsparams]*)	Converte uma string de caractere contendo dígitos para um número no formato especificado pelo modelo de formato opciona *fmt*. O parâmetro nlsparams tem a mesma finalidade nesta função como na função TO_CHAR para a conversão de número.
TO_DATE(*carac*,[*fmt*],[*nlsparams*])	Converte uma string de caractere representando uma data para um valor de data de acordo com o *fmt* especificado. Se *fmt* for omitido, o formato é DD-MON-YY. O parâmetro nlsparams possui a mesma finalidade na função TO_CHAR para a conversão de data.

Conversão Explícita de Tipo de Dados (continuação)

Observação: A lista de funções mencionadas nesta lição é um subconjunto das funções de conversão disponíveis.

Função TO_CHAR com Datas

```
TO_CHAR(data, 'fmt')
```

O modelo de formato:

• Deve estar entre aspas simples e fazer distinção entre maiúsculas e minúsculas

• Pode incluir qualquer elemento de formato de data válido

• Tem um elemento fm para remover espaços preenchidos ou suprimir zeros à esquerda

• É separado do valor de data por uma vírgula

Exibindo uma Data em um Formato Específico

Anteriormente, todos os valores de data Oracle eram exibidos no formato DD-MON-YY. A função

TO_CHAR permite converter uma data a partir desse formato default para um especificado por você.

Diretrizes

• O modelo de formato deve estar entre aspas simples e fazer distinção entre maiúsculas e minúsculas.

• O modelo de formato pode incluir qualquer elemento de formato de data válido. Certifique-se de separar o valor da data do modelo de formato por uma vírgula.

• Os nomes de dias e meses na saída são preenchidos automaticamente por espaços.

• Para remover os espaços em branco ou suprimir os zeros à esquerda, use o modo de preenchimento do elemento fm.

• Você pode redimensionar o tamanho da exibição do campo de caractere resultante com o comando COLUMN do SQL*Plus.

• A largura da coluna resultante é, por padrão, de 80 caracteres.

```
SQL> SELECT COD_EMP, TO_CHAR(hiredate, 'MM/YY') Month_Hired
  2   FROM  EMPREGADO
  3   WHERE NOME =      'BLAKE';
```

Elementos de Modelo de Formato de Data

YYYY	Ano completo em números
YEAR	Ano por extenso
MM	Valor de dois dígitos para mês
MONTH	Abreviação de três letras do dia da semana
DY	Abreviação de três letras do dia da semana
DAY	Nome completo do dia

Elementos de Exemplo de Formatos de Data Válidos

Elemento	Descrição
SCC ou CC	Século; Prefixos S data AC com -
Anos em datas YYYY ou SYYYY	Ano; Prefixos S data AC com -
YYY ou YY ou Y	Últimos três, dois ou um dígitos do ano
Y,YYY	Ano com vírgula nesta posição
IYYY, IYY, IY, I	Quatro, três, dois ou um dígito do ano com base no padrão ISO
SYEAR ou YEAR	Ano inteiro; Prefixos S data AC com -
BC ou AD	Indicador AC/DC
B.C. ou A.D.	Indicador com pontos AC/DC
Q	Trimestre do ano
MM	Mês, valor de dois dígitos
MONTH	Nome do mês preenchido com espaços limitado a nove caracteres
MON	Nome do mês, abreviação de três letras
RM	Mês em números romanos
WW ou W	Semana do ano ou mês
DDD ou DD ou D	Dia do ano, mês ou semana
DAY	Nome do dia preenchido com espaços limitado a nove caracteres
DY	Nome do dia; abreviação de três letras
J	Dia do calendário juliano; o número de dias desde 31de dezembro 4713 BC

Elementos de Modelo de Formato de Data

• Elementos de hora formatam a parte de hora da data.

HH24:MI:SS AM	15:45:32 PM

• Adicionar strings de caractere incluindo-as entres aspas.

DD "of" MONTH	12 of OCTOBER

• Sufixos de número escrevem os números por extenso.

ddspth	fourteenth

Formatos de Hora

Use os formatos listados tabelas a seguir para exibir informações de hora e literais e alterar numerais para números escritos.

Elemento	Descrição
AM ou PM	Indicador meridiano
A.M. ou P.M.	Indicador meridiano com pontos
HH ou HH12 ou HH24	Horas do dia ou hora (1 a 12) ou hora (0 a 23)
MI	Minuto (0 a 59)
SS	Segundo (0 a 59)
SSSSS	Segundos após a meia-noite (0–86399)

Outros Formatos

Elemento	Descrição
/ . ,	A pontuação é reproduzida no resultado
"of the"	A string entre aspas é reproduzida no resultado

Especificando Sufixos para Influenciar Exibição de Número

Elemento	Descrição
TH	Número ordinal (por exemplo, DDTH para 4TH)
SP	Número por extenso (por exemplo, DDSP para FOUR)
SPTH ou THSP	Números ordinais por extenso (por exemplo, DDSPTH

Usando a Função TO_CHAR com Datas

```
SQL> SELECT NOME,
2     TO_CHAR(hiredate, 'fmDD Month YYYY') HIREDATE
3     FROM  EMPREGADO;

NOME        HIREDATE
----------  -----------------
KING        17 November 1981
BLAKE       1 May 1981
CLARK       9 June 1981
JONES       2 April 1981
MARTIN      28 September 1981
ALLEN       20 February 1981
...
14 rows selected.
```

Função TO_CHAR com Datas

A instrução SQL no slide exibe o nome e as datas de admissão para todos os funcionários. A data de admissão aparece como 17 November 1981.

Exemplo

Modificar o exemplo do slide para exibir as datas em um formato que apareça como Seventh of February 1981 08:00:00 AM.

```
SQL> SELECT NOME,
  2         TO_CHAR(hiredate, 'fmDdspth "of" Month YYYY
fmHH:MI:SS AM')
  3         HIREDATE
  4    FROM  EMPREGADO;


NOME  HIREDATE
---------- ----------------------------------------------
KING  Seventeenth of November 1981 12:00:00 AM
BLAKE First of May 1981 12:00:00 AM
...
14 rows selected.
```

Note que o mês segue o formato do modelo de formato especificado, ou seja, a primeira letra em maiúscula e o restante em minúscula.

Função TO_CHAR com Números

```
TO_CHAR(número, 'fmt')
```

Use estes formatos com a função TO_CHAR para exibir um valor de número como um caractere:

9	Representa um número
0	Forces a zero to be displayed
$	Coloca um sinal de dólar flutuante
L	Usa o símbolo da moeda local flutuante
.	Imprime um ponto decimal
,	Imprime um indicador de milhar

Função TO_CHAR com Números

Ao trabalhar com valores de número tais como strings de caractere, você deve converter esse números para o tipo de dados de caracter usando a função TO_CHAR , que traduz o tipo de dados de NUMBER para tipo de dados VARCHAR2. Essa técnica é particularmente útil com concatenação.

Elementos de Formato de Número

Se estiver convertendo um número para tipo de dados de caractere, você pode usar os seguintes elementos:

Element	Descrição	Exemplo	Resultado
9	Posição numérica (número de 9s determinam o tamanho da exibição)	999999	1234
0	Exibe zeros à esquerda	099999	001234
$	Sinal de dólar flutuante	$999999	$1234
L	Símbolo da moeda local flutuante	L999999	FF1234
.	Ponto decimal na posição especificada	999999.99	1234.00
,	Vírgula na posição especificada	999,999	1,234
MI	Sinais de menos à direita (valores negativos)	999999MI	1234-
PR	Coloca números negativos entre parênteses	999999PR	<1234>
EEEE	Notação científica (formato deve especificar	99.999EEEE	1.234E+03
V	Multiplica por 10 *n* vezes (*n* = número de 9s	9999V99	123400
B	Exibe valores de zero como espaço, não 0	B9999.99	1234.00

Usando a Função TO_CHAR com Números

```
SQL> SELECT TO_CHAR(sal,'$99,999') SALARY
  2   FROM  EMPREGADO
  3   WHERE NOME =      'SCOTT';


SALARY
--------
$3,000
```

Diretrizes

- O Oracle Server exibe uma string com sinais numéricos (#) no lugar de um número inteiro cujos dígitos excedam o número de dígitos fornecidos no modelo de formato.
- O Oracle Server arredonda os valores decimais armazenados para o número de espaços decimais fornecidos no modelo de formato.

Funções TO_NUMBER e TO_DATE

- Converter uma string de caractere para um formato de número usando a função TO_NUMBER

```
TO_NUMBER(carac[, 'fmt'])
```

- Converter uma string de caractere para um formato de data usando a função TO_DATE

```
TO_DATE(carac[, 'fmt'])
```

Funções TO_NUMBER e TO_DATE

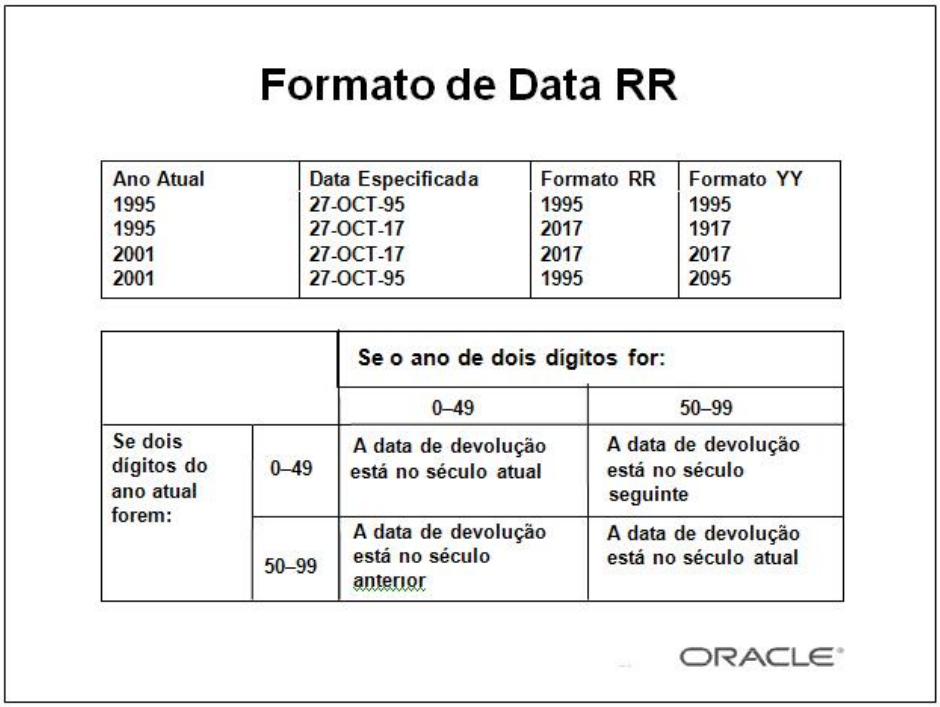
Talvez deseje converter uma string de caractere para um número ou data. Para completar essa tarefa, use as funções TO_NUMBER ou TO_DATE. O modelo de formato que escolher será baseado nos elementos de formato demonstrados anteriormente.

Exemplo

Exibir os nomes e as datas de admissão para todos os funcionários contratados em 22 de fevereiro de 1981.

```
SQL> SELECT NOME, hiredate
```

```
  2    FROM  EMPREGADO
  3    WHERE hiredate = TO_DATE('February 22, 1981', 'Month
dd,    YYYY');


NOME  HIREDATE
----------  --------
WARD  22-FEB-81
```



Formato de Data RR

Ano Atual	Data Especificada	Formato RR	Formato YY
1995	27-OCT-95	1995	1995
1995	27-OCT-17	2017	1917
2001	27-OCT-17	2017	2017
2001	27-OCT-95	1995	2095

		Se o ano de dois dígitos for:	
		0–49	50–99
Se dois dígitos do ano atual forem:	0–49	A data de devolução está no século atual	A data de devolução está no século seguinte
	50–99	A data de devolução está no século anterior	A data de devolução está no século atual

ORACLE

O Elemento de Formato de Data RR

O formato de data RR é similar ao elemento YY, porém ele permite especificar séculos diferentes. Você pode usar um elemento de formato de data RR no lugar de YY, para que o século do valor de devolução varie de acordo com o ano de dois dígitos especificado e com os últimos dois dígitos do ano atual. A tabela do slide resume o comportamento do elemento RR.

Ano Atual	Data Fornecida	Interpretada (RR)	Interpretada (YY)
1994	27-OCT-95	1995	1995
1994	27-OCT-17	2017	1917

2001	27-OCT-17	2017	2017

Função NVL

Converte nulo para um valor real

• Os tipos de dados que podem ser usados são data, caractere e número.

• Os tipos de dados devem corresponder com

– NVL(comm,0)

– NVL(hiredate,'01-JAN-97')

– NVL(OCUP,'No OCUP Yet')

A Função NVL

Para converter um valor nulo para um valor real, use a função NVL.

Syntax

```
NVL (expr1, expr2)
```

onde: *expr1* é o valor de origem ou expressão que pode conter nulo
expr2 é o valor de destino para a conversão de nulo

Você pode usar a função NVL para converter qualquer tipo de dados, porém o valor de devolução é sempre o mesmo do tipo de dados da expr1.

Conversões NVL para Vários Tipos de Dados

Tipo de dados	Exemplo de Conversão
NUMBER	NVL(*number_column*,9)
DATE	NVL(*date_column*, '01-JAN-95')
CHAR ou VARCHAR2	NVL(*character_column*, 'Unavailable')

Usando a Função NVL

```
SQL> SELECT NOME, sal, comm, (sal*12)+NVL(comm,0)
2     FROM  EMPREGADO;


NOME         SAL    COMM   (SAL*12)+NVL(COMM,0)
---------- ---------    ---------   --------------------
KING         5000                    60000
BLAKE        2850                    34200
CLARK        2450                    29400
JONES        2975                    35700
MARTIN       1250         1400       16400
ALLEN        1600         300        19500
...
14 rows selected.
```

Função NVL

Para calcular a remuneração anual de todos os funcionários, você precisa multiplicar o salário mensal por 12 e, em seguida, adicionar a comissão.

```
SQL> SELECT NOME, sal, comm, (sal*12)+comm
2     FROM  EMPREGADO;


NOME         SAL    COMM  (SAL*12)+COMM
---------- --------- --------- -------------
KING         5000
BLAKE        2850
CLARK        2450
JONES        2975
MARTIN       1250  1400         16400
...
14 rows selected.
```

Note que a remuneração anual é calculada somente para aqueles funcionários que recebem uma comissão. Se qualquer valor de coluna na expressão for nulo, o resultado será nulo. Para calcular valores para todos os funcionários, você deve converter o valor nulo para um número antes de aplicar o operador aritmético. No exemplo do slide, a função para NVL é usada para converter valores nulos para zero.

Função DECODE

Facilita pesquisas condicionais realizando o trabalho de uma instrução CASE ou IF- THEN-ELSE

```
DECODE(col/express, pesquisa1, resultado1
                        [, pesquisa2, resultado2,...,]
                        [, default])
```

A Função DECODE

A função DECODE decodifica uma expressão de um modo similar à lógica IF-THEN-ELSE usada em diversas linguagens. A função DECODE decodifica a expressão após compará-la a cada valor de pesquisa. Se a expressão for a mesma da pesquisa, o resultado é retornado.

Se o valor default for omitido, será retornado um valor nulo onde um valor de pesquisa não corresponde a quaisquer valores de resultado.

Usando a Função DECODE

```
SQL> SELECT OCUP, sal,
  2   DECODE(OCUP, 'ANALYST', SAL*1.1,
  3          'CLERK',   SAL*1.15,
  4          'MANAGER', SAL*1.20,
  5                     SAL)
  6   REVISED_SALARY
  7   FROM  EMPREGADO;


OCUP         SAL REVISED_SALARY
--------- --------- --------------
PRESIDENT    5000  5000
MANAGER      2850  3420
MANAGER      2450  2940
...
14 rows selected.
```

Usando a Função DECODE

Na instrução SQL acima, o valor de OCUP está decodificado. Se OCUP for ANALYST, o aumento de salário é de 10%; se OCUP for CLERK, o aumento de salário é de 15%; se OCUP for MANAGER, o

aumento de salário é de 20%. Para todas as outras funções de trabalho, não há aumento de salário.

A mesma instrução pode ser escrita como uma instrução IF-THEN-ELSE:

```
IF OCUP = 'ANALYST'     THEN  sal = sal*1.1
IF OCUP = 'CLERK'       THEN  sal = sal*1.15
IF OCUP = 'MANAGER'     THEN  sal = sal*1.20
ELSE sal = sal
```

Usando a Função DECODE

Exibir a taxa de imposto aplicável para cada funcionário do departamento 30.

```
SQL> SELECT NOME, sal,
  2   DECODE(TRUNC(sal/1000, 0),
  3   0, 0.00,
  4   1, 0.09,
  5   2, 0.20,
  6   3, 0.30,
  7   4, 0.40,
  8   5, 0.42,
  9   6, 0.44,
 10   0.45) TAX_RATE
 11   FROM  EMPREGADO
 12   WHERE COD_DEPTO = 30;
```

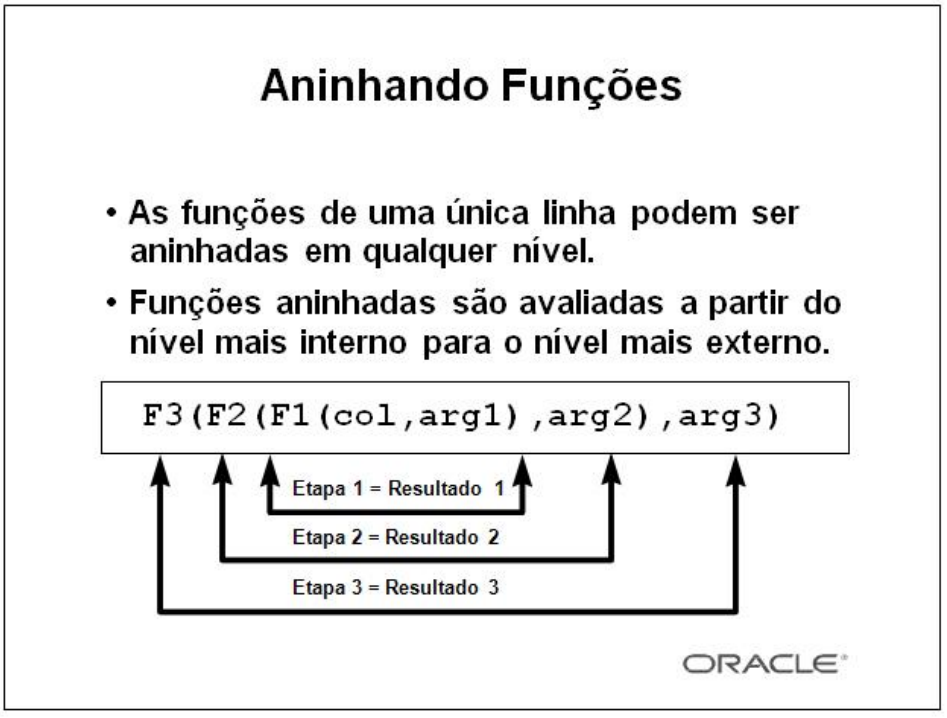
Exemplo

O slide mostra outro exemplo usando a função DECODE. Neste exemplo, determinamos a taxa de imposto para cada funcionário do departamento 30 com base no salário mensal. As taxas de imposto são mencionadas por valores na tabela a seguir.

Faixa Salarial Mensal	Taxa
US$0,00 - 999,99	0%
US$1.000,00 - 1.999,99	9%
US$2.000,00 - 2.999,99	20%
US$3.000,00 - 3.999,99	30%
US$4.000,00 - 4.999,99	40%
US$5.000,00 - 2.999,99	42%
US$6.000,00 - 6.999,99	44%

US$7.000,00 ou superior	45%

```
NOME         SAL    TAX_RATE
---------- --------- ---------
BLAKE        2850   .2
MARTIN       1250   .09
ALLEN        1600   .09
TURNER       1500   .09
...
6 rows selected.
```



Aninhando Funções

As funções de uma única linha podem ser aninhadas em qualquer nível. As funções aninhadas são avaliadas desde o nível mais interno até o mais externo. Alguns exemplos a seguir mostram a flexibilidade dessas funções.

Aninhando Funções

```
SQL> SELECT NOME,
  2   NVL(TO_CHAR(mgr),'No Manager')
  3   FROM  EMPREGADO
  4   WHERE mgr IS      NULL;

NOME  NVL(TO_CHAR(MGR),'NOMANAGER')
---------- ------------------------------
KING  No Manager
```

Aninhando Funções (continuação)

O exemplo do slide exibe o presidente da empresa, que não possui nenhum gerente. A avaliação da instrução SQL envolve duas etapas:

1. Avaliar a função interna a fim de converter um valor de número para uma string de caractere.
   - –Result1 = TO_CHAR(mgr)
2. Avaliar a função externa a fim de substituir o valor nulo com uma string de texto.
   - –NVL(Result1, 'No Manager')

A expressão inteira transforma-se no cabeçalho da coluna, pois não foi fornecido nenhum apelido de coluna.

Exemplo

Exibir a data da próxima sexta-feira, que fica seis meses após a data de admissão. A data resultante deve aparecer como Friday, March 12th, 1982. Ordenar os resultados por data de admissão.

```
SQL>  SELECT      TO_CHAR(NEXT_DAY(ADD_MONTHS
2                 (hiredate, 6), 'FRIDAY'),
3                 'fmday, Month, ddth, YYYY')
4                 "Next 6 Month Review"
5     FROM        EMPREGADO
6     ORDER BY    hiredate;
```

Sumário

Use as funções para realizar o seguinte:

- Executar cálculos usando dados
- Modificar itens de dados individuais
- Manipular saída para grupos de linhas

• Alterar formatos de data para exibição

• Converter tipos de dados de coluna

Funções de Uma Única Linha

As funções de uma única linha podem ser aninhadas em qualquer nível. As funções de uma única linha podem manipular o seguinte:

- Dados de caractere: LOWER, UPPER, INITCAP, CONCAT, SUBSTR, INSTR, LENGTH
- Dados de número: ROUND, TRUNC, MOD
- Dados de data: MONTHS_BETWEEN, ADD_MONTHS, NEXT_DAY, LAST_DAY, ROUND, TRUNC
- Valores de data podem também usar operadores aritméticos.
- As funções de conversão podem converter valores de caractere, de data e numérico: TO_CHAR, TO_DATE, TO_NUMBER

SYSDATE e DUAL

SYSDATE é uma função de data que retorna a data e a hora atual. Costuma-se selecionar SYSDATE em uma tabela fictícia chamada DUAL.

Capítulo 05

Exibindo Dados de Várias Tabelas

Objetivos

Depois de completar esta lição, você poderá fazer o seguinte:

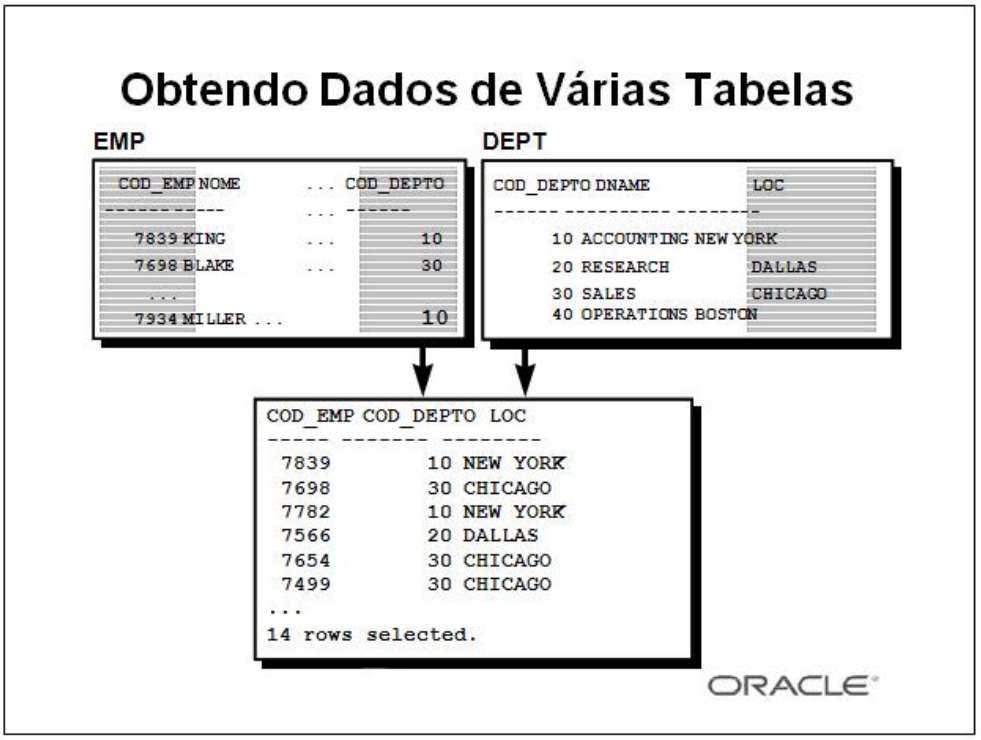
• Criar instruções SELECT para obter acesso aos dados a partir de mais de uma tabela usando as junções idênticas e não-idênticas

• Visualizar dados que, em geral, não correspondem a uma condição de junção usando junções externas

• Unindo uma tabela a ela mesma

Objetivo da Lição

Esta lição aborda como obter dados a partir de uma ou mais tabelas, usando diferentes métodos disponíveis.



Dados de Várias Tabelas

Algumas vezes é necessário utilizar dados a partir de uma ou mais tabelas. No exemplo do slide, o relatório exibe dados a partir de duas tabelas separadas.

- COD_EMP existe na tabela EMP.
- COD_DEPTO existe nas tabelas EMP e DEPT.
- LOC existe na tabela DEPT.

Para produzir o relatório, você precisa vincular as tabelas EMP e DEPT e ter acesso aos dados das duas.

O Que É uma Junção?

Use uma junção para consultar dados a partir de uma ou mais tabelas.

```
SELECT      tabela1.coluna, tabela2.coluna
FROM        tabela1, tabela2
WHERE       tabela1.coluna1 = tabela2.coluna2;
```

• Criar uma condição de junção na cláusula WHERE.

• Prefixar o nome da coluna com o nome da tabela quando o mesmo nome da coluna aparecer em mais de uma tabela.

Definindo Junções

Quando são necessários dados de uma ou mais tabelas no banco de dados, usa-se uma condição de junção. As linhas de uma tabela podem ser unidas às linhas de outra tabela de acordo com os valores comuns existentes nas colunas correspondentes, isto é, em geral colunas de chave primária e estrangeira.

Para exibir dados a partir de uma ou mais tabelas relacionadas, crie uma condição de junção simples na cláusula WHERE.

Na sintaxe:

tabela1.coluna denota a tabela e a coluna a partir das quais recupera-se os dados

tabela1.coluna1 = tabela2.coluna2 é a condição que junta (ou relaciona) tabelas

Diretrizes

• Ao criar uma instrução SELECT que una tabelas, anteceda o nome da coluna com o nome da tabela a fim de esclarecer e avançar o acesso ao banco de dados.

• Caso apareça o mesmo nome da coluna em mais de uma tabela, o nome da coluna deve estar prefixado com o nome da tabela.

• Para juntar n tabelas, é necessário um mínimo de (n-1) condições de junção. Assim, para juntar quatro tabelas, é necessário um mínimo de três junções. Esta regra pode não se aplicar se sua tabela possuir uma chave primária concatenada, no caso de mais de uma coluna ser necessária para identificar exclusivamente cada linha.

Para obter mais informações, consulte o Oracle Server SQL Reference Manual, Release 8, "SELECT".

Produto Cartesiano

• Um produto cartesiano é formado quando:

– Uma condição de junção estiver omitida

– Uma condição de junção estiver inválida

– Todas as linhas na primeira tabela estão unidas a todas as linhas da segunda tabela

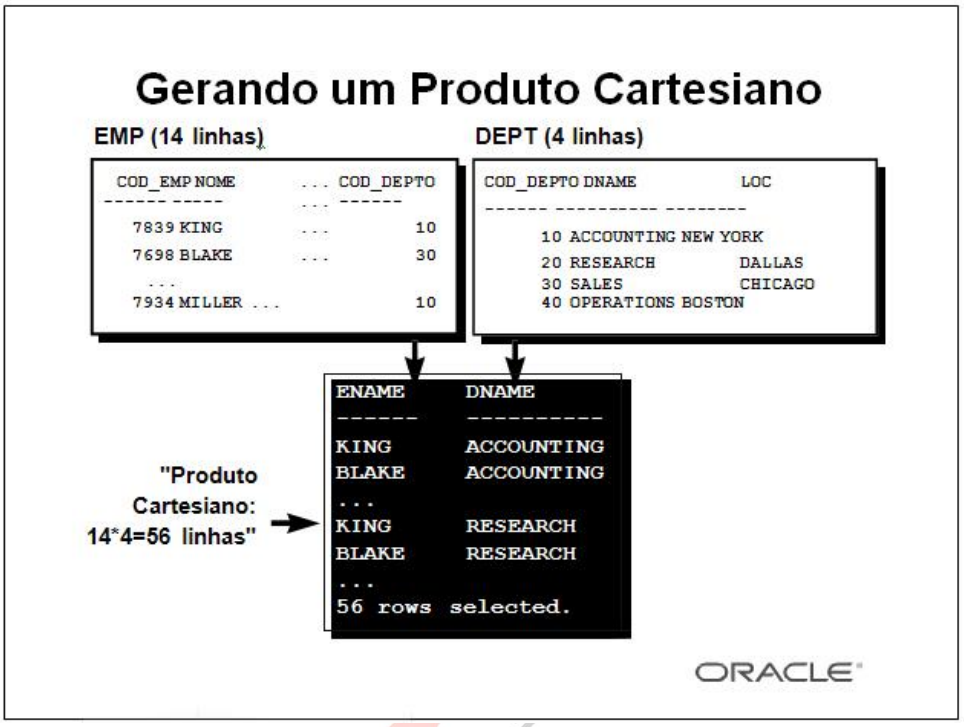
• Para evitar um produto Cartesiano, sempre inclua uma condição de junção válida em uma cláusula WHERE.

Produto Cartesiano

Quando uma condição de junção for completamente inválida ou omitida, o resultado é um produto Cartesiano no qual serão exibidas todas as combinações de linhas. Todas as linhas da primeira tabela estão unidas a todas as linhas da segunda.

Um produto Cartesiano tende a gerar um grande número de linhas e seu resultado raramente é útil.

Você deve sempre incluir uma condição de junção em uma cláusula WHERE, a menos que tenha uma necessidade específica de combinar todas as linhas de todas as tabelas.

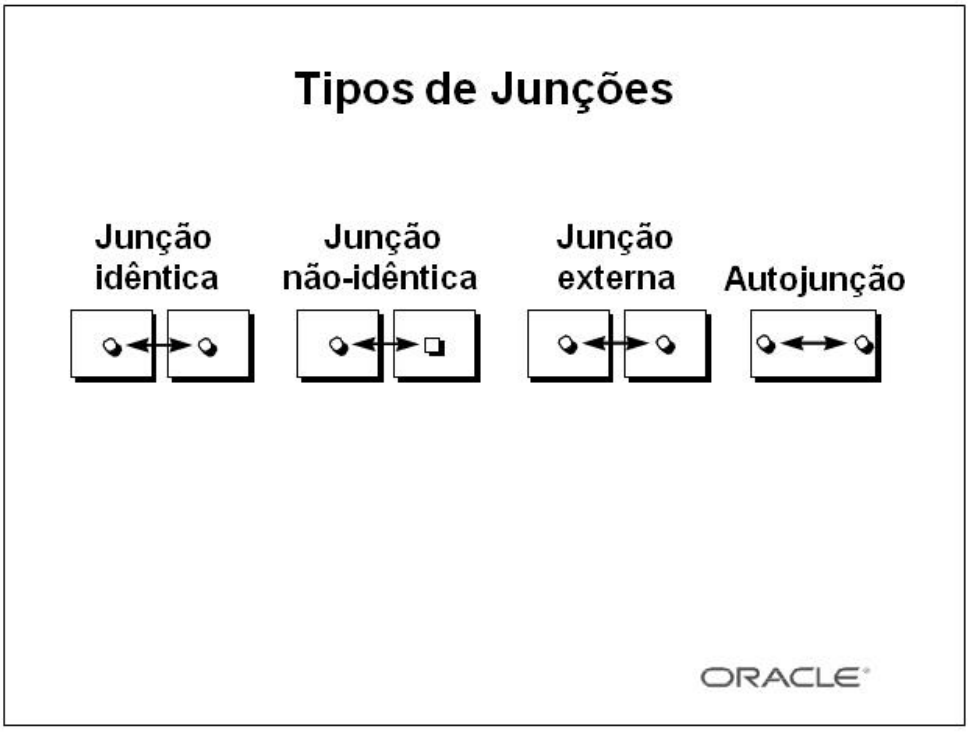


Produto Cartesiano (continuação)

Gera-se um produto Cartesiano caso uma condição de junção seja omitida. O exemplo do slide exibe

o nome do funcionário e do departamento a partir das tabelas EMP e DEPT. Porque nenhuma cláusula WHERE foi especificada, todas as linhas (14 linhas) da tabela EMP são unidas a todas as linhas (4 linhas) na tabela DEPT, gerando dessa forma 56 linhas na saída.

```
SQL> SELECT NOME, dname
  2   FROM  EMPREGADO, dept;

NOME       DNAME
---------- --------------
KING       ACCOUNTING
BLAKE      ACCOUNTING
...

KING       RESEARCH
BLAKE      RESEARCH
...
56 rows selected.
```



Tipos de Junções

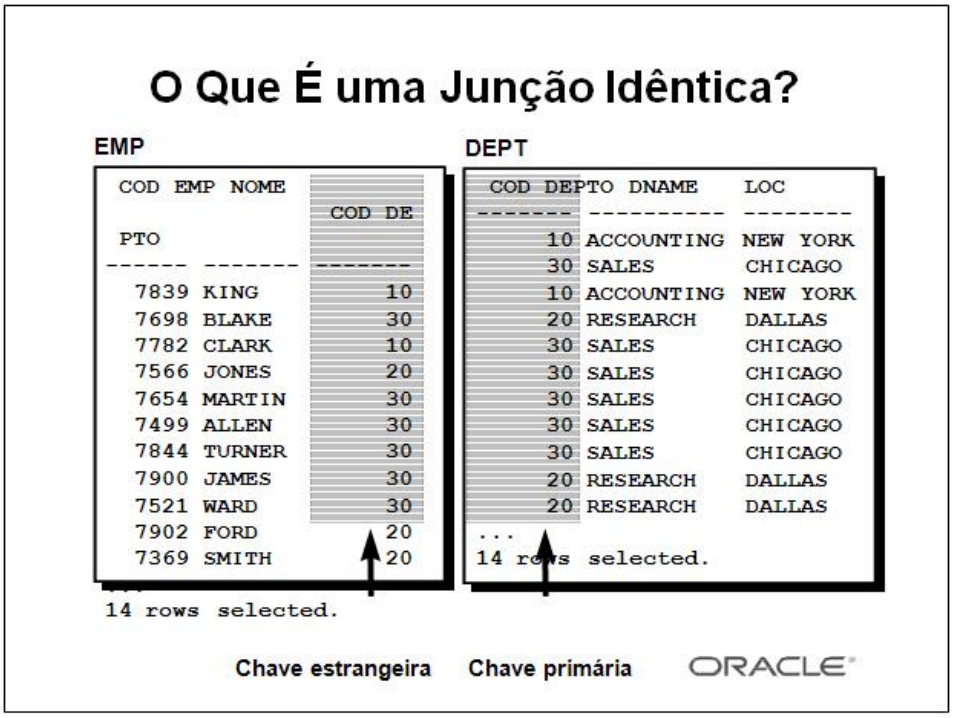
Há dois tipos principais de condições de junção:

- Junções idênticas
- Junções não-idênticas

Métodos de junção adicional incluem o seguinte:

- Junções externas
- Autojunções
- Operadores de conjunto

Observação: Os operadores de conjunto não são abordados neste curso. Eles são abordados em outro curso SQL.



