

---

# **Banco de Dados Oracle 10g: Fundamentos de SQL I**

**Volume I • Guia do Aluno**

---

D17108BP10

Produção 1.0

Junho 2004

D39572

**ORACLE®**

## **Autor**

Nancy Greenberg

## **Revisores e Colaboradores Técnicos**

Wayne Abbott  
Christian Bauwens  
Perry Benson  
Brian Boxx  
Zarko Cesljas  
Dairy Chan  
Laszlo Czinkoczki  
Marjolein Dekkers  
Matthew Gregory  
Stefan Grenstad  
Joel Goodman  
Rosita Hanoman  
Sushma Jagannath  
Angelika Krupp  
Christopher Lawless  
Marcelo Manzano  
Isabelle Marchand  
Malika Marghadi  
Valli Pataballa  
Elspeth Payne  
Ligia Jasmin Robayo  
Bryan Roberts  
Helen Robertson  
Lata Shivaprasad  
John Soltani  
Priya Vennapusa  
Ken Woolfe

## **Editor**

Nita K. Brozowski

**Copyright © 2004, Oracle. Todos os direitos reservados.**

Esta documentação contém informações de propriedade da Oracle Corporation. Ela é fornecida sob um contrato de licença que contém restrições quanto ao uso e à divulgação, além de ser protegida pela legislação de direitos autorais. É proibida a engenharia reversa do software. Se esta documentação for distribuída a uma Agência Governamental subordinada ao Departamento de Defesa dos EUA, ela terá direitos restritos e o seguinte aviso deverá ser aplicado:

### **Aviso de Direitos Restritos**

A utilização, a duplicação ou a divulgação pelo governo estará sujeita às restrições impostas a um software comercial e deverão ser aplicadas as leis federais relativas a um software com direitos restritos, como definidos no subparágrafo (c)(1)(ii) de DFARS 252.227-7013, Rights in Technical Data and Computer Software (Direitos sobre Dados Técnicos e Software de Computadores) (outubro de 1988).

Este material, ou parte dele, não poderá ser copiado de qualquer forma ou por qualquer meio sem a prévia permissão expressa por escrito da Oracle Corporation. Qualquer outra cópia constituirá uma violação da legislação de direitos autorais e poderá resultar em indenizações civis e/ou criminais.

Se esta documentação for distribuída a uma Agência Governamental que não pertença ao Departamento de Defesa dos EUA, ela terá "direitos restritos", conforme definido no FAR 52.227-14, Rights in Data-General (Direitos Gerais sobre Dados), incluindo Alternate III (Alternativa III) (junho de 1987).

As informações contidas neste documento estão sujeitas a alterações sem aviso prévio. Se você encontrar algum problema na documentação, envie ao departamento Worldwide Education Services uma descrição de tal problema por escrito. Oracle Corporation, 500 Oracle Parkway, Box SB-6, Redwood Shores, CA 94065 - USA. Distribuidor no Brasil: Oracle do Brasil Sistemas Ltda. Rua José Guerra, 127, São Paulo, SP - 04719-030 - Brasil - CNPJ: 59.456.277/0001-76. A Oracle Corporation não garante que esta documentação esteja isenta de erros.

Oracle e todas as referências a produtos da Oracle são marcas comerciais ou registradas da Oracle Corporation.

Todos os outros nomes de empresas e produtos são usados com o único propósito de identificação e podem ser marcas comerciais dos respectivos proprietários.

# Conteúdo

## Prefácio

### Introdução

Objetivos da Lição	I-2
Metas do Curso	I-3
Oracle10g	I-4
Banco de Dados Oracle 10g	I-6
Oracle Application Server 10g	I-7
Oracle Enterprise Manager 10g Grid Control	I-8
Sistemas de Gerenciamento de Banco de Dados Relacional e Banco de Dados Relacional de Objeto	I-9
Plataforma Oracle para a Internet	I-10
Ciclo de Vida de Desenvolvimento do Sistema	I-11
Armazenamento de Dados em Diferentes Mídias	I-13
Conceito de Banco de Dados Relacional	I-14
Definição de um Banco de Dados Relacional	I-15
Modelos de Dados	I-16
Modelo de Relacionamento entre Entidades	I-17
Convenções de Modelagem de Relacionamento entre Entidades	I-19
Relacionando Várias Tabelas	I-21
Terminologia do Banco de Dados Relacional	I-23
Propriedades do Banco de Dados Relacional	I-25
Comunicando-se com um RDBMS por Meio de SQL	I-26
Sistema de Gerenciamento de Banco de Dados Relacional da Oracle	I-27
Instruções SQL	I-28
Tabelas Usadas no Curso	I-29
Sumário	I-30

### 1 Recuperando Dados com a Instrução SQL **SELECT**

Objetivos	1-2
Recursos de Instruções SQL <b>SELECT</b>	1-3
Instrução <b>SELECT</b> Básica	1-4
Selecionando Todas as Colunas	1-5
Selecionando Colunas Específicas	1-6
Criando Instruções SQL	1-7
Defaults de Cabeçalhos de Colunas	1-8
Expressões Aritméticas	1-9
Usando Operadores Aritméticos	1-10
Precedência de Operadores	1-11
Definindo um Valor Nulo	1-12
Valores Nulos em Expressões Aritméticas	1-13
Definindo um Apelido de Coluna	1-14
Usando Apelidos de Colunas	1-15
Operador de Concatenação	1-16
Strings de Caracteres Literais	1-17
Usando Strings de Caracteres Literais	1-18

Operador de Aspas (q) Alternativo	1-19
Linhas Duplicadas	1-20
Interação entre SQL e <i>iSQL*Plus</i>	1-21
Instruções SQL e Comandos <i>iSQL*Plus</i>	1-22
Visão Geral do <i>iSQL*Plus</i>	1-23
Efetuando Login no <i>iSQL*Plus</i>	1-24
Ambiente <i>iSQL*Plus</i>	1-25
Exibindo a Estrutura de Tabelas	1-26
Interagindo com Arquivos de Script	1-28
Página History do <i>iSQL*Plus</i>	1-32
Definindo Preferências do <i>iSQL*Plus</i>	1-34
Definindo a Preferência de Localização da Saída	1-35
Sumário	1-36
Exercício 1: Visão Geral	1-37

## **2 Restringindo e Classificando Dados**

Objetivos	2-2
Limitando Linhas por Seleção	2-3
Limitando as Linhas Seleccionadas	2-4
Usando a Cláusula <code>WHERE</code>	2-5
Strings de Caracteres e Datas	2-6
Condições de Comparação	2-7
Usando Condições de Comparação	2-8
Usando a Condição <code>BETWEEN</code>	2-9
Usando a Condição <code>IN</code>	2-10
Usando a Condição <code>LIKE</code>	2-11
Usando as Condições <code>NULL</code>	2-13
Condições Lógicas	2-14
Usando o Operador <code>AND</code>	2-15
Usando o Operador <code>OR</code>	2-16
Usando o Operador <code>NOT</code>	2-17
Regras de Precedência	2-18
Usando a Cláusula <code>ORDER BY</code>	2-20
Classificação	2-21
Variáveis de Substituição	2-22
Usando a Variável de Substituição <code>&amp;</code>	2-24
Valores de Caractere e Data com Variáveis de Substituição	2-26
Especificando Nomes de Colunas, Expressões e Texto	2-27
Usando a Variável de Substituição <code>&amp;&amp;</code>	2-28
Usando o Comando <code>DEFINE</code> do <i>iSQL*Plus</i>	2-29
Usando o Comando <code>VERIFY</code>	2-30
Sumário	2-31
Exercício 2: Visão Geral	2-32

### **3 Usando Functions de uma Única Linha para Personalizar a Saída**

Objetivos 3-2

Functions SQL 3-3

Dois Tipos de Functions SQL 3-4

Functions de uma Única Linha 3-5

Functions de Caractere 3-7

Functions de Manipulação de Maiúsculas e Minúsculas 3-9

Usando Functions de Manipulação de Maiúsculas e Minúsculas 3-10

Functions de Manipulação de Caracteres 3-11

Usando as Functions de Manipulação de Caracteres 3-12

Functions de Número 3-13

Usando a Function ROUND 3-14

Usando a Function TRUNC 3-15

Usando a Function MOD 3-16

Trabalhando com Datas 3-17

Aritmética com Datas 3-20

Usando Operadores Aritméticos com Datas 3-21

Functions de Data 3-22

Usando Functions de Data 3-23

Exercício 3: Visão Geral da Parte 1 3-25

Functions de Conversão 3-26

Conversão Implícita de Tipos de Dados 3-27

Conversão Explícita de Tipos de Dados 3-29

Usando a Function TO\_CHAR com Datas 3-32

Elementos do Modelo de Formato de Data 3-33

Usando a Function TO\_CHAR com Datas 3-37

Usando a Function TO\_CHAR com Números 3-38

Usando as Functions TO\_NUMBER e TO\_DATE 3-41

Formato de Data RR 3-43

Exemplo do Formato de Data RR 3-44

Aninhando Functions 3-45

Functions Gerais 3-47

Function NVL 3-48

Usando a Function NVL 3-49

Usando a Function NVL2 3-50

Usando a Function NULLIF 3-51

Usando a Function COALESCE 3-52

Expressões Condicionais 3-54

Expressão CASE 3-55

Usando a Expressão CASE 3-56

Function DECODE 3-57

Usando a Function DECODE 3-58

Sumário 3-60

Exercício 3: Visão Geral da Parte 2 3-61

## 4 Gerando Relatórios de Dados Agregados com as Functions de Grupo

- Objetivos 4-2
- O Que São Functions de Grupo? 4-3
- Tipos de Functions de Grupo 4-4
- Functions de Grupo: Sintaxe 4-5
- Usando as Functions AVG e SUM 4-6
- Usando as Functions MIN e MAX 4-7
- Usando a Function COUNT 4-8
- Usando a Palavra-Chave DISTINCT 4-9
- Functions de Grupo e Valores Nulos 4-10
- Criando Grupos de Dados 4-11
- Criando Grupos de Dados: Sintaxe da Cláusula GROUP BY 4-12
- Usando a Cláusula GROUP BY 4-13
- Agrupando por Mais de Uma Coluna 4-15
- Usando a Cláusula GROUP BY em Várias Colunas 4-16
- Consultas Inválidas Usando Functions de Grupo 4-17
- Restringindo Resultados de Grupos 4-19
- Restringindo Resultados de Grupos com a Cláusula HAVING 4-20
- Usando a Cláusula HAVING 4-21
- Aninhando Functions de Grupo 4-23
- Sumário 4-24
- Exercício 4: Visão Geral 4-25

## 5 Exibindo Dados de Várias Tabelas

- Objetivos 5-2
- Obtendo Dados de Várias Tabelas 5-3
- Tipos de Joins 5-4
- Unindo Tabelas com a Sintaxe SQL:1999 5-5
- Criando Joins Naturais 5-6
- Recuperando Registros com Joins Naturais 5-7
- Criando Joins com a Cláusula USING 5-8
- Unindo Nomes de Colunas 5-9
- Recuperando Registros com a Cláusula USING 5-10
- Qualificando Nomes de Colunas Ambíguos 5-11
- Usando Apelidos de Tabelas 5-12
- Criando Joins com a Cláusula ON 5-13
- Recuperando Registros com a Cláusula ON 5-14
- Auto-Joins Usando a Cláusula ON 5-15
- Aplicando Outras Condições a uma Join 5-17
- Criando Joins Tridimensionais com a Cláusula ON 5-18
- Não-Equijoins 5-19
- Recuperando Registros com Não-Equijoins 5-20
- Joins Externas 5-21
- Joins Internas e Externas 5-22
- LEFT OUTER JOIN 5-23
- RIGHT OUTER JOIN 5-24

FULL OUTER JOIN	5-25
Produtos Cartesianos	5-26
Gerando um Produto Cartesiano	5-27
Criando Joins Cruzadas	5-28
Sumário	5-29
Exercício 5: Visão Geral	5-30

## **6 Usando Subconsultas para Solucionar Consultas**

Objetivos	6-2
Usando uma Subconsulta para Solucionar um Problema	6-3
Sintaxe da Subconsulta	6-4
Usando uma Subconsulta	6-5
Diretrizes de Uso de Subconsultas	6-6
Tipos de Subconsultas	6-7
Subconsultas de uma Única Linha	6-8
Executando Subconsultas de uma Única Linha	6-9
Usando Functions de Grupo em uma Subconsulta	6-10
A Cláusula HAVING com Subconsultas	6-11
O Que Está Errado Nesta Instrução?	6-12
Esta Instrução Retornará Linhas?	6-13
Subconsultas de Várias Linhas	6-14
Usando o Operador ANY em Subconsultas de Várias Linhas	6-15
Usando o Operador ALL em Subconsultas de Várias Linhas	6-16
Valores Nulos em uma Subconsulta	6-17
Sumário	6-19
Exercício 6: Visão Geral	6-20

## **7 Usando os Operadores de Conjunto**

Objetivos	7-2
Operadores de Conjunto	7-3
Tabelas Usadas Nesta Lição	7-4
Operador UNION	7-8
Usando o Operador UNION	7-9
Operador UNION ALL	7-11
Usando o Operador UNION ALL	7-12
Operador INTERSECT	7-13
Usando o Operador INTERSECT	7-14
Operador MINUS	7-15
Diretrizes de Operadores de Conjunto	7-17
O Servidor Oracle e os Operadores de Conjunto	7-18
Correspondência entre Instruções SELECT	7-19
Correspondência entre Instruções SELECT: Exemplo	7-20
Controlando a Ordem das Linhas	7-21
Sumário	7-23
Exercício 7: Visão Geral	7-24

## 8 Manipulando Dados

- Objetivos 8-2
- Data Manipulation Language 8-3
- Adicionando uma Nova Linha a uma Tabela 8-4
- Sintaxe da Instrução `INSERT` 8-5
- Inserindo Novas Linhas 8-6
- Inserindo Linhas com Valores Nulos 8-7
- Inserindo Valores Especiais 8-8
- Inserindo Valores de Data Específicos 8-9
- Criando um Script 8-10
- Copiando Linhas de Outra Tabela 8-11
- Alterando Dados de uma Tabela 8-12
- Sintaxe da Instrução `UPDATE` 8-13
- Atualizando Linhas de uma Tabela 8-14
- Atualizando Duas Colunas com uma Subconsulta 8-15
- Atualizando Linhas com Base em Outra Tabela 8-16
- Removendo uma Linha de uma Tabela 8-17
- Instrução `DELETE` 8-18
- Deletando Linhas de uma Tabela 8-19
- Deletando Linhas com Base em Outra Tabela 8-20
- Instrução `TRUNCATE` 8-21
- Usando uma Subconsulta em uma Instrução `INSERT` 8-22
- Transações de Banco de Dados 8-24
- Vantagens das Instruções `COMMIT` e `ROLLBACK` 8-26
- Controlando Transações 8-27
- Fazendo Rollback de Alterações até um Marcador 8-28
- Processamento de Transação Implícita 8-29
- Estado dos Dados antes de `COMMIT` ou `ROLLBACK` 8-31
- Estado dos Dados após `COMMIT` 8-32
- Submetendo Dados a Commit 8-33
- Estado dos Dados após `ROLLBACK` 8-34
- Rollback no Nível de Instrução 8-36
- Consistência de Leitura 8-37
- Implementação da Consistência de Leitura 8-38
- Sumário 8-39
- Exercício 8: Visão Geral 8-40

## 9 Usando Instruções DDL para Criar e Gerenciar Tabelas

- Objetivos 9-2
- Objetos de Banco de Dados 9-3
- Regras de Nomeação 9-4
- Instrução `CREATE TABLE` 9-5
- Fazendo Referência a Tabelas de Outro Usuário 9-6
- Opção `DEFAULT` 9-7
- Criando Tabelas 9-8
- Tipos de Dados 9-9
- Tipos de Dados de Data/Horário 9-11