Junções idênticas

Para determinar o nome do departamento de um funcionário, compare o valor na coluna COD_DEPTO na tabela EMP com os valores COD_DEPTO da tabela DEPT. O relacionamento entre as tabelas EMP e

DEPT é uma junção idêntica — ou seja, os valores da coluna COD_DEPTO das duas tabelas devem ser iguais. Com freqüência, essa junção envolve complementos de chave primária e estrangeira.

Observação: As junções idênticas também são chamadas de junções simples ou junções internas.

Recuperando Registros com Junções Idênticas

```
SQL> SELECT EMPREGADO.COD_EMP,      EMPREGADO.NOME,
EMPREGADO.COD_DEPTO,
2           dept.COD_DEPTO, dept.loc
3     FROM  EMPREGADO, dept
4     WHERE EMPREGADO.COD_DEPTO=dept.COD_DEPTO;

COD_EMP NOME       COD_DEPTO COD_DEPTO LOC
```

```
----- ------ ------ ------ ---------
NEW YORK
CHICAGO
NEW YORK
DALLAS
...
14 rows selected.
```

Recuperando Registros com Junções Idênticas

No exemplo do slide:

- A cláusula SELECT especifica os nomes de coluna a recuperar:
  - nome do funcionário, número do funcionário e número do departamento, que são as colunas na tabela EMP
  - número do departamento, nome do departamento e localização, que são as colunas na tabela DEPT
- A cláusula FROM especifica as duas tabelas que o banco de dados deve acessar:
  - tabela EMP
  - tabela DEPT
- A cláusula WHERE especifica como as tabelas serão unidas: EMP.COD_DEPTO=DEPT.COD_DEPTO

Porque a coluna COD_DEPTO é comum às duas tabelas, ela deve estar prefixada pelo nome da tabela a fim de evitar ambigüidade.

Qualificando Nomes de Coluna Ambíguos

• Use os prefixos de tabela para qualificar nomes de coluna que estão em várias tabelas.

• Melhore o desempenho usando os prefixos de tabela.

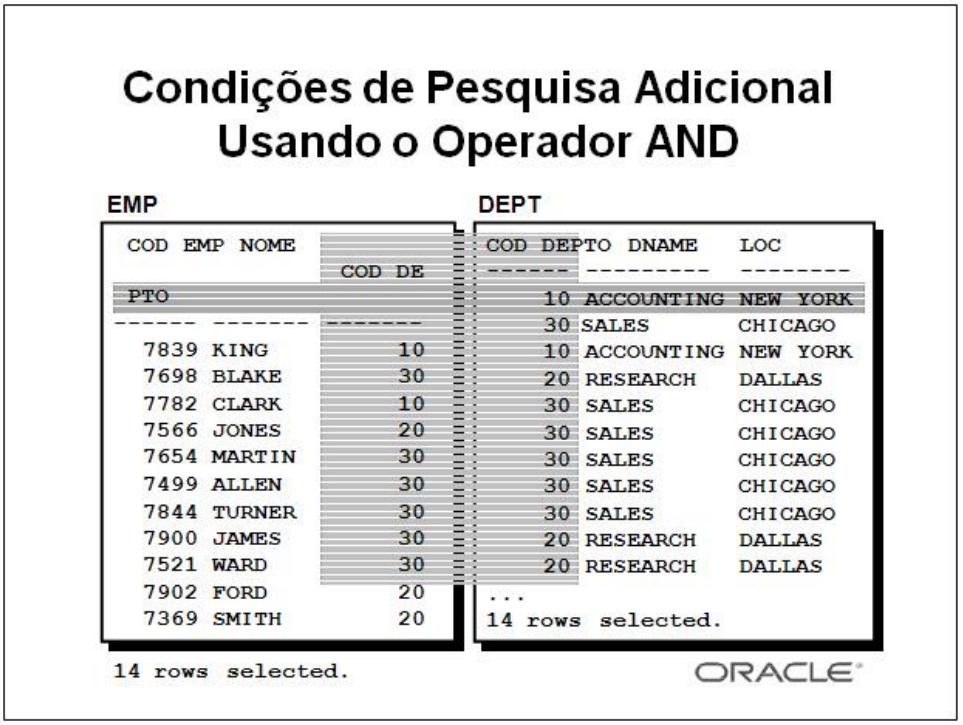
• Diferencie colunas que possuem nomes idênticos, mas que residam em tabelas diferentes usando apelidos de coluna.

Qualificando Nomes de Coluna Ambíguos

Você precisa qualificar os nomes das colunas na cláusula WHERE com o nome da coluna a fim de evitar ambigüidade. Sem os prefixos de tabela, a coluna COD_DEPTO pode vir tanto da tabela DEPT quanto da EMP. É necessário adicionar o prefixo da tabela para executar a consulta.

Se não houver nenhum nome de coluna comum entre as duas tabelas, não haverá necessidade de qualificar as colunas. No entanto, você obterá um melhor desempenho usando o prefixo da tabela, pois informa ao Oracle Server exatamente onde ir para localizar colunas.

A necessidade de qualificar nomes de coluna ambíguos é também aplicável às colunas que podem estar ambíguas em outras cláusulas, tais como a cláusula SELECT ou a ORDER BY.



Condições de Pesquisa Adicional

Além da junção, é possível ter critérios para a cláusula WHERE. Por exemplo, para exibir o número de funcionário, o nome, o número de departamento e a localização do departamento do funcionário

King, você precisa de uma condição adicional na cláusula WHERE.

```
SQL> SELECT COD_EMP, NOME, EMPREGADO.COD_DEPTO, loc
  2   FROM  EMPREGADO, dept
  3   WHERE EMPREGADO.COD_DEPTO = dept.COD_DEPTO
  4   AND   INITCAP(NOME) = 'King';
```

```
COD_EMP NOME                 COD_DEPTO LOC
--------- ----------    --------- -------------
7839 KING                      10 NEW YORK
```

Usando Apelidos de Tabela

Simplifique consultas usando apelidos de tabela.

```
SQL> SELECT EMPREGADO.COD_EMP,      EMPREGADO.NOME,
EMPREGADO.COD_DEPTO,

2           dept.COD_DEPTO, dept.loc
3      FROM  EMPREGADO EMPREGADO, dept
4      WHERE EMPREGADO.COD_DEPTO=dept.COD_DEPTO;


SQL> SELECT e.COD_EMP,  e.NOME, e.COD_DEPTO,
2      d.COD_DEPTO, d.loc
```

Apelidos de Tabela

Qualificar nomes de coluna com nomes de tabela pode consumir muito tempo, principalmente se os nomes da tabela forem extensos. Você pode usar apelidos de tabela no lugar de nomes de tabela. Da mesma forma que um apelido de coluna fornece um outro nome à coluna, um apelido de tabela fornece um outro nome à tabela. Os apelidos de tabela ajudam a manter o código SQL menor, usando dessa

forma menos memória.

No exemplo, note como os apelidos de tabela são identificados na cláusula FROM. O nome de tabela é

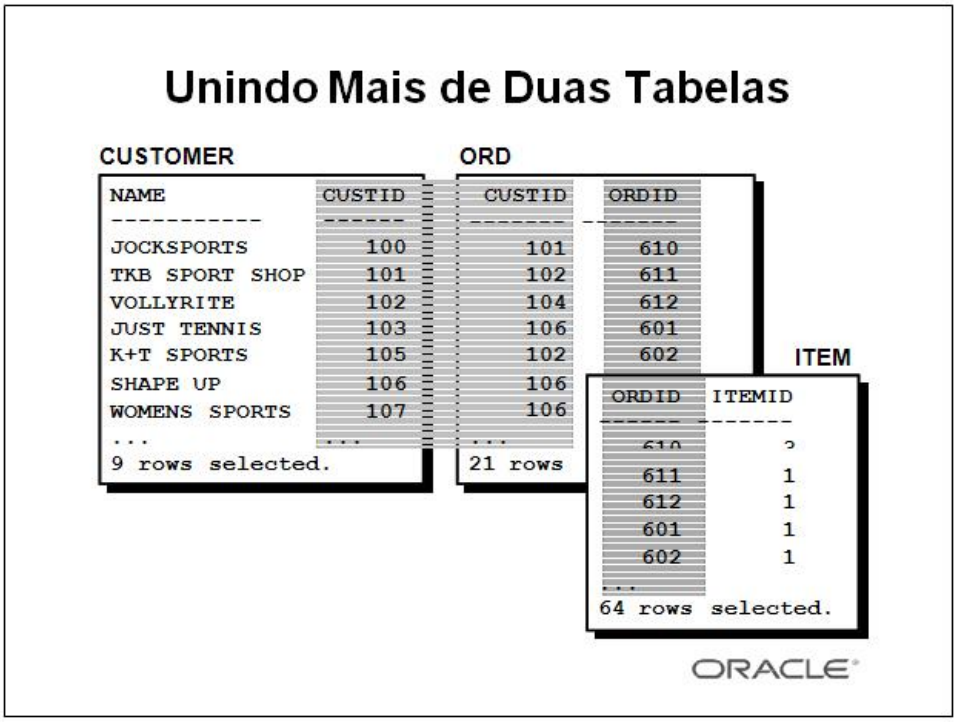
especificado integralmente, seguido de um espaço e, em seguida, do apelido da tabela.

A tabela EMP recebeu o apelido E, enquanto a tabela DEPT tem o apelido D.

Diretrizes

- Os apelidos de tabela podem ter um tamanho de até 30 caracteres, porém quanto menores, melhores.
- Se um apelido de tabela for usado para um determinado nome de tabela na cláusula FROM, então aquele apelido de tabela deve ser substituído para o nome da tabela em toda a instrução SELECT.
- Apelidos de tabela devem ser significativos.

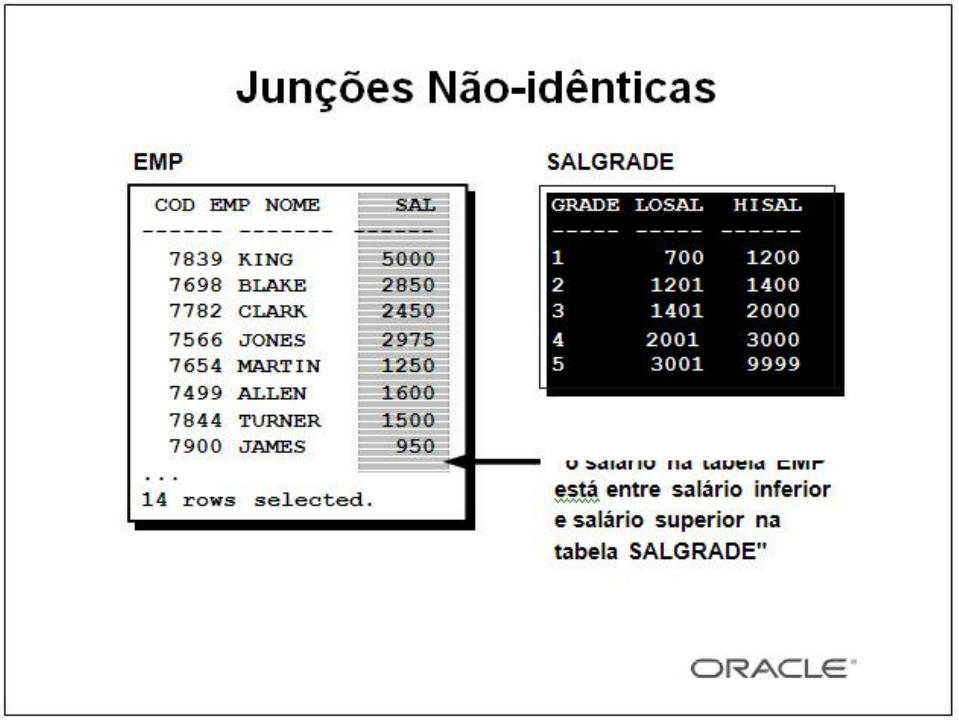
• O apelido de tabela é válido somente para a instrução SELECT atual.



Condições de Pesquisa Adicional

Algumas vezes você pode precisar unir mais de duas tabelas. Por exemplo, para exibir o nome, a colocação das ordens, os números de item, o total para cada item e o total para cada ordem para o cliente TKB SPORT SHOP, você terá de unir as tabelas CUSTOMER, ORD e ITEM.

```
SQL> SELECT c.name, o.ordid, i.itemid, i.itemtot, o.total
  2  FROM customer c, ord o, item i
  3  WHERE c.custid = o.custid
  4   AND   o.ordid =   i.ordid
  5   AND   c.name = 'TKB SPORT SHOP';

NAME  ORDID ITEMID      ITEMTOT     TOTAL
------------ --------- --------- --------- ---------
TKB SPORT SHOP    610   3      58    101.4
TKB SPORT SHOP    610   1      35    101.4
TKB SPORT SHOP    610   2     8.4    101.4
```



Junções Não-Idênticas

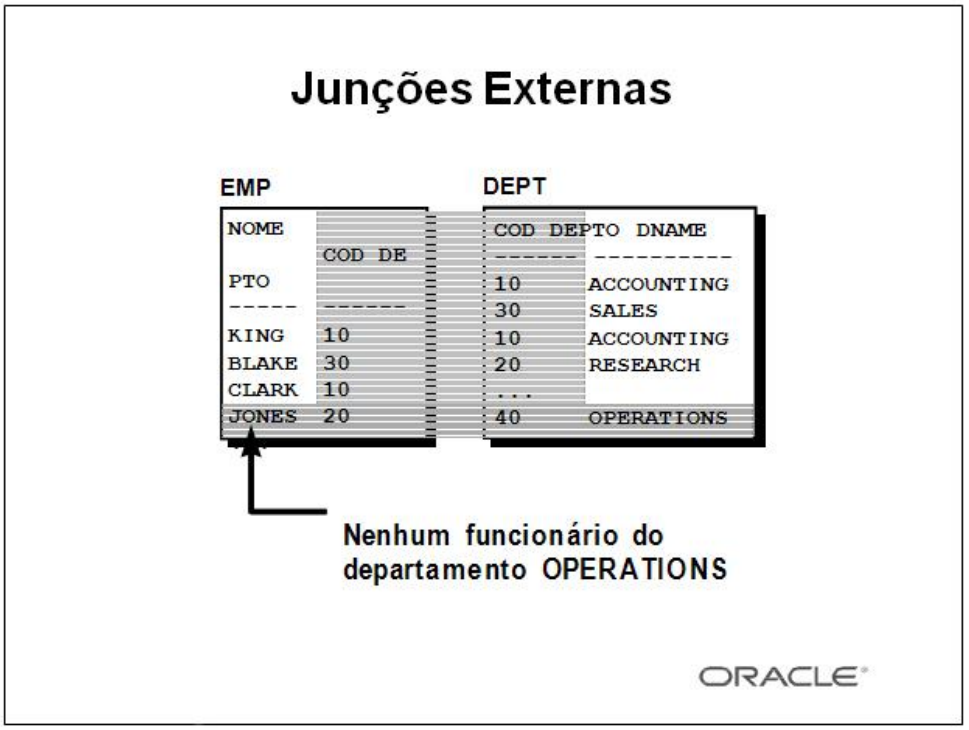
O relacionamento entre a tabela EMP e a tabela SALGRADE é uma junção não-idêntica, o que significa que nenhuma coluna da tabela EMP corresponde diretamente a uma coluna da tabela SALGRADE. O relacionamento entre as duas tabelas é que a coluna SAL da tabela EMP está entre a coluna LOSAL e HISAL da tabela SALGRADE. O relacionamento é obtido usando um outro operador que não o igual (=).

Recuperando Registros com Junções Não-idênticas

```
SQL > SELECT e.name, e.sal, s.grade
   2  FROM    EMPREGADO salgrade s
   3  WHERE  e.sal
   4  BETWEEN     s.losal AND s.hisal;


NOME         SAL          GRADE
----------  ---------   ---------

JAMES        950          1
SMITH        800          1
ADAMS        1100         1
...
14 rows selected.
```

Junções Não-Idênticas (continuação)
O exemplo do slide cria uma junção não-idêntica para avaliar uma classificação de salário do funcionário.
O salário deve estar entre qualquer par de faixas salariais inferior ou superior.
É importante notar que todos os funcionários aparecem exatamente uma vez quando esta consulta é executada. Nenhum funcionário é repetido na lista. Há dois motivos para isto:

•Nenhuma das linhas na tabela de classificação salarial possui classificações que se sobrepõem. Isto é, o valor do salário para um funcionário pode estar somente entre valores salariais superiores e inferiores de uma das linhas tabela de classificação salarial.
•Todos os salários dos funcionários estão entre limites fornecidos pela tabela de classificação salarial. Ou seja, nenhum funcionário ganha menos que o valor contido na coluna LOSAL ou mais que o valor mais alto contido na coluna HISAL.
Observação: Outros operadores tais como <= e >= podem ser utilizados, porém BETWEEN é o mais simples. Lembre-se de especificar o primeiro valor inferior e o último valor superior ao usar BETWEEN. Foram especificados apelidos de tabela por motivos de desempenho, não por causa da possibilidade de ambigüidade.



Retornando Registros sem Correspondência Direta com as Junções Externas

Se uma linha não satisfizer uma condição de junção, a linha não aparecerá no resultado da consulta. Por exemplo, na condição de junção idêntica das tabelas EMP e DEPT, o departamento OPERATIONS não aparece porque ninguém trabalha neste departamento.

```
SQL> SELECT e.NOME, e.COD_DEPTO, d.dname
  2   FROM  EMPREGADO e,      dept d
  3   WHERE e.COD_DEPTO = d.COD_DEPTO;

NOME        COD_DEPTO DNAME
----------  --------- -------------
KING        10    ACCOUNTING
BLAKE       30    SALES
CLARK       10    ACCOUNTING
JONES       20    RESEARCH
...
ALLEN       30    SALES
TURNER      30    SALES
```

```
JAMES        30     SALES
...
14 rows selected.
```

Junções Externas

• Use uma junção externa para consultar também todas as linhas que em geral não atendem à condição de junção.

• O operador de junção externo é um sinal de adição (+).

```
SELECT tabela1.coluna, tabela2.coluna
FROM  tabela1, tabela2
WHERE tabela1.coluna(+) = tabela2.coluna;


SELECT tabela1.coluna , tabela2.coluna
FROM  tabela1, tabela2
WHERE tabela1.coluna = tabela2.coluna(+);
```

Retornando Registros sem Correspondência Direta com as Junções Externas
(continuação)

A(s) linha(s) ausente(s) pode(m) ser retornada(s) se um operador de junção externa for utilizado na condição de junção. O operador é um sinal de adição entre parênteses (+), *e é colocado ao "lado" da junção que está com informação insuficiente*. Este operador possui o efeito de criar uma ou mais

linhas nulas, para qual uma ou mais linhas da tabela não-deficiente pode ser unida. Na sintaxe:

tabela1.coluna = é a condição que une (ou relaciona) as tabelas.

tabela2.coluna (+) é o símbolo de junção externa, que pode ser colocado em

qualquer lado da condição da cláusula WHERE, mas não dos dois lados (Coloque o símbolo de junção externa seguindo o nome da coluna na tabela sem linhas correspondidas.)

Usando Junções Externas

```
SQL> SELECT e.NOME, d.COD_DEPTO, d.dname
  2   FROM  EMPREGADO e,      dept d
  3   WHERE e.COD_DEPTO(+) =  d.COD_DEPTO
  4   ORDER BY    e.COD_DEPTO;


NOME  COD_DEPTO DNAME
---------- --------- -------------
KING  10 ACCOUNTING
CLARK 10 ACCOUNTING
...
      40 OPERATIONS


15 rows selected.
```

Retornando Registros sem Correspondência Direta com as Junções Externas (continuação)

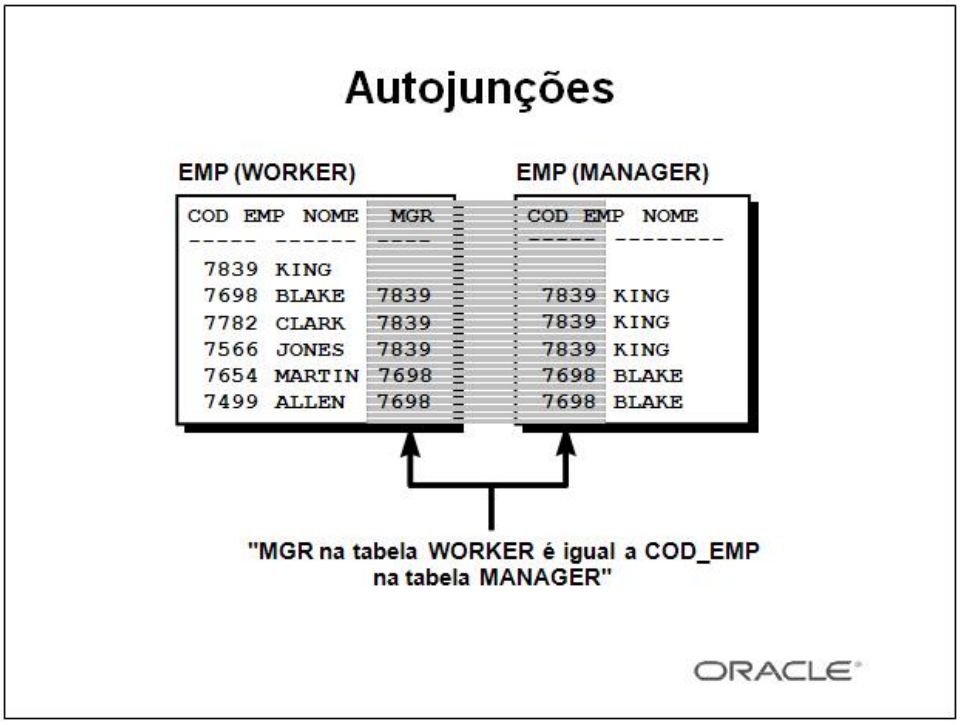
O exemplo do slide exibe números e nomes para todos os departamentos. O departamento

OPERATIONS, que não possui nenhum funcionário, também é exibido.

Restrições da Junção Externa

•O operador da junção externa pode aparecer somente de um lado da expressão — o lado que possui informações ausentes. Ele retorna estas linhas de uma tabela que não possui correspondência direta em outra tabela.

•Uma condição envolvendo uma junção externa não pode usar o operador IN ou estar vinculada a outra condição pelo operador OR.



Unindo uma Tabela a Ela Mesma

Algumas vezes será necessário unir uma tabela a ela mesma. Para localizar o nome do gerente de cada funcionário, é necessário unir a tabela EMP a ela mesma ou executar uma autojunção. Por exemplo, para localizar o nome do gerente de Blake, é necessário:

•Localizar Blake na tabela EMP consultando a coluna NOME.

•Localizar o número do gerente de Blake consultando a coluna MGR. O número do gerente de

Blake é 7839.

•Localizar o nome do gerente como o COD_EMP 7839 consultando a coluna NOME. O número do funcionário King é 7839, então King é gerente de Blake.

Neste processo, você analisa a tabela duas vezes. Na primeira vez, você consulta a tabela para

localizar Blake na coluna NOME e o valor MGR de 7839. Na segunda vez, você consulta a coluna COD_EMP para localizar 7839 e a coluna NOME para localizar King.

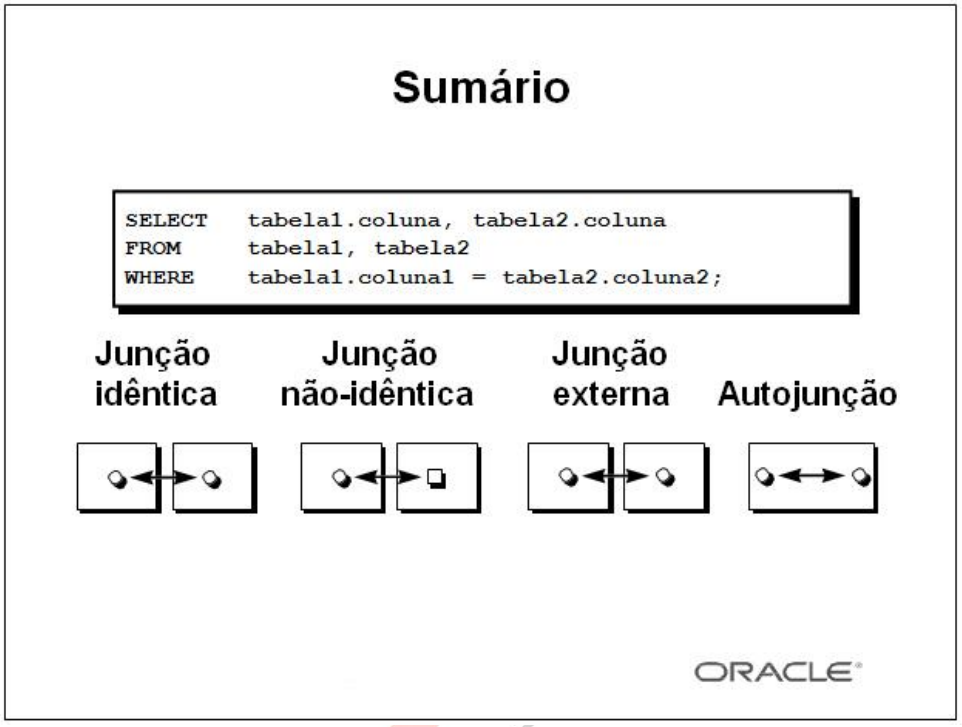
Unindo uma Tabela a Ela Mesma

```
SQL> SELECT worker.NOME||' works for '||manager.NOME
  2   FROM  EMPREGADO worker, EMPREGADO manager
  3   WHERE worker.mgr = manager.COD_EMP;


WORKER.NOME||'WORKSFOR'||MANAG
------------------------------
BLAKE works for KING
CLARK works for KING
JONES works for KING
MARTIN works for BLAKE
...
13 rows selected.
```

Unindo uma Tabela a Ela Mesma (continuação)

O exemplo do slide une a tabela EMP a ela mesma. Para simular duas tabelas na cláusula FROM, há dois apelidos, chamados WORKER e MANAGER, para a mesma tabela EMP.

Neste exemplo, a cláusula WHERE contém a junção que significa "em que lugar o número de gerente do trabalhador corresponde ao número do funcionário para o gerente".



Sumário

Há vários modos para juntar tabelas. O elemento comum, no entanto, é aquele que deseja vincular através de uma condição na cláusula WHERE. O método que você escolher será baseado nas estruturas de dados que estiver usando.

```
SELECT      tabela1.coluna, tabela2.coluna
FROM        tabela1, tabela2
WHERE       tabela1.coluna1 = tabela2.coluna2;
```

Capítulo 06

Agregando Dados Usando Funções de Grupo

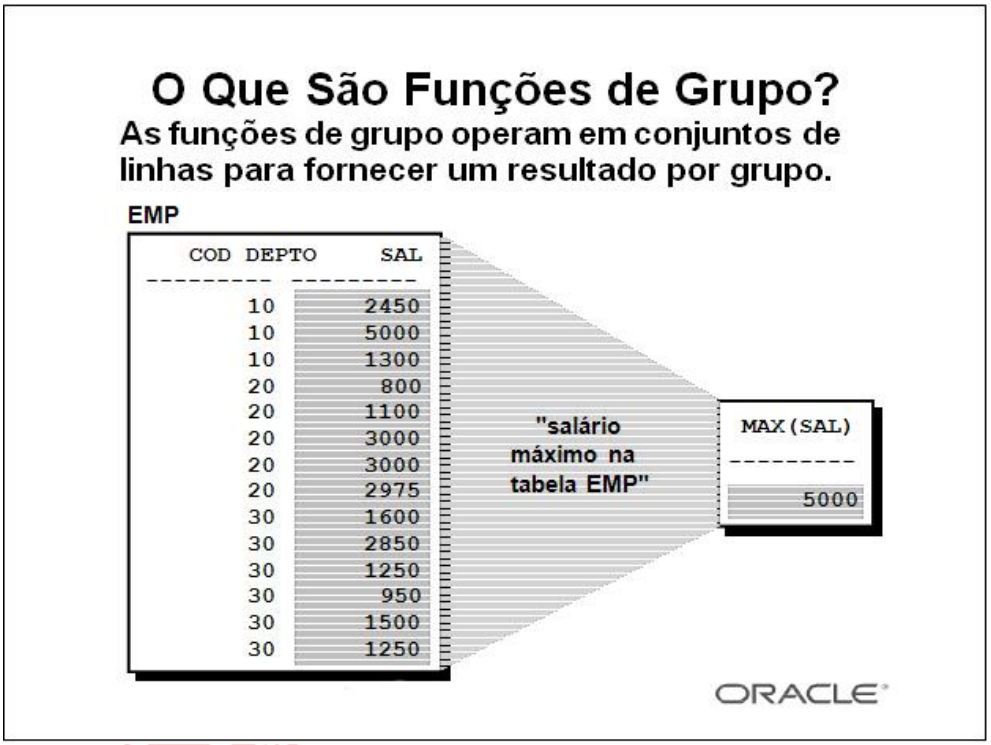
Objetivos

Depois de completar esta lição, você poderá fazer o seguinte:

- Identificar as funções de grupo disponíveis
- Descrever o uso de funções de grupo
- Agrupar dados usando a cláusula GROUP BY
- Incluir ou excluir linhas agrupadas usando a cláusula HAVING

Objetivo da Lição

Esta lição aborda funções. Visa obter informações sumariadas, tais como médias, para grupos de linhas. Discute como agrupar linhas de uma tabela em conjuntos menores e como especificar critérios de pesquisa para grupos de linhas.



Funções de Grupo

De modo diferente das funções de uma única linha, as funções de grupo operam em conjuntos de linhas para fornecer um resultado por grupo. Esses conjuntos podem ser a tabela inteira ou a tabela dividida em grupos.

Tipos de Funções de Grupo

- AVG
- COUNT

- MAX
- MIN
- STDDEV
- SUM
- VARIANCE

Funções de Grupo (continuação)
Cada uma das funções aceita um argumento. A tabela a seguir identifica as opções que podem ser usadas na sintaxe:

Função	Descrição
AVG([DISTINCT|ALL]*n*)	Valor médio de *n*, ignorando valores nulos
COUNT({*|[DISTINCT|ALL]*expr*})	Número de linhas, onde *expr* avalia para algo diferente de nulo (Conte todas as linhas selecionadas usando *, inclusive duplicadas e linhas com nulos.)
MAX([DISTINCT|ALL]*expr*)	Valor máximo de *expr*, ignorando valores nulos
MIN([DISTINCT|ALL]*expr*)	Valor mínimo de *expr*, ignorando valores nulos
STDDEV([DISTINCT|ALL]*x*)	Desvio padrão de *n*, ignorando valores nulos
SUM([DISTINCT|ALL]*n*)	Valores somados de *n*, ignorando valores nulos
VARIANCE([DISTINCT|ALL]*x*)	Variação de *n*, ignorando valores nulos

Usando Funções de Grupo

```
SELECT      [coluna,] group_function(coluna)
FROM        tabela
[WHERE      condição]
[GROUP BY   coluna]
[ORDER BY   coluna];
```

Diretrizes para o Uso de Funções de Grupo

•DISTINCT faz com que a função considere somente valores não-duplicados; ALL faz com que ela considere cada valor, inclusive duplicados. O default é ALL e, portanto, não precisa ser especificado.

•Os tipos de dados para os argumentos podem ser CHAR, VARCHAR2, NUMBER ou DATE, onde expr está listado.

•Todas as funções de grupo, exceto COUNT(*), ignoram valores nulos. Para substituir um valor por valores nulos, use a função NVL.

•O Oracle Server classifica implicitamente a definição do resultado em ordem crescente quando usa uma cláusula GROUP BY. Para sobrepor essa ordenação default, DESC pode ser usado em uma cláusula ORDER BY.

Usando Funções AVG e SUM

Você pode usar AVG e SUM para dados numéricos.

```
SQL> SELECT AVG(sal), MAX(sal),
  2         MIN(sal), SUM(sal)
  3    FROM  EMPREGADO
  4    WHERE OCUP LIKE   'SALES%';


AVG(SAL)    MAX(SAL)    MIN(SAL)    SUM(SAL)
--------  ---------  ---------  ---------
1400        1600        1250        5600
```

Funções de Grupo

Você pode usar as funções AVG, SUM, MIN e MAX com colunas que possam armazenar dados numéricos. O exemplo no slide exibe os salários maior, médio, menor e a soma dos salários mensais de todos os vendedores.

Usando Funções MIN e MAX

Você pode usar MIN e MAX para qualquer tipo de dados.

```
SQL> SELECT MIN(hiredate), MAX(hiredate)
  2    FROM  EMPREGADO;

MIN(HIRED   MAX(HIRED
---------   ---------
17-DEC-80   12-JAN-83
```

Funções de Grupo (continuação)

Você pode usar as funções MAX e MIN para qualquer tipo de dados. O exemplo no slide exibe o funcionário mais antigo e o mais novo.

O exemplo a seguir exibe o primeiro e o último nome de funcionário em uma lista alfabética de todos os funcionários.

```
SQL> SELECT MIN(NOME), MAX(NOME)
  2   FROM  EMPREGADO;


MIN(NOME)    MAX(NOME)
----------   ----------
ADAMS        WARD
```

Observação: As funções AVG, SUM, VARIANCE e STDDEV só podem ser usadas com tipos de dados numéricos.

Usando a Função COUNT

COUNT(*) retorna o número de linhas em uma tabela.

```
SQL>  SELECT      COUNT(*)
  2   FROM      EMPREGAD
  3   WHERE     deptno = 30;

COUNT(*)

---------
```

A Função COUNT
A Função COUNT tem dois formatos:
•*COUNT(*)*
•*COUNT(expr)*

COUNT(*) retorna o número de linhas em uma tabela, inclusive linhas duplicadas e linhas contendo valores nulos em qualquer uma das colunas. Se uma cláusula WHERE estiver incluída na instrução SELECT, COUNT(*) retornará o número de linhas que satisfizer a condição na cláusula WHERE.

Entretanto, COUNT(expr) retorna o número de linhas não nulas na coluna identificada por expr.
O exemplo no slide exibe o número de funcionários no departamento 30.

Usando a Função COUNT

COUNT(expr) retorna o número de linhas não nulas.

```
SQL>    SELECT      COUNT(comm)
   2   FROM       EMPREGAD
   3   WHERE      deptno = 30;


   4  COUNT(COMM)

-----------
```

A Função COUNT (continuação)

O exemplo no slide exibe o número de funcionários no departamento 30 que podem receber uma comissão. Observe que o resultado fornece o número total de quatro linhas porque dois funcionários no departamento 30 não podem receber a comissão e, portanto, a coluna COMM contém um valor nulo.

Exemplo

Exiba o número de departamentos na tabela EMP.

```
SQL> SELECT COUNT(COD_DEPTO)
  2    FROM  EMPREGADO;

COUNT(COD_DEPTO)
----------------
              14
```

Exiba o número de departamentos distintos na tabela EMP.

```
SQL> SELECT COUNT(DISTINCT (COD_DEPTO))
  2    FROM  EMPREGADO;

COUNT(DISTINCT(COD_DEPTO))
--------------------------
                         3
```

Funções de Grupo e Valores Nulos

As funções de grupo ignoram valores nulos na coluna.

```
SQL> SELECT AVG(comm)
2      FROM  EMPREGADO;

AVG(COMM)
---------
550
```

Funções de Grupo e Valores Nulos

Todas as funções de grupo, com a exceção de COUNT (*), ignoram valores nulos na coluna. No exemplo do slide, a média é calculada com base somente nas linhas da tabela em que um valor válido é armazenado na coluna COMM. A média é calculada como o total da comissão sendo paga a todos os funcionários dividido pelo número de funcionários recebendo a comissão (4).