Incluindo Constraints	9-17
Diretrizes de Constraints	9-18
Definindo Constraints	9-19
Constraint NOT NULL	9-21
Constraint UNIQUE	9-22
Constraint PRIMARY KEY	9-24
Constraint FOREIGN KEY	9-25
Constraint FOREIGN KEY: Palavras-chave	9-27
Constraint CHECK	9-28
CREATE TABLE: Exemplo	9-29
Violando Constraints	9-30
Criando uma Tabela com uma Subconsulta	9-32
Instrução ALTER TABLE	9-34
Eliminando uma Tabela	9-35
Sumário	9-36
Exercício 9: Visão Geral	9-37

## **10 Criando Outros Objetos de Esquema**

Objetivos	10-2
Objetos de Banco de Dados	10-3
O Que É uma View?	10-4
Vantagens das Views	10-5
Views Simples e Complexas	10-6
Criando uma View	10-7
Recuperando Dados de uma View	10-10
Modificando uma View	10-11
Criando uma View Complexa	10-12
Regras para Executar Operações DML em uma View	10-13
Usando a Cláusula WITH CHECK OPTION	10-16
Negando Operações DML	10-17
Removendo uma View	10-19
Exercício 10: Visão Geral da Parte 1	10-20
Seqüências	10-21
Instrução CREATE SEQUENCE: Sintaxe	10-23
Criando uma Seqüência	10-24
Pseudocolunas NEXTVAL e CURRVAL	10-25
Usando uma Seqüência	10-27
Armazenando Valores de Seqüência em Cache	10-28
Modificando uma Seqüência	10-29
Diretrizes para Modificar uma Seqüência	10-30
Índices	10-31
Como Criar Índices?	10-33
Criando um Índice	10-34
Diretrizes para Criar Índices	10-35
Removendo um Índice	10-36

Sinônimos 10-37  
 Criando e Removendo Sinônimos 10-39  
 Sumário 10-40  
 Exercício 10: Visão Geral da Parte 2 10-41

## **11 Gerenciando Objetos com Views de Dicionário de Dados**

Objetivos 11-2  
 O Dicionário de Dados 11-3  
 Estrutura do Dicionário de Dados 11-4  
 Como Usar as Views de Dicionário 11-6  
 View USER\_OBJECTS 11-7  
 Informações sobre Tabelas 11-9  
 Informações sobre Colunas 11-10  
 Informações sobre Constraints 11-12  
 Informações sobre Views 11-15  
 Informações sobre Seqüências 11-16  
 Informações sobre Sinônimos 11-18  
 Adicionando Comentários a uma Tabela 11-19  
 Sumário 11-20  
 Exercício 11: Visão Geral 11-21

### **A Soluções dos Exercícios**

### **B Dados e Descrições de Tabelas**

### **C Sintaxe de Join Oracle**

### **D Usando o SQL\*Plus**

#### **Índice**

#### **Exercícios Adicionais**

#### **Exercícios Adicionais: Dados e Descrições de Tabelas**

#### **Exercícios Adicionais: Soluções**

---

# Prefácio

---



## **Perfil**

### **Antes de Iniciar o Curso**

Antes de iniciar este curso, você deverá ser capaz de usar uma GUI (interface gráfica do usuário). Como pré-requisito, é necessário que o aluno tenha familiaridade com os conceitos e as técnicas de processamento de dados.

### **Organização deste Curso**

O curso *Banco de Dados Oracle 10g: Fundamentos de SQL I* é orientado por instrutor e inclui palestras e exercícios práticos. As sessões de demonstração on-line e os exercícios reforçam as técnicas e os conceitos apresentados.

## Publicações Relacionadas

### Publicações Oracle

<b>Título</b>	<b>Part Number (Número do Componente)</b>
<i>Oracle® Database Reference 10g Release 1 (10.1)</i>	B10755-01
<i>Oracle® Database SQL Reference 10g Release 1 (10.1)</i>	B10759-01
<i>Oracle® Database Concepts 10g Release 1 (10.1)</i>	B10743-01
<i>Oracle® Database Application Developer's Guide - Fundamentals 10g Release 1 (10.1)</i>	B10795-01
<i>SQL*Plus® User's Guide and Reference</i>	B12170-01

### Publicações Adicionais

- Boletins de releases de sistemas
- Guias de instalação e do usuário
- Arquivos *readme*
- Artigos do IOUG (International Oracle User's Group)
- *Oracle Magazine*

## Convenções Tipográficas

A seguir, são fornecidas duas listas de convenções tipográficas usadas especificamente no texto ou no código.

### Convenções Tipográficas do Texto

Convenção	Objeto ou Termo	Exemplo
Letras maiúsculas	Comandos, functions, nomes de colunas, nomes de tabelas, objetos PL/SQL, esquemas	Use o comando <code>SELECT</code> para exibir informações armazenadas na coluna <code>LAST_NAME</code> da tabela <code>EMPLOYEES</code> .
Letras minúsculas, itálico	Nomes de arquivos, variáveis de sintaxe, nomes de usuários, senhas	<b>em que:</b> <i>role</i> é o nome da atribuição a ser criada.
Inicial maiúscula	Trigger e nomes de botão	Designe um trigger When-Validate-Item para o bloco ORD. Escolha Cancel.
Itálico	Livros, nomes de cursos e manuais, palavras ou frases enfatizadas	Para obter mais informações sobre o assunto, consulte o <i>Oracle SQL Reference Manual</i> Não salve as alterações no banco de dados.
Aspas	Títulos de módulos da lição mencionados em um curso	Este assunto é abordado na Lição 3, “Trabalhando com Objetos”.

## Convenções Tipográficas (continuação)

### Convenções Tipográficas do Código

Convenção	Objeto ou Termo	Exemplo
Letras maiúsculas	Comandos, functions	<code>SELECT employee_id FROM employees;</code>
Letras minúsculas, itálico	Variáveis de sintaxe	<code>CREATE ROLE role;</code>
Inicial maiúscula	Triggers de forms	<code>Form module: ORD Trigger level: S_ITEM.QUANTITY item Trigger name: When-Validate-Item . . .</code>
Letras minúsculas	Nomes de colunas, nomes de tabelas, nomes de arquivos, objetos PL/SQL	<code>. . . OG_ACTIVATE_LAYER (OG_GET_LAYER ('prod_pie_layer')) . . . SELECT last_name FROM employees;</code>
Negrito	Texto a ser especificado por um usuário	<code>CREATE USER scott IDENTIFIED BY tiger;</code>



# I

## Introdução

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

# Objetivos da Lição

**Ao concluir esta lição, você será capaz de:**

- **Listar os recursos do Oracle10g**
- **Discutir os aspectos teóricos e físicos de um banco de dados relacional**
- **Descrever a implementação Oracle do RDBMS e do ORDBMS**
- **Compreender as metas do curso**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Objetivos

Nesta lição, você compreenderá o RDBMS (Relational Database Management System) e o ORDBMS (Object Relational Database Management System). Também serão apresentadas informações sobre:

- Instruções SQL específicas do Oracle
- O *iSQL\**Plus, que é um ambiente usado para executar instruções SQL e para fins de formatação e geração de relatórios

# Metas do Curso

**Após concluir este curso, você será capaz de:**

- **Identificar os principais componentes estruturais do Banco de Dados Oracle 10g**
- **Recuperar dados contidos em linhas e colunas de tabelas com a instrução `SELECT`**
- **Criar relatórios de dados classificados e restritos**
- **Utilizar functions SQL para gerar e recuperar dados personalizados**
- **Executar instruções DML (data manipulation language) para atualizar dados no Banco de Dados Oracle 10g**
- **Obter metadados por meio de consultas às views de dicionário**