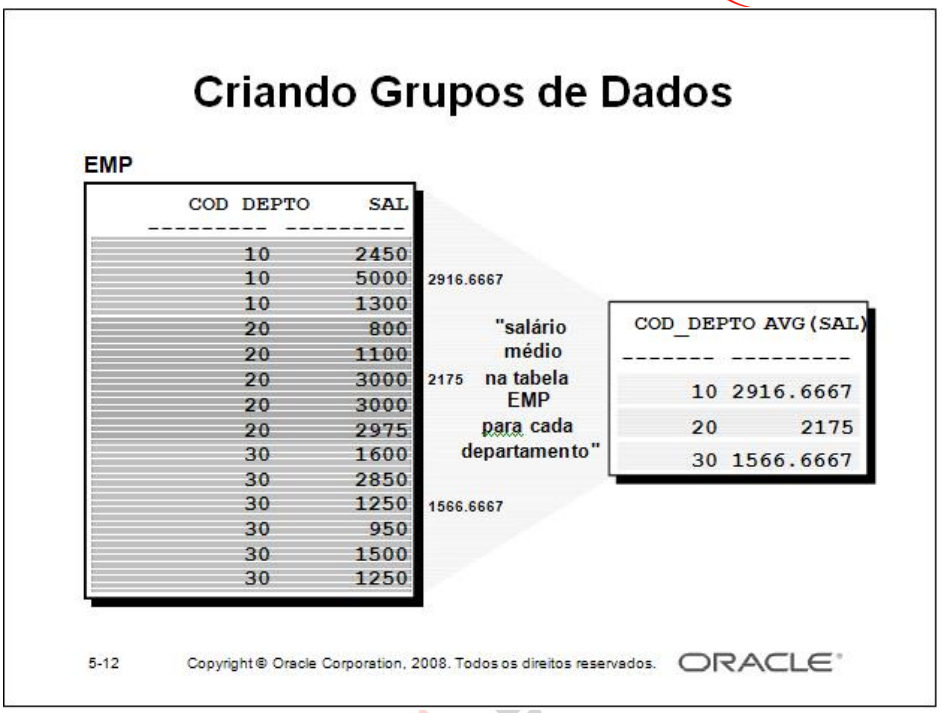
Usando a Função NVL com Funções de Grupo

A função NVL força as funções de grupo a incluírem valores nulos.

```
SQL> SELECT AVG(NVL(comm,0))
  2    FROM  EMPREGADO;


AVG(NVL(COMM,0))
----------------
157.14286
```

Funções de Grupo e Valores Nulos (continuação)

A função NVL força as funções de grupo a incluírem valores nulos. No exemplo do slide, a média é calculada com base em todas as linhas na tabela, independentemente de os valores nulos estarem armazenados na coluna COMM. A média é calculada como o total da comissão sendo paga a todos os funcionários dividido pelo número total de funcionários na empresa (14).



Grupos de Dados

Até este momento, todas as funções de grupo trataram a tabela como um grande grupo de informações. Às vezes, é necessário dividir a tabela de informações em grupos menores. Isso pode ser feito usando a cláusula GROUP BY.

Criando Grupos de Dados: Cláusula GROUP BY

```
SELECT      coluna, group_function(coluna)
FROM        tabela
[WHERE      condição]
[GROUP BY   group_by_expression]
[ORDER BY   coluna];
```

Divida linhas de uma tabela em grupos menores usando a cláusula GROUP BY.

A Cláusula GROUP BY

Você pode usar a cláusula GROUP BY para dividir as linhas de uma tabela em grupos. Em seguida, pode usar as funções de grupo para retornar informações sumárias para cada grupo.

Na sintaxe:

group_by_expression especifica colunas cujos valores determinam a base para agrupar linhas

Diretrizes

- Se você incluir uma função de grupo em uma cláusula SELECT, não poderá selecionar resultados individuais, a menos que a coluna individual apareça na cláusula GROUP BY. Se você não conseguir incluir a lista de colunas, uma mensagem de erro será exibida.
- Ao usar uma cláusula WHERE, você pode excluir linhas com antecedência antes de dividi-las em grupos.
- Você deve incluir as colunas na cláusula GROUP BY.
- Não é possível usar o apelido de coluna na cláusula GROUP BY.
- Por default, as linhas são classificadas por ordem crescente das colunas incluídas na lista

GROUP BY. Isso pode ser sobreposto usando a cláusula ORDER BY.

Usando a Cláusula GROUP BY

Todas as colunas na lista SELECT que não estejam em funções de grupo devem estar na cláusula GROUP BY.

```
SQL> SELECT COD_DEPTO, AVG(sal)
  2  FROM   EMPREGADO
  3  GROUP BY deptno;

COD_DEPTO   AVG(SAL)
--------- ---------
10 2916.6667
20    2175
30 1566.6667
```

A Cláusula GROUP BY (continuação)

Quando utilizar a cláusula GROUP BY, certifique-se de que todas as colunas na lista SELECT que não estejam nas funções de grupo estejam incluídas na cláusula GROUP BY. O exemplo no slide exibe o número do departamento e o salário médio para cada

departamento. Essa instrução SELECT, que contém uma cláusula GROUP BY, é avaliada da seguinte forma:

•A cláusula SELECT especifica as colunas a serem recuperadas:

– A coluna de números de departamento na tabela EMP

– A média de todos os salários no grupo que você especificou na cláusula GROUP BY

•A cláusula FROM especifica as tabelas que o banco de dados deve acessar:

a tabela EMP.

•A cláusula WHERE especifica as linhas a serem recuperadas. Já que não há uma cláusula

WHERE, todas as linhas são recuperadas por default.

•A cláusula GROUP BY especifica como as linhas devem ser agrupadas. As linhas são agrupadas pelo número do departamento, de forma que a função AVG que esteja sendo aplicada à coluna de salários calcule o salário médio para cada departamento.

Usando a Cláusula GROUP BY

A coluna GROUP BY não precisa estar na lista SELECT.

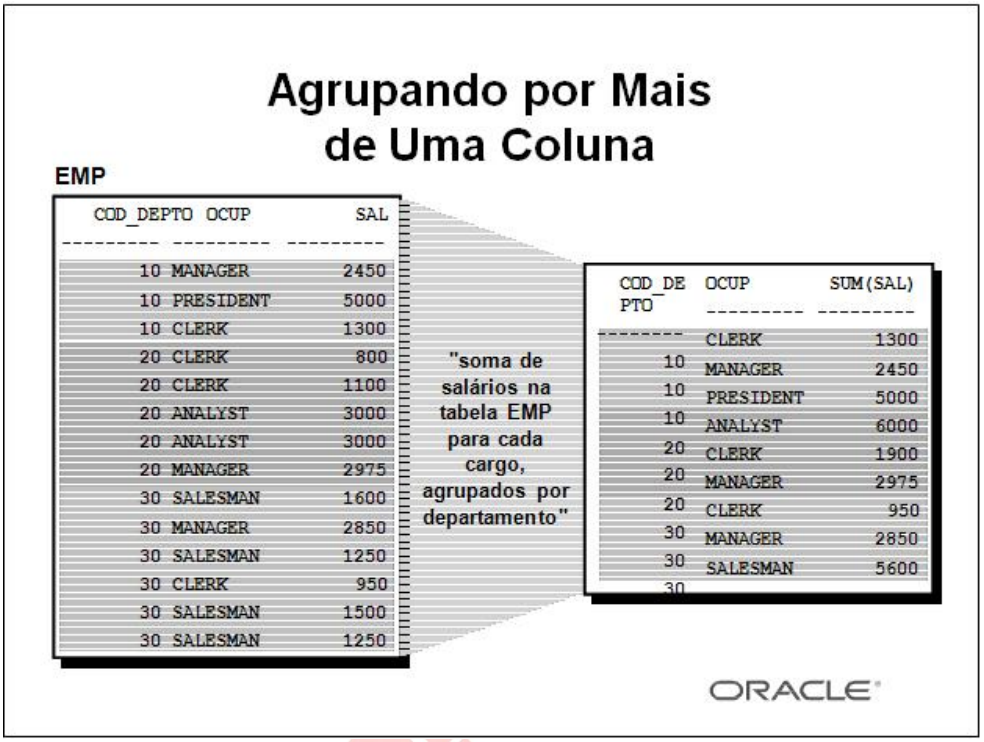
```
SQL> SELECT AVG(sal)
2      FROM  EMPREGADO
3      GROUP BY    COD_DEPTO;
```

A Cláusula GROUP BY (continuação)

A coluna GROUP BY não precisa estar na cláusula SELECT. Por exemplo, a instrução SELECT no slide exibe os salários médios para cada departamento sem exibir os respectivos números dos departamentos. Sem os números dos departamentos, entretanto, os resultados não parecem significativos.

Você pode usar a função de grupo na cláusula ORDER BY.

```
SQL> SELECT COD_DEPTO, AVG(sal)
2      FROM  EMPREGADO
3      GROUP BY    COD_DEPTO
4      ORDER BY    AVG(sal);

COD_DEPTO   AVG(SAL)
---------- ------------
30     1566.6667
20     2175
10     2916.6667
```



Grupos Dentro de Grupos

Às vezes há necessidade de ver os resultados para grupos dentro de grupos. O slide mostra um relatório que exibe o salário total sendo pago a cada cargo, dentro de cada departamento.

A tabela EMP é agrupada primeiro pelo número do departamento e, dentro desse agrupamento, é agrupada pelo cargo. Por exemplo, os dois escriturários no departamento 20 estão agrupados e um único resultado (salário total) é produzido para todos os vendedores dentro do grupo.

Usando a Cláusula GROUP BY em Várias Colunas

```
SQL> SELECT COD_DEPTO, OCUP, sum(sal)
  2   FROM  EMPREGADO
  3   GROUP BY    COD_DEPTO, OCUP;
```

Grupos Dentro de Grupos (continuação)

É possível retornar resultados sumários para grupos e subgrupos listando mais de uma coluna GROUP BY. Você pode determinar a ordem de classificação default dos resultados pela ordem das colunas na cláusula GROUP BY. A instrução SELECT no slide, que contém uma cláusula GROUP BY, é avaliada da seguinte forma:

•A cláusula SELECT especifica a coluna a ser recuperada:

–O número do departamento na tabela EMP

–O cargo na tabela EMP

–A soma de todos os salários no grupo que você especificou na cláusula GROUP BY

•A cláusula FROM especifica as tabelas que o banco de dados deve acessar: a tabela EMP.

•A cláusula GROUP BY especifica como você deve agrupar as linhas:

–Primeiro, as linhas são agrupadas pelo número do departamento

–Em seguida, dentro dos grupos de números de departamentos, as linhas são agrupadas pelo cargo

Dessa forma, a função SUM é aplicada à coluna de salários para todos os cargos dentro de cada grupo de números de departamentos.

Consultas Ilegais Usando Funções de Grupo

Qualquer coluna ou expressão na lista SELECT
que não seja uma função agregada deve estar na cláusula GROUP BY.

```
SQL> SELECT COD_DEPTO, COUNT(NOME)
2      FROM  EMPREGADO;


SELECT COD_DEPTO, COUNT(NOME)
*
ERROR at line 1:
ORA-00937: Nenhuma função de grupo de grupo único
(Not a single-group group function)
```

Consultas Ilegais Usando Funções de Grupo

Sempre que você usar uma mistura de itens individuais (COD_DEPTO) e funções de grupo (COUNT) na mesma instrução SELECT, deve incluir uma cláusula GROUP BY que especifique os itens individuais (neste caso, COD_DEPTO). Se a cláusula GROUP BY estiver ausente, aparecerá a mensagem de erro "Nenhuma função de grupo de grupo único" e um asterisco (*) apontará para a coluna que contiver o erro. Você pode corrigir o erro no slide ao adicionar a cláusula GROUP BY.

```
SQL> SELECT COD_DEPTO,COUNT(NOME)
  2   FROM  EMPREGADO
  3   GROUP BY    COD_DEPTO;

COD_DEPTO   COUNT(NOME)
---------- ------------
      10     3
      20     5
      30     6
```

Qualquer coluna ou expressão na lista SELECT que não seja uma função agregada deve estar na cláusula GROUP BY.

Consultas Ilegais Usando Funções de Grupo

- Não é possível usar a cláusula WHERE para restringir grupos.
- Use a cláusula HAVING para restringir grupos.

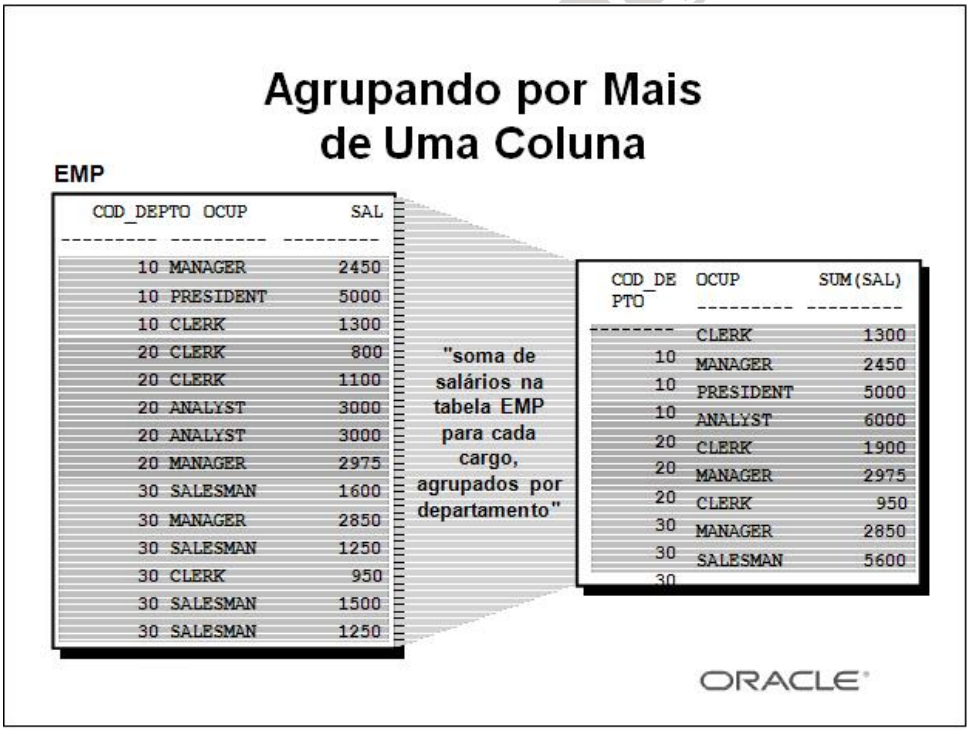
```
SQL> SELECT COD_DEPTO, AVG(sal)
  2   FROM  EMPREGADO
  3   WHERE AVG(sal) > 2000
  4   GROUP BY COD_DEPTO;


WHERE AVG(sal) > 2000
*
ERROR at line 3:
ORA-00934: A função de grupo não é permitida aqui
(Group function is not allowed here)
```

Consultas Ilegais Usando Funções de Grupo (continuação)

A cláusula WHERE não pode ser usada para restringir grupos. A instrução SELECT no slide resulta em um erro porque usa a cláusula WHERE para restringir a exibição de salários médios dos departamentos que tenham um salário médio maior do que US$ 2.000.

Você pode corrigir o erro no slide usando a cláusula HAVING para restringir grupos.

```
SQL> SELECT COD_DEPTO, AVG(sal)
2      FROM  EMPREGADO
3      GROUP BY    COD_DEPTO
4      HAVING      AVG(sal) > 2000;

COD_DEPTO   AVG(SAL)
---------- --------------
10     2916.6667
20     2175
```



Restringindo Resultados do Grupo

Da mesma forma que você usa a cláusula WHERE para restringir as linhas que seleciona, pode usar a cláusula HAVING para

restringir grupos. Para localizar o salário máximo de cada departamento, mas mostrar apenas os departamentos que tenham um salário máximo de mais do que US$ 2.900, faça o seguinte:

•Localize o salário médio para cada departamento ao agrupar por número do departamento.

•Restrinja os grupos aos departamentos com um salário máximo maior do que US$ 2.900.

Excluindo Resultados do Grupo: Cláusula HAVING

Use a cláusula HAVING para restringir grupos

• As linhas são agrupadas.

• A função de grupo é aplicada.

• Os grupos que correspondem à cláusula HAVING são exibidos.

```
SELECT      coluna, group_function
FROM        tabela
[WHERE      condição]
[GROUP BY   group_by_expression]
[HAVING     group_condition]
[ORDER BY   coluna];
```

A Cláusula HAVING

Use a cláusula HAVING para especificar quais grupos serão exibidos. Portanto, restrinja ainda mais os grupos com base nas informações agregadas. Na sintaxe:

group_condition restringe os grupos de linhas retornados aos grupos para os quais a condição especificada seja TRUE

O Oracle Server executa as seguintes etapas quando você usa a cláusula HAVING:

•As linhas são agrupadas.

•A função de grupo é aplicada ao grupo.

•Os grupos que correspondem aos critérios na cláusula HAVING são exibidos.

A cláusula HAVING pode preceder a cláusula GROUP BY, mas recomenda-se que você coloque a cláusula GROUP BY primeiro, por ser mais lógico. Os grupos são formados e as funções de grupo

são calculadas antes de a cláusula HAVING ser aplicada aos grupos na lista SELECT.

Usando a Cláusula HAVING

```
SQL> SELECT COD_DEPTO, max(sal)
  2   FROM  EMPREGADO
  3   GROUP BY    COD_DEPTO
  4   HAVING      max(sal)>2900;

COD_DEPTO   MAX(SAL)
--------- ---------
       10        5000
       20        3000
```

A Cláusula HAVING (continuação)

O exemplo no slide exibe números de departamentos e o salário máximo para os departamentos cujo salário máximo seja maior do que US$ 2.900.

Você pode usar a cláusula GROUP BY sem usar uma função de grupo na lista SELECT.

Se você restringir linhas baseado no resultado de uma função de grupo, deve ter uma cláusula

GROUP BY assim como a cláusula HAVING.

O exemplo a seguir exibe os números de departamentos e o salário médio para os departamentos cujo salário máximo seja maior do que US$ 2.900:

```
SQL> SELECT COD_DEPTO, AVG(sal)
  2   FROM  EMPREGADO
  3   GROUP BY    COD_DEPTO
  4   HAVING      MAX(sal) >  2900;


COD_DEPTO   AVG(SAL)
--------- ---------
10          2916.6667
20          2175
```

Usando a Cláusula HAVING

```
SQL> SELECT OCUP, SUM(sal) PAYROLL
  2   FROM  EMPREGADO
  3   WHERE OCUP NOT LIKE 'SALES%'
  4   GROUP BY    OCUP


OCUP         PAYROLL
---------  ---------
ANALYST      6000
MANAGER      8275
```

A Cláusula HAVING (continuação)
O exemplo no slide exibe o cargo e o salário mensal total para cada cargo com uma folha de pagamento total excedendo US$ 5.000. O exemplo exclui vendedores e classifica a lista pelo salário mensal total.

Aninhando Funções de Grupo

Exiba o salário médio máximo.

```
SQL> SELECT max(avg(sal))
  2   FROM      EMPREGADO
  3   GROUP BY deptno;


MAX(AVG(SAL))
-------------
      2916.6667
```

Aninhando Funções de Grupo

As funções de grupo podem ser aninhadas até uma profundidade de dois. O exemplo no slide exibe o salário médio máximo.

Sumário

```
SELECT      coluna, group_function(coluna)
FROM        tabela
[WHERE      condição]
```

```
[GROUP BY   group_by_expression]
[HAVING     group_condition]
[ORDER BY   coluna];
```

Ordem de avaliação das cláusulas:

- cláusula WHERE
- cláusula GROUP BY
- cláusula HAVING

Sumário

Sete funções de grupo estão disponíveis no SQL:

- AVG
- COUNT
- MAX
- MIN
- SUM
- STDDEV
- VARIANCE

Você pode criar subgrupos usando a cláusula GROUP BY. Os grupos podem ser excluídos usando a cláusula HAVING.
Coloque as cláusulas HAVING e GROUP BY após a cláusula WHERE em uma instrução. Coloque a cláusula ORDER BY por último.

O Oracle Server avalia as cláusulas na seguinte ordem:

- Se a instrução contém uma cláusula WHERE, o servidor estabelece as linhas do candidato.
- O servidor identifica os grupos especificados na cláusula GROUP BY.
- A cláusula HAVING restringe ainda mais os grupos de resultado que não atendam aos critérios do grupo na cláusula HAVING.

Capítulo 07

Subconsultas

Objetivos

Depois de completar esta lição, você poderá fazer o seguinte:

• Descrever os tipos de problemas que as subconsultas podem resolver

• Definir as subconsultas

• Listar os tipos de subconsultas

• Criar subconsultas de uma única linha e de várias linhas

Objetivo da Lição

Nesta lição, você aprenderá sobre os recursos mais avançados da instrução SELECT. Você pode criar subconsultas na cláusula WHERE de outra instrução SQL para obter valores baseados em um valor condicional desconhecido. Esta lição abrange as subconsultas de uma única linha e de várias linhas.

Usando uma Subconsulta para Resolver um Problema

"Quem tem um salário maior que o de Jones?"

Consulta principal

"Que funcionários têm um salário maior que o salário de Jones?"

Subconsulta

"Qual é o salário de Jones?"

Usando uma Subconsulta para Resolver um Problema

Suponha que você deseje criar uma consulta para descobrir quem recebe um salário maior que o de Jones.

Para resolver esse problema, são necessárias duas consultas: uma consulta para descobrir quanto Jones recebe e outra para descobrir quem recebe mais que essa quantia.

Você pode resolver esse problema combinando as duas consultas, colocando uma consulta dentro da outra consulta.

A consulta interna ou a subconsulta retorna um valor que é usado pela consulta externa ou pela consulta principal. Usar uma subconsulta equivale a executar duas consultas seqüenciais e usar o resultado da primeira como o valor de pesquisa na segunda consulta.

Subconsultas

```
SELECT      select_list
FROM        tabela
WHERE       operador expr
                  (SELECT     select_list
```

```
                    FROM  tabela);
```

• A subconsulta (consulta interna) é executada uma vez antes da consulta principal.

• O resultado da subconsulta é usado pela consulta principal (consulta externa).

Subconsultas

Uma subconsulta é uma instrução SELECT que é incorporada em uma cláusula de outra instrução SELECT. Você pode desenvolver instruções sofisticadas a partir de instruções simples usando subconsultas. Elas podem ser muito úteis quando você precisar selecionar linhas de uma tabela com uma condição que dependa dos dados na própria tabela.

É possível colocar a subconsulta em várias cláusulas SQL:

- cláusula WHERE
- cláusula HAVING
- cláusula FROM Na sintaxe:

operador inclui um operador de comparação tal como >, = ou IN

Observação: Os operadores de comparação subdividem-se em duas classes: operadores de uma única linha (>, =, >=, <, <>, <=) e operadores de várias linhas (IN, ANY, ALL).

A subconsulta geralmente é chamada de instrução SELECT, sub-SELECT ou SELECT interna aninhada. A subconsulta normalmente é executada primeiro e sua saída é usada para concluir a condição de consulta da consulta principal ou externa.

Usando uma Subconsulta

```
SQL> SELECT NOME
2     FROM  EMPREGADO   2975
3     WHERE sal >
4           (SELECT sal
5           FROM  EMPREGADO
6           WHERE COD_EMP=7566);

NOME
----------
KING
FORD
```

SCOTT

Usando uma Subconsulta

No slide, a consulta interna determina o salário do funcionário 7566. A consulta externa obtém o resultado da consulta interna e o utiliza para exibir todos os funcionários que recebem mais que essa quantia.

Diretrizes para o Uso de Subconsultas

• Coloque as subconsultas entre parênteses.

• Coloque as subconsultas no lado direito do operador de comparação.

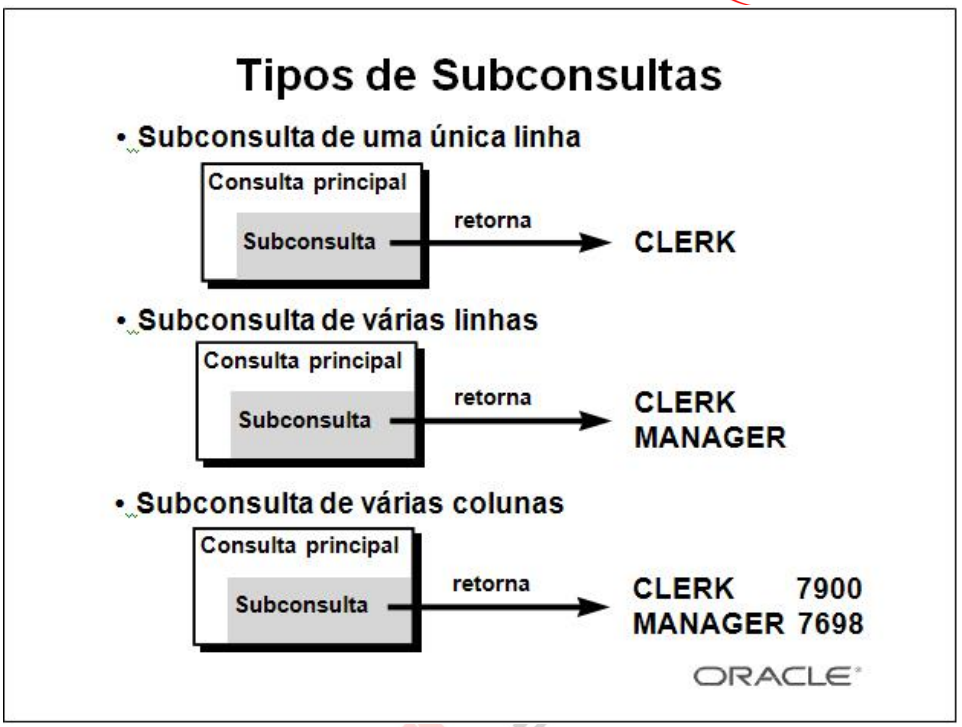
• Não adicione uma cláusula ORDER BY a uma subconsulta.

• Use operadores de uma única linha com subconsultas de uma única linha.

• Use operadores de várias linhas com subconsultas de várias linhas.

Diretrizes para o Uso de Subconsultas

•Uma subconsulta deve estar entre parênteses.
•Uma subconsulta deve aparecer do lado direito do operador de comparação.
•As subconsultas não podem conter uma cláusula ORDER BY. Só é possível haver uma cláusula ORDER BY para uma instrução SELECT e, se estiver especificado, ela deve ser a última cláusula na instrução SELECT principal.
•São usadas duas classes de operadores de comparação nas subconsultas: operadores de uma única linha e operadores de várias linhas.



Tipos de Subconsultas

•Subconsultas de uma única linha: consultas que retornam somente uma linha da instrução

SELECT interna

•Subconsultas de várias linhas: consultas que retornam mais de uma linha da instrução

SELECT interna

•Subconsultas de várias colunas: consultas que retornam mais de uma coluna da instrução

SELECT interna

Subconsultas de uma Única Linha

- Retorne somente uma linha
- Use operadores de comparação de uma única linha

Operador	Significado
=	Igual a
>	Maior do que
>=	Maior do que ou igual a
<	Menor do que
<=	Menor ou igual a
<>	Diferente de

Subconsultas de uma Única Linha

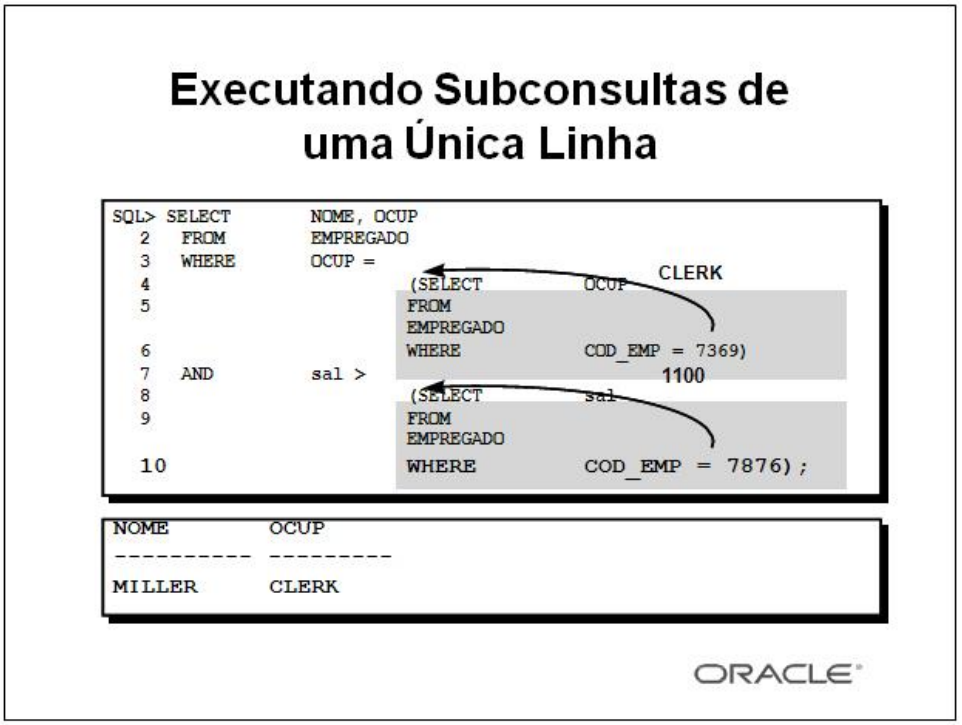
A subconsulta de uma única linha retorna uma linha da instrução SELECT interna. Esse tipo de subconsulta usa um operador de uma única linha. O slide fornece uma lista de operadores de uma única linha.

Exemplo

Exiba os funcionários cujo cargo seja o mesmo do funcionário 7369.

```
SQL> SELECT NOME, OCUP
  2   FROM  EMPREGADO
  3   WHERE OCUP =

  4               (SELECT     OCUP
  5               FROM  EMPREGADO
  6               WHERE COD_EMP =   7369);


NOME        OCUP
---------- ---------
```

```
JAMES        CLERK
SMITH        CLERK
ADAMS        CLERK
MILLER       CLERK
```



Executando Subconsultas de uma Única Linha

A instrução SELECT pode ser considerada um bloco de consulta. O exemplo no slide exibe funcionários cujo cargo é o mesmo que o do funcionário 7369 e cujo salário é maior que o do funcionário 7876.

O exemplo consiste em três blocos de consulta: a consulta externa e duas consultas internas. Os blocos de consulta interna são executados primeiro, produzindo os resultados da consulta: CLERK e 1100, respectivamente. O bloco de consulta externa é então processado e usa os valores retornados pelas consultas internas para completar suas condições de pesquisa.

Ambas as consultas internas retornam valores únicos (CLERK e 1100, respectivamente). Portanto, essa instrução SQL é chamada de subconsulta de uma única linha.

Observação: As consultas externa e interna podem obter dados de tabelas diferentes.

Usando Funções de Grupo em uma Subconsulta

```
SQL> SELECT NOME, OCUP, sal
  2   FROM  EMPREGADO   800
  3   WHERE sal =
  4               (SELECT     MIN(sal)
  5               FROM  EMPREGADO);


NOME        OCUP        SAL
---------- --------- ---------
SMITH       CLERK       800
```

Usando Funções de Grupo em uma Subconsulta

É possível exibir dados de uma consulta principal usando uma função de grupo em uma subconsulta para retornar uma única linha. A subconsulta fica entre parênteses e é colocada após o operador de comparação.

O exemplo no slide exibe o nome do funcionário, o cargo do funcionário e o salário de todos os funcionários cujo salário seja igual ao salário mínimo. A função de grupo MIN retorna um único valor (800) para a consulta externa.

Cláusula HAVING com Subconsultas

• O Oracle Server primeiro executa as subconsultas.

• O Oracle Server retorna os resultados para
a cláusula HAVING da consulta principal.

```
SQL> SELECT COD_DEPTO, MIN(sal)
  2 FROM    EMPREGADO
  3 GROUP BY      COD_DEPTO
  4 HAVING  MIN(sal) >  800
  5         (SELECT     MIN(sal)
  6         FROM  EMPREGADO
  7         WHERE COD_DEPTO = 20);
```

Cláusula HAVING com Subconsultas

Você pode usar as subconsultas não só na cláusula WHERE, mas também na cláusula HAVING.

O Oracle Server executa a subconsulta e os resultados são retornados para a cláusula HAVING da consulta principal.

A instrução SQL no slide exibe todos os departamentos que tenham um salário mínimo maior que o do departamento 20.

```
COD_DEPTO   MIN(SAL)
--------- ---------
10          1300
30          950
```

Exemplo

Localize o cargo com o menor salário médio.

```
SQL> SELECT OCUP, AVG(sal)
  2   FROM  EMPREGADO
  3   GROUP BY    OCUP
  4   HAVING      AVG(sal) = (SELECT      MIN(AVG(sal))
  5   FROM  EMP
  6   GROUP BY    OCUP);
```

O que Há de Errado com esta Instrução?

```
SQL> SELECT COD_EMP, NOME
  2   FROM  EMPREGADO
  3   WHERE sal =
  4         (SELECT     MIN(sal)
  5         FROM  EMPREGADO
  6         GROUP BY    COD_DEPTO);


ERROR:
ORA-01427: A subconsulta de uma única linha retorna mais de
uma linha (Single-row subquery returns more than one row)

no rows selected
```

Erros em Subconsultas

Um erro comum em subconsultas é o retorno de mais de uma linha para uma subconsulta de uma única linha.

Na instrução SQL do slide, a subconsulta contém uma cláusula GROUP BY (COD_DEPTO), o que indica que a subconsulta retornará várias linhas, uma para cada grupo que localizar. Nesse caso, o resultado da subconsulta será 800, 1300 e 950.

A consulta externa obtém os resultados da subconsulta (800, 950, 1300) e os utiliza na sua cláusula WHERE. A cláusula WHERE contém um operador igual (=), um operador de comparação de uma única linha que espera somente um valor. O operador = não pode aceitar mais de um valor da subconsulta e, portanto, gera o erro.

Para corrigir esse erro, altere o operador = para IN

Esta Instrução Irá Funcionar?

```
SQL> SELECT NOME, OCUP
2     FROM  EMPREGADO
3     WHERE OCUP =
4     (SELECT OCUP
5     FROM  EMPREGADO
6     WHERE NOME='SMYTHE');


no rows selected
```

Problemas nas Subconsultas

Um problema comum nas subconsultas é nenhuma linha ser retornada pela consulta interna.

Na instrução SQL do slide, a subconsulta contém uma cláusula WHERE (NOME='SMYTHE'). Supostamente, a intenção é localizar o funcionário cujo nome seja Smythe. A instrução parece ser correta, mas não seleciona linhas ao ser executada.

O problema é que Smythe não foi escrito corretamente. Não há nenhum funcionário chamado Smythe. Dessa forma, a subconsulta não retorna nenhuma linha. A consulta externa obtém os resultados da subconsulta (nula) e os utiliza na sua cláusula WHERE. A consulta externa não localiza nenhum funcionário com um cargo igual a nulo e, portanto, não retorna nenhuma linha.