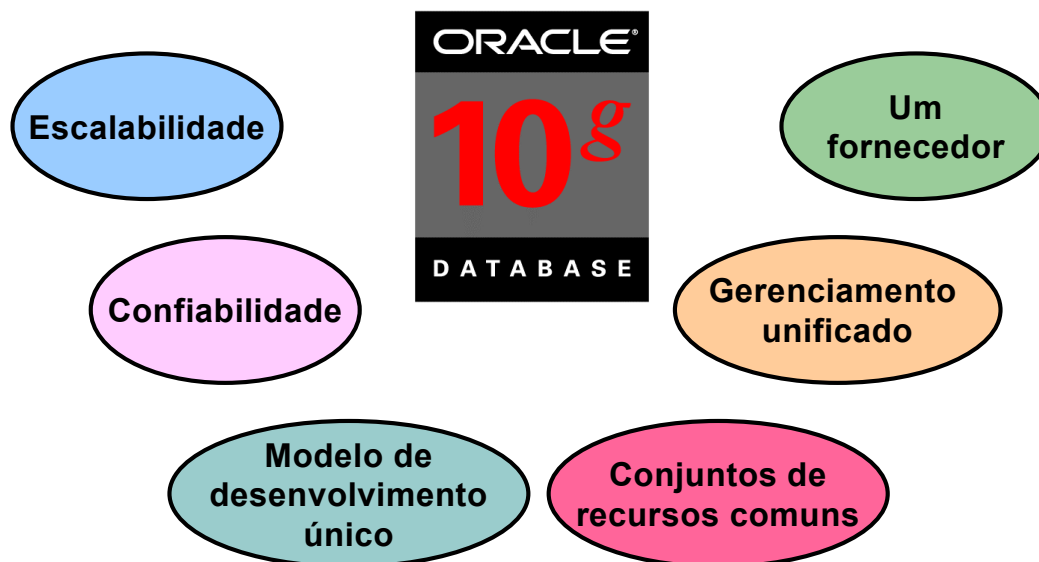
ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Metas do Curso

Este curso apresenta a tecnologia de banco de dados Oracle 10g. No curso, você aprenderá os conceitos básicos de bancos de dados relacionais e a avançada linguagem de programação SQL. O curso fornece as habilidades essenciais em SQL que permitem criar consultas em uma ou mais tabelas, manipular dados em tabelas, criar objetos de banco de dados e consultar metadados.

# Oracle10g



ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Recursos do Oracle10g

A release Oracle10g oferece uma infra-estrutura completa e de alto desempenho que inclui:

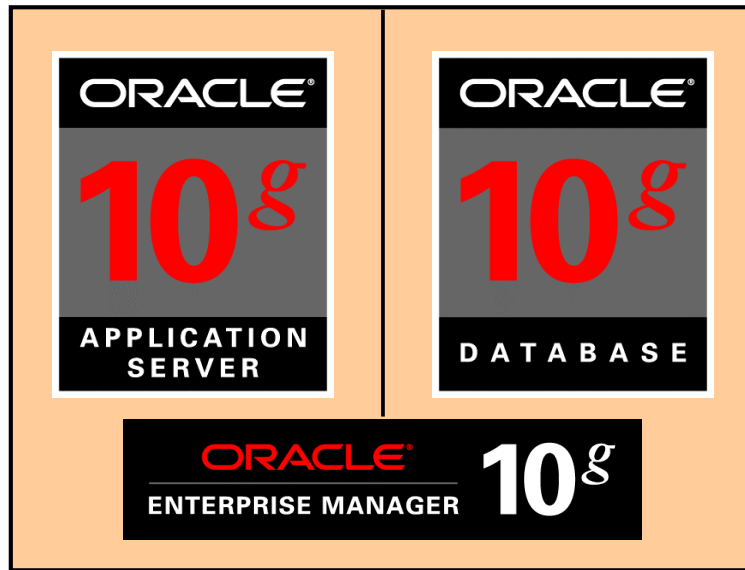
- Escalabilidade de departamentos até sites de e-business da empresa
- Arquitetura segura, disponível, confiável e avançada
- Um modelo de desenvolvimento; opções de disponibilização fácil
- Aproveitamento do conjunto de recursos atuais de uma organização por meio da plataforma Oracle (incluindo SQL, PL/SQL, Java e XML)
- Uma interface de gerenciamento para todas as aplicações
- Tecnologias padrão do setor; sem bloqueios proprietários

Além de oferecer as vantagens relacionadas acima, a release Oracle10g contém o banco de dados para a grade. A computação em grade pode diminuir drasticamente o custo de computação, aumentar a disponibilidade dos recursos de computação e proporcionar mais produtividade e qualidade.

A idéia básica da computação em grade é a noção de computação como um serviço público, em analogia à rede de energia elétrica ou à rede telefônica. Como cliente da grade, você não se importa com o local onde os dados são mantidos ou onde é realizada a computação. Você deseja que a computação seja concluída e que as informações sejam fornecidas quando especificado. No lado servidor, a grade refere-se a virtualização e provisionamento. Você agrupa todos os recursos, provisiona-os dinamicamente com base nas necessidades da sua empresa e, como consequência, obtém maior eficiência na utilização desses recursos.

Development Program (WDP) eKit materials are provided for WDP in-class use only. Copying eKit materials is strictly prohibited and is in violation of Oracle copyright. All WDP students must receive eKit code marks with their name and e-mail. Contact OracleWDP\_ww@oracle.com if you have not received your personalized eKit.

# Oracle10g



ORACLE

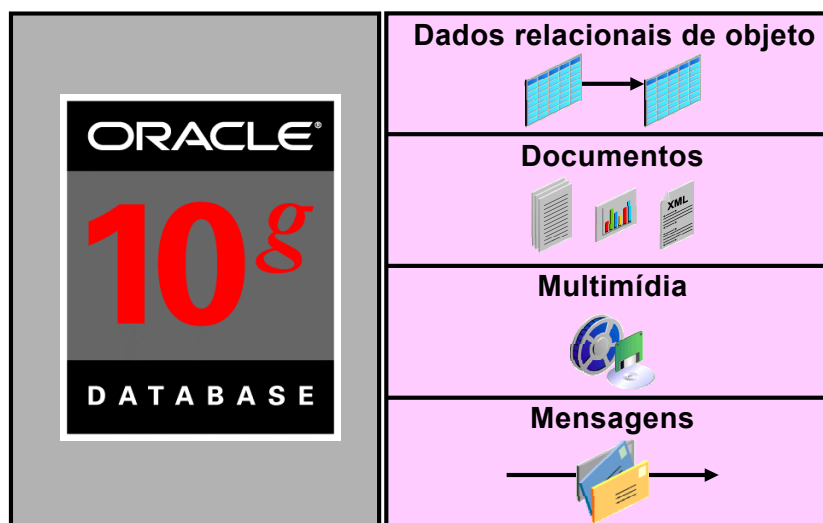
Copyright © 2004, Oracle. Todos os direitos reservados.

## Oracle10g

Estes são os três produtos da infra-estrutura em grade da release Oracle10g:

- Oracle Database 10g
- Oracle Application Server 10g
- Oracle Enterprise Manager 10g Grid Control

# Oracle Database 10g



ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Oracle Database 10g

O Oracle Database 10g foi projetado para armazenar e gerenciar informações empresariais. Além de eliminar custos de gerenciamento, ele oferece um serviço de alta qualidade. A redução dos requisitos de configuração e gerenciamento associada ao ajuste automático de SQL diminuirá drasticamente o custo de manutenção do ambiente.

O Oracle Database 10g é um dos produtos da infra-estrutura em grade da release Oracle 10g. A computação em grade está relacionada ao conceito de computação como um serviço público. Os clientes não precisam saber onde os dados são mantidos nem em qual computador estão armazenados. Basta ser capaz de solicitar informações ou a computação dos dados e recebê-los como desejado.