Subconsultas de Várias Linhas

- Retorne mais de uma linha
- Use operadores de comparação de várias linhas

Operador	Significado
IN	Igual a qualquer membro na lista
ANY	Compare o valor a cada valor retornado pela subconsulta
ALL	Compare o valor a todo valor retornado pela subconsulta

Subconsultas de Várias Linhas

As subconsultas que retornam mais de uma linha chamam-se subconsultas de várias linhas. Você pode usar um operador de várias linhas, em vez de um operador de uma única linha, com uma subconsulta de várias linhas. O operador de várias linhas espera um ou mais valores.

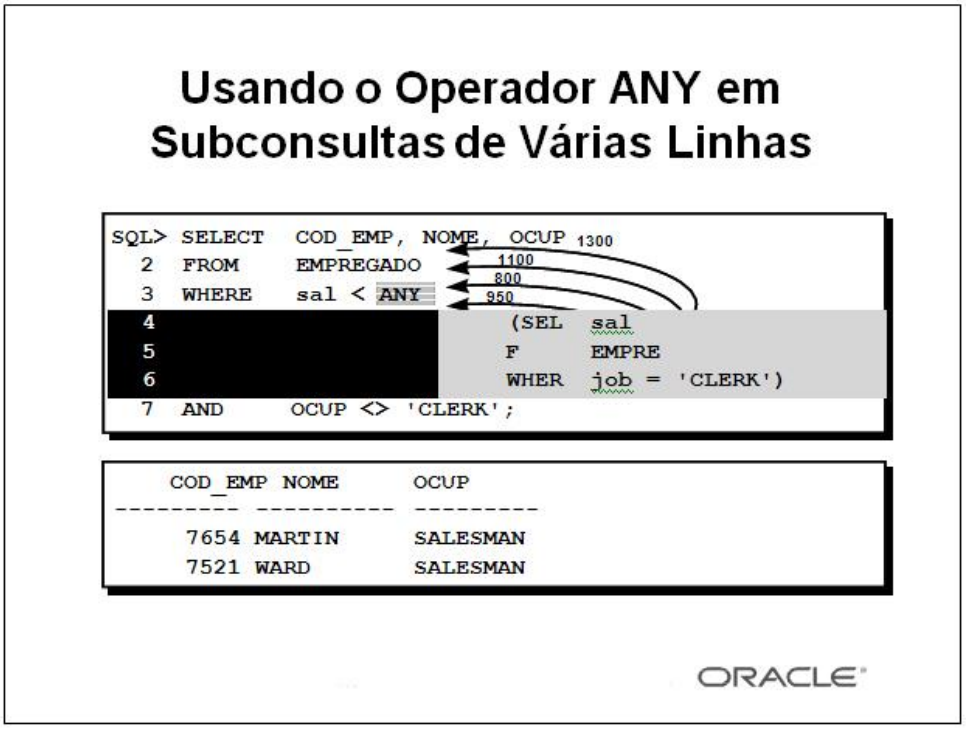
```
SQL> SELECT NOME, sal, COD_DEPTO
  2   FROM  EMPREGADO
  3   WHERE sal IN      (SELECT     MIN(sal)
  4   FROM  EMPREGADO
  5   GROUP BY    COD_DEPTO);
```

Exemplo

Localize os funcionários que recebam o mesmo salário que o salário mínimo dos departamentos.

A consulta interna é executada primeiro, produzindo um resultado de consulta que contenha três linhas: 800, 950, 1300. O bloco da consulta principal é processado em seguida e usa os valores retornados pela consulta interna para completar sua condição de pesquisa. Na verdade, a consulta principal pareceria da seguinte forma para o Oracle Server:

```
SQL> SELECT NOME, sal, COD_DEPTO
  2   FROM  EMPREGADO
  3   WHERE sal IN      (800, 950,  1300);
```

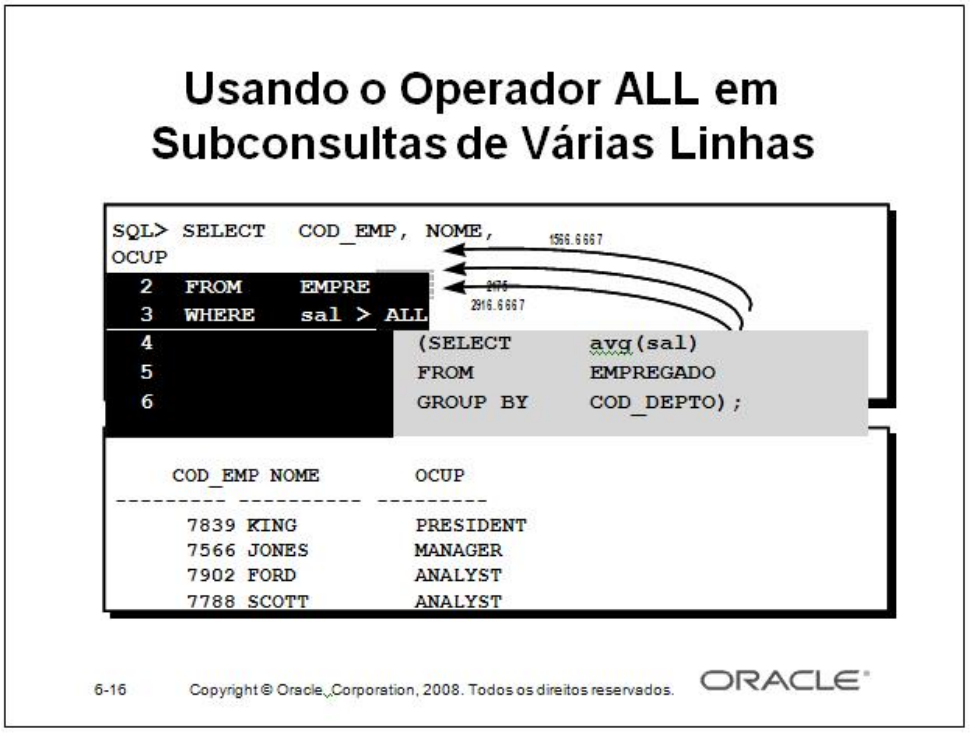


Subconsultas de Várias Linhas (continuação)

O operador ANY (e o operador sinônimo SOME) compara um valor a cada valor retornado por uma subconsulta. O exemplo no slide exibe funcionários cujo salário é menor que o de qualquer escriturário e que não são escriturários. O salário máximo que um escriturário recebe é US$ 1.300.

A instrução SQL exibe todos os funcionários que não são escriturários, mas recebem menos que US$ 1.300.

<ANY significa menos do que o máximo. >ANY significa mais do que o mínimo. =ANY equivale a IN.



Subconsultas de Várias Linhas (continuação)

O operador ALL compara um valor a todo valor retornado por uma subconsulta. O exemplo no slide exibe funcionários cujo salário seja maior que os salários médios de todos os departamentos. O salário médio mais alto de um departamento é US$ 2.916,66, portanto a consulta retorna os funcionários cujo salário seja maior que US$ 2.916,66.

>ALL significa mais do que o máximo e <ALL significa menos do que o mínimo.

O operador NOT pode ser usado com os operadores IN, ANY e ALL.

Sumário

As subconsultas são úteis quando uma consulta baseia-se em valores desconhecidos.

```
SELECT      select_list
FROM        tabela
WHERE       operador expr
                  (SELECT select_list
                  FROM  tabela);
```

Sumário

Uma subconsulta é uma instrução SELECT que é incorporada em uma cláusula de outra instrução SQL. As subconsultas são úteis quando uma consulta baseia-se em critérios de seleção com valores intermediários desconhecidos.

As subconsultas têm as seguintes características:

•Podem passar uma linha de dados para uma instrução principal que contenha um operador de uma única linha, tal como =, <>, >, >=, < ou <=

•Podem passar várias linhas de dados para uma instrução principal que contenha um operador de várias linhas, tal como IN

•São processadas primeiro pelo Oracle Server, e a cláusula WHERE ou HAVING usa os resultados

•Podem conter funções de grupo

Subconsultas de Várias Colunas

Objetivos

Depois de completar esta lição, você poderá fazer o seguinte:

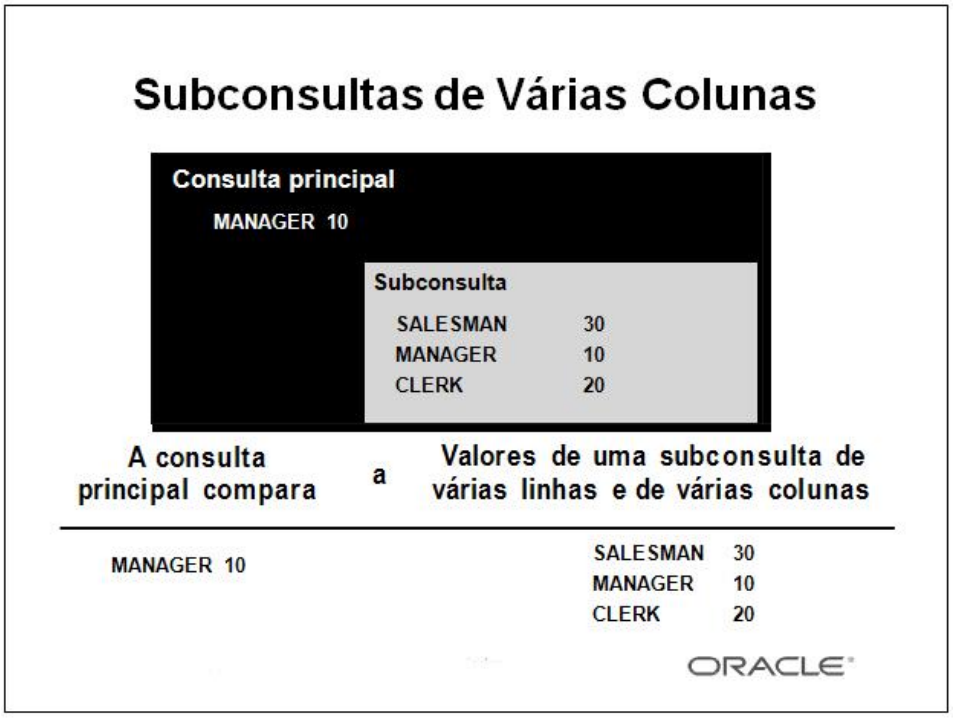
• Criar uma subconsulta de várias colunas

• Descrever e explicar o comportamento de subconsultas quando valores nulos forem recuperados

• Criar uma subconsulta em uma cláusula FROM

Objetivo da Lição

Nesta lição, você aprenderá a criar subconsultas de várias colunas e subconsultas na cláusula FROM de uma instrução SELECT.



Subconsultas de Várias Colunas

Até este momento você tem criado subconsultas de uma única linha e subconsultas de várias linhas, onde somente uma coluna é comparada na cláusula WHERE ou na cláusula HAVING da instrução SELECT. Para comparar duas ou mais colunas, você deverá criar uma cláusula WHERE composta usando operadores lógicos. As subconsultas de várias colunas permitem que você combine condições WHERE duplicadas em uma única cláusula WHERE.

Sintaxe

```
SELECT      coluna, coluna,   ...
FROM  tabela

WHERE (coluna, coluna, ...) IN
(SELECT coluna, coluna, ... FROM    tabela
WHERE condição);
```

Usando Subconsultas deVárias Colunas

Exiba a ID da ordem, a ID do produto e a quantidade de itens na tabela de itens que corresponde à ID do produto e à quantidade de um item na ordem 605.

```
SQL> SELECT ordid, prodid, qty
  2   FROM  item
  3   WHERE (prodid, qty) IN
  4   (SELECT prodid, qty
  5   FROM  item
  6   WHERE ordid = 605)
  7   AND   ordid <> 605;
```

Usando Subconsultas de Várias Colunas

O exemplo no slide é o de uma subconsulta de várias colunas porque a subconsulta retorna mais de uma coluna. Ele compara os valores na coluna PRODID e na coluna QTY de cada linha do candidato na tabela ITEM aos valores na coluna PRODID e na coluna QTY dos itens na ordem 605.

Primeiro, execute a subconsulta para ver os valores PRODID e QTY de cada item na ordem 605.

```
SQL> SELECT prodid, qty
  2   FROM  item
  3   WHERE ordid =    605;


PRODID     QTY
---------- ----------
100861     100
100870     500
100890     5
101860     50
101863     100
102130     10

6 rows selected.
```

Usando Subconsultas de Várias Colunas

Exiba o número da ordem, o número do produto e a quantidade de qualquer item em que o número do produto e a quantidade correspondam ao número do produto e à quantidade de um item na ordem 605.

```
SQL> SELECT ordid, prodid, qty
  2   FROM  item
  3   WHERE (prodid, qty) IN
  4   (SELECT prodid, qty
  5   FROM  item
  6   WHERE ordid = 605)
  7   AND   ordid <> 605;
```

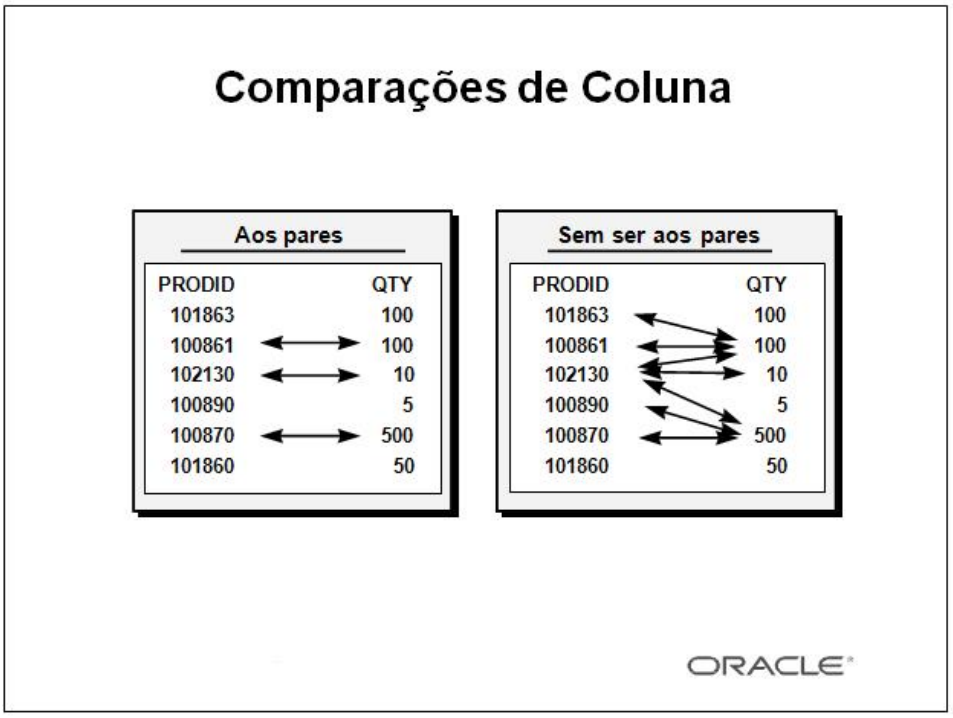
Usando Subconsultas de Várias Colunas (continuação)

Quando a instrução SQL no slide é executada, o servidor Oracle compara os valores nas colunas PRODID e QTY e retorna as ordens em que o número e a quantidade do produto correspondem ao número do produto e à quantidade desse item na ordem 605.
A saída da instrução SQL é:

```
ORDID PRODID      QTY
----- ---------   ---------
617   100861      100
617   100870      500
616   102130      10
```

A saída mostra que há três itens em outras ordens que contêm o mesmo número do produto e a mesma quantidade que um item na ordem 605. Por exemplo, a ordem 617 solicitou uma quantidade

500 do produto 100870. A ordem 605 também solicitou uma quantidade 500 do produto 100870. Portanto, essas linhas do candidato são parte da saída.



Comparações aos Pares Versus Comparações que Não Sejam aos Pares

As comparações de coluna em consultas de várias colunas podem ser comparações aos pares ou não.

O slide mostra os números do produto e as quantidades dos itens na ordem 605.

No exemplo do slide anterior, uma comparação aos pares foi executada na cláusula WHERE. Cada linha do candidato na instrução SELECT deve ter o mesmo número do produto e a mesma quantidade que um item na ordem 605. Isso é ilustrado no lado esquerdo do slide acima. As setas indicam que o número do produto e a quantidade em uma linha do candidato correspondem a um número do produto e a uma quantidade de um item na ordem 605.

Uma subconsulta de várias colunas também pode ser uma comparação que não seja aos pares. Se você desejar uma comparação que não seja aos pares (um produto cruzado), você deverá usar uma cláusula WHERE com várias condições. Uma linha do candidato deve corresponder às várias condições na cláusula WHERE, mas os valores são comparados individualmente. Uma linha do candidato deve corresponder a algum número do produto na ordem

605, assim como a alguma quantidade na ordem 605, mas esses valores não precisam estar na mesma linha. Isso é ilustrado no lado direito do slide. Por exemplo, o produto 102130 aparece em outras ordens, uma ordem correspondendo à quantidade na ordem 605 (10) e outra ordem tendo uma quantidade de 500. As setas exibem uma amostra das várias quantidades solicitadas para determinado produto.

Subconsulta de Comparação que Não Seja aos Pares

Exiba o número da ordem, o número do produto e a quantidade de qualquer item em que o número do produto e a quantidade correspondam a qualquer número do produto e quantidade de um item na ordem 605.

```
SQL> SELECT ordid, prodid, qty
2     FROM  item
3     WHERE prodid      IN    (SELECT    prodid
4                       FROM  item
5                       WHERE ordid =    605)
6     AND   qty   IN    (SELECT    qty
7                       FROM  item
8                       WHERE ordid =    605)
9     AND   ordid <> 605;
```

Subconsulta de Comparação que Não Seja aos Pares

O exemplo no slide faz uma comparação das colunas sem ser aos pares. Ele exibe o número da ordem, o número do produto e a quantidade de qualquer item em que o número do produto e a quantidade correspondam a qualquer número do produto e quantidade de um item na ordem 605. A ordem 605 não é incluída na saída.

Subconsulta que Não Seja aos Pares

```
ORDID PRODID      QTY
----- --------- ---------

609   100870      5
616   100861      10
616   102130      10
621   100861      10
```

```
618    100870       10
618    100861       50
616    100870       50
617    100861       100
619    102130       100
615    100870       100
617    101860       100
621    100870       100
617    102130       100
. . .
16 rows selected.
```

Subconsulta que Não Seja aos Pares

Os resultados da subconsulta sem ser aos pares são mostrados no slide. Dezesseis linhas do candidato na tabela ITEM correspondem às várias condições na cláusula WHERE.

Por exemplo, um item da ordem 621 é retornado da instrução SQL. Um produto na ordem 621(número do produto 100861) corresponde a um produto de um item na ordem 605. A quantidade do produto 100861 na ordem 621 (10) corresponde à quantidade em outro item na ordem 605(a quantidade do produto 102130).

Valores Nulos em uma Subconsulta

```
SQL> SELECT FUNCIONARIO.NOME
2      FROM  EMPREGADO FUNCIONARIO
3      WHERE FUNCIONARIO.COD_EMP NOT IN
4      (SELECT manager.mgr
5      FROM  EMPREGADO manager);
no rows selected.
```

Retornando Nulos no Conjunto Resultante de uma Subconsulta

A instrução SQL no slide tenta exibir todos os funcionários que não tenham nenhum subordinado. Logicamente, essa instrução SQL deveria ter retornado oito linhas. Entretanto, a instrução SQL não retorna nenhuma linha. Um dos valores retornados pela consulta interna é um valor nulo e, portanto, a consulta inteira não retorna nenhuma linha. O motivo é que todas as condições que comparam um valor nulo resultam em um nulo. Dessa forma, sempre que houver a possibilidade de valores nulos serem parte do conjunto resultante de uma subconsulta, o operador NOT IN não será usado. O operador NOT IN é equivalente a !=ALL.

Observe que o valor nulo como parte do conjunto resultante de uma subconsulta não será um problema se você estiver usando o operador IN. O operador IN é equivalente a =ANY. Por exemplo, para exibir os funcionários que têm subordinados, use a seguinte instrução SQL:

```
SQL> SELECT FUNCIONARIO.NOME
2     FROM  EMPREGADO FUNCIONARIO
3     WHERE FUNCIONARIO.COD_EMP IN (SELECT manager.mgr
4                 FROM  EMPREGADO manager);


NOME
----------
KING
...
6 rows selected.
```

Usando uma Subconsulta na Cláusula FROM

```
SQL> SELECT a.NOME, a.sal, a.COD_DEPTO, b.salavg
2     FROM  EMPREGADO a, (SELECT   COD_DEPTO, avg(sal)
salavg
3     FROM  EMPREGADO
4     GROUP BY COD_DEPTO) b
5     WHERE a.COD_DEPTO = b.COD_DEPTO
6     AND   a.sal > b.salavg;


NOME  SAL   COD_DEPTO   SALAVG
---------- --------- --------- ----------
KING  5000  10    2916.6667
JONES 2975  20    2175
SCOTT 3000  20    2175
...
6 rows selected.
```

Usando uma Subconsulta na Cláusula FROM

Você pode usar uma subconsulta na cláusula FROM de uma instrução SELECT, o que se parece muito com a forma de utilizar as views. Uma subconsulta na cláusula FROM de uma instrução SELECT define uma origem de dados para essa, e somente essa, instrução SELECT em particular. O exemplo no slide exibe nomes de funcionários, salários, números de departamento e salários médios de

todos os funcionários que recebem mais que o salário médio nos seus departamentos.

Sumário

• Uma subconsulta de várias colunas retorna mais de uma coluna.

• As comparações de coluna em comparações de várias colunas podem ser aos pares ou não.

• Uma subconsulta de várias colunas também pode ser usada na cláusula FROM de uma instrução SELECT.

Sumário

As subconsultas de várias colunas permitem que você combine condições WHERE duplicadas em uma única cláusula WHERE. As comparações de coluna em consultas de várias colunas podem ser comparações aos pares ou não. Você pode usar uma subconsulta para definir uma tabela que seja operada por uma consulta contida. Para isso, coloque a subconsulta na cláusula FROM da consulta contida, como se fosse um nome de tabela.

Capítulo 08

Manipulação de Dados

Objetivos

Depois de completar esta lição, você poderá fazer o seguinte:

- Descrever cada instrução DML
- Inserir linhas em uma tabela
- Atualizar linhas em uma tabela
- Deletar linhas de uma tabela
- Controlar transações

Objetivo da Lição

Nesta lição, você aprenderá como inserir linhas, atualizar e deletar linhas existentes em uma tabela. Você também aprenderá como controlar transações com as instruções COMMIT, SAVEPOINT e ROLLBACK.

DML (Data Manipulation Language)

• Uma instrução DML é executada quando você:

– Adiciona novas linhas a uma tabela

– Modifica linhas existentes em uma tabela

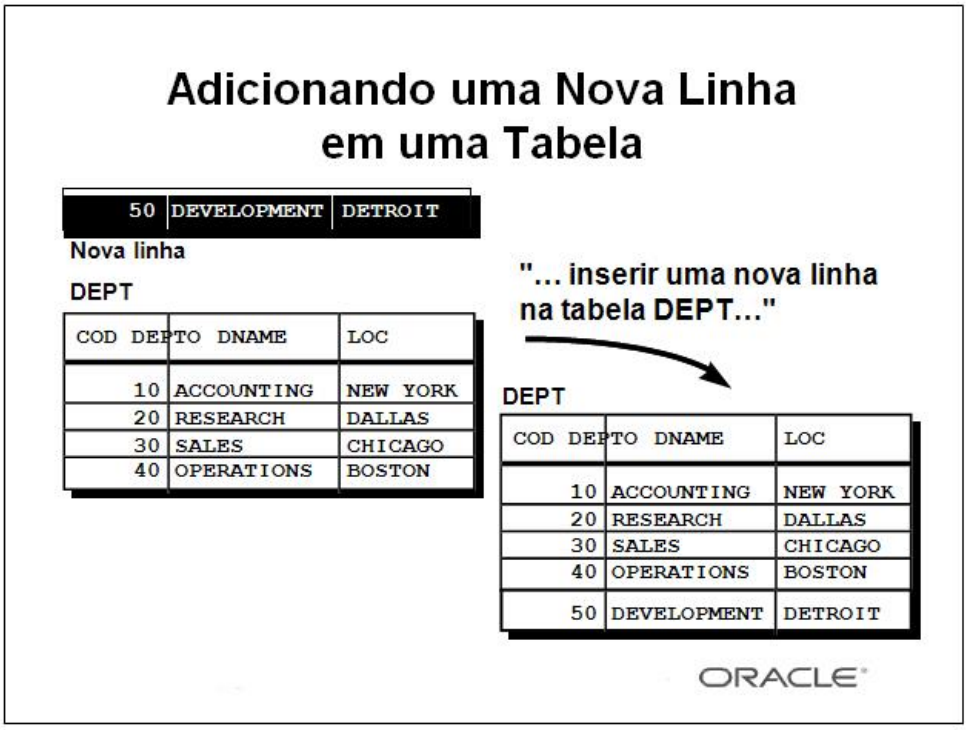
– Remove linhas existentes de uma tabela

• Uma transação consiste em um conjunto de instruções DML que formam uma unidade lógica de trabalho.

DML (Data Manipulation Language)

A DML (Data Manipulation Language) é uma parte essencial do SQL. Quando você quiser adicionar, atualizar ou deletar dados no banco de dados, execute uma instrução DML. Um conjunto de instruções DML que formam uma unidade lógica de trabalho é chamada de transação.

Considere um banco de dados bancário. Quando o cliente de um banco transfere dinheiro de uma conta de poupança para uma conta corrente, a transação consiste em três operações separadas: diminuir a conta de poupança, aumentar a conta corrente e registrar a transação no lançamento da transação. O Oracle Server deve garantir que todas as três instruções SQL sejam executadas para manter as contas com um saldo apropriado. Quando algo impedir que uma das instruções na transação seja executada, as outras instruções da transação deverão ser desfeitas.



Adicionando uma Nova Linha em uma Tabela

O gráfico do slide adiciona um novo departamento à tabela DEPT.

A Instrução INSERT

• Adicione novas linhas em uma tabela usando a instrução INSERT.

```
INSERT INTO tabela [(coluna [, coluna...])]
VALUES      (valor [, valor...]);
```

• Somente uma linha é inserida por vez com esta sintaxe.

Adicionando uma nova linha em uma Tabela (continuação)

Você pode adicionar novas linhas à tabela emitindo a instrução INSERT.

Na sintaxe:

tabela	é o nome da tabela is the name of the table
coluna	é o nome da coluna a ser preenchida

valor é o valor correspondente para a coluna

Observação: Esta instrução com a cláusula VALUES adiciona somente uma linha por vez a uma tabela.

Inserindo Novas Linhas

• Insira uma nova linha contendo valores para cada coluna.

• Liste valores na ordem default das colunas na tabela.

• Liste opcionalmente as colunas na cláusula INSERT.

```
SQL> INSERT INTO  dept (COD_DEPTO, dname, loc)
  2   VALUES      (50, 'DEVELOPMENT', 'DETROIT');
  1 row created.
```

• Coloque os valores de data e caractere entre aspas simples.

Adicionando uma nova linha em uma Tabela (continuação)

Como você pode inserir uma nova linha que contenha valores para cada coluna, a lista de colunas não é requerida na cláusula INSERT. Entretanto, se você não usar a lista de colunas, os valores deverão ser listados de acordo com a ordem default das colunas na tabela.

```
SQL> DESCRIBE     dept

Name                              Null?       Type
--------------------------------- --------- ------------
COD_DEPTO                         NOT NULL    NUMBER(2)
DNAME                                         VARCHAR2(14)
LOC                                           VARCHAR2(13)
```

Para clareza, use a lista de colunas na cláusula INSERT.
Coloque os valores de dados e caracteres entre aspas simples, não coloque valores numéricos entre aspas simples.

Inserindo Linhas com Valores Nulos

• Método implícito: Omita a coluna da lista de colunas.

```
SQL> INSERT INTO  dept (COD_DEPTO, dname )
  2   VALUES      (60, 'MIS');
  1  row created.
```

• Método explícito: Especifique a palavra- chave NULL.

```
SQL> INSERT INTO  dept
  2   VALUES      (70, 'FINANCE', NULL);
  1  row created.
```

Métodos para Inserir Valores Nulos

Método	Descrição
Implícito	Omite a coluna da lista de colunas.
Explícito	Especifica a palavra-chave NULL na lista VALUES. Especifica a string vazia ('') na lista VALUES; para strings de caractere e datas somente.

Certifique-se de que a coluna de destino permita valores nulos verificando o status Null? a partir do comando DESCRIBE do SQL*Plus.

O Oracle Server impõe automaticamente todas as restrições de integridade de dados, tipos de dados e faixas de dados. Qualquer coluna que não esteja listada explicitamente obtém um valor nulo na nova linha.

Inserindo Valores Especiais

A função SYSDATE registra a data e hora atuais.

```
SQL> INSERT INTO  EMPREGADO (COD_EMP, NOME, OCUP,
  2               mgr, hiredate, sal, comm,
  3               COD_DEPTO)
  4   VALUES      (7196, 'GREEN', 'SALESMAN',
  5               7782, SYSDATE, 2000, NULL,
  6               10);
1 row created.
```

Inserindo Valores Especiais Usando Funções SQL

Você pode usar pseudocolunas para inserir valores especiais em sua tabela.

O exemplo do slide registra informações para o funcionário Green na tabela EMP. Ele fornece a data e hora atuais na coluna HIREDATE. Ele usa a função SYSDATE para data e hora atuais.

Você também poderá usar a função USER ao inserir linhas em uma tabela. A função USER registra o nome de usuário atual.

Confirmando Adições à Tabela

```
SQL> SELECT COD_EMP, NOME, OCUP, hiredate, comm
  2   FROM  EMPREGADO
  3   WHERE COD_EMP =   7196;


COD_EMP     NOME  OCUP  HIREDATE    COMM

---------   ---------- --------- --------- ---------
7196        GREEN SALESMAN    01-DEC-97
```

Inserindo Valores Específicos de Data

• Adicionar um novo funcionário.

```
SQL> INSERT INTO EMPREGADO
  2   VALUES      (2296,'AROMANO','SALESMAN',7782,
  3         TO_DATE('FEB 3, 1997', 'MON DD, YYYY'),

  4   1300, NULL, 10);
  1 row created.
```

• Verifique sua adição.

```
COD_EMP NOME      OCUP  MGR   HIREDATE    SAL COMM
COD_DEPTO
----- ------- -------- ---- --------- ---- ---- ------
2296  AROMANO SALESMAN 7782 03-FEB-97 1300      10
```

Inserindo Valores Específicos de Data e Hora

O formato DD-MON-YY é geralmente usado para inserir um valor de data. Com este formato, lembre-se que o século atual é o default. Como a data também contém informações sobre a hora, a hora default é meia-noite (00:00:00).

Se uma data tiver que ser informada em um formato diferente do default — por exemplo, outro século e/ou uma hora específica — você deve usar a função TO_DATE.

O exemplo no slide registra informações para o funcionário Aromano na tabela EMP. Ele define a coluna HIREDATE como 3 de fevereiro de 1997.

Se o formato RR estiver definido, o século pode não ser o atual.

Inserindo Valores Usando Variáveis de Substituição

Crie um script interativo usando parâmetros de substituição do SQL*Plus.

```
SQL> INSERT INTO  dept (COD_DEPTO, dname, loc)
  2   VALUES      (&DEPARTAMENTO_id,
  3         '&DEPARTAMENTO_name', '&location');


Enter value for DEPARTAMENTO_id: 80
Enter value for DEPARTAMENTO_name: EDUCATION Enter value
for location: ATLANTA

1 row created.
```

Inserindo Valores Usando Variáveis de Substituição

Você pode produzir uma instrução INSERT que permite que o usuário adicione valores interativamente usando as variáveis de substituição do SQL*Plus.

O exemplo do slide registra informações para um departamento na tabela DEPT. Ele solicita ao usuário o número do departamento, nome do departamento e localização.

Para valores de data e caractere, o "e" comercial e o nome da variável aparecem entre aspas simples.

Criando um Script com Prompts Personalizados

• ACCEPT armazena o valor em uma variável.

• PROMPT exibe o texto personalizado.

```
ACCEPT      DEPARTAMENTO_id PROMPT 'Please enter the
            - DEPARTAMENTO number:'
ACCEPT      DEPARTAMENTO_name PROMPT 'Please enter
            - the DEPARTAMENTO name:'
ACCEPT      location PROMPT 'Please enter the
            - location:'
INSERT INTO dept (COD_DEPTO, dname, loc)
VALUES      (&DEPARTAMENTO_id, '&DEPARTAMENTO_name',
            '&location');
```

Criando um Script para Manipular Dados

Você pode salvar o seu comando com variáveis de substituição em um arquivo e executá-lo. Cada vez que você executar o comando, ele irá solicitar novos valores. Personalize os prompts usando o comando ACCEPT do SQL*Plus.

O exemplo no slide registra informações para um departamento na tabela DEPT. Ele solicita ao usuário o número do departamento, o nome do departamento e a localização usando mensagens de prompt personalizadas.

```
Please enter the DEPARTAMENTO number: 90
Please enter the DEPARTAMENTO name: PAYROLL
Please enter the location: HOUSTON

1 row created.
```

Não preceda o parâmetro de substituição do SQL*Plus com o "e" comercial (&) ao fazer referência a ele no comando ACCEPT. Use um traço (-) para continuar com um comando do SQL*Plus na próxima linha.

Copiando Linhas a partir de Outra Tabela

• Crie a instrução INSERT com uma subconsulta.

```
SQL> INSERT INTO managers(id, name, salary, hiredate)
```

```
2   SELECT COD_EMP, NOME, sal, hiredate
3   FROM  EMPREGADO
4   WHERE OCUP = 'MANAGER';
3 rows created.
```

• Não use a cláusula VALUES.

• Faça a correspondência do número de colunas na cláusula INSERT com o número de colunas na subconsulta.

Copiando Linhas a partir de Outra Tabela

Você pode usar a instrução INSERT para adicionar linhas a uma tabela onde os valores são derivados de tabelas existentes. No lugar da cláusula VALUES, use uma subconsulta.

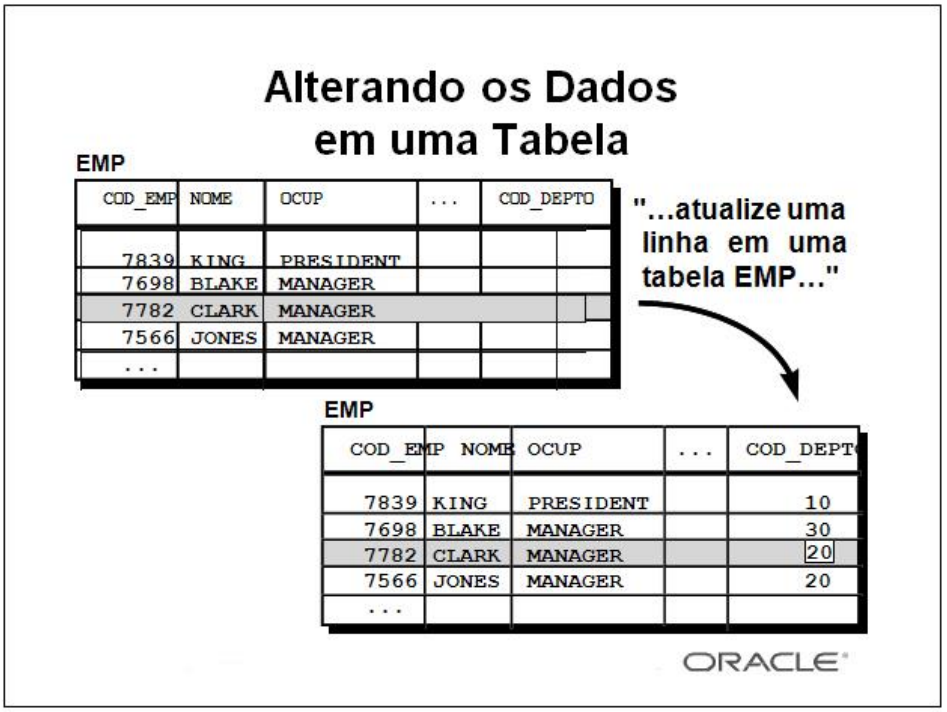
Sintaxe

```
INSERT INTO tabela [ coluna (, coluna) ]
subconsulta;
```

onde: *tabela* é o nome da tabela
coluna é o nome da coluna a ser preenchida
subconsulta é a subconsulta que retorna linhas na tabela

Para obter mais informações, consulte o Oracle Server SQL Reference, Release 8, "SELECT". seção
Subqueries.