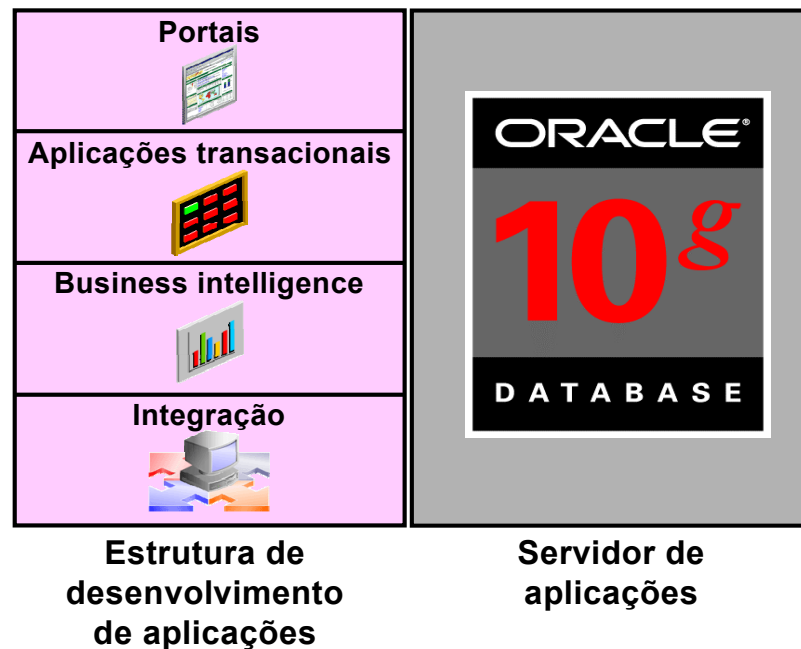
O Oracle Database 10g gerencia todos os seus dados. Não são apenas os dados relacionais de objeto que o banco de dados de uma empresa deve gerenciar. Ele também pode gerenciar dados não estruturados como:

- Planilhas
- Documentos do Word
- Apresentações do PowerPoint
- XML
- Tipos de dados multimídia, como MP3, elementos gráficos, vídeo e outros

Os dados nem precisam estar no banco de dados. O Oracle Database 10g contém serviços que permitem armazenar metadados sobre as informações mantidas em sistemas de arquivos. Você pode usar o servidor de banco de dados para gerenciar e fornecer informações onde quer que ele esteja localizado.

Development Program (WDP) eKit materials are provided for WDP in-class use only. Copying eKit materials is strictly prohibited and is in violation of Oracle copyright. All WDP eKit materials must remain in the eKit container with their name and e-mail. Contact OracleWDP\_ww@oracle.com if you have not received your personalized eKit.

# Oracle Application Server 10g



Copyright © 2004, Oracle. Todos os direitos reservados.

ORACLE

## Oracle Application Server 10g

O Oracle Application Server 10g fornece uma plataforma de infra-estrutura completa para o desenvolvimento e a disponibilização de aplicações empresariais. Ele integra várias funções, incluindo um ambiente de runtime de serviços Web e J2EE, um portal empresarial, um broker de integração empresarial, business intelligence, armazenamento em cache na Web e serviços de gerenciamento de identidades.

O Oracle Application Server 10g contém novos recursos de computação em grade, que se baseiam no sucesso do Oracle9i Application Server, com centenas de clientes que executam aplicações empresariais de produção.

O Oracle Application Server 10g é o único servidor de aplicações que contém serviços para as diversas aplicações de servidor que você executará, incluindo:

- Portais ou Web sites
- Aplicações transacionais Java
- Aplicações de business intelligence

Ele também possibilita a integração entre usuários, aplicações e dados em toda a organização.

# Oracle Enterprise Manager 10g Grid Control

- **Provisionamento de software**
- **Monitoramento no nível de serviços de aplicações**



ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

## Oracle Enterprise Manager 10g Grid Control

O Oracle Enterprise Manager 10g Grid Control é a console de gerenciamento completa, integrada e central, bem como a estrutura subjacente que automatiza as tarefas administrativas nos conjuntos de sistemas em um ambiente de grade. Com ele, é possível agrupar vários nós de hardware, bancos de dados, servidores de aplicações e outros destinos em entidades lógicas únicas. O Grid Control permite o escalonamento com uma grade em crescimento por meio da execução de jobs, da imposição de políticas padrão, do monitoramento do desempenho e da automação de várias outras tarefas em um grupo de destinos, e não em diversos sistemas individualmente.

### Provisionamento de Software

Com o Grid Control, o Oracle 10g automatiza a instalação, a configuração e a clonagem do Application Server 10g e do Database 10g em vários nós. O Oracle Enterprise Manager contém uma estrutura comum para provisionamento e gerenciamento de software, permitindo aos administradores criar, configurar, disponibilizar e utilizar novos servidores com novas instâncias do servidor de aplicações e do banco de dados conforme necessário.

### Monitoramento no Nível de Serviços de Aplicações

Assim como um usuário, o Oracle Grid Control avalia a disponibilidade e o desempenho da infra-estrutura de grade como um todo, e não como unidades de armazenamento, caixas de processamento, bancos de dados e servidores de aplicações isolados.



# Sistemas de Gerenciamento de Banco de Dados Relacional e Banco de Dados Relacional de Objeto

- **Modelo relacional e modelo relacional de objeto**
- **Tipos de dados e objetos definidos pelo usuário**
- **Compatibilidade integral com o banco de dados relacional**
- **Suporte a objetos grandes e multimídia**
- **Recursos de servidor de banco de dados de alta qualidade**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Sobre o Servidor Oracle

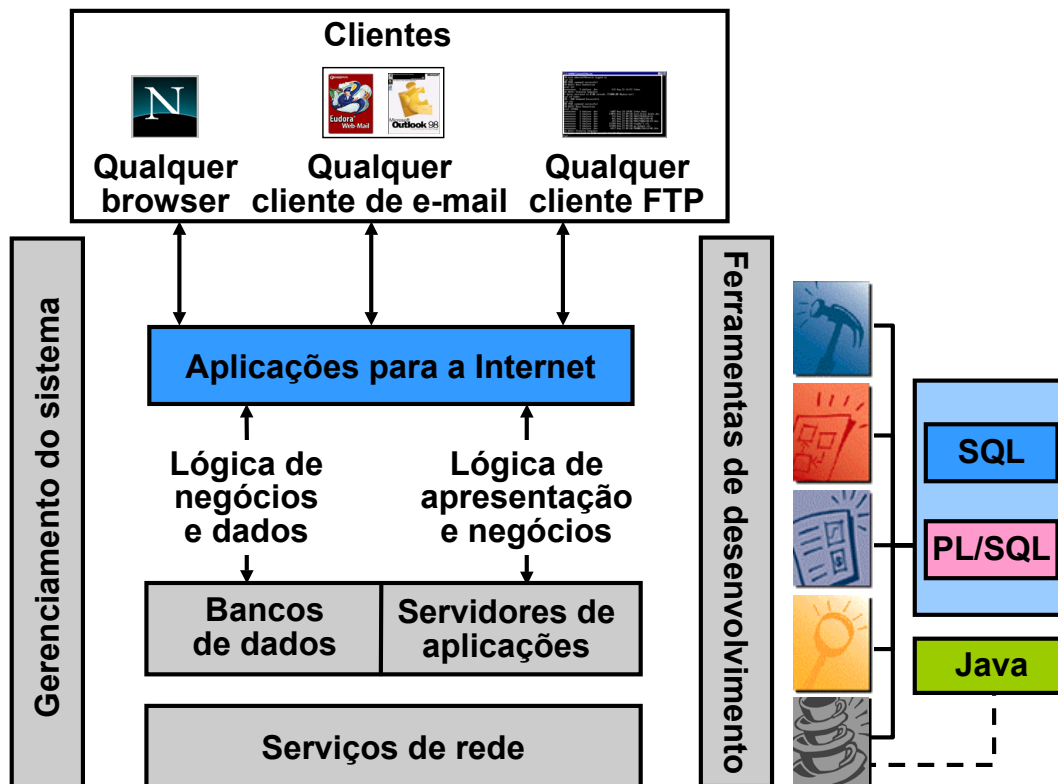
O servidor Oracle suporta modelos relacionais e modelos relacionais de objeto.