O número de colunas e seus tipos de dados na lista de colunas da cláusula INSERT devem coincidir com o número de valores e seus tipos de dados na subconsulta.



Alterando os Dados em uma Tabela

O gráfico do slide altera o número do departamento de Clark de 10 para 20.

A instrução UPDATE

• Modifique linhas existentes com a instrução UPDATE.

```
UPDATE      tabela
SET         coluna = valor [, coluna = valor, ...]
[WHERE      condição];
```

• Atualize mais de uma linha por vez, se necessário.

Atualizando Linhas

Você pode modificar linhas existentes usando a instrução UPDATE. Na sintaxe acima:

tabela	é o nome da tabela
coluna	é o nome da coluna a ser preenchida
valor	é o valor correspondente ou subconsulta para a coluna
condição	identifica as linhas a serem atualizadas e é composto de nomes de colunas expressões, constantes, subconsultas e operadores de comparação

Confirme a operação de atualização consultando a tabela para exibir as linhas atualizadas.

Para obter mais informações, consulte o Oracle Server SQL Reference, Release 8, "UPDATE".

Observação: Em geral, use a chave primária para identificar uma única linha. Várias linhas poderão ser atualizadas inesperadamente se outras colunas forem usadas. Por exemplo, identificar uma única linha na tabela EMP por nome é perigoso porque mais de um funcionário pode ter o mesmo nome.

Atualizando Linhas em uma Tabela

• Uma linha ou linhas específicas são modificadas quando você especifica a cláusula WHERE.

```
SQL> UPDATE EMPREGADO
  2   SET   COD_DEPTO = 20
  3   WHERE COD_EMP = 7782;
  1 row updated.
```

• Todas as linhas na tabela são modificadas quando você omite a cláusula WHERE.

```
SQL> UPDATE FUNCIONARIO
  2   SET   COD_DEPTO = 20;
14 rows updated.
```

Atualizando Linhas (continuação)

A instrução UPDATE modifica linhas específicas, se a cláusula WHERE for especificada.

O exemplo do slide transfere o funcionário 7782 (Clark) para o departamento 20. Se você omitir a cláusula WHERE, todas as linhas na tabela serão modificadas.

```
SQL> SELECT NOME, COD_DEPTO
2     FROM  FUNCIONARIO;


NOME        COD_DEPTO
---------- ---------
KING        20
BLAKE       20
CLARK       20
JONES       20
MARTIN      20
ALLEN       20
TURNER      20
...
14 rows selected.
```

Observação: A tabela FUNCIONARIO possui os mesmos dados que a tabela EMP.

Atualizando com Subconsulta de Várias Colunas

Atualize o cargo e o departamento do funcionário 7698 para coincidir com o do funcionário 7499.

```
SQL> UPDATE EMPREGADO
  2   SET    (OCUP, COD_DEPTO) =
  3                (SELECT OCUP, COD_DEPTO
  4                 FROM  EMPREGADO
  5                 WHERE COD_EMP = 7499)
  6   WHERE COD_EMP = 7698;
1 row updated.
```

Atualizando com Subconsulta de Várias Colunas

Subconsultas de várias colunas podem ser implementadas na cláusula SET de uma instrução UPDATE.

Sintaxe

```
UPDATE tabela
SET   (coluna, coluna, ...) =
           (SELECT     coluna, coluna, ...
           FROM  tabela
```

```
            WHERE condição)
WHERE condição;
```

Atualizando Linhas Baseadas em Outra Tabela

Use subconsultas em instruções UPDATE para atualizar linhas em uma tabela baseada em valores de outra tabela.

```
SQL> UPDATE FUNCIONARIO
  2   SET   COD_DEPTO = (SELECT     COD_DEPTO
  3               FROM  EMPREGADO
  4               WHERE COD_EMP =   7788)
  5   WHERE OCUP  =     (SELECT     OCUP
  6               FROM  EMPREGADO
  7               WHERE COD_EMP = 7788);
  2 rows updated.
```

Atualizando Linhas Baseadas em Outra Tabela

Você pode usar subconsultas em instruções UPDATE para atualizar linhas em uma tabela. O exemplo no slide atualiza a tabela FUNCIONARIO baseada nos valores da tabela EMP. Isso altera o número do departamento de todos os funcionários com o cargo do funcionário 7788 para o número do departamento atual do funcionário 7788.

Atualizando Linhas: Erro de Restrição de Integridade

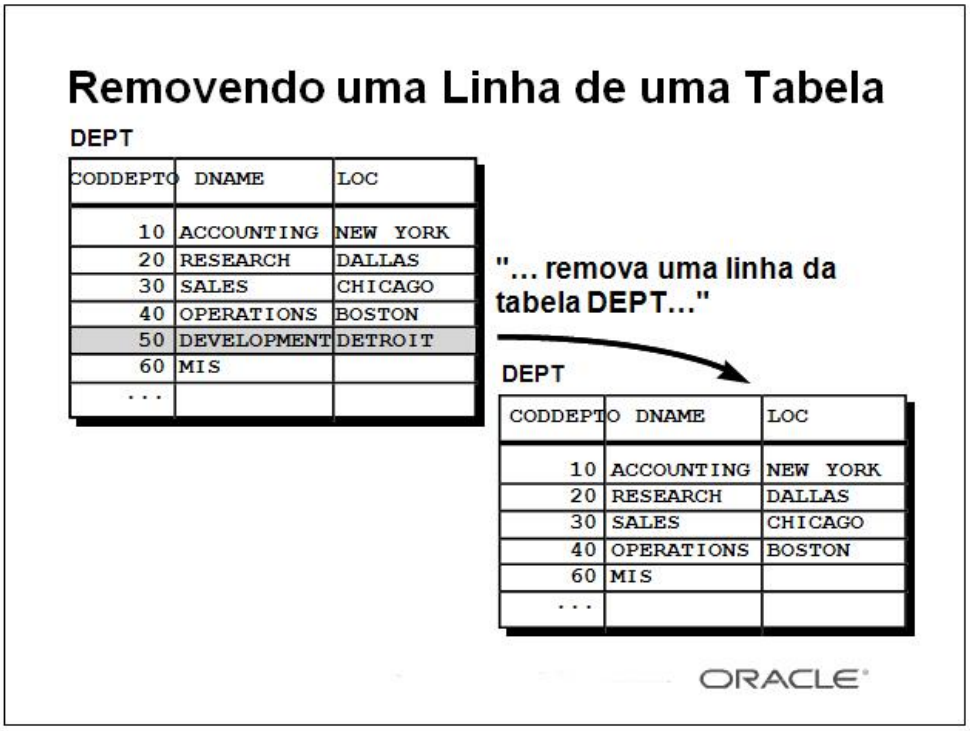
```
SQL> UPDATE EMPREGADO
  2   SET   COD_DEPTO = 55
  3   WHERE COD_DEPTO = 10;


UPDATE EMPREGADO
*
ERROR at line 1:
ORA-02291: integrity constraint (USR.EMP_COD_DEPTO_FK)
violated - parent key not found
```

Erro de Restrição de Integridade

Se tentar atualizar um registro com um valor vinculado a uma restrição de integridade, você causará um erro.

No exemplo no slide, o número de departamento 55 não existe na tabela mãe, DEPT, assim você não receberá a violação da chave mãe ORA-02291.

Observação: As restrições de integridade garantem que os dados obedeçam a um conjunto predefinido de regras. Uma lição subseqüente irá abordar as restrições de integridade com mais detalhes.



Removendo uma Linha de uma Tabela

O gráfico do slide remove o departamento DEVELOPMENT da tabela DEPT (pressupondo que não há restrições definidas na tabela DEPT).

A Instrução DELETE

Você pode remover linhas existentes de uma tabela usando a instrução DELETE.

```
DELETE [FROM]     tabela
[WHERE      condição];
```

Deletando Linhas

Você pode remover linhas existentes usando a instrução DELETE.

Na sintaxe:

tabela	é o nome da tabela
condição	identifica as linhas a serem deletadas e é composta de nomes de colunas, expressões, constantes, subconsultas e operadores de comparação

Para obter mais informações, consulte o Oracle Server SQL Reference, Release 8, "DELETE".

Deletando Linhas de uma Tabela

• Linhas específicas são deletadas quando você especifica a cláusula WHERE.

```
SQL> DELETE FROM  DEPARTAMENTO
  2  WHERE  dname = 'DEVELOPMENT';
  1 row deleted.
```

• Todas as linhas na tabela serão deletadas se você omitir a cláusula WHERE.

```
SQL> DELETE FROM  DEPARTAMENTO;
  4 rows deleted.
```

Deletando Linhas (continuação)

Você pode deletar determinadas linhas especificando a cláusula WHERE na instrução DELETE. O exemplo do slide deleta o departamento DEVELOPMENT da tabela DEPARTAMENTO. Você pode confirmar essa operação exibindo as linhas deletadas usando a instrução SELECT.

```
SQL> SELECT *
  2   FROM  DEPARTAMENTO
  3   WHERE dname = 'DEVELOPMENT';
```

```
no rows selected.
```

Exemplo

Remova todos os funcionários que iniciaram após 1º de janeiro de 1997.

```
SQL> DELETE FROM  EMPREGADO
  2   WHERE hiredate > TO_DATE('01.01.1997', 'DD.MM.YYYY');
  1  row deleted.
```

Se você omitir a cláusula WHERE, todas as linhas na tabela serão deletadas. O segundo exemplo no slide deleta todas as linhas da tabela DEPARTAMENTO porque nenhuma cláusula WHERE foi especificada.

Observação: A tabela DEPARTAMENTO possui os mesmos dados que a tabela DEPT.

Deletando Linhas Baseadas em Outra Tabela

Use subconsultas em instruções DELETE
para remover linhas de uma tabela baseadas em valores de outra tabela.

```
SQL> DELETE FROM  FUNCIONARIO
  2   WHERE COD_DEPTO =
  3               (SELECT     COD_DEPTO
  4               FROM  dept
  5               WHERE dname ='SALES');
  6 rows deleted.
```

Deletando Linhas Baseadas em Outra Tabela

Você pode usar subconsultas para deletar linhas de uma tabela baseadas em valores de outra tabela.

O exemplo no slide deleta todos os funcionários que estejam no departamento 30. A subconsulta procura a tabela DEPT para localizar o número de departamento para o departamento SALES. A subconsulta então alimenta o número de departamento para a consulta principal, que deleta linhas de dados da tabela FUNCIONARIO baseada nesse número de departamento.

Deletando Linhas: Erro de Restrição de Integridade

```
SQL> DELETE FROM  dept
  2   WHERE COD_DEPTO = 10;

DELETE FROM dept
* ERROR at line 1:
ORA-02292: integrity constraint (USR.EMP_COD_DEPTO_FK)
violated - child record found
```

Erro de Restrição de Integridade

Se tentar deletar um registro com um valor vinculado a uma restrição de integridade, você causará um erro.

O exemplo no slide tenta deletar o número de departamento 10 da tabela DEPT, o que resulta em um erro porque o número de departamento é usado como uma chave estrangeira na tabela EMP. Se o registro pai que você tentar deletar tiver registros filhos, você receberá a violação de registro filho encontrada ORA-02292.

Transações de Banco de Dados

Consistem de uma das seguintes instruções:

- Instruções DML que fazem uma alteração consistente nos dados
- Uma instrução DDL
- Uma instrução DCL

Transações de Banco de Dados

O Oracle Server garante consistência de dados baseada em transações. As transações lhe dão mais flexibilidade e controle ao alterar dados e garantem a consistência de dados no caso de falha do usuário ou do sistema.

As transações consistem de instruções DML que façam uma alteração consistente nos dados. Por exemplo, uma transferência de fundos entre duas contas deve incluir o débito em uma conta e o crédito em outra com mesma quantia. Ambas as ações devem falhar ou ter êxito juntas. O crédito não deve ser processado sem o débito.

Tipos de Transação

Tipo	Descrição
Data manipulation language (DML)	Consiste de qualquer número de instruções DML que o Oracle Server trata como uma única entidade ou unidade lógica de trabalho
Data definition language (DDL)	Consiste de apenas uma instrução DDL
Data control language (DCL)	Consiste de apenas uma instrução DCL

Transações de Banco de Dados

• Começa quando for executada a primeira instrução SQL executável

• Termina com um dos seguintes eventos:

– COMMIT ou ROLLBACK é emitida

– Instrução DDL ou DCL é executada (commit automático)

– O usuário sai

– O sistema cai

Quando uma Transação Inicia e Termina?

Uma transação inicia quando a primeira instrução SQL executável for encontrada e termina quando ocorrer uma das seguinte situações:

- Uma instrução COMMIT ou ROLLBACK for emitida
- Uma instrução DDL, como CREATE, for emitida
- Uma instrução DCL for emitida
- O usuário sair do SQL*Plus
- Houver uma falha no computador ou o sistema cair

Depois que uma transação termina, a próxima instrução SQL executável automaticamente inicia a próxima transação.

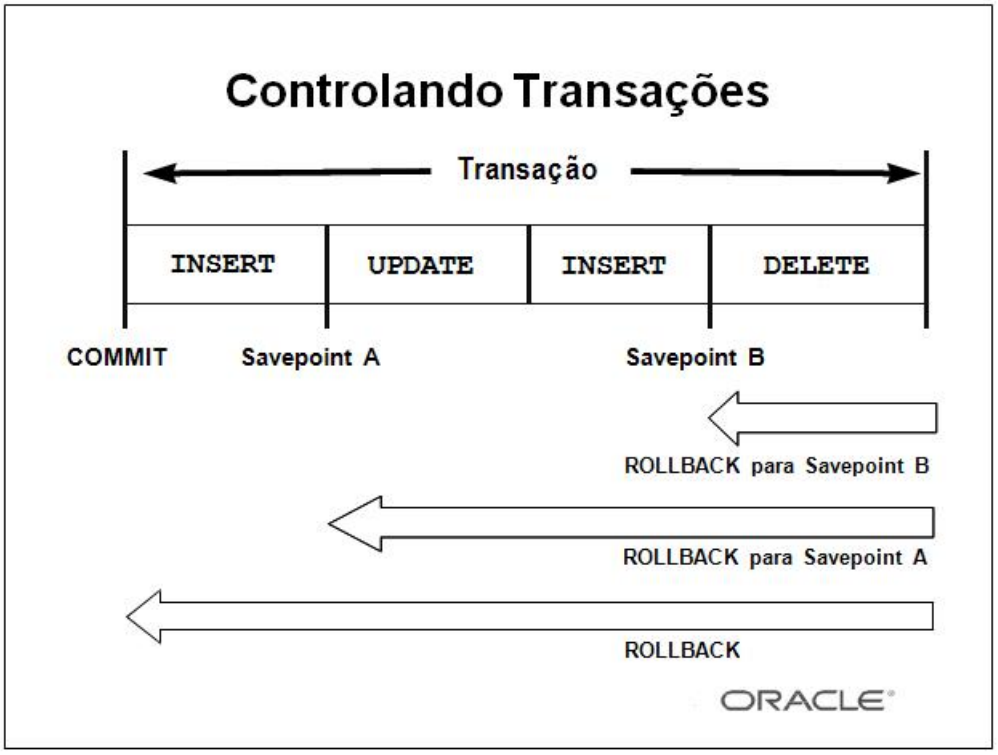
Uma instrução DDL ou DCL é automaticamente processada e, portanto, finaliza implicitamente uma transação.

Vantagens das Instruções COMMIT e ROLLBACK

• Garantir consistência de dados

• Visualizar alterações nos dados antes de fazer as alterações permanentemente

• Agrupar operações relacionadas logicamente



Instruções de Controle de Transação Explícita

Você pode controlar a lógica das transações usando as instruções COMMIT, SAVEPOINT e ROLLBACK.

Instrução	Descrição
COMMIT	Finaliza a transação atual tornando permanentes todas as alterações de dados pendentes
SAVEPOINT *nome*	Marca um ponto de gravação dentro da transação atual
ROLLBACK [TO *SAVEPOINT nome*]	ROLLBACK finaliza a transação atual descartando todas as alterações de dados pendentes; ROLLBACK TO SAVEPOINT descarta a transação atual para o ponto de gravação específico, descartando assim o ponto de gravação e quaisquer alterações subseqüentes. Se você omitir essa cláusula, a instrução ROLLBACK descarta toda a transação.

Observação: SAVEPOINT não é uma SQL padrão de ANSI.

Processando Transações Implícitas

• Um commit automático ocorre sob as seguintes circunstâncias:

– A instrução DDL é emitida

– A instrução DCL é emitida

– A saída normal do SQL*Plus, sem emitir explicitamente COMMIT ou ROLLBACK

• Um rollback automático ocorre quando há uma finalização anormal do SQL*Plus ou queda do sistema.

Processando Transações Implícitas

Status	Circunstâncias
Processamento automático	As instruções DDL ou DCL são emitidas Saída normal do SQL*Plus, sem emitir explicitamente COMMIT ou ROLLBACK
Rollback automático	Finalização anormal do SQL*Plus ou queda do sistema

Observação: Há um terceiro comando disponível no SQL*Plus. O comando AUTOCOMMIT pode ser alternado entre ON ou OFF. Se for definido como ON, cada instrução DML individual será processada assim que for executada. Você não pode fazer roll back das alterações. Se for definido como OFF, COMMIT poderá ser emitido explicitamente. Além disso, COMMIT é emitido quando uma instrução DLL é emitida ou quando você sai do SQL*Plus.

Falhas do Sistema

Quando uma transação é interrompida por uma falha do sistema, faz-se automaticamente roll back de toda a transação. Isso impede que o erro faça alterações não desejadas nos dados e retorna as tabelas ao seu estado no momento do último commit. Dessa forma, o Oracle Server protege a integridade das tabelas.

Estado dos Dados Antes de COMMIT ou ROLLBACK

• O estado anterior dos dados pode ser recuperado.

• O usuário atual pode revisar os resultados das operações DML usando a instrução SELECT.

• Outros usuários não poderão ver os resultados das instruções DML do usuário atual.

• As linhas afetadas são bloqueadas, outros usuários não poderão alterar os dados dentro das linhas afetadas.

Submetendo Alterações a Commit

Todas as alterações feitas nos dados durante a transação são temporárias até que a transação seja processada.

Estado dos dados antes que COMMIT ou ROLLBACK seja emitido:

•As operações de manipulação de dados afetam primeiramente o buffer do banco de dados, assim, o estado anterior dos dados pode ser recuperado.

•O usuário atual pode revisar os resultados das operações de manipulação de dados consultando as tabelas.

•Os outros usuários não poderão exibir os resultados das operações de manipulação de dados feitas pelo usuário atual. O Oracle Server institui a consistência na leitura para garantir que cada usuário veja os dados como eram no último commit.

•As linhas afetadas são bloqueadas, os outros usuários não poderão alterar os dados nas linhas afetadas.

Estado dos Dados Após COMMIT

• As alterações nos dados são feitas permanentemente no banco de dados.

• O estado anterior dos dados é perdido permanentemente.

• Todos os usuários podem ver os resultados.

• As linhas afetadas são desbloqueadas, essas linhas estão disponíveis para serem manipuladas por outros usuários.

• Todos os savepoints são apagados.

Submetendo Alterações a Commit (continuação)

Torne permanentes todas as alterações pendentes usando a instrução COMMIT. Após COMMIT:

O estado dos dados após emitir uma COMMIT:

•As alterações nos dados são gravadas no banco de dados.

•O estado anterior dos dados é perdido permanentemente.

•Todos os usuários podem exibir os resultados da transação.

•As linhas afetadas são desbloqueadas, as linhas estão agora disponíveis para outros usuários executarem as novas alterações nos dados.

•Todos os savepoints são apagados.

Submetendo Dados a Commit

• Fazer as alterações.

```
SQL> UPDATE EMPREGADO
  2   SET   COD_DEPTO = 10
  3   WHERE COD_EMP = 7782;
  1 row updated.
```

• Submeter alterações a commit.

```
SQL> COMMIT;
Commit complete.
```

Submetendo Alterações a Commit (continuação)

O exemplo do slide atualiza a tabela EMP e define o número de departamento para o funcionário 7782 (Clark) como 10. Ele depois torna a alteração permanente emitindo a instrução COMMIT.

Exemplo

Crie um novo departamento ADVERTISING com pelo menos um funcionário. Torne permanentes as alterações nos dados.

```
SQL> INSERT INTO DEPARTAMENTO(COD_DEPTO, dname, loc)
  2   VALUES      (50, 'ADVERTISING', 'MIAMI');
  1 row created.

SQL> UPDATE FUNCIONARIO
  2   SET   COD_DEPTO = 50
  3   WHERE COD_EMP = 7876;
  1 row updated.

SQL> COMMIT;
Commit complete.
```

Estado dos Dados Após ROLLBACK

Descarte todas as alterações pendentes usando a instrução ROLLBACK.

- As alterações nos dados são desfeitas.
- O estado anterior dos dados é restaurado.
- As linhas afetadas são desbloqueadas.

```
SQL> DELETE FROM  FUNCIONARIO;
14 rows deleted.

SQL> ROLLBACK;
Rollback complete.
```

Fazendo Roll Back de Alterações

Descarte todas as alterações pendentes usando a instrução ROLLBACK.

Após ROLLBACK:

- As alterações nos dados são desfeitas.
- O estado anterior dos dados é restaurado.
- As linhas afetadas são desbloqueadas.

Exemplo

Ao tentar remover um registro da tabela TEST, você pode acidentalmente esvaziar a tabela. Você pode corrigir o erro, emitir novamente a instrução apropriada e tornar permanentes as alterações dos dados.

```
SQL> DELETE FROM  test;
25,000 rows deleted.
SQL> ROLLBACK; Rollback complete.
SQL> DELETE FROM  test
2     WHERE id = 100;
1 row deleted.
SQL> SELECT *
2     FROM  test
3     WHERE id = 100;
No rows selected.
SQL> COMMIT; Commit complete.
Introdução ao Oracle: SQL e PL/SQL  9-32
```

Fazendo Roll Back de Alterações para um Marcador

- Crie um marcador em uma transação atual usando a instrução SAVEPOINT.

- Faça roll back do marcador usando a instrução ROLLBACK TO SAVEPOINT.

```
SQL> UPDATE...
SQL> SAVEPOINT update_done; Savepoint created.
SQL> INSERT...
SQL> ROLLBACK TO update_done; Rollback complete.
```

Fazendo Roll Back de Alterações para um Savepoint

Você pode criar um marcador na transação atual usando a instrução SAVEPOINT. Dessa forma, a transação poderá ser dividida

em seções menores. Você poderá então descartar as alterações pendentes até aquele marcador usando a instrução ROLLBACK TO SAVEPOINT.

Se você criar um segundo savepoint com o mesmo nome que o anterior, o anterior será deletado.

Rollback no Nível da Instrução

• Se uma única instrução DML falhar durante a execução, será feito roll back somente dessa instrução.

• O Oracle Server implementa um savepoint implícito.

• Todas as outras alterações são mantidas.

• O usuário deve finalizar as transações explicitamente usando uma instrução COMMIT ou ROLLBACK.

Rollback no Nível da Instrução

Parte da transação pode ser descartada por um rollback implícito se for detectado um erro na execução da instrução. Se uma única instrução DML falhar durante a execução de uma transação, seu efeito será desfeito por um rollback no nível da instrução, mas as alterações feitas pelas instruções

DML anteriores na transação não serão descartadas. Somente o usuário poderá fazer o roll back ou processá-las.

O Oracle emite uma instrução COMMIT implícita antes e após qualquer instrução DDL (Data Definition Language). Assim, mesmo que a instrução DDL não seja executada com êxito, você não poderá fazer roll back da instrução anterior porque o servidor emitiu um commit.

Finalize suas transações explicitamente executando uma instrução COMMIT ou ROLLBACK.

Consistência na Leitura

• A consistência na leitura garante sempre uma exibição consistente dos dados.

• As alterações feitas por um usuário não entram em conflito com as alterações feitas por outro usuário.

• A consistência na leitura garante que nos mesmos dados:

– Os leitores não esperem pelos autores

– Os autores não esperem pelos leitores

Consistência na Leitura

Os usuários de bancos de dados fazem dois tipos de acesso ao banco de dados:

•Operações de leitura (instrução SELECT)

•Operações de gravação (instruções INSERT, UPDATE, DELETE) Você precisa de consistência na leitura para que ocorra o seguinte:

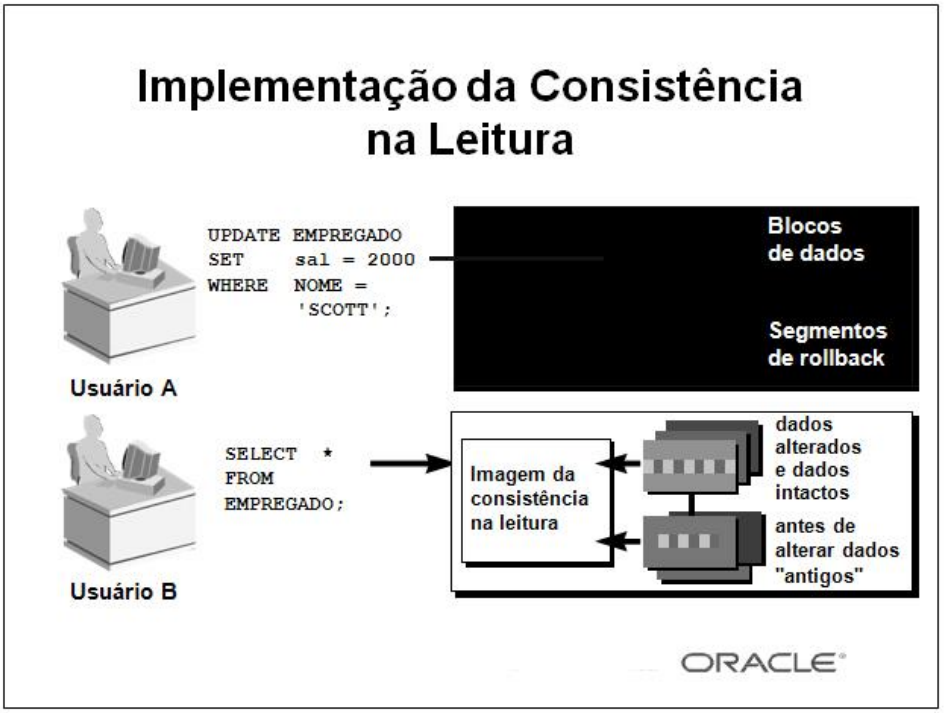
•O leitor e autor do banco de dados tenham garantia de uma exibição consistente dos dados.

•Os leitores não vejam os dados que estejam sendo alterados.

•Os autores tenham garantia de que as alterações no banco de dados sejam feitas de forma consistente.

•As alterações feitas por um autor não interrompam ou entrem em conflito com as alterações que outro autor esteja fazendo.

O objetivo da consistência na leitura é garantir que cada usuário veja os dados como eles eram no último commit, antes da operação DML iniciar.



Implementação da Consistência na Leitura

A consistência na leitura é uma implementação automática. Ela mantém uma cópia parcial do banco

de dados em segmentos de rollback.

Quando uma operação de inserção, atualização ou exclusão é feita no banco de dados, o Oracle

Server tira uma cópia dos dados antes de serem alterados e os grava no segmento de rollback.

Todos os leitores, exceto o que fez a alteração, ainda visualizam o banco de dados como era antes do início das alterações, eles visualizam um "instantâneo" dos segmentos de rollback dos dados.

Antes das alterações serem processadas no banco de dados, somente o usuário que está modificando os dados vê o banco de dados com as alterações, todas as outras pessoas vêem o instantâneo no segmento de rollback. Isso garante que os leitores dos dados leiam dados consistentes que não estejam sendo alterados no momento.

Quando uma instrução DML é processada, a alteração feita no banco de dados torna-se visível a qualquer pessoa que execute a

instrução SELECT. O espaço ocupado pelos dados "antigos" no arquivo do segmento de rollback é liberado para ser utilizado novamente.

Se for feito o roll back da transação, as alterações serão desfeitas.

•A versão original, mais antiga, dos dados no segmento de rollback é gravada de volta na tabela.

•Todos os usuários vêem o banco de dados como ele era antes da transação iniciar.

Bloqueando

Bloqueios do Oracle:

• Impedem a interação destrutiva entre transações simultâneas

• Não requerem ação do usuário

• Usam automaticamente o nível mais baixo de restrição

• São mantidos durante a duração da transação

• Há dois modos básicos:
– Exclusivo
– Compartilhado

O Que São Bloqueios?

Bloqueios são mecanismos que impedem a interação destrutiva entre transações que acessem o mesmo recurso: um objeto de usuário (como tabelas ou linhas) ou objetos do sistema não visíveis aos usuários (como estruturas de dados compartilhados e linhas de dicionários de dados).

Como o Oracle Bloqueia os Dados

O bloqueio em um banco de dados do Oracle é totalmente automático e não requer ação do usuário.

O bloqueio implícito ocorre para todas as instruções SQL exceto SELECT. O mecanismo de bloqueio default do Oracle automaticamente usa o nível mais inferior da restrição aplicável, fornecendo assim

O maior grau de simultaneidade e máxima integridade de dados. O Oracle também permite que o usuário bloqueie os dados manualmente.

Modos de Bloqueio

O Oracle usa dois modos de bloqueio em um banco de dados de vários usuários:

Modo de Bloqueio	Descrição
exclusivo	Impede que um recurso seja compartilhado. A primeira transação que bloquear um recurso exclusivamente será a única alteração que poderá alterá-lo até ser liberado.
bloqueio compartilhado	Permite que o recurso seja compartilhado. Vários usuários que leiam os dados podem compartilhar esses dados, mantendo os bloqueios compartilhados para impedir o acesso simultâneo por um escritor (que precisa de um bloqueio exclusivo). Várias transações podem adquirir bloqueios compartilhados no mesmo.

Sumário

Instrução	Descrição
INSERT	Adiciona uma nova linha à tabela
UPDATE	Modifica linhas existentes na tabela
DELETE	Remove linhas existentes da tabela
COMMIT	Torna permanente todas as alterações pendentes
SAVEPOINT	Permite um rollback no marcador do savepoint
ROLLBACK	Descarta todas as alterações nos dados pendentes

Sumário

Manipule dados no banco de dados do Oracle usando as instruções INSERT, UPDATE e DELETE. Controle as alterações nos dados usando as instruções COMMIT, SAVEPOINT e ROLLBACK.

O Oracle Server garante uma exibição consistente dos dados sempre. O bloqueio pode ser implícito ou explícito.

Capítulo 09

Criando e Gerenciando Tabelas

Objetivos

Depois de completar esta lição, você poderá fazer o seguinte:

• Descrever os principais objetos do banco de dados

• Criar tabelas

• Descrever os tipos de dados que podem ser usados ao especificar a definição da coluna

• Alterar definições de tabela

• Eliminar, renomear e truncar tabelas

Objetivo da Lição
Nesta lição, você aprenderá sobre os principais objetos do banco de dados e o relacionamento entre eles. Você também aprenderá como criar, alterar e eliminar tabelas.

Objetos do Banco de Dados

Objeto	Descrição
Tabela	Unidade básica de armazenamento, composta de linhas uma ou mais tabelas
View	Representa logicamente subconjuntos de dados de uma ou mais tabelas
Sequência	Gera valores de chave primária
Índice	Melhora o desempenho de algumas consultas
Sinônimo	Atribui nomes alternativos a objetos

Objetos do Banco de Dados

Um banco de dados Oracle pode conter várias estruturas de dados. Cada estrutura deve ser descrita no projeto do banco de dados para que possa ser criada durante o estágio de desenvolvimento do banco de dados.

- Tabela: Armazena dados
- View: Subconjunto de dados de uma ou mais tabelas
- Sequência: Gera valores de chave primária
- Índice: Melhora o desempenho de algumas consultas
- Sinônimo: Atribui nomes alternativos a objetos

Estruturas de Tabela do Oracle

- As tabelas podem ser criadas a qualquer momento, até mesmo quando os usuários estiverem usando o banco de dados.
- Não é necessário especificar o tamanho de nenhuma tabela. O tamanho é definido pela quantidade de espaço alocada no banco de dados como um todo. Entretanto, é importante estimar a quantidade de espaço que uma tabela usará.
- A estrutura da tabela pode ser modificada on-line.

Observação: Há mais objetos de banco de dados disponíveis que não são abordados neste curso.

Convenções para Nomeação

• Deve começar com uma letra

• Pode ter de 1 a 30 caracteres

• Deve conter somente A–Z, a–z, 0–9, _, $ e #

• Não deve duplicar o nome de outro objeto de propriedade do mesmo usuário

• Não deve ser uma palavra reservada pelo Oracle Server

Regras para Nomeação

Nomeie tabelas e colunas do banco de dados de acordo com as regras default de nomeação de qualquer objeto do banco de dados Oracle:

•Nomes de tabela e de colunas devem começar com uma letra e podem ter de 1 a 30 caracteres.

•Os nomes devem conter somente os caracteres A-Z, a–z, 0–9, _ (sublinhado), $ e # (caracteres legais, mas evite usá-los).

•Os nomes não devem duplicar o nome de outro objeto de propriedade do mesmo usuário do Oracle Server.

•Os nomes não devem ser uma palavra reservada do Oracle Server.

Diretrizes para Nomeação

•Use nomes descritivos para tabelas e outros objetos do banco de dados.

•Nomeie a mesma entidade de modo consistente em diferentes tabelas. Por exemplo, a coluna do número do departamento é chamada COD_DEPTO nas tabelas EMP e DEPT.

Observação: Os nomes não fazem distinção entre maiúsculas de minúsculas. Por exemplo, EMP é tratada com o mesmo nome que eMP ou eMp.

Para obter mais informações, consulte o Oracle Server SQL Reference, Release 8, "Object Names and Qualifiers".

A Instrução CREATE TABLE

• Você deve ter:

– privilégio CREATE TABLE

– Uma área de armazenamento

```
CREATE [GLOBAL TEMPORARY] TABLE [esquema.]tabela
            (tipo de dados da coluna
            [DEFAULT expr][, ...]);
```

• Especifique:

– Nome da tabela

– Nome da coluna, tipo de dados da coluna e tamanho da coluna

A Instrução CREATE TABLE

Crie tabelas para armazenar dados executando a instrução SQL CREATE TABLE. Ela é uma das instruções DDL (data definition language) a ser abordada nas próximas lições. As instruções DDL são um subconjunto de instruções SQL usadas para criar, modificar ou remover estruturas do banco de dados Oracle. Essas instruções possuem um efeito imediato no banco de dados e também registram informações no dicionário de dados.

Para criar uma tabela, o usuário deve ter o privilégio CREATE TABLE e uma área de armazenamento na qual criar objetos. O administrador do banco de dados usa instruções DCL (data control language), que serão abordadas mais tarde.

Na sintaxe:

GLOBAL TEMPORARY	especifica que a tabela é temporária e que sua definição está visível em todas as sessões. Os dados em uma tabela temporária são visíveis somente na sessão que insere dados na tabela.
esquema	é o mesmo do nome do proprietário
tabela	é o nome da tabela
DEFAULT expr	especifica um valor default se um valor estiver omitido na instrução INSERT
coluna	é o nome da coluna
tipo de dados	é o tipo de dados e o comprimento da coluna

Fazendo Referência a Tabelas de Outro Usuário

• As tabelas que pertencem a outros usuários não estão no esquema do usuário.

• Você deve usar o nome do proprietário como um prefixo da tabela.

Fazendo Referência a Tabelas de Outro Usuário

Um esquema é um conjunto de objetos. Os objetos de esquema são as estruturas lógicas que fazem referência diretamente aos dados no banco de dados. Os objetos de esquema incluem tabelas, views, sinônimos, sequências, procedimentos armazenados, índices, clusters e vínculos com o banco de dados.

Se uma tabela não pertencer ao usuário, o nome do proprietário deve ser prefixado à tabela.

A Opção DEFAULT

• Especifique um valor default para uma coluna durante uma inserção.

```
… hiredate DATE DEFAULT SYSDATE, …
```

• Valores legais são um valor literal, expressão ou função SQL.

• Valores ilegais são outro nome da coluna ou pseudocoluna.

• O tipo de dados default deve corresponder ao tipo de dados da coluna.

A Opção DEFAULT

Um valor default pode ser dado à uma coluna usando-se a opção DEFAULT. Essa opção impede que valores nulos entrem nas colunas se uma linha for inserida sem um valor para a coluna. O valor default pode ser um literal, uma expressão ou uma função SQL, como SYSDATE e USER, mas o valor não pode ser o nome de outra coluna ou pseudocoluna como NEXTVAL ou CURRVAL. A expressão default deve corresponder ao tipo de dados da coluna.

Criando Tabelas

• Crie a tabela.

```
SQL> CREATE TABLE dept
  2         (COD_DEPTO  NUMBER(2),
  3         dname VARCHAR2(14),
  4         loc   VARCHAR2(13));
 Table created.
```

• Confirme a criação da tabela.

```
SQL> DESCRIBE dept

Name                           Null?       Type
------------------------------ -------- ---------
COD_DEPTO                                  NUMBER(2)
DNAME                                      VARCHAR2(14)
LOC                                        VARCHAR2(13)
```

Criando Tabelas

O exemplo no slide cria a tabela DEPT, com três colunas — chamadas, COD_DEPTO, DNAME e LOC. Ele também confirma a criação da tabela emitindo o comando DESCRIBE.

Como criar uma tabela é uma instrução DDL, um commit automático ocorre quando essa instrução é executada.

Tabelas no Banco de Dados Oracle

• Tabelas do Usuário

– Conjunto de tabelas criadas e mantidas pelo usuário

– Contêm informações sobre o usuário

• Dicionário de dados

– Conjunto de tabelas criadas e mantidas pelo Oracle server

– Contêm informações sobre o banco de dados

Tabelas no Banco de Dados Oracle

Tabelas do usuário são tabelas criadas pelo usuário, como EMP. Há outro conjunto de tabelas e views no banco de dados Oracle conhecido como o dicionário de dados. Esse conjunto é criado e mantido pelo Oracle Server e contém informações sobre o banco de dados.

Todas as tabelas de dicionário de dados são de propriedade do usuário SYS. As tabelas-base dificilmente são acessadas pelo usuário porque a informação contida nelas não é de fácil compreensão. Assim, os usuários geralmente acessam as views do dicionário de dados porque as informações são apresentadas em um formato mais fácil de entender. As informações armazenadas no dicionário de dados incluem nomes dos usuários do Oracle Server, privilégios concedidos a usuários, nomes de objeto do banco de dados, restrições de tabela e informações sobre auditoria.

Há quatro categorias de views do dicionário de dados, cada categoria contém um prefixo distinto que reflete o uso pretendido.

Prefixo	Descrição
USER_	Estas views contêm informações sobre objetos de propriedade do usuário.
ALL_	Estas views contêm informações sobre todas as tabelas (de objeto e relacionais) acessíveis ao usuário.
DBA_	Estas views são restritas. Estas views somente podem ser acessadas por pessoas que tenham sido atribuídas o DBA total.
V$_	Estas views contêm informações sobre views de desempenho dinâmico, desempenho do servidor do banco de dados e bloqueio.

Consultando o Dicionário de Dados

• Descreva tabelas de propriedade do usuário.

```
SQL> SELECT *
  2   FROM  user_tables;
```

• Exiba tipos de objetos distintos de propriedade do usuário.

```
SQL> SELECT DISTINCT object_type
  2   FROM  user_objects;
```

• Exiba tabelas, views, sinônimos e sequências de propriedade do usuário.

```
SQL> SELECT *
  2   FROM  user_catalog;
```

Consultando o Dicionário de Dados

Você pode consultar as tabelas de dicionário de dados para exibir vários objetos de banco de dados de sua propriedade. As tabelas de dicionário de dados freqüentemente usadas são:

•USER_TABLES
•USER_OBJECTS
•USER_CATALOG

Observação: USER_CATALOG possui um sinônimo chamado CAT. Você pode usar esse sinônimo no lugar de USER_CATALOG nas instruções SQL.

```
SQL> SELECT *
  2   FROM  CAT;
```

Tipos de Dados

Tipo de Dados	Descrição
VARCHAR2(tamanho)	Dados de caractere de comprimento variável
CHAR(tamanho)	Dados de caractere de comprimento fixo
NUMBER(p,s)	Dados numéricos de comprimento variável
DATE	Valores de data e hora
LONG	Dados de caractere de comprimento variável até 2 gigabytes
CLOB	Dados de caractere de um byte de até
4 gigabytes	
RAW e LONG RAW	Dados binários brutos
BLOB	Dados binários de até 4 gigabytes
BFILE	Dados binários armazenados em um arquivo externo de até 4 gigabytes

Tipos de Dados

Tipo de Dados	Descrição
VARCHAR2(*tamanho*)	Dados de caractere de comprimento variável (Um *tamanho* máximo deve ser especificado). O *tamanho* default e mínimo é 1; o *tamanho* máximo é 4000.)
CHAR(*tamanho*)	Dados de caractere de comprimento fixo de bytes de *tamanho* de comprimento (O *tamanho* default e mínimo é 1; o *tamanho* máximo é 2000.)
NUMBER(*p,s*)	Número contendo a precisão *p* e a escala *s* (A precisão é o número total de dígitos decimais e a escala é o número de dígitos à direita do ponto decimal. A precisão pode variar de 1 a 38 e a escala, de -84 a 127.)
DATE	Valores de data e hora entre 1º de janeiro, 4712 A.C. e 31 de dezembro, 9999 D.C.
LONG	Dados de caractere de comprimento variável até 2 gigabytes
CLOB	Dados de caractere de um byte de até 4 gigabytes

Tipos de Dados

Tipo de Dados	Descrição
VARCHAR2(tamanho)	Dados de caractere de comprimento variável
CHAR(tamanho)	Dados de caractere de comprimento fixo
NUMBER(p,s)	Dados numéricos de comprimento variável
DATE	Valores de data e hora
LONG	Dados de caractere de comprimento variável até 2 gigabytes
CLOB	Dados de caractere de um byte de até 4 gigabytes
RAW e LONG RAW	Dados binários brutos
BLOB	Dados binários de até 4 gigabytes
BFILE	Dados binários armazenados em um arquivo externo de até 4 gigabytes

Tipos de dados (continuação)

Tipo de Dados	Descrição
RAW(*tamanho*)	Dados binários brutos de *tamanho* de comprimento (Um *tamanho* máximo deve ser especificado. O *tamanho* máximo é 2000.)
LONG RAW	Dados binários brutos de comprimento variável de até 2 gigabytes
BLOB	Dados binários de até 4 gigabytes
BFILE	Dados binários armazenados em um arquivo externo; até 4 gigabytes

Criando uma Tabela Usando uma Subconsulta

• Crie uma tabela e insira linhas combinando a instrução CREATE TABLE e a opção da
subconsulta AS.

```
CREATE TABLE tabela
[(coluna, coluna...)] AS subconsulta;
```

• Faça a correspondência do número de
colunas especificadas com o número
de colunas da subconsulta.

• Defina colunas com nomes de colunas e valores default.

Criando uma Tabela a Partir de Linhas em Outra Tabela

Um segundo método para criar uma tabela é aplicar a cláusula da subconsulta AS para criar a tabela e inserir linhas retornadas da subconsulta.

Na sintaxe:

tabela é o nome da tabela.
coluna é o nome da coluna, valor default e restrição de integridade.
subconsulta é a instrução SELECT que define o conjunto de linhas a serem inseridas sua nova tabela.

Diretrizes

•A tabela será criada com os nomes de coluna especificados e as linhas recuperadas pela instrução SELECT serão inseridas na tabela.

•A definição da coluna pode conter somente o nome da coluna e o valor default.

•Se forem dadas especificações para a coluna, o número de colunas deve ser igual ao número de colunas na lista SELECT da subconsulta.

•Se não forem dadas especificações para a coluna, os nomes de coluna da tabela serão os mesmos que os nomes de coluna na subconsulta.

Criando uma Tabela

Usando uma Subconsulta

```
SQL> CREATE TABLE dept30
  2   AS
  3   SELECT      COD_EMP, NOME, sal*12 ANNSAL, hiredate
  4   FROM  EMPREGADO
  5   WHERE deptno =    30;
Table created.


SQL> DESCRIBE dept30


Name                           Null?      Type
------------------------------ --------   -----
COD_EMP                        NOT NULL   NUMBER(4)
NOME                                      VARCHAR2(10)
ANNSAL                                    NUMBER
HIREDATE                                  DATE
```

Criando uma Tabela a Partir de Linhas em Outra Tabela (continuação)

O exemplo do slide cria uma tabela, DEPT30, que contém detalhes sobre todos os funcionários que trabalham no departamento 30. Observe que os dados da tabela DEPT30 vêm da tabela EMP.

Você pode verificar a existência de uma tabela do banco de dados e verificar as definições da coluna usando o comando DESCRIBE do SQL*Plus.

Forneça um apelido para a coluna ao selecionar uma expressão.

A Instrução ALTER TABLE

Use a instrução ALTER TABLE para:

- Adicionar uma nova coluna
- Modificar uma coluna existente
- Definir um valor default para a nova coluna

```
ALTER TABLE tabela
ADD         (tipo de dados da coluna [DEFAULT expr]
            [, tipo de dados da coluna ]...);


ALTER TABLE tabela
MODIFY      (tipo de dados da coluna [DEFAULT expr]
            [, tipo de dados da coluna ]...);
```

Instrução ALTER TABLE
Após criar suas tabelas, você talvez precise alterar as estruturas da tabela porque omitiu uma coluna ou a definição da coluna precisa ser alterada. Você pode fazer isso usando a instrução ALTER TABLE.

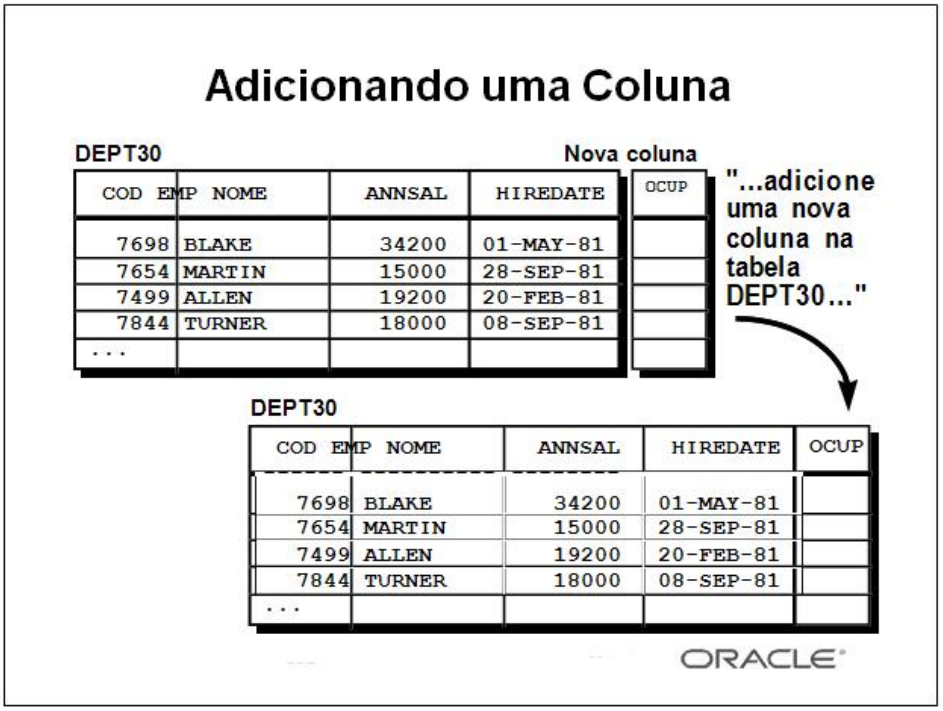
Você pode adicionar colunas a uma tabela usando a instrução ALTER TABLE com a cláusula ADD.

Na sintaxe:

tabela	é o nome da tabela
coluna	é o nome da nova coluna
tipo de dados	é o tipo de dados e o comprimento da nova coluna
DEFAULT expr	especifica o valor default para a nova coluna

Você pode modificar colunas existentes em uma tabela usando a instrução ALTER TABLE com a cláusula MODIFY.

Observação: O slide fornece a sintaxe abreviada para ALTER TABLE. ALTER TABLE será abordada com mais detalhes em uma lição mais adiante.



Adicionando uma Coluna

O gráfico adiciona a coluna OCUP à tabela DEPT30. Observe que a nova coluna torna-se a última coluna na tabela.

Adicionando uma Coluna

- Use a cláusula ADD para adicionar colunas.

```
SQL> ALTER TABLE dept30
  2    ADD    (OCUP VARCHAR2(9));
 Table altered.
```

- A coluna nova torna-se a última coluna.

```
COD_EMP NOME       ANNSAL HIREDATE   OCUP
--------- ---------- --------- --------- ----
7698  BLAKE        34200 01-MAY-81
7654  MARTIN       15000 28-SEP-81
7499  ALLEN        19200 20-FEB-81
7844  TURNER       18000 08-SEP-81
...
6 rows selected.
```

Diretrizes para Adicionar uma Coluna

•Você pode adicionar ou modificar colunas, mas não pode removê-las de uma tabela.

•Você não pode especificar onde a coluna deve aparecer. A coluna nova torna-se a última coluna.

O exemplo no slide adiciona uma coluna chamada OCUP à tabela DEPT30. A coluna OCUP torna-se a última coluna na tabela.

Observação: Se uma tabela já contiver linhas quando uma coluna for adicionada, então a nova coluna será inicialmente nula para todas as linhas.

Modificando uma Coluna

• Você pode alterar um tipo de dados, tamanho e valor default de uma coluna.

```
SQL> ALTER TABLE  dept30
2     MODIFY      (NOME VARCHAR2(15)); Table altered.
```

• Uma alteração no valor default afeta somente as inserções subseqüentes à
tabela.

Modificando uma Coluna

Você pode modificar uma definição de coluna usando a instrução ALTER TABLE com a cláusula MODIFY. A modificação da coluna pode incluir alterações ao tipo de dados, tamanho e valor default da coluna.

Diretrizes

•Aumente a largura ou precisão de uma coluna numérica.

•Diminua a largura de uma coluna se ela contiver somente valores nulos e se a tabela não tiver linhas.

•Altere o tipo de dados se a coluna contiver valores nulos.

•Converta uma coluna CHAR para o tipo de dados VARCHAR2 ou converta uma coluna VARCHAR2 para o tipo de dados CHAR se a coluna contiver valores nulos ou se você não alterar o tamanho.

•Uma alteração no valor default de uma coluna afeta somente as inserções subseqüentes à tabela.

Eliminando uma Coluna

Use a cláusula DROP COLUMN para eliminar colunas que você não precisa mais na
tabela.

```
SQL> ALTER TABLE  dept30
  2   DROP  COLUMN      OCUP ; Table altered.
```

Eliminando uma Coluna

Você pode eliminar uma coluna de uma tabela usando a instrução ALTER TABLE com a cláusula DROP COLUMN. Esse é um recurso disponível a partir do Oracle.

Diretrizes

•A coluna pode ou não conter dados.
•Somente uma coluna pode ser eliminada por vez.
•A tabela deve permanecer com pelo menos uma coluna após ser alterada.
•Depois que a coluna for eliminada, não poderá ser recuperada.

Opção SET UNUSED

• Use a opção SET UNUSED para marcar uma ou mais colunas como não usadas.

• Use a opção DROP UNUSED COLUMNS para remover as colunas marcadas como UNUSED.

```
ALTER TABLE tabela

SET   UNUSED (coluna);

OR

ALTER TABLE tabela
```

```
SET UNUSED COLUMN coluna;


ALTER TABLE tabela
DROP  UNUSED COLUMNS;
```

Opção SET UNUSED

A opção SET UNUSED marca uma ou mais colunas como não usadas para que possam ser eliminadas quando a demanda nos recursos do sistema for menor. Esse recurso está disponível no Oracle. Ao especificar esta cláusula você, na verdade, não remove as colunas de destino de cada linha na tabela (ou seja, não restaura o espaço em disco usado por essas colunas). Por isso, o tempo

de resposta é mais rápido do que seria se você executasse a cláusula DROP. As colunas não usadas são tratadas como se fossem eliminadas, embora os dados da coluna permaneçam nas linhas da tabela. Após uma coluna ter sido marcada como não usada, você não terá acesso a essa coluna .

Uma consulta "SELECT *" não recuperará os dados de colunas não usadas. Além disso, os nomes e tipos de colunas marcadas como não usadas não serão exibidos durante DESCRIBE e você poderá adicionar à tabela uma nova coluna com o mesmo nome que uma coluna não usada.

Opção DROP UNUSED COLUMNS

DROP UNUSED COLUMNS remove da tabela todas as colunas marcadas atualmente como não usadas. Você pode usar essa instrução quando quiser recuperar o espaço em disco extra de colunas não usadas na tabela. Se a tabela não contiver colunas não usadas, a instrução retornará sem erros.

```
SQL> ALTER TABLE  dept30
  2   SET  UNUSED (NOME);
Table altered.

SQL> ALTER TABLE  dept30
  2   DROP  UNUSED COLUMNS;
Table altered.
```

Eliminando uma Tabela

• Todos os dados e estrutura da tabela serão excluídos.

• Todas as transações pendentes sofrerão commit.

• Todos os índices serão eliminados.

• Você não pode fazer roll back desta instrução.

```
SQL> DROP TABLE dept30;
Table dropped.
```

Eliminando uma Tabela

A instrução DROP TABLE remove a definição de uma tabela do Oracle. Quando você elimina uma tabela, o banco de dados perde todos os dados na tabela e todos os índices associados a ela.

Sintaxe

```
DROP TABLE tabela;
```

onde: *tabela* é o nome da tabela

Diretrizes

•Todos os dados são deletados da tabela.
•As views e sinônimos permanecerão, mas serão inválidos.
•Todas as transações pendentes sofrerão commit.
•Somente o criador da tabela ou um usuário com o privilégio DROP ANY TABLE poderá
remover uma tabela.

A instrução DROP TABLE, uma vez executada, é irreversível. O Oracle Server não questiona a ação quando você emite a instrução DROP TABLE. Se você possuir tal tabela ou tiver um privilégio de
nível superior, então a tabela será imediatamente removida. Todas as instruções DDL emitem um
commit, tornando assim a transação permanente.

Alterando o Nome de um Objeto

• Para alterar o nome de uma tabela, view, sequência ou sinônimo, execute a instrução RNOME.

```
SQL> RNOME dept
TO DEPARTAMENTO; Table rNOMEd.
```

• Você deve ser o proprietário do objeto.

Renomeando uma Tabela

As instruções DDL adicionais incluem a instrução RNOME, que é usada para renomear uma tabela, view, sequência ou sinônimo.

Sintaxe

```
RNOME old_name    TO    new_name;
```

onde: *old_name* é o nome antigo da tabela, da view, da sequência ou do sinônimo

new_name é o novo nome da tabela, da view, da sequência ou do sinônimo

Você deve ser o proprietário do objeto que renomear.

Truncando uma Tabela

• A Instrução TRUNCATE TABLE:

– Remove todas as linhas de uma tabela

– Libera o espaço de armazenamento usado por esta tabela

```
SQL> TRUNCATE TABLE
DEPARTAMENTO; Table truncated.
```

• Você não pode fazer roll back da remoção da linha ao usar TRUNCATE.

• Como alternativa, você pode remover as linhas usando a instrução DELETE.

Truncando uma Tabela
Outra instrução DDL é a instrução TRUNCATE TABLE, que é usada para remover todas as linhas de uma tabela e para liberar o espaço de armazenamento usado por essa tabela. Ao usar a instrução TRUNCATE TABLE, você não pode fazer roll back da remoção de linha.

Sintaxe

```
TRUNCATE    TABLE tabela;
```

onde: *tabela* é o nome da tabela

Você deve ser o proprietário da tabela ou ter privilégios de sistema DELETE TABLE para truncar a tabela.
A instrução DELETE também pode remover todas as linhas de uma tabela, mas não libera o espaço de armazenamento.

Adicionando Comentários a uma Tabela

• Você pode adicionar comentários a uma tabela ou coluna usando a instrução COMMENT.

```
SQL> COMMENT ON TABLE EMPREGADO
  2   IS 'FUNCIONARIO Information'; Comment created.
```

• Os comentários podem ser exibidos através das views do dicionário de datas.

– ALL_COL_COMMENTS

– USER_COL_COMMENTS

– ALL_TAB_COMMENTS

– USER_TAB_COMMENTS

Adicionando um Comentário a uma Tabela

Você pode adicionar um comentário de até 2.000 bytes sobre uma coluna, tabela, view ou instantâneo usando a instrução COMMENT. O comentário é armazenado no dicionário de dados e pode ser visto em uma das seguintes views do dicionário de datas na coluna COMMENTS:

•ALL_COL_COMMENTS

•USER_COL_COMMENTS
•ALL_TAB_COMMENTS
•USER_TAB_COMMENTS

Sintaxe

```
COMMENT ON TABLE tabela | COLUMN tabela.coluna
IS 'text';
```

onde: *tabela* é o nome da tabela
coluna é o nome da coluna em uma tabela
texto é o texto do comentário

Você pode eliminar um comentário do banco de dados definindo-o como uma string vazia ('').

```
SQL> COMMENT ON TABLE   EMPREGADO IS '';
```

Sumário

Instrução	Descrição
CREATE TABLE	Cria uma tabela
ALTER TABLE	Modifica as estruturas da tabela
DROP TABLE	Remove as linhas e estrutura da tabela
RNOME	Altera o nome de uma tabela, view, sequência ou sinônimo
TRUNCATE	Remove todas as linhas de uma tabela e libera o espaço de armazenamento
COMMENT	Adiciona comentários a uma tabela ou view

CREATE TABLE
•Cria uma tabela.
•Cria uma tabela baseada em outra tabela usando uma subconsulta.

ALTER TABLE
•Modifica as estruturas da tabela.
•Altera larguras da coluna, tipos de dados da coluna e adiciona colunas.

DROP TABLE
•Remove linhas e estrutura de uma tabela.

•Uma vez executada, não é possível fazer roll back dessa instrução.

RNOME
•Renomeia uma tabela, view, sequência ou sinônimo.

TRUNCATE
•Remove todas as linhas de uma tabela e libera o espaço de armazenamento usado pela tabela.
•A instrução DELETE remove somente linhas.

COMMENT
•Adiciona um comentário a uma tabela ou coluna.
•Consulta o dicionário de dados para ver o comentário.

Capítulo 10

Incluindo Restrições

Objetivos

Depois de completar esta lição, você poderá fazer o seguinte:

- Descrever restrições
- Criar e manter restrições

Objetivo da Lição
Nesta lição, você aprenderá como implementar regras comerciais incluindo restrições de integridade.

O Que São Restrições?

• As restrições impõem regras no nível da tabela.

• As restrições evitam que uma tabela seja deletada se houver dependências.

• Os seguintes tipos de restrição são válidos no Oracle:
– NOT NULL
– UNIQUE
– PRIMARY KEY
– FOREIGN KEY
– CHECK

Restrições

O Oracle Server usa restrições para impedir que dados inválidos sejam digitados nas tabelas. Use restrições para fazer o seguinte:

•Impor regras no nível da tabela sempre que uma linha for inserida, atualizada ou deletada da tabela. A restrição deve ser satisfeita para a operação ser bem-sucedida.

•Impedir que uma tabela seja deletada se houver dependências de outras tabelas.

•Fornecer regras para ferramentas do Oracle, como o Oracle Developer.

Restrições de Integridade de Dados

Restrição	Descrição
NOT NULL	Especifica que esta coluna não pode conter um valor nulo
UNIQUE	Especifica uma coluna ou combinação de colunas cujos valores devem ser exclusivos para todas as linhas na tabela
PRIMARY KEY	Identifica exclusivamente cada linha da tabela
FOREIGN KEY	Estabelece e impõe um relacionamento de chave estrangeira entre a coluna e a coluna da tabela referenciada
CHECK	Especifica uma condição que deve ser verdadeira

Para obter mais informações, consulte o Oracle Server SQL Reference, Release 8, "CONSTRAINT Clause".

Diretrizes sobre Restrições

• Nomeie uma restrição ou o Oracle Server gerará um nome usando o formato SYS_Cn.

• Crie uma restrição:

– No momento em que a tabela for criada

– Depois que a tabela tiver sido criada

• Defina uma restrição no nível da coluna ou da tabela.

• Exiba uma restrição no dicionário de dados.

Diretrizes sobre Restrições

Todas as restrições são armazenadas no dicionário de dados. Será mais fácil fazer referência às restrições se você der a elas um nome significativo. Os nomes de restrições devem seguir as regras padrão para nomeação de objeto. Se você não nomear a restrição, o Oracle gerará um nome com o formato SYS_Cn, onde n é um número inteiro para criar um nome de restrição exclusivo.

As restrições podem ser definidas quando a tabela for criada ou depois de ter sido criada.

Você pode ver as restrições definidas para uma tabela específica olhando na tabela do dicionário de dados USER_CONSTRAINTS.

Definindo Restrições

```
CREATE TABLE [esquema.]tabela
            (tipo de dados da coluna [DEFAULT expr]
            [column_constraint],
            ...
            [table_constraint][,...]);

CREATE TABLE EMPREGADO(
            COD_EMP     NUMBER(4),
```

```
NOME  VARCHAR2(10),
...
COD_DEPTO   NUMBER(2) NOT
NULL, CONSTRAINT
EMPREGADO_COD_EMP_pk
      PRIMARY KEY (COD_EMP));
```

Definindo Restrições

O slide fornece a sintaxe para definir restrições ao criar uma tabela.

Na sintaxe:

esquema	é igual ao nome do proprietário
tabela	é o nome da tabela
DEFAULT expr	especifica um valor default se um valor estiver omitido na instrução
INSERT	
coluna	é o nome da coluna
tipo de dados	é o tipo de dados e o comprimento da coluna
column_constraint	é uma restrição de integridade como parte da definição da coluna
table_constraint	é uma restrição de integridade como parte da definição da tabela

Para obter mais informações, consulte o Oracle Server SQL Reference, Release 8, "CREATE TABLE".

Definindo Restrições

• Nível de restrição da coluna

```
coluna [CONSTRAINT constraint_name] constraint_type,
```

• Nível de restrição da tabela

```
coluna,...
[CONSTRAINT constraint_name] constraint_type
(coluna, ...),
```

Definindo Restrições (continuação)

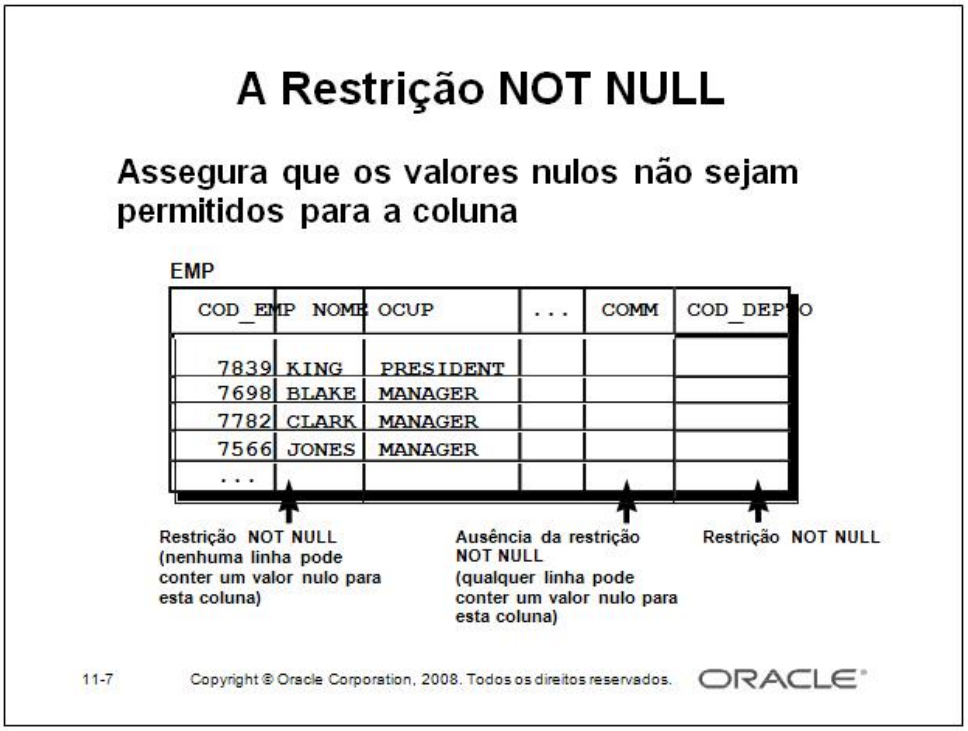
As restrições geralmente são criadas ao mesmo tempo em que é criada a tabela. As restrições podem ser adicionadas a uma tabela depois de sua criação e podem ser desativadas temporariamente.

As restrições podem ser definidas em um ou dois níveis.

Nível da Restrição	Descrição
Column	Faz referência a uma única coluna e é definida dentro de uma especificação para a coluna à qual pertence
Table	Faz referência a uma ou mais colunas e é definida separadamente das definições da coluna na tabela; pode definir quaisquer restrições exceto NOT NULL

Na sintaxe:

constraint_name é o nome da restrição
constraint_type é o tipo da restrição



A Restrição NOT NULL

A restrição NOT NULL assegura que os valores nulos não sejam permitidos na coluna. As colunas sem uma restrição NOT NULL podem conter valores nulos por default.

A Restrição NOT NULL

Definida no nível da coluna

```
SQL> CREATE TABLE EMPREGADO(
   2  empno       NUMBER(4),
   3  ename       VARCHAR2(10) NOT  NULL,
   4  job         VARCHAR2(9),
   5  mgr         NUMBER(4),
   6  hiredate    DATE,
   7  sal         NUMBER(7,2),
   8  comm        NUMBER(7,2),
   9  COD_DEPTO   NUMBER(7,2) NOT NULL);
```

A Restrição NOT NULL (continuação)

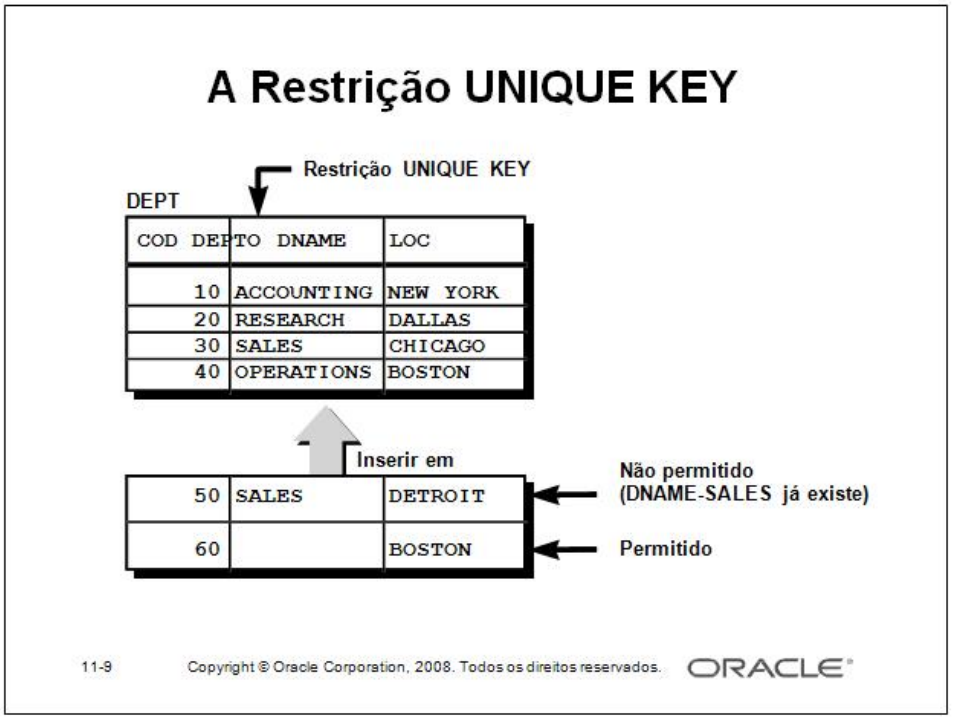
A restrição NOT NULL pode ser especificada somente no nível da coluna, não no da tabela.

O exemplo do slide aplica a restrição NOT NULL às colunas NOME e COD_DEPTO da tabela EMP. Como essas restrições não possuem nome, o Oracle Server criará nomes para elas.

Você pode especificar o nome da restrição ao especificá-la.

```
... COD_DEPTO NUMBER(7,2)

CONSTRAINT EMPREGADO_COD_DEPTO_nn NOT NULL...
```

Observação: Todos os exemplos de restrição descritos nesta lição podem não estar presentes nas tabelas de exemplo fornecidas com o curso. Se desejar, essas restrições podem ser adicionadas às tabelas.



A Restrição UNIQUE KEY

Uma restrição de integridade UNIQUE KEY requer que cada valor em uma coluna ou conjunto de colunas (chave) seja exclusivo — ou seja, duas linhas de uma tabela não podem ter valores duplicados em uma coluna específica ou conjunto de colunas. A coluna (ou conjunto de colunas) incluída na definição da restrição UNIQUE KEY é chamada de chave exclusiva. Se a chave UNIQUE contiver mais de uma coluna, tal grupo de colunas é considerado uma chave exclusiva composta.

As restrições UNIQUE KEY permitem a entrada de valores nulos a menos que você defina as restrições NOT NULL para as mesmas colunas. Na realidade, qualquer número de linhas pode incluir valores nulos para colunas sem restrições NOT NULL porque os valores nulos não são considerados. Um valor nulo em uma coluna (ou em todas as colunas de uma chave UNIQUE composta) sempre satisfaz uma restrição de UNIQUE KEY.

Observação: Por causa do mecanismo de pesquisa por restrições UNIQUE em mais de uma coluna, você não pode ter valores idênticos nas colunas não-nulas de uma restrição de UNIQUE KEY composta parcialmente nula.