O servidor estende os recursos de modelagem de dados para suportar um modelo de banco de dados relacional de objeto que inclui programação orientada a objeto, tipos de dados complexos, objetos de negócios complexos e compatibilidade integral com o mundo relacional.

Ele contém vários recursos para proporcionar melhor desempenho e funcionalidade de aplicações OLTP (On-Line Transaction Processing), como o compartilhamento mais eficiente de estruturas de dados durante o runtime, caches de buffer maiores e constraints adiáveis. As aplicações de data warehouse são beneficiadas por melhorias como a execução paralela de operações de inserção, atualização e deleção; o particionamento; e a otimização de consultas em paralelo. Operando na estrutura da NCA (Network Computing Architecture), o modelo Oracle suporta aplicações cliente/servidor e aplicações baseadas na Web distribuídas e com várias camadas.

Para obter mais informações sobre o modelo relacional e o modelo relacional de objeto, consulte o manual *Database Concepts*.

# Plataforma Oracle para a Internet



ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Plataforma Oracle para a Internet

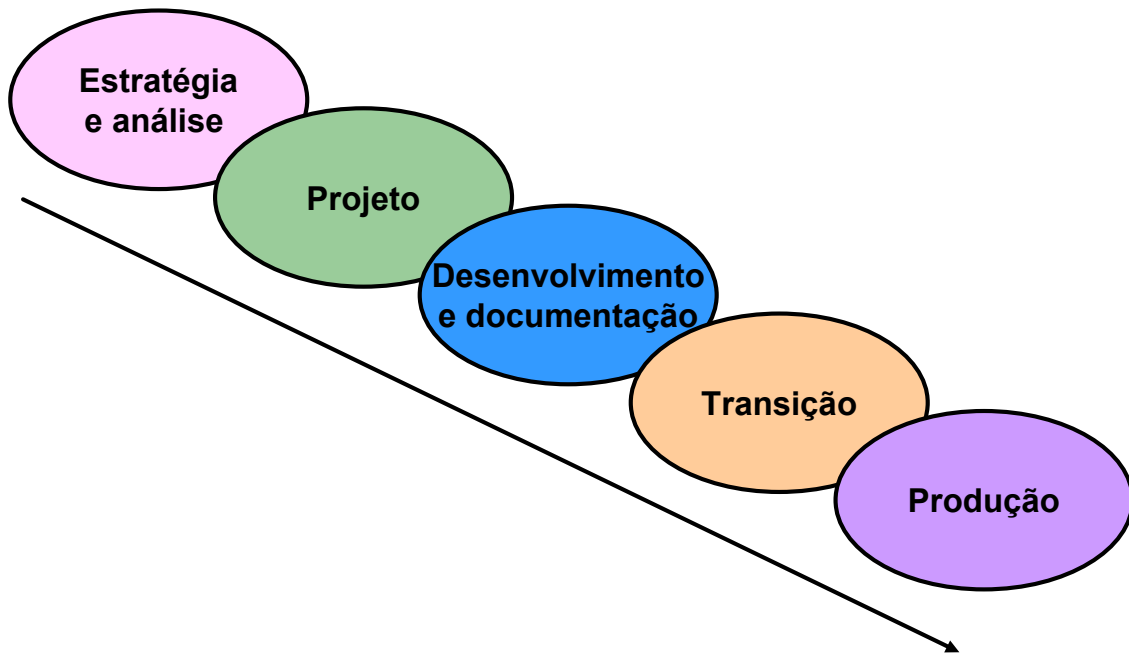
Para desenvolver uma aplicação de e-commerce, é necessário um produto que possa armazenar e gerenciar dados, um produto que possa proporcionar um ambiente de runtime para as aplicações que implementam a lógica de negócios e um produto que possa monitorar e diagnosticar a aplicação após a sua integração. Os produtos Oracle 10g abordados fornecem todos os componentes necessários ao desenvolvimento da sua aplicação.

A Oracle oferece uma plataforma completa para a Internet de alto desempenho que permite desenvolver aplicações de e-commerce e data warehouse. A Plataforma Oracle para a Internet integrada contém todos os elementos necessários ao desenvolvimento, à disponibilização e ao gerenciamento de aplicações para a Internet, incluindo estes três elementos principais:

- Clientes baseados em browser para processar a apresentação
- Servidores de aplicações para executar a lógica de negócios e fornecer a lógica de apresentação a clientes baseados em browser
- Bancos de dados para executar a lógica de negócios com uso intensivo de banco de dados e para fornecer dados

A Oracle oferece uma grande variedade de ferramentas avançadas de desenvolvimento orientadas por GUI (interface gráfica do usuário) para criar aplicações de negócios, bem como um amplo conjunto de aplicações de software para diversos setores e áreas de negócios. O Oracle Developer Suite contém ferramentas para o desenvolvimento de forms e relatórios, e para a criação de data warehouses. É possível criar stored procedures, functions e packages com SQL, PL/SQL ou Java.

# Ciclo de Vida de Desenvolvimento do Sistema



ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

## 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 sistemática e completa do desenvolvimento de um banco de dados transforma requisitos de informações de negócios em um banco de dados operacional.

### Fase de Estratégia e Análise

- Estude e analise os requisitos de negócios. Entreviste os usuários e os gerentes para identificar os requisitos de informações. Incorpore os objetivos das aplicações e da empresa, assim como as futuras especificações do sistema.
- Desenvolva modelos do sistema. Transforme a narrativa em uma representação gráfica das regras e necessidades de informações de negócios. Confirme e refine o modelo com os analistas e os especialistas.

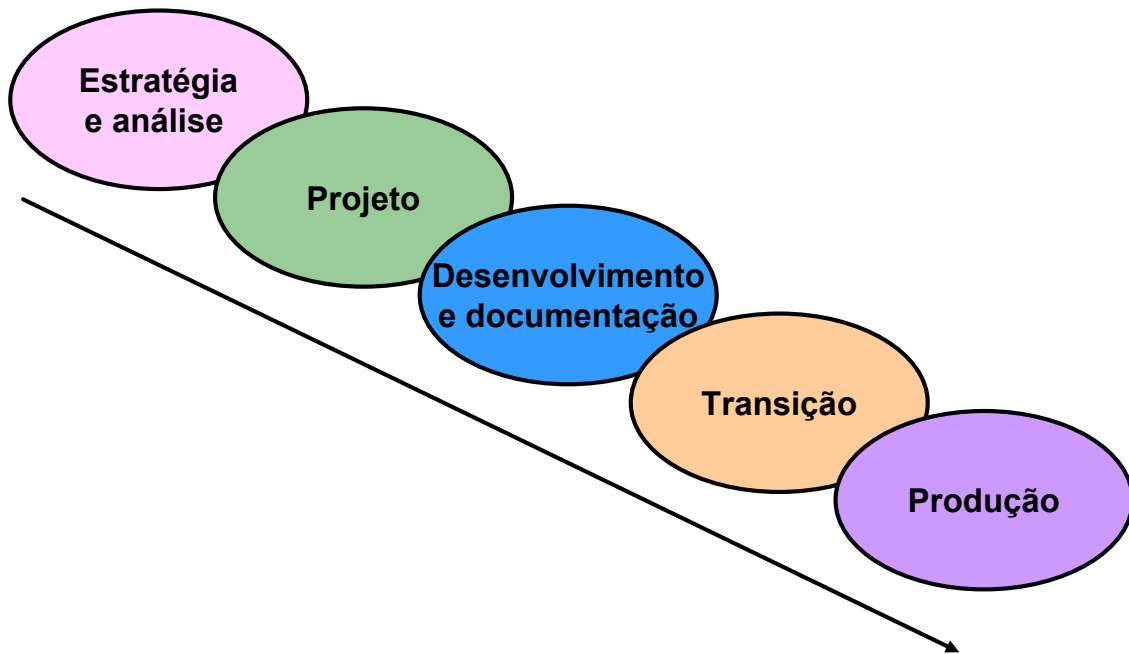
### Fase de Projeto

Projete o banco de dados com base no modelo desenvolvido na fase de estratégia e análise.