A Restrição UNIQUE KEY

Definida no nível da tabela ou da coluna

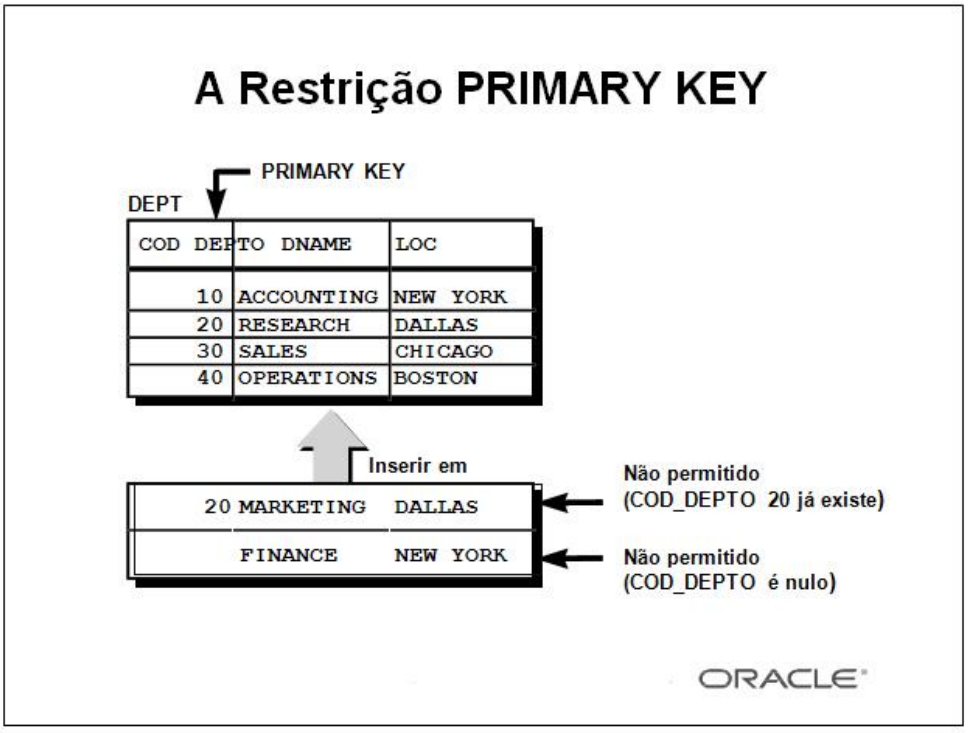
```
SQL> CREATE TABLE dept(
  2   COD_DEPTO   NUMBER(2),
  3   dname VARCHAR2(14),
  4   loc   VARCHAR2(13),
  5   CONSTRAINT  dept_dname_uk UNIQUE(dname));
```

A Restrição UNIQUE KEY (continuação)

Restrições UNIQUE KEY podem ser definidas no nível da coluna ou da tabela. Uma chave exclusiva composta é criada usando-se a definição no nível da tabela.

O exemplo no slide aplica a restrição UNIQUE KEY à coluna DNAME da tabela DEPT. O nome da restrição é DEPT_DNAME_UK.

Observação: O Oracle Server impõe a restrição UNIQUE KEY implicitamente criando um índice exclusivo na chave exclusiva.



A Restrição PRIMARY KEY

Uma restrição PRIMARY KEY cria uma chave primária para a tabela. Somente uma chave primária pode ser criada para cada tabela. A restrição PRIMARY KEY é uma coluna ou conjunto de colunas que identifica exclusivamente cada linha em uma tabela. Essa restrição impõe a exclusividade da coluna ou combinação de colunas e assegura que nenhuma coluna que seja parte da chave primária possa conter um valor nulo.

A Restrição PRIMARY KEY

Definida no nível da tabela ou da coluna

```
SQL> CREATE TABLE dept(
  2    deptno      NUMBER(2),
  3    dname VARCHAR2(14),
  4    loc   VARCHAR2(13),
  5    CONSTRAINT  dept_dname_uk UNIQUE    (dname),
  6    CONSTRAINT dept_COD_DEPTO_pk PRIMARY
KEY(COD_DEPTO));
```

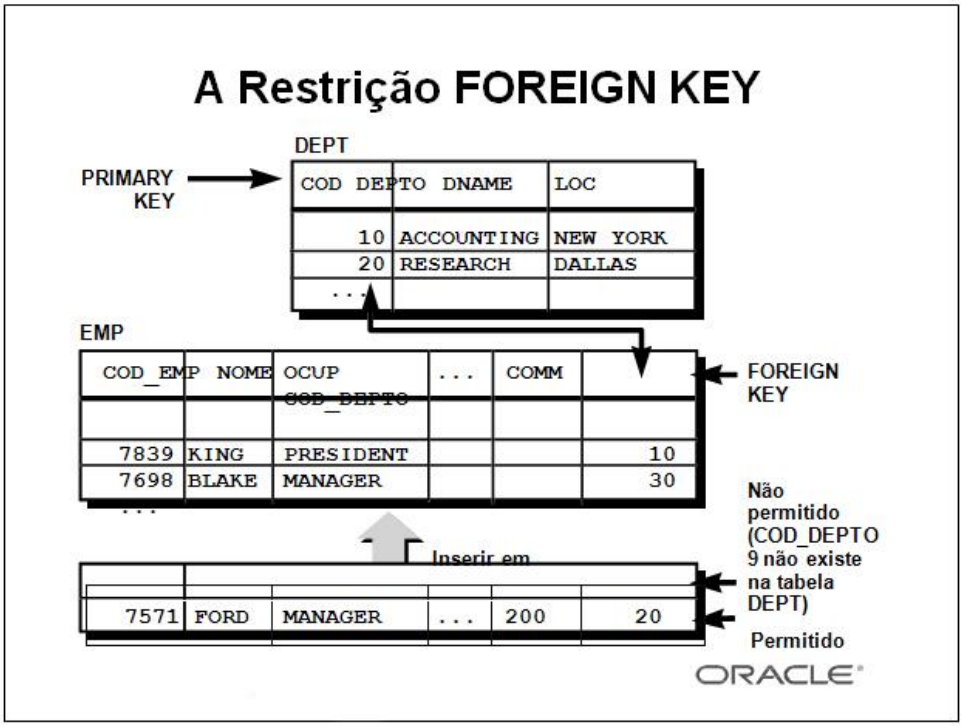
A Restrição PRIMARY KEY (continuação)

As restrições PRIMARY KEY podem ser definidas no nível da coluna ou da tabela. Uma PRIMARY KEY composta é criada usando-se a definição no nível da tabela.

O exemplo no slide define uma restrição PRIMARY KEY na coluna COD_DEPTO da tabela DEPT.

O nome da restrição é DEPT_COD_DEPTO_PK.

Observação: Um índice UNIQUE é automaticamente criado para uma coluna PRIMARY KEY.



A Restrição FOREIGN KEY

FOREIGN KEY, ou uma restrição de integridade referencial, designa uma coluna ou combinação de colunas como a chave estrangeira e estabelece um relacionamento entre a chave primária ou uma

chave exclusiva na mesma tabela ou uma tabela diferente. No exemplo do slide, COD_DEPTO foi definida como a chave estrangeira na tabela EMP (tabela filha ou dependente); ela faz referência à coluna COD_DEPTO da tabela DEPT (tabela mãe ou referenciada).

Um valor de chave estrangeira deve corresponder a um valor existente na tabela mãe ou ser NULL.

As chaves estrangeiras são baseadas nos valores dos dados, sendo puramente lógicas, e não ponteiros físicos.

A Restrição FOREIGN KEY

Definida no nível da tabela ou da coluna

```
SQL> CREATE TABLE EMPREGADO(
  2   COD_EMP     NUMBER(4),
  3   NOME  VARCHAR2(10) NOT  NULL,
  4   OCUP  VARCHAR2(9),
  5   mgr   NUMBER(4),
  6   hiredate    DATE,
  7   sal   NUMBER(7,2),
  8   comm  NUMBER(7,2),
  9   COD_DEPTO   NUMBER(7,2) NOT NULL,
 10   CONSTRAINT EMPREGADO_COD_DEPTO_fk FOREIGN KEY
(COD_DEPTO)
 11   REFERENCES dept (COD_DEPTO));
```

A Restrição FOREIGN KEY (continuação)

As restrições FOREIGN KEY podem ser definidas no nível da restrição da tabela ou coluna. Uma chave estrangeira composta deve ser criada usando-se a definição no nível da tabela.

O exemplo no slide define uma restrição FOREIGN KEY na coluna COD_DEPTO da tabela EMP, usando uma sintaxe no nível da tabela. O nome da restrição é EMP_COD_DEPTO_FK.

A chave estrangeira também pode ser definida no nível da coluna, desde que a restrição seja baseada em uma única coluna. A sintaxe é diferente pois as palavras-chave FOREIGN KEY não aparecem.

Por exemplo:

```
SQL> CREATE TABLE EMPREGADO

(….

COD_DEPTO NUMBER(2) CONSTRAINT EMPREGADO_COD_DEPTO_fk
REFERENCES

                              dept(COD_DEPTO),

….

);
```

Palavras-chave da Restrição FOREIGN KEY

• FOREIGN KEY: Define a coluna na tabela filha no nível de restrição da tabela

• REFERENCES: Identifica a tabela e a coluna na tabela mãe

• ON DELETE CASCADE: Permite exclusão na tabela mãe e das linhas dependentes na
tabela filha

A Restrição FOREIGN KEY (continuação)

A chave estrangeira é definida na tabela filha e a tabela contendo a coluna referenciada é a tabela mãe. A chave estrangeira é definida usando-se uma combinação das seguintes palavras-chave:

•FOREIGN KEY é usada para definir a coluna na tabela filha no nível de restrição da tabela.

•REFERENCES identifica a tabela e a coluna na tabela mãe.

•ON DELETE CASCADE indica que quando a linha na tabela mãe é deletada, as linhas dependentes na tabela filha também serão deletadas.

Sem a opção ON DELETE CASCADE, a linha na tabela mãe não pode ser deletada se for feita referência a ela na tabela filha.

A Restrição CHECK

• Define uma condição que cada linha deve satisfazer

• Expressões que não são permitidas:
– Referências às pseudocolunas CURRVAL, NEXTVAL, LEVEL, e ROWNUM
– Chamadas para as funções SYSDATE, UID, USER e USERENV
– Consultas que se referem a outros valores em outras linhas

```
..., COD_DEPTO    NUMBER(2),
CONSTRAINT
EMPREGADO_COD_DEPTO_ck
      CHECK (COD_DEPTO BETWEEN 10 AND 99),...
```

A Restrição CHECK

A restrição CHECK define uma condição que cada linha deve satisfazer. A condição pode usar as mesmas construções que as condições de consulta, com as seguintes exceções:

•Referências às pseudocolunas CURRVAL, NEXTVAL, LEVEL e ROWNUM

•Chamadas para as funções SYSDATE, UID, USER e USERENV

•Consultas que se referem a outros valores em outras linhas

Uma única coluna pode ter várias restrições CHECK que fazem referência à coluna na sua definição. Não há limite no número de restrições CHECK que você pode definir em uma coluna.

As restrições CHECK podem ser definidas no nível da coluna ou da tabela.

Adicionando uma Restrição

```
ALTER TABLE tabela
ADD [CONSTRAINT restrição] tipo (coluna);
```

• Adicione ou elimine, mas não modifique uma restrição

• Ative ou desative restrições

• Adicione uma restrição NOT NULL usando a cláusula MODIFY

Adicionando uma Restrição

Você pode adicionar uma restrição para tabelas existentes usando uma instrução ALTER TABLE com a cláusula ADD.

Na sintaxe:

tabela	é o nome da tabela
restrição	é o nome da restrição
tipo	é o tipo da restrição
coluna	é o nome da coluna afetada pela restrição

A sintaxe do nome da restrição é opcional, embora recomendada. Se você não nomear suas restrições,o sistema criará nomes de restrição.

Diretrizes
•Você pode adicionar, eliminar, ativar ou desativar uma restrição, mas não pode modificar sua estrutura.
•Você pode adicionar uma restrição NOT NULL a uma coluna existente usando a cláusula
MODIFY da instrução ALTER TABLE.

Observação: Você pode definir uma coluna NOT NULL somente se a tabela não contiver linhas, porque os dados não podem ser especificados para linhas existentes ao mesmo tempo em que a coluna é adicionada.

Adicionando uma Restrição

Adicione uma restrição FOREIGN KEY à tabela EMP indicando que um gerente já deve existir como um funcionário válido na tabela EMP.

```
SQL> ALTER TABLE  EMPREGADO
  2   ADD CONSTRAINT    EMPREGADO_mgr_fk
  3   FOREIGN KEY(mgr) REFERENCES EMPREGADO(COD_EMP); Table
altered.
```

Adicionando uma Restrição (continuação)
O exemplo no slide cria uma restrição FOREIGN KEY na tabela EMP. A restrição assegura que um gerente existe como um funcionário válido na tabela EMP.

Eliminando uma Restrição

• Remova a restrição do gerente da tabela
EMP.

```
SQL> ALTER TABLE  EMPREGADO
  2   DROP CONSTRAINT   EMPREGADO_mgr_fk; Table altered.
```

• Remova a restrição PRIMARY KEY na tabela DEPT e elimine a restrição FOREIGN KEY associada na coluna EMP.COD_DEPTO.

```
SQL> ALTER TABLE  dept
  2   DROP PRIMARY KEY CASCADE; Table altered.
```

Eliminando uma Restrição

Para eliminar uma restrição, você pode identificar o nome da restrição a partir das views do dicionário de dados USER_CONSTRAINTS e USER_CONS_COLUMNS. Em seguida, use a instrução ALTER TABLE com a cláusula DROP. A opção CASCADE da cláusula DROP faz com que quaisquer restrições dependentes sejam eliminadas.

Sintaxe

```
ALTER TABLE tabela
DROP  PRIMARY KEY | UNIQUE (coluna) |
      CONSTRAINT  restrição   [CASCADE];
```

onde: *tabela* é o nome da tabela
coluna é o nome da coluna afetada pela restrição
restrição é o nome da restrição

Quando você elimina uma restrição de integridade, essa restrição não é mais imposta pelo Oracle Server e não fica mais disponível no dicionário de dados.

Desativando Restrições

• Execute a cláusula DISABLE da instrução
ALTER TABLE para desativar uma restrição de integridade.

• Aplique a opção CASCADE para desativar restrições de integridade dependentes.

```
SQL> ALTER TABLE              EMPREGADO
  2   DISABLE CONSTRAINT      EMPREGADO_COD_EMP_pk
CASCADE; Table altered.
```

Desativando uma Restrição

Você pode desativar uma restrição sem eliminá-la ou recriá-la usando a instrução ALTER TABLE com a cláusula DISABLE.

Sintaxe

```
ALTER TABLE tabela
DISABLE     CONSTRAINT restrição [CASCADE];
```

onde: *tabela* é o nome da tabela
restrição é o nome da restrição

Diretrizes
•Você pode usar a cláusula DISABLE nas instruções CREATE TABLE e ALTER TABLE.
•A cláusula CASCADE desativa restrições de integridade dependentes.

Ativando Restrições

• Ative uma restrição de integridade atualmente desativada na definição da tabela usando a cláusula ENABLE.

```
SQL> ALTER TABLE  EMPREGADO
  2   ENABLE CONSTRAINT
      EMPREGADO_COD_EMP_pk; Table altered.
```

• Um índice UNIQUE ou PRIMARY KEY é automaticamente criado se você ativar uma restrição UNIQUE KEY ou PRIMARY KEY.

Ativando uma Restrição
Você pode ativar uma restrição sem eliminá-la ou recriá-la usando a instrução ALTER TABLE com a cláusula ENABLE.

Sintaxe

```
ALTER TABLE tabela
ENABLE      CONSTRAINT restrição;
```

onde: *tabela* é o nome da tabela
restrição é o nome da restrição

Diretrizes
•Se você ativar uma restrição, essa restrição será aplicada a todos os dados na tabela. Todos os dados na tabela devem ajustar-se à restrição.
•Se você ativar uma restrição UNIQUE KEY ou PRIMARY KEY, um índice UNIQUE ou

PRIMARY KEY é automaticamente criado.

•Você pode usar a cláusula ENABLE nas instruções CREATE TABLE e ALTER TABLE.

Restrições em Cascata

• A cláusula CASCADE CONSTRAINTS é usada junto com a cláusula DROP COLUMN.

• A cláusula CASCADE CONSTRAINTS elimina todas as restrições de integridade referenciais que se referem às chaves exclusiva e primária definidas nas colunas eliminadas.

CONSTRAINTS em cascata
Esta instrução ilustra o uso da cláusula CASCADE CONSTRAINTS. Suponha que a tabela test1 seja criada da seguinte forma:

```
SQL> CREATE TABLE test1 (

2     pk NUMBER PRIMARY KEY,

3     fk NUMBER,

4     col1 NUMBER,

5     col2 NUMBER,

6     CONSTRAINT fk_constraint FOREIGN KEY (fk) REFERENCES
test1,

7     CONSTRAINT ck1 CHECK (pk > 0 and col1 > 0),

8     CONSTRAINT ck2 CHECK (col2 > 0));
Um erro será retornado para as seguintes instruções:
SQL> ALTER TABLE test1 DROP (pk); -- pk é uma chave mãe
SQL> ALTER TABLE test1 DROP (col1); -- c1 é referenciada
pela restrição de várias colunas ck1
```

Restrições em Cascata

A cláusula CASCADE CONSTRAINTS
também elimina todas as restrições de várias colunas definidas nas colunas eliminadas.

CONSTRAINTS em cascata (continuação)
Quando as seguintes instruções são submetidas, a coluna ok, a restrição de chave primária, a restrição
de chave estrangeira, a restrição fk e a restrição de verificação ck1 são eliminadas:

```
SQL> ALTER TABLE test1 DROP (pk) CASCADE CONSTRAINTS;
```

Se todas as colunas referenciadas pelas restrições definidas nas colunas eliminadas também forem eliminadas, então CASCADE CONSTRAINTS não é requerida. Por exemplo, pressupondo que nenhuma outra restrição referencial de outras tabelas refere-se à coluna PK, então é válido submeter a seguinte instrução sem a cláusula CASCADE CONSTRAINTS:

```
SQL> ALTER TABLE test1 DROP (pk, fk, col1);
```

Verificando Restrições

Consulte a tabela USER_CONSTRAINTS para ver todos os nomes e definições de restrição.

```
SQL>  SELECT constraint_name, constraint_type,
  2   search_condition
  3   FROM  user_constraints
  4   WHERE table_name = 'EMP';


CONSTRAINT_NAME              C SEARCH_CONDITION
---------------------------- - --------------------------
SYS_C00674                   C COD_EMP IS NOT NULL
SYS_C00675                   C COD_DEPTO IS NOT NULL
EMP_COD_EMP_PK               P
...
```

Verificando Restrições

Após criar uma tabela, você pode confirmar sua existência emitindo um comando DESCRIBE. A única restrição que você pode verificar é a restrição NOT NULL. Para ver todas as restrições em sua tabela, consulte a tabela USER_CONSTRAINTS.

O exemplo no slide exibe todas as restrições da tabela EMP.

Observação: As restrições que não forem nomeadas pelo proprietário da tabela recebem o nome de restrição atribuído pelo sistema. No tipo de restrição, C significa CHECK, P significa PRIMARY KEY, R significa integridade referencial e U significa chave UNIQUE. Observe que a restrição NOT NULL é, na verdade, uma restrição CHECK.

Verificando Colunas Associadas com Restrições

Visualize as colunas associadas aos nomes de restrição na view USER_CONS_COLUMNS.

```
SQL> SELECT constraint_name, column_name
  2   FROM  user_cons_columns
  3   WHERE table_name = 'EMP';

CONSTRAINT_NAME              COLUMN_NAME
-------------------------    --------------------
EMP_COD_DEPTO_FK             COD_DEPTO
EMP_COD_EMP_PK               COD_EMP
EMP_MGR_FK                   MGR
SYS_C00674                   COD_EMP
SYS_C00675                   COD_DEPTO
```

Verificando Restrições (continuação)

Você pode ver os nomes das colunas envolvidas em restrições consultando a view do dicionário de dados USER_CONS_COLUMNS. Essa view é especialmente útil para restrições que usam o nome atribuído pelo sistema.

Sumário

• Crie os seguintes tipos de restrições:

– NOT NULL

– UNIQUE

– PRIMARY KEY

– FOREIGN KEY

– CHECK

• Consulte a tabela USER_CONSTRAINTS para ver todos os nomes e definições de restrição.

Sumário

O Oracle Server usa restrições para impedir que dados inválidos sejam digitados nas tabelas. Os seguintes tipos de restrições são válidos:

- NOT NULL
- UNIQUE
- PRIMARY KEY
- FOREIGN KEY
- CHECK

Você pode consultar a tabela USER_CONSTRAINTS para ver todos os nomes e definições de restrição.

Capítulo 11

Criando Views

Objetivos

Depois de completar esta lição, você poderá fazer o seguinte:

- Descrever uma view
- Criar uma view
- Recuperar dados através de uma view
- Alterar a definição de uma view
- Inserir, atualizar e deletar dados através de uma view
- Eliminar uma view

Objetivo da Lição

Nesta lição, você aprenderá a criar e usar views. Você também aprenderá a consultar o objeto relevante do dicionário de dados para recuperar informações sobre views.

Objetivos

Depois de completar esta lição, você poderá fazer o seguinte:

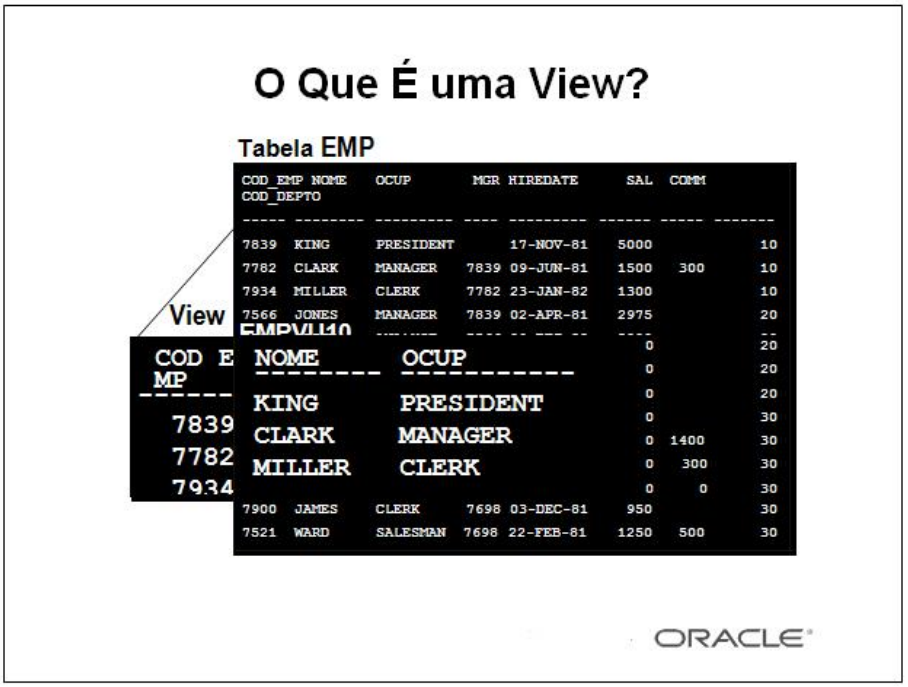
- Descrever uma view em linha
- Executar a Análise "Top-N"

Objetivo da Lição

Nesta lição, você também aprenderá a criar e usar views em linha e executar a análise Top-N usando views em linha.

Objetos de Banco de Dados

Objeto	Descrição
Tabela	Unidade básica de armazenamento; composto de linhas e colunas
View	Representa logicamente subconjuntos de dados de uma ou mais tabelas
Sequência	Gera valores de chave primária
Índice	Melhora o desempenho de algumas consultas
Sinônimo	Nome alternativo para um objeto



O Que É uma View?

Você pode apresentar combinações ou subconjuntos lógicos de dados criando views de tabelas.

Uma view é uma tabela lógica baseada em uma tabela ou outra view. Uma view não contém dados próprios mas é como uma janela através da qual os dados das tabelas podem ser vistos ou alterados.

As tabelas nas quais uma view é baseada são chamadas tabelas-base. A view é armazenada como uma instrução SELECT no dicionário de dados.

Por Que Usar Views?

- Para restringir o acesso a dados
- Para facilitar as consultas complexas
- Para permitir a independência dos dados

• Para apresentar diferentes views dos mesmos dados

Vantagens das Views

• As views restringem o acesso a dados porque uma view pode exibir colunas seletivas a partir da tabela.
• As views permitem que usuários façam consultas simples para recuperar resultados de consultas complexas. Por exemplo, as views permitem que usuários consultem informações de várias tabelas sem saber como criar uma instrução de junção.
• As views permitem a independência de dados para usuários ad hoc e programas aplicativos. Uma view pode ser usada para recuperar os dados de várias tabelas.
• As views fornecem acesso aos dados a grupos de usuários de acordo com seus critérios em particular.

Para obter mais informações, consulte o Oracle Server SQL Reference, Release 8, "CREATE VIEW".

Views Simples e Views Complexas

Recurso	Views Simples	Views Complexas
Número de tabelas	Uma	Uma ou mais
Contém funções	Não	Sim
Contém grupos de dados	Não	Sim
DML através da view	Sim	Nem sempre

Views Simples versus Complexas

Há duas classificações para views: simples e complexa. A diferença básica está relacionada às operações DML (inserir, atualizar e deletar).

- Uma view simples é uma que:
  - Cria dados a partir de somente uma tabela
  - Não contém funções ou grupos de dados
  - Pode executar a DML através da view

- Uma view complexa é uma que:
  - Cria dados a partir de várias tabelas
  - Contém funções ou grupos de dados
  - Nem sempre executa a DML através da view

Criando uma View

• Embuta uma subconsulta na instrução CREATE VIEW.

```
View CREATE [OR REPLACE] [FORCE|NOFORCE] VIEW
[(apelido[, apelido]...)] AS subconsulta
[WITH CHECK OPTION [CONSTRAINT restrição]]
[WITH READ ONLY];
```

• A subconsulta pode conter uma sintaxe SELECT complexa.

• A subconsulta não pode conter uma cláusula ORDER BY.

Criando uma View
Você pode criar uma view incorporando uma subconsulta na instrução CREATE VIEW.

Na sintaxe:

OR REPLACE	recria a view se ela já existir
FORCE	cria a view independentemente das tabelas-base existirem ou não
NOFORCE	cria a view somente se as tabelas-base existirem (default)
view	é o nome da view
apelido	especifica nomes para as expressões selecionadas pela consulta da view (O número de apelidos deve corresponder ao número de expressões selecionadas pela view.)
subconsulta	é uma instrução SELECT completa (Você pode usar apelidos para as colunas na lista SELECT.)
WITH CHECK OPTION	especifica que somente linhas acessíveis à view podem ser inseridas ou atualizadas
restrição	é o nome atribuído à restrição CHECK OPTION
WITH READ ONLY	assegura que as operações DML não possam ser executadas nesta view

Criando uma View

• Crie uma view, EMPVU10, que contenha detalhes sobre os funcionários no departamento 10.

```
SQL> CREATE VIEW  empvu10
  2   AS SELECT   COD_EMP, NOME, OCUP
  3   FROM  EMPREGADO
  4   WHERE COD_DEPTO = 10;
View created.
```

• Descreva a estrutura da view usando o comando DESCRIBE do SQL*Plus.

```
SQL> DESCRIBE empvu10
```

Criando uma View (continuação)

O exemplo no slide cria uma view que contém o número, nome e cargo para todos os funcionários no departamento 10.

Você pode exibir a estrutura da view usando o comando DESCRIBE do SQL*Plus.

```
Name                          Null?    Type
----------------------------- -------- ------------
COD_EMP                       NOT NULL NUMBER(4)
NOME                                   VARCHAR2(10)
OCUP                                   VARCHAR2(9)
```

Diretrizes para criar uma view:

- A subconsulta que define uma view pode conter sintaxe SELECT complexa, incluindo junções, grupos e subconsultas.
- A subconsulta que define a view não pode conter uma cláusula ORDER BY. A cláusula ORDER BY é especificada quando você recupera dados da view.
- Se você não especificar um nome de restrição para uma view criada com CHECK OPTION, o sistema irá atribuir um nome default no formato SYS_Cn.
- Você pode usar a opção OR REPLACE para alterar a definição da view sem eliminá-la e recriá-la ou reconceder-lhe os privilégios de objeto anteriormente concedidos.

Criando uma View

- Crie uma view usando apelidos de coluna na subconsulta.

```
SQL> CREATE VIEW  salvu30
  2   AS SELECT   COD_EMP FUNCIONARIO_NUMBER, NOME NAME,
  3   sal SALARY
  4   FROM  EMPREGADO
  5   WHERE COD_DEPTO = 30;
View created.
```

- Selecione as colunas a partir desta view pelos nomes de apelidos dados.

Criando uma View (continuação)

Você pode controlar os nomes de coluna incluindo apelidos de coluna na subconsulta.

O exemplo no slide cria uma view contendo o número do funcionário (COD_EMP) com o apelido FUNCIONARIO_NUMBER, o

nome (NOME) com o apelido NAME e o salário (sal) com o apelido SALARY para o departamento 30.

Como alternativa, você pode controlar os nomes de coluna incluindo apelidos de coluna na cláusula

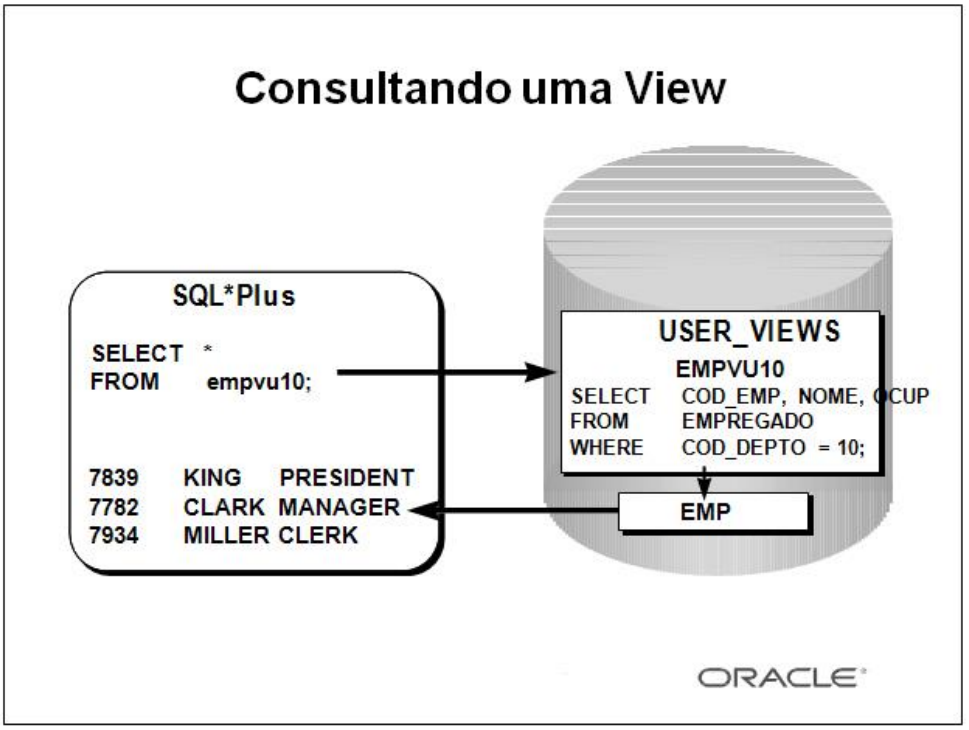
CREATE VIEW.

Recuperando Dados de uma View

```
SQL>  SELECT *
  2   FROM  salvu30;


FUNCIONARIO_NUMBER NAME SALARY
------------------ ---------- ---------
              7698 BLAKE  2850
              7654 MARTIN 1250
              7499 ALLEN  1600
              7844 TURNER 1500
              7900 JAMES  950
              7521 WARD   1250

6 rows selected.
```

Recuperando Dados de uma View

Você pode recuperar os dados de uma view como faria de qualquer tabela. Você pode exibir o conteúdo de toda a view ou somente ver linhas e colunas específicas.



Views no Dicionário de Dados

Depois que a view for criada, você pode consultar a tabela do dicionário de dados chamada USER_VIEWS para ver o nome e a definição da view. O texto da instrução SELECT que constitui a view é armazenado em uma coluna LONG.

Acesso a Dados Usando Views

Quando você acessa dados, usando uma view, o Oracle Server executa as seguintes operações:

1.Recupera a definição da view da tabela do dicionário de dados USER_VIEWS.

2.Verifica os privilégios de acesso para a tabela-base da view.

3.Converte a consulta da view em uma operação equivalente nas tabelas ou tabela-base subjacentes. Em outras palavras, os dados são recuperados a partir da(s) tabela(s)-base, ou uma atualização é feita nela(s).

Modificando uma View

• Modificar a view EMPVU10 usando a cláusula CREATE OR REPLACE VIEW. Adicionar um apelido para cada nome de coluna.

```
SQL> CREATE OR REPLACE VIEW empvu10
  2   (FUNCIONARIO_number, FUNCIONARIO_name, OCUP_title)
  3   AS SELECT   COD_EMP, NOME, OCUP
  4   FROM  EMPREGADO
  5   WHERE COD_DEPTO = 10;
View created.
```

• Os apelidos de coluna na cláusula CREATE VIEW estão listados na mesma ordem que as colunas na subconsulta.

Modificando uma View

A opção OR REPLACE permite que uma view seja criada mesmo que uma já exista com esse nome, substituindo, assim, a versão antiga da view para o seu proprietário. Isso significa que a view poderá ser alterada sem eliminar, recriar e reconceder os privilégios de objeto.

Observação: Ao atribuir apelidos de coluna na cláusula CREATE VIEW, lembre-se de que os apelidos estão listados na mesma ordem que as colunas na subconsulta.

Criando uma View Complexa

Criar uma view complexa que contenha funções de grupo para exibir os valores a partir de duas tabelas.

```
SQL> CREATE VIEW  dept_sum_vu
  2   (name, minsal, maxsal, avgsal)
  3   AS SELECT   d.dname, MIN(e.sal), MAX(e.sal),
  4         AVG(e.sal)
  5   FROM  EMPREGADO e, dept d
  6   WHERE e.COD_DEPTO =     d.COD_DEPTO
  7   GROUP BY    d.dname;

View created.
```

Criando uma View Complexa

O exemplo no slide cria uma view complexa de dos nomes de departamento, salário mínimo e máximo e o salário médio por

departamento. Note que os nomes alternativos foram especificados para a view. Esse é um requisito se uma coluna da view derivar de uma função ou expressão.

Você poderá ver a estrutura da view usando o comando DESCRIBE do SQL*Plus. Exibir o conteúdo da view emitindo uma instrução SELECT.

```
SQL> SELECT *

2     FROM  dept_sum_vu;


NAME       MINSAL  MAXSAL      AVGSAL

-------------- --------- --------- ---------

ACCOUNTING  1300   5000        2916.6667
RESEARCH    800    3000        2175
SALES       950    2850        1566.6667
```

Regras para Executar Operações DML em uma View

• Você poderá executar as operações DML em views simples.

• Você não poderá remover uma linha se a view contiver:

– Funções de grupo

– Uma cláusula GROUP BY

– A palavra-chave DISTINCT

– A palavra-chave da pseudocoluna ROWNUM

Executando Operações DML em uma View

Você poderá executar as operações DML em dados pela view se essas operações seguirem determinadas regras.