### Fase de Criação e Documentação

- Desenvolva o protótipo do sistema. Crie e execute os comandos para produzir as tabelas e os objetos de suporte do banco de dados.
- Desenvolva a documentação do usuário, o texto da ajuda e os manuais de operação para suportar o uso e a operação do sistema.

# Ciclo de Vida de Desenvolvimento do Sistema



Copyright © 2004, Oracle. Todos os direitos reservados.

ORACLE

## Ciclo de Vida de Desenvolvimento do Sistema (continuação)

### Fase de Transição

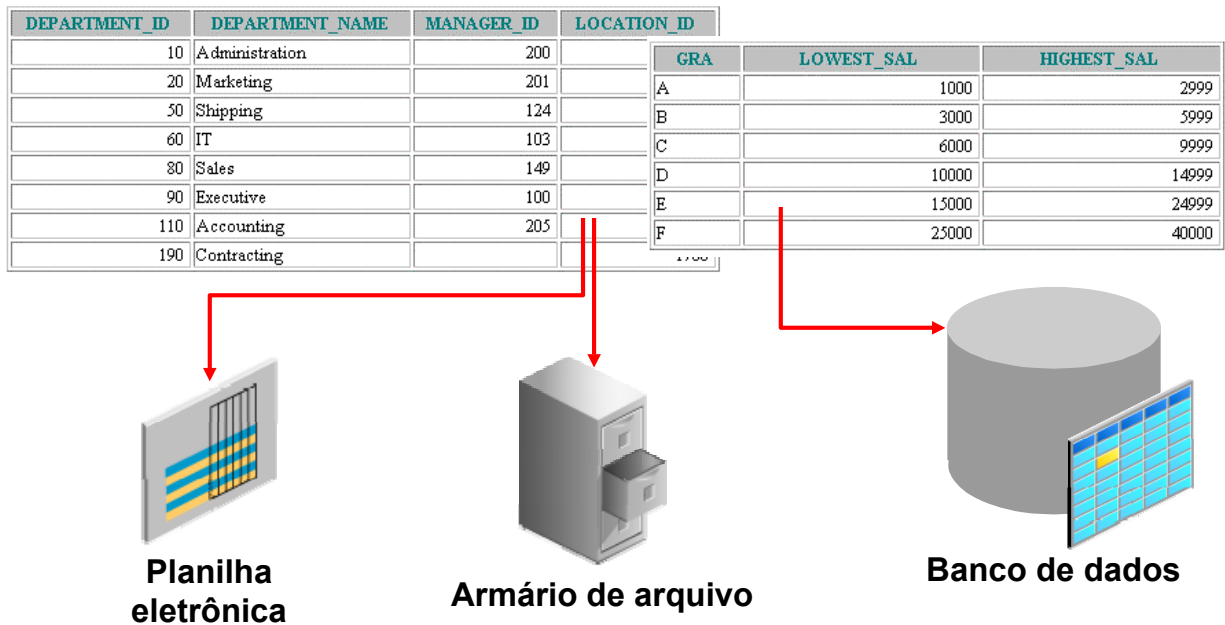
Refine o protótipo. Passe uma aplicação para a fase de produção com testes de aceitação pelos usuários, conversão dos dados existentes e operações paralelas. Faça todas as modificações necessárias.

### Fase de Produção

Apresente o sistema aos usuários. Opere o sistema de produção. Monitore seu desempenho e, depois, aprimore e refine o sistema.

**Observação:** As diversas fases do ciclo de desenvolvimento do sistema podem ser realizadas de forma iterativa. Este curso concentra-se na fase de criação do ciclo.

# Armazenamento de Dados em Diferentes Mídias



ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Armazenando Informações

Toda organização tem necessidades de informações. Uma biblioteca mantém uma lista de membros, livros, datas de vencimento e multas. Uma empresa precisa guardar 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 armário de arquivo ou dados armazenados em planilhas eletrônicas ou em bancos de dados.

Um *banco de dados* é um conjunto organizado de informações.

Para gerenciar bancos de dados, é necessário um DBMS (Database Management System). Um DBMS é um programa que armazena, recupera e modifica dados de bancos de dados sob demanda. Existem quatro tipos principais de bancos de dados: *hierárquico*, *de rede*, *relacional* e (recentemente) *relacional de objeto*.

# Conceito de Banco de Dados Relacional

- O Dr. E. F. Codd propôs o modelo relacional para sistemas de banco de dados em 1970.
- Ele é a base para o RDBMS.
- O modelo relacional é composto de:
  - Um conjunto de objetos ou relações
  - Um conjunto de operadores para agir sobre as relações
  - Integridade de dados para precisão e consistência

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

## Modelo Relacional

Os princípios do modelo relacional foram descritos primeiramente pelo Dr. E. F. Codd em um trabalho de junho de 1970 intitulado "A Relational Model of Data for Large Shared Data Banks". Nesse trabalho, o Dr. Codd propunha o modelo relacional para sistemas de banco de dados.

Os modelos comuns usados na época eram o hierárquico e o de rede, ou até mesmo estruturas de dados simples de flat files. Logo depois, o RDBMS (Relational Database Management System) tornou-se muito popular, especialmente pela facilidade de uso e flexibilidade em termos de estrutura. Além disso, vários fornecedores inovadores, como a Oracle, complementaram o RDBMS com um conjunto de produtos eficientes para usuários e desenvolvedores de aplicações, que compunham uma solução integral.

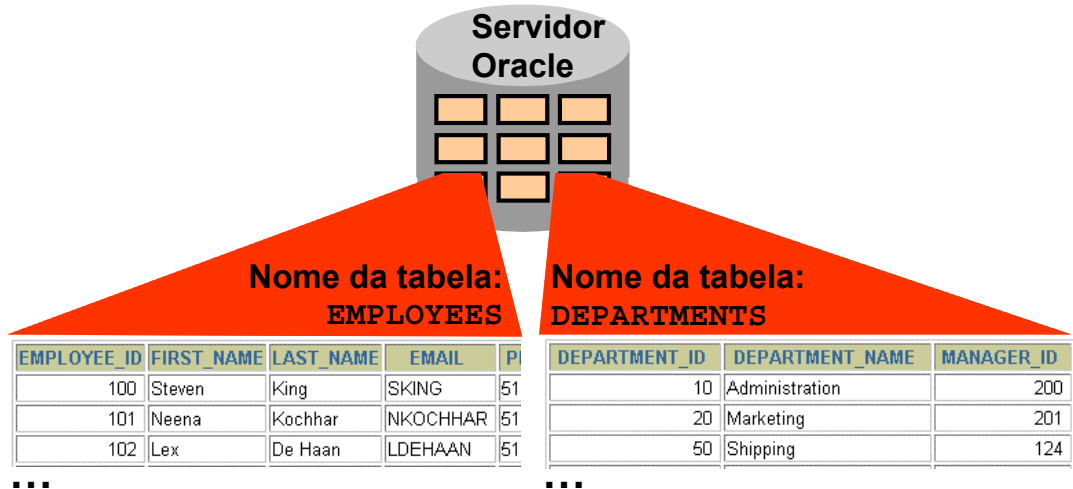
## Componentes do Modelo Relacional

- Conjuntos de objetos ou relações que armazenam os dados
- Um conjunto de operadores que age sobre as relações para produzir outras relações
- Integridade de dados para precisão e consistência

Para obter mais informações, consulte *An Introduction to Database Systems, Eighth Edition* (Addison-Wesley: 2004), escrito por Chris Date.

# Definição de um Banco de Dados Relacional

Um banco de dados relacional é um conjunto de relações ou tabelas de duas dimensões.