Você poderá remover uma linha da view, exceto se ela contiver:

- Funções de grupo
- Uma cláusula GROUP BY
- A palavra-chave da pseudocoluna ROWNUM

Regras para Executar Operações DML em uma View

• Você não poderá modificar dados em uma view se eles contiverem:

– Uma das condições mencionadas no slide anterior
– Colunas definidas por expressões
– A pseudocoluna ROWNUM
• Você não poderá adicionar dados se:

– A view contiver uma das condições mencionadas acima ou no slide anterior

– Houver colunas NOT NULL nas tabelas-base que não forem selecionadas pela view

Executando Operações DML em uma View (continuação)

Você poderá modificar os dados em uma view, exceto se ela contiver uma das condições mencionadas no slide anterior e uma dos seguintes:

•Colunas definidas por expressões — por exemplo, SAL * 12
•A pseudocoluna ROWNUM

Você poderá adicionar dados por uma view, exceto se ela contiver uma das condições acima e houver colunas NOT NULL, sem um valor default, na tabela-base que não forem selecionadas pela view.

Todos os valores necessários devem estar presentes na Lembre-se de que você esta adicionando valores diretamente na tabela subjacente na view.

Para obter mais informações, consulte o 0racle8 Server SQL Reference, Release 8, "CREATE VIEW".

Usando a Cláusula WITH CHECK OPTION

• Você poderá garantir que a DML na view continue no domínio da view usando a cláusula WITH CHECK OPTION.

```
SQL> CREATE OR REPLACE VIEW empvu20
  2   AS SELECT   *
  3   FROM  EMPREGADO
  4   WHERE deptno =    20
  5   WITH CHECK OPTION CONSTRAINT empvu20_ck; View
created.
```

• Qualquer tentativa de alteração do número do departamento para qualquer linha na view falhará porque ela violará a restrição WITH CHECK OPTION.

Usando a Cláusula WITH CHECK OPTION

É possível executar verificações de integridade referencial pelas views. Você deverá também impor restrições no nível de banco de dados. A view poderá ser usada para proteger a integridade dos dados, mas o uso é muito restrito.

A cláusula WITH CHECK OPTION especifica que INSERTS e UPDATES executados pela view não têm permissão de criar linhas que a view não possa selecionar e, portanto, ela permite restrições de integridade e verificações de validação de dados a serem impostas aos dados que estiverem sendo inseridos ou atualizados.

Se houver uma tentativa de executar operações DML em linhas que a view não selecionou, será exibido um erro, com o nome da restrição, se ele tiver sido especificado.

```
SQL> UPDATE empvu20
  2   SET   COD_DEPTO = 10
  3   WHERE COD_EMP = 7788;
update empvu20
*
ERRO na linha 1: (ERROR at line1)
ORA-01402: violação na cláusula where WITH CHECK OPTION na
view
(view WITH CHECK OPTION where-clause violation)
```

Observação: Nenhuma linha é atualizada porque se o número do departamento fosse alterado para 10, a view não seria mais capaz de enxergar o funcionário. Por isso, com a cláusula WITH CHECK OPTION, a view poderá ver apenas funcionários do departamento 20 e não permitirá que o número de departamento para esses funcionários seja alterado na view.

Negando Operações DML

• Você poderá assegurar que nenhuma operação DML ocorra através da adição da opção WITH READ ONLY à definição da sua view.

```
SQL> CREATE OR REPLACE VIEW empvu10
  2   (FUNCIONARIO_number, FUNCIONARIO_name, OCUP_title)
  3   AS SELECT   COD_EMP, NOME, OCUP
  4   FROM  EMPREGADO
  5   WHERE COD_DEPTO = 10
  6   WITH READ ONLY; View created.
```

• Qualquer tentativa de executar uma DML em uma linha na view resultará em erro no
Oracle Server.

Negando Operações DML
Você poderá garantir que nenhuma operação DML ocorra na sua view criando-a com a opção WITH READ ONLY. O exemplo no slide modifica a view EMPVU10 para impedir que as operações DML na view.
Quaisquer tentativas de remover uma linha da view resultará em erro.

```
SQL> DELETE FROM  empvu10
  2   WHERE FUNCIONARIO_number = 7782;
DELETE FROM empvu10
*
ERRO na linha 1 (ERROR at line 1:)
ORA-01752: Não é possível deletar da view sem exatamente
uma chave preservada tabela (Cannot delete from view
without exactly one key- preserved table)
```

Quaisquer tentativas de inserir uma linha ou modificá-la usando a view resultará em erro no Oracle Server -01733: `não é permitida coluna virtual aqui (virtual column not allowed here).`

Removendo uma View

Remover uma view sem perder dados porque uma view está baseada em tabelas
subjacentes no banco de dados.

```
DROP VIEW view;

SQL> DROP VIEW empvu10;
View dropped.
```

Removendo uma View

Você deve usar a instrução DROP VIEW para remover uma view. A instrução remove a definição da view do banco de dados. A eliminação de views não tem efeito nas tabelas nas quais ela é baseada.

As views ou outras aplicações baseadas em views deletadas tornam-se inválidas. Apenas o criador ou usuário com o privilégio DROP ANY VIEW poderá remover uma view.

Na sintaxe:
view é o nome da view

Views Em Linha

• Uma view em linha é uma subconsulta subjacente com um apelido (nome de correlação) que pode ser usado em uma instrução SQL.

• Uma view em linha é similar ao uso de uma subconsulta nomeada na cláusula FROM
da consulta principal.

• Uma view em linha não é um objeto de esquema.

Views Em Linha

Uma view em linha na cláusula FROM de uma instrução SELECT define uma fonte de dados para a instrução SELECT. No exemplo abaixo, a view em linha b retorna os detalhes de todos os números do departamento e o salário máximo para cada departamento da tabela EMP. A cláusula WHERE a.COD_DEPTO

= b.COD_DEPTO AND a.sal < b.maxsal de consulta principal exibe os nomes dos funcionários, salário,

números do departamento e os salários máximos de todos os funcionários que ganham menos que o salário máximo em seus departamentos.

```
SQL> SELECT a.NOME, a.sal, a.COD_DEPTO, b.maxsal
```

```
  2   FROM  EMPREGADO a, (SELECT    COD_DEPTO, max(sal)
maxsal
  3   FROM  EMPREGADO
  4   GROUP BY    COD_DEPTO) b
  5   WHERE a.COD_DEPTO = b.COD_DEPTO
  6   AND   a.sal < b.maxsal;

NOME        SAL         COD_DEPTO   MAXSAL
---------- ---------    ---------   ---------
CLARK       2450        10          5000
MILLER      1300        10          5000
...
TURNER      1500        30          2850
JAMES       950         30          2850

10 rows selected.
```

Análise "Top-N"

• As consultas Top-N pedem os maiores ou menores valores n de uma coluna.

– Quais são os dez produtos mais vendidos?

– Quais são os dez produtos menos vendidos?

• Tanto o conjunto dos maiores quanto dos menores valores são considerados consultas Top-N.

Análise "Top-N"

As consultas Top-N são úteis em cenários onde existe a necessidade de exibir apenas os maiores ou menores registros n de uma tabela baseada em uma condição. Esse conjunto de resultados podem ser para análise posterior. Por exemplo, ao usar as análises Top-N você poderá executar os seguintes

tipos de consultas:

•Os três maiores salários na empresa

•Os quatro mais novos contratados da empresa

•Os dois melhores representantes de vendas que venderam mais produtos

•Os três melhores produtos que mais venderam nos últimos seis meses

Executando a Análise "Top-N"

A estrutura de nível mais elevado de uma consulta de análise Top-N é:

```
SQL> SELECT [column_list], ROWNUM
  2  FROM   (SELECT [column_list] FROM table
  3          ORDER BY Top-N_column)
  4 WHERE   ROWNUM <=   N
```

Executando a Análise "Top-N"

As consultas Top-N usam uma estrutura de consultas aninhadas consistentes com os elementos descritos abaixo:

•Uma subconsulta ou uma view em linha para gerar a lista classificada de dados. A subconsulta ou a view em linha inclui a cláusula ORDER BY para assegurar que a classificação esteja na ordem desejada. Para os resultados recuperando os maiores valores, é necessário um parâmetro DESC.

•Uma consulta externa para limitar o número de linhas no conjunto final de resultados.

A consulta externa inclui os seguintes componentes:

–A pseudocoluna ROWNUM, que atribui um valor seqüencial iniciando com 1 para cada uma das linhas retornadas da subconsulta.

–A cláusula WHERE, que especifica as linhas n a serem retornadas. A cláusula externa

WHERE deve usar um operador < ou <=.

Exemplo de Análise "Top-N"

Para exibir os nomes dos funcionários que recebem os três maiores salários e seus nomes na tabela EMP.

```
SQL> SELECT ROWNUM as RANK, NOME, sal
  2   FROM  (SELECT NOME,sal FROM EMPREGADO
  3          ORDER BY sal DESC)
  4   WHERE ROWNUM <= 3;
```

RANK	NOME	SAL
1	KING	5000
2	SCOTT	3000
3	FORD	3000

Exemplo de Análise "Top-N"

O exemplo no slide descreve como exibir os nomes e os salários dos três funcionários mais bem pagos na tabela EMP. A subconsulta retorna os detalhes de todos os nomes de funcionários e salários da tabela EMP, classificados em ordem decrescente de salários. A cláusula WHERE ROWNUM < 3 da consulta principal garante que apenas os três primeiros registros do conjunto de resultados serão exibidos.

Eis aqui um outro exemplo de análises Top-N que usa uma view em linha. O exemplo abaixo usa a view em linha E para exibir os quatro funcionários mais antigos na empresa.

```
SQL> SELECT ROWNUM as SENIOR,E.NOME, E.hiredate

  2   FROM  (SELECT NOME,hiredate FROM EMPREGADO

  3   ORDER BY hiredate)E

  4   WHERE rownum <= 4;

SENIOR NOME HIREDATE

--------- ---------- ---------

1 SMITH     17-DEC-80

2 ALLEN     20-FEB-81

3 WARD      22-FEB-81

4 JONES     02-APR-81
```

Sumário

• Uma view é criada a partir de dados em outras tabelas ou views.

• Uma view fornece todas as vantagens a seguir:

– Restringe o acesso a bancos de dados

– Simplifica as consultas

– Permite a independência de dados

– Exibe várias views dos mesmos dados

– Pode ser eliminada sem remover os dados subjacentes

O Que É uma View?

Uma view é baseada em uma tabela ou em uma outra view e age como uma janela através da qual os dados nas tabelas podem ser vistos ou alterados. Uma view não contém dados. A definição de view

está armazenada no dicionário de dados. Você poderá ver a definição de view na tabela de dicionário

de dados USER_VIEWS.

Vantagens das Views

•Restringem o acesso a bancos de dados
•Simplificam as consultas
•Permite a independência de dados
•Exibem várias views dos mesmos dados
•Podem ser removidas sem afetar os dados subjacentes

Opções de Views

•Pode ser uma view simples baseada em uma tabela
•Pode ser uma view complexa baseada em mais de uma tabela ou pode conter grupos de funções
•Pode ser substituída se um dos mesmos nomes existir
•Pode conter uma restrição de verificação
•Pode ser somente para leitura

Sumário

• Uma view em linha é uma subconsulta com um nome apelido.

- As análises "Top-N" podem ser executadas usando-se:
  - Subconsulta
  - Consulta externa

Capítulo 12

Outros Objetos do Banco de Dados

Objetivos

Depois de completar esta lição, você poderá fazer o seguinte:

• Descrever alguns objetos do banco de dados e seus usos

• Criar, manter e usar sequências

•. Criar e manter índices

• Criar sinônimos particulares e públicos

Objetivo da Lição

Nesta lição, você aprenderá a criar e manter alguns dos outros objetos do banco de dados que são normalmente utilizados. Esses objetos incluem sequências, índices e sinônimo.

Objetos do Banco de Dados

Objeto	Descrição
Tabela	Unidade básica de armazenamento, composta de linhas e colunas
View	Representa logicamente subconjuntos de dados de uma ou mais tabelas
Sequência	Gera valores de chave primária
Índice	Melhora o desempenho de algumas consultas
Sinônimo	Nome alternativo para um objeto

Objetos do Banco de Dados

Muitas aplicações necessitam usar números exclusivos como valores de chave primária. Você pode elaborar o código na aplicação para tratar essa necessidade ou usar uma sequência para gerar números exclusivos.

Para melhorar o desempenho de algumas consultas, você deve considerar criar um índice. Você também pode usar os índices para aplicar exclusividade a uma coluna ou um conjunto de colunas.

É possível fornecer nomes alternativos para objetos usando sinônimos.

O Que É uma Sequência?

- Gera números exclusivos automaticamente
- É um objeto compartilhável
- É geralmente usada para criar um valor de chave primária
- Substitui o código de aplicação
- Acelera a eficácia do acesso a valores de sequência quando estão em cache na memória

O Que é uma Sequência?

Uma sequência é um objeto do banco de dados criado pelo usuário que pode ser compartilhado por vários usuários para gerar números inteiros exclusivos. Você pode usar as sequências para gerar valores de chave primária automaticamente.

Um uso comum para as sequências é criar um valor de chave primária, que deve ser exclusivo para cada linha. A sequência é gerada e incrementada (ou diminuída) por uma rotina Oracle interna. Esse objeto pode economizar tempo, pois é capaz de reduzir a quantidade de código de aplicação necessária para criar uma rotina de geração de sequências.

Os números de sequência são armazenados e gerados de modo independente das tabelas. Portanto, a mesma sequência pode ser usada para várias tabelas.

A Instrução CREATE SEQUENCE

Defina uma sequência para gerar números sequenciais automaticamente.

```
CREATE SEQUENCE sequence
            [INCREMENT BY n]
            [START WITH n]
            [{MAXVALUE n | NOMAXVALUE}]
            [{MINVALUE n | NOMINVALUE}]
            [{CYCLE | NOCYCLE}]
            [{CACHE n | NOCACHE}];
```

Criando uma Sequência

Gere números de sequência automaticamente usando a instrução CREATE SEQUENCE.

Na sintaxe:

Sequência	é o nome do gerador da sequência
INCREMENT BY *n*	especifica o intervalo entre números de sequência onde *n* é um número inteiro (Se essa cláusula for omitida, a sequência será incrementada em 1.)
START WITH *n*	especifica o primeiro número de sequência a ser gerado

	(Se essa cláusula for omitida, a sequência começará com 1.)
	MAXVALUE *n* especifica o valor máximo que a sequência pode gerar
NOMAXVALUE	especifica um valor máximo de 10^27 para uma sequência crescente e –1 para uma sequência decrescente (Essa é a opção default.)
MINVALUE *n*	especifica o valor de sequência mínimo
NOMINVALUE	especifica um valor mínimo de 1 para uma sequência crescente e –(10^26) para uma sequência decrescente (Essa é a opção default.)

Criando uma Sequência (continuação)

CYCLE | NOCYCLE	especifica que a sequência continue a gerar valores após alcançar seu valor máximo ou mínimo ou não gere valores adicionais (NOCYCLE é a opção default.)
CACHE *n* | NOCACHE	especifica quantos valores o Oracle Server alocará previamente e manterá na memória (Por default, o Oracle Server colocará em cache 20 valores.)

Criando uma Sequência

• Crie uma sequência chamada DEPT_COD_DEPTO para ser usada na chave primária da tabela DEPT.

• Não use a opção CYCLE.

```
SQL> CREATE SEQUENCE dept_COD_DEPTO
  2   INCREMENT BY 1
  3   START WITH 91
  4   MAXVALUE 100
  5   NOCACHE
  6   NOCYCLE; Sequence created.
```

Criando uma Sequência (continuação)

O exemplo no slide cria uma sequência chamada DEPT_COD_DEPTO para ser usada na coluna COD_DEPTO da tabela DEPT. A sequência começa em 91, não permite cache e não permite o ciclo da sequência.

Não use a opção CYCLE se a sequência for utilizada para gerar valores de chave primária, a menos que você tenha um mecanismo confiável que expurgue linhas antigas mais rápido do que o ciclo da sequência.

Para obter mais informações, consulte o Oracle Server SQL Reference, Release 8, "CREATE SEQUENCE".

Confirmando Sequências

• Verifique seus valores de sequência na tabela do dicionário de dados USER_SEQUENCES.

```
SQL> SELECT sequence_name, min_value, max_value,
  2         increment_by, last_number
  3   FROM  user_sequences;
```

• A coluna LAST_NUMBER exibe o próximo número de sequência disponível.

Confirmando sequências

Após você criar sua sequência, ela é documentada no dicionário de dados. Já que uma sequência é um objeto do banco de dados, você pode identificá-la na tabela do dicionário de dados USER_OBJECTS.

Também é possível confirmar as configurações da sequência selecionando a partir da tabela do dicionário de dados USER_SEQUENCES.

```
SEQUENCE_NAME     MIN_VALUE MAX_VALUE INCREMENT_BY
LAST_NUMBER

-------------- ----------- --------- ------------ ---------
--
CUSTID                    1 1.000E+27            1       109
DEPT_COD_DEPTO            1       100            1        91
ORDID                     1 1.000E+27            1       622
PRODID                    1 1.000E+27            1
      200381
```

Pseudocolunas NEXTVAL e CURRVAL

• NEXTVAL retorna o próximo valor de sequência disponível.

Retorna um valor exclusivo sempre que é feita referência a ele, até mesmo por usuários diferentes.

• CURRVAL obtém o valor de sequência atual. NEXTVAL deve ser emitido para essa
sequência antes que CURRVAL contenha
um valor.

Usando uma Sequência

Após criar a sequência, você poderá usá-la para gerar números sequenciais a serem usados nas suas tabelas. Faça referência aos valores de sequência usando as pseudocolunas NEXTVAL e CURRVAL.

Pseudocolunas NEXTVAL e CURRVAL

A pseudocoluna NEXTVAL é usada para extrair números de sequência sucessivos de uma sequência especificada. Você deve qualificar NEXTVAL com o nome da sequência. Quando você faz

referência à SEQUENCE.NEXTVAL, um novo número de sequência é gerado e o número de sequência atual é colocado em CURRVAL.

A pseudocoluna CURRVAL é usada para fazer referência a um número de sequência que o usuário atual acabou de gerar. NEXTVAL deve ser usado para gerar um número de sequência na sessão do usuário atual antes que seja feita referência à CURRVAL. Você deve qualificar CURRVAL com o nome da sequência. Quando é feita referência à sequência. CURRVAL, o último valor retornado ao processo desse usuário é exibido.

Pseudocolunas NEXTVAL e CURRVAL

• NEXTVAL retorna o próximo valor de sequência disponível.

Retorna um valor exclusivo sempre que é feita referência a ele, até mesmo por usuários diferentes.

• CURRVAL obtém o valor de sequência atual. NEXTVAL deve ser emitido para essa
sequência antes que CURRVAL contenha
um valor.

Regras para Usar NEXTVAL e CURRVAL

Você pode usar NEXTVAL e CURRVAL nos seguintes casos:

•Na lista SELECT de uma instrução SELECT que não seja parte de uma subconsulta

•Na lista SELECT de uma subconsulta em uma instrução INSERT

•Na cláusula VALUES de uma instrução INSERT

•Na cláusula SET de uma instrução UPDATE

Você não pode usar NEXTVAL e CURRVAL nos seguintes casos:

•Na lista SELECT de uma view

•Em uma instrução SELECT com a palavra-chave DISTINCT

•Em uma instrução SELECT com as cláusulas GROUP BY, HAVING ou ORDER BY

•Em uma subconsulta de uma instrução SELECT, DELETE ou UPDATE

•Em uma expressão DEFAULT de uma instrução CREATE TABLE ou ALTER TABLE Para obter mais informações, consulte o

Oracle Server SQL Reference, Release 8, seção "Pseudocolumns" e "CREATE SEQUENCE".

Usando uma Sequência

• Insira um novo departamento chamado "MARKETING" em San Diego.

```
SQL> INSERT INTO  dept(COD_DEPTO, dname, loc)
  2   VALUES      (dept_COD_DEPTO.NEXTVAL,
  3   'MARKETING', 'SAN DIEGO');
  1 row created.
```

• Visualize o valor atual para a sequência DEPT_COD_DEPTO.

```
SQL> SELECT dept_COD_DEPTO.CURRVAL
2     FROM  dual;
```

Usando uma Sequência

O exemplo no slide insere um novo departamento na tabela DEPT. Ele usa a sequência

DEPT_COD_DEPTO para gerar um novo número de departamento. Você pode exibir o valor atual da sequência:

```
SQL> SELECT dept_COD_DEPTO.CURRVAL
  2   FROM  dual;

CURRVAL

-------

     91
```

Suponha agora que você deseje admitir funcionários para o novo departamento. A instrução INSERT

que pode ser executada repetidamente para todos os novos funcionários pode incluir o seguinte código:

```
SQL> INSERT INTO EMPREGADO ...
  2   VALUES (EMPREGADO_COD_EMP.NEXTVAL,
dept_COD_DEPTO.CURRVAL,
...);
```

Observação: O exemplo acima pressupõe que uma sequência EMP_COD_EMP já tenha sido criada para gerar um novo número de funcionário.

Usando uma Sequência

• Colocar valores de sequência em cache na memória permite um acesso mais rápido a esses valores.

• Podem ocorrer intervalos em valores de sequência quando:

– Ocorrer um rollback

– O sistema falhar

– Uma sequência for usada em outra tabela

• Visualize a próxima sequência disponível, se tiver sido criada com NOCACHE, consultando
a tabela USER_SEQUENCES.

Colocando Valores de Sequência em Cache

Coloque sequências em cache na memória para permitir um acesso mais rápido aos valores de sequência. O cache é preenchido na primeira referência à sequência. Cada solicitação para o próximo valor de sequência é recuperada na sequência em cache. Após a última sequência ser usada, a próxima solicitação para a sequência baixa outro cache de sequências para a memória.

Cuidado com Gaps na sua Sequência

Apesar de os geradores de sequência emitirem números de sequência sem gaps, essa ação ocorre independentemente de um commit ou rollback. Portanto, se você fizer o rollback de uma instrução que contenha uma sequência, o número será perdido.

Outro evento que pode causar gaps na sequência é uma falha do sistema. Se a sequência colocar valores em cache na memória, esses valores serão perdidos em caso de falha do sistema.

Já que as sequências não estão diretamente ligadas às tabelas, a mesma sequência pode ser usada para várias tabelas. Se isso ocorrer, cada tabela poderá contar gaps nos números sequenciais.

Visualizando o Próximo Valor de Sequência Disponível Sem Incrementá-lo

Se a sequência tiver sido criada com NOCACHE, será possível visualizar o próximo valor de sequência disponível sem incrementá-lo ao consultar a tabela USER_SEQUENCES.

Modificando uma Sequência

Altere o valor de incremento, o valor máximo,
o valor mínimo, a opção de ciclo ou a opção de cache.

```
SQL> ALTER SEQUENCE dept_COD_DEPTO
  2          INCREMENT BY 1
  3          MAXVALUE 999999
  4          NOCACHE
  5          NOCYCLE; Sequence altered.
```

Alterando uma Sequência

Se você alcançar o limite MAXVALUE para sua sequência, nenhum valor adicional da sequência será alocado e ocorrerá um erro indicando que a sequência excede o MAXVALUE. Para continuar a usar a sequência, você pode modificá-la usando a instrução ALTER SEQUENCE.

Sintaxe

```
ALTER SEQUENCE    sequência
[INCREMENT BY n]
[{MAXVALUE n  | NOMAXVALUE}]
[{MINVALUE n  | NOMINVALUE}]
[{CYCLE  | NOCYCLE}]
[{CACHE n  | NOCACHE}];
```

onde: *sequência* é o nome do gerador da sequência

Para obter mais informações, consulte o Oracle Server SQL Reference, Release 8, "ALTER SEQUENCE".

Diretrizes para Modificar uma Sequência

• Você deve ser o proprietário ou possuir o privilégio ALTER para a sequência.

• Somente os números de sequência futuras são afetados.

• A sequência deve ser eliminada e
recriada para reiniciar a sequência em um número diferente.

• Alguma validação é executada.

Diretrizes
• Você deve ser o proprietário ou possuir o privilégio ALTER para a sequência a fim de modificá-la.
• Somente os números de sequência futuros são afetados pela instrução ALTER SEQUENCE.
• A opção START WITH não pode ser alterada usando ALTER SEQUENCE. A sequência deve ser eliminada e recriada para reiniciar a sequência em um número difere.
• Alguma validação é executada. Por exemplo, não é possível impor um novo MAXVALUE menor do que o número de sequência atual.

```
SQL> ALTER SEQUENCE dept_COD_DEPTO
  2   INCREMENT BY 1
  3   MAXVALUE 90
  4   NOCACHE
  5   NOCYCLE;
ALTER SEQUENCE dept_COD_DEPTO
*
ERROR at line 1:
ORA-04009: Não é possível tornar MAXVALUE menor do que o
valor atual (MAXVALUE cannot be made to be less than the
current value)
```

Removendo uma Sequência

• Remova uma sequência do dicionário de dados usando a instrução DROP SEQUENCE.

• Após remover a sequência, você não poderá mais fazer referência à ela.

```
SQL> DROP SEQUENCE
dept_COD_DEPTO; Sequence
dropped.
```

Removendo uma Sequência

Para remover uma sequência do dicionário de dados, use a instrução DROP SEQUENCE. Você deve ser o proprietário da sequência ou possuir o privilégio DROP ANY SEQUENCE para removê-la.

Sintaxe

```
DROP  SEQUENCE sequência;
```

onde: *sequência* é o nome do gerador da sequência

Para obter mais informações, consulte o Oracle Server SQL Reference, Release 8, "DROP SEQUENCE".

O Que É um Índice?

• É um objeto de esquema

• É usado pelo Oracle Server para acelerar a recuperação de linhas usando um ponteiro

• Pode reduzir a E/S do disco usando um método rápido de acesso a caminhos para localizar os dados rapidamente

• É independente da tabela que indexa

• É usado e mantido automaticamente pelo Oracle Server

O Que é um Índice?

Um índice do Oracle Server é um objeto de esquema que pode acelerar a recuperação de linhas usando um ponteiro. Os índices podem ser criados explícita ou automaticamente. Se não houver um índice na coluna, ocorrerá uma análise em toda a tabela.

Um índice fornece acesso direto e rápido às linhas em uma tabela. Seu objetivo é reduzir a necessidade de E/S do disco usando um caminho indexado para localizar dados rapidamente. O índice é usado e mantido automaticamente pelo Oracle Server. Após a criação de um índice, não é necessária nenhuma atividade direta do usuário.

Os índices são lógica e fisicamente independentes da tabela que indexam. Isso significa que eles podem ser criados e eliminados a qualquer momento e não têm nenhum efeito sobre as tabelas-base ou outros índices.

Observação: Quando você elimina uma tabela, os índices correspondentes também são eliminados. Para obter mais informações, consulte o Oracle Server Concepts Manual, Release 8, seção "Schema Objects", tópico "Indexes".

Como os Índices são Criados?

• Automaticamente: Um índice exclusivo é criado automaticamente quando você define uma restrição PRIMARY KEY ou UNIQUE em uma definição de tabela.

• Manualmente: Os usuários podem criar índices não-exclusivos em colunas para acelerar o tempo de acesso às linhas.

Como os Índices são Criados?

É possível criar dois tipos de índices. Um tipo é um índice exclusivo. O Oracle Server cria esse índice automaticamente quando você define que uma coluna de uma tabela tenha uma restrição PRIMARY KEY ou UNIQUE KEY. O nome do índice é o nome dado à restrição.

O outro tipo de índice que um usuário pode criar é um índice não-exclusivo. Por exemplo, você pode criar um índice da coluna FOREIGN KEY para uma junção em uma consulta a fim de aumentar a velocidade de recuperação.

Criando um Índice

• Crie um índice em uma ou mais colunas.

```
CREATE INDEX índice
ON tabela (coluna[, coluna]...);
```

• Aumente a velocidade do acesso de consulta na coluna NOME da tabela EMP.

```
SQL> CREATE  INDEX EMPREGADO_NOME_idx
  2    ON     EMPREGADO(NOME); Index created.
```

Criando um Índice

Crie um índice em uma ou mais colunas emitindo uma instrução CREATE INDEX.

Na sintaxe:

índice é o nome do índice
tabela é o nome da tabela
coluna é o nome da coluna na tabela a ser indexada

Para obter mais informações, consulte o Oracle Server SQL Reference, Release 8, "CREATE INDEX".

Quando Criar um Índice

• Quando a coluna for usada freqüentemente na cláusula WHERE ou em uma condição de junção.

• Quando a coluna contiver uma ampla faixa de valores.

• Quando a coluna contiver um grande número de valores nulos.

• Quando duas ou mais colunas forem usadas juntas com freqüência em uma cláusula WHERE ou em uma condição de junção.

• Quando a tabela for grande e se esperar que a maioria das consultas recupere menos que 2 a 4% das linhas.

Mais Nem Sempre é Melhor

Ter mais índices em uma tabela não significa que as consultas serão aceleradas. Cada operação DML

que seja submetida a commit em uma tabela com índices significa que os índices devem ser atualizados. Quanto mais índices associados a uma tabela você tiver, maior será o esforço feito pelo Oracle Server para atualizar todos os índices após uma DML.

Quando Criar um Índice

- Quando a coluna for usada freqüentemente na cláusula WHERE ou em uma condição de junção.
- Quando a coluna contiver uma ampla faixa de valores.
- Quando a coluna contiver um grande número de valores nulos.
- Quando duas ou mais colunas forem usadas juntas com freqüência em uma cláusula WHERE ou em uma condição de junção.
- Quando a tabela for grande e se esperar que a maioria das consultas recupere menos que 2 a 4% das linhas.

Lembre-se de que, para aplicar exclusividade, você deve definir uma restrição exclusiva na definição da tabela. Em seguida, um índice exclusivo será criado automaticamente.

Quando Não Criar um Índice

- Quando a tabela for pequena.

- Quando as colunas não forem utilizadas com freqüência como uma condição na consulta.

- Quando se esperar que a maioria das consultas recupere mais que 2 a 4% das linhas.

- Quando a tabela for atualizada com freqüência.

Quando Não Criar um Índice

- Quando a tabela for pequena.
- Quando as colunas não forem utilizadas com freqüência como uma condição na consulta.
- Quando se esperar que a maioria das consultas recupere mais que 2 a 4% das linhas.
- Quando a tabela for atualizada com freqüência. Se você tiver um ou mais índices em uma tabela, as instruções DML que acessarem a tabela serão mais lentas.

Confirmando Índices

• A view do dicionário de dados USER_INDEXES contém o nome do índice e sua exclusividade.

• A view USER_IND_COLUMNS contém os nomes do índice, da tabela e da coluna.

```
SQL> SELECT ic.index_name, ic.column_name,
  2         ic.column_position col_pos,ix.uniqueness
  3   FROM  user_indexes ix, user_ind_columns ic
  4   WHERE ic.index_name = ix.index_name
  5   AND   ic.table_name = 'EMP';
```

Confirmando Índices

Confirme a existência de índices na view do dicionário de dados USER_INDEXES. Também é possível checar as colunas envolvidas em um índice consultando a view USER_IND_COLUMNS.

O exemplo no slide exibe todos os índices anteriormente criados, os nomes de coluna afetados e a exclusividade na tabela EMP.

```
INDEX_NAME        COLUMN_NAME       COL_POS UNIQUENESS

----------------- ------------- ------- ----------

EMP_COD_EMP_PK    COD_EMP             1 UNIQUE
EMP_NOME_IDX      NOME                1 NONUNIQUE
```

Observação: A saída foi formatada.

Índices Baseados em Função

• Um índice baseado em função é aquele que se baseia em expressões.

• Uma expressão de índice é elaborada a partir de colunas de tabela, constantes, funções
SQL e funções definidas pelo usuário.

```
SQL> CREATE TABLE test (col1 NUMBER);
```

```
SQL> CREATE INDEX test_index on test(col1,col1+10);

SQL> SELECT col1+10 FROM test;
```

Índice Baseado em Função
Os índices baseados em função definidos com as palavras-chave UPPER(column_name) ou LOWER(column_name) aceitam pesquisas sem a distinção entre maiúsculas e minúsculas. Por exemplo, o índice a seguir:

```
SQL> CREATE INDEX uppercase_idx ON EMPREGADO (UPPER(NOME));
```

Facilita o processamento de consultas como:

```
SQL> SELECT * FROM EMPREGADO WHERE UPPER(NOME) = 'KING';
```

Para garantir que o Oracle use o índice em vez de desempenhar uma análise em toda a tabela, certifique-se de que o valor da função não seja nulo em consultas subseqüentes. Por exemplo, a instrução abaixo certamente usará o índice, mas sem a cláusula WHERE o Oracle executará uma análise em toda a tabela.

```
SQL> SELECT * FROM EMPREGADO
  2   WHERE UPPER (NOME) IS NOT NULL
  3   ORDER BY UPPER (NOME);
```

O Oracle trata os índices com colunas marcadas como DESC como índices baseados em função. As colunas marcadas como DESC são classificadas em ordem decrescente.

Removendo um Índice

• Remova um índice do dicionário de dados.

```
SQL> DROP INDEX index;
```

• Remova o índice EMP_NOME_IDX do dicionário de dados.

```
SQL> DROP INDEX EMPREGADO_NOME_idx; Index dropped.
```

• Para eliminar um índice, você precisa ser o proprietário do índice ou possuir o privilégio

DROP ANY INDEX.

Removendo um Índice

Você não pode modificar índices. Para alterar um índice, você deve eliminá-lo e, em seguida, recriá-lo. Remova uma definição de índice do dicionário de dados emitindo a instrução DROP INDEX. Para eliminar um índice, você precisa ser o proprietário do índice ou possuir o privilégio DROP ANY INDEX.

Na sintaxe:

índice é o nome do índice

Sinônimos

Simplifique o acesso aos objetos criando um sinônimo (outro nome para um objeto).

• Consulte uma tabela de propriedade de outro usuário.

• Abrevie nomes de objeto longos.

```
CREATE [PUBLIC] SYNONYM sinônimo
FOR   objeto;
```

Criando um Sinônimo para um Objeto

Para consultar uma tabela de propriedade de outro usuário, é necessário preceder o nome da tabela com o nome do usuário que a criou, seguido por um ponto. A criação de um sinônimo elimina a necessidade de qualificar o nome do objeto com o esquema e fornece um nome alternativo para uma tabela, view, sequência, um procedimento ou outros objetos. Esse método pode ser especialmente útil com nomes de objeto longos, tais como views.

Na sintaxe:

PUBLIC cria um sinônimo acessível a todos os usuários
sinônimo é o nome do sinônimo a ser criado
objeto identifica o objeto para o qual o sinônimo é criado

Diretrizes

• O objeto não pode estar contido em um pacote.

• Um nome de sinônimo particular deve ser distinto de todos os outros objetos de propriedade do mesmo usuário.

Para obter mais informações, consulte o Oracle Server SQL Reference, Release 8, "CREATE SYNONYM".

Criando e Removendo Sinônimos

• Crie um nome abreviado para a view DEPT_SUM_VU.

```
SQL> CREATE SYNONYM d_sum
  2   FOR   dept_sum_vu; Synonym Created.
```

• Elimine um sinônimo.

```
SQL> DROP SYNONYM d_sum; Synonym dropped.
```

Criando um Sinônimo para um Objeto (continuação)

O exemplo no slide cria um sinônimo para a view DEPT_SUM_VU para referência mais rápida.

A DBA pode criar um sinônimo público acessível a todos os usuários. O exemplo a seguir cria um sinônimo público chamado DEPT para a tabela DEPT de Alice:

```
SQL> CREATE PUBLIC SYNONYM    dept
  2   FOR   alice.dept;
Synonym created.
```

Removendo um Sinônimo

Para eliminar um sinônimo, use a instrução DROP SYNONYM. Somente a DBA pode eliminar um sinônimo público.

```
SQL> DROP SYNONYM dept;
Synonym dropped.
```

Para obter mais informações, consulte o Oracle Server SQL Reference, Release 8, "DROP SYNONYM".