ORACLE

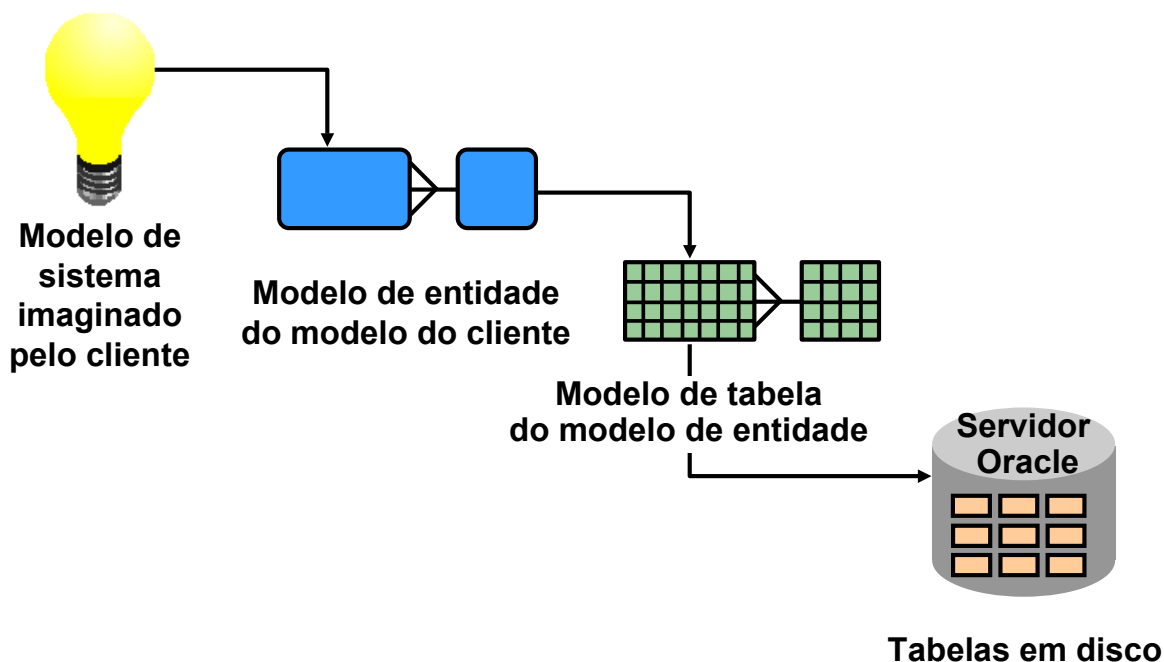
Copyright © 2004, Oracle. Todos os direitos reservados.

## Definição de um Banco de Dados Relacional

Um banco de dados relacional usa relações ou tabelas de duas dimensões para armazenar informações.

Por exemplo, é possível armazenar informações sobre todos os funcionários de uma empresa. Em um banco de dados relacional, você cria diversas tabelas para armazenar diferentes informações sobre os funcionários, como uma tabela de funcionários, uma tabela de departamentos e uma tabela de salários.

## Modelos de Dados



ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Modelos de Dados

Os modelos constituem a base do projeto. Os engenheiros desenvolvem o modelo de um carro para aperfeiçoar os detalhes antes de produzi-lo. Da mesma forma, os designers de sistemas desenvolvem modelos para explorar idéias e compreender melhor o projeto do banco de dados.

### Finalidade dos Modelos

Os modelos ajudam a comunicar os 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 atenda a vários desses usos, seja compreendido por um usuário final e contenha detalhes suficientes para que um desenvolvedor crie um sistema de banco de dados.



# Modelo de Relacionamento entre Entidades

- Crie um diagrama de relacionamento entre entidades a partir de especificações de negócios ou narrativas:



- Cenário
  - "... Designe um ou mais funcionários a um departamento. ..."
  - "... Ainda não foram designados funcionários a alguns departamentos. ..."

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Modelagem de ER

Em um sistema eficiente, os dados são divididos em entidades ou categorias discretas. Um modelo de ER (Entity Relationship) é uma ilustração de várias entidades em uma empresa e dos relacionamentos entre elas. Um modelo de ER é derivado de especificações de negócios ou narrativas e é criado durante a fase de análise do ciclo de vida de desenvolvimento do sistema. Os modelos de ER separam as informações necessárias a uma empresa das atividades realizadas por ela. Embora as empresas possam mudar de atividades, o tipo de informações tende a permanecer constante. Portanto, as estruturas de dados também tendem a permanecer constantes.

## Modelagem de ER (continuação)

### Vantagens da Modelagem de ER

- Documenta as informações para a organização em um formato simples e preciso
- Possibilita uma visão clara do escopo dos requisitos de informações
- Fornece um mapa ilustrado e de fácil compreensão do projeto do banco de dados
- Oferece uma estrutura eficiente para integrar várias aplicações

### Componentes Principais

- **Entidade:** Algo importante sobre o qual são necessárias informações. Como exemplos, podemos citar departamentos, funcionários e pedidos.
- **Atributo:** Algo que descreve ou qualifica uma entidade. Por exemplo, para a entidade de funcionário, os atributos serão o número, o nome, o cargo, a data de admissão, o número do departamento e outros dados sobre o funcionário. Cada um desses atributos é obrigatório ou opcional. Esse estado é denominado *opcionalidade*.
- **Relacionamento:** Uma associação nomeada entre entidades que exhibe a opcionalidade e o grau. Como exemplos, podemos citar as associações entre funcionários e departamentos, e entre pedidos e itens.

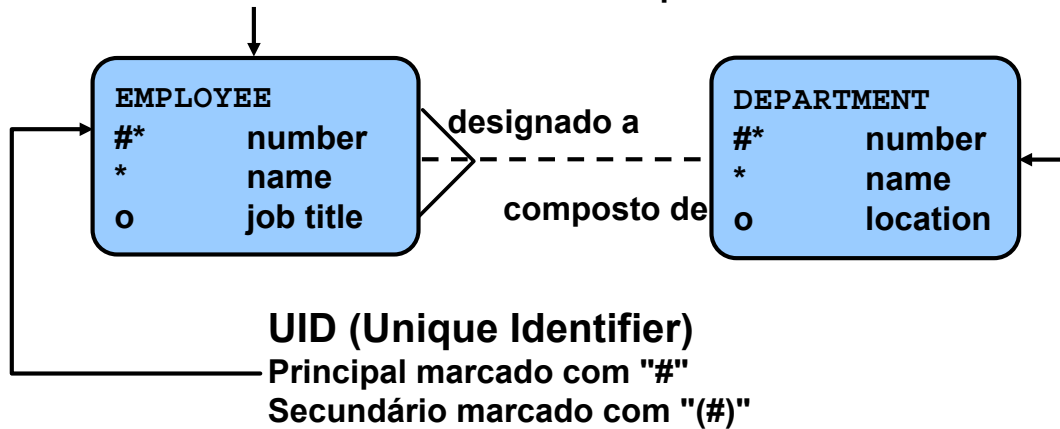
# Convenções de Modelagem de Relacionamento entre Entidades

## Entidade

- Nome exclusivo, singular
- Letras maiúsculas
- Caixa editável
- Sinônimo entre parênteses

## Atributo

- Nome singular
- Letras minúsculas
- Obrigatório marcado com \*
- Opcional marcado com "o"



ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

## Convenções da Modelagem de ER

### Entidades

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

- Nome de entidade exclusivo, singular
- Nome de entidade em maiúsculas
- Caixa editável
- Nomes de sinônimos opcionais em maiúsculas entre parênteses: ( )

### Atributos

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

- Nome singular em minúsculas
- Tag de asterisco (\*) para atributos obrigatórios (isto é, valores que *devem* ser conhecidos)
- Tag de letra "o" para atributos opcionais (isto é, valores que *podem* ser conhecidos)

### Relacionamentos

Símbolo	Descrição
Linha tracejada	Elemento opcional que indica probabilidade
Linha sólida	Elemento necessário que indica obrigatoriedade
Pé-de-galinha	Elemento de grau que indica um ou mais
Linha única	Elemento de grau que indica apenas um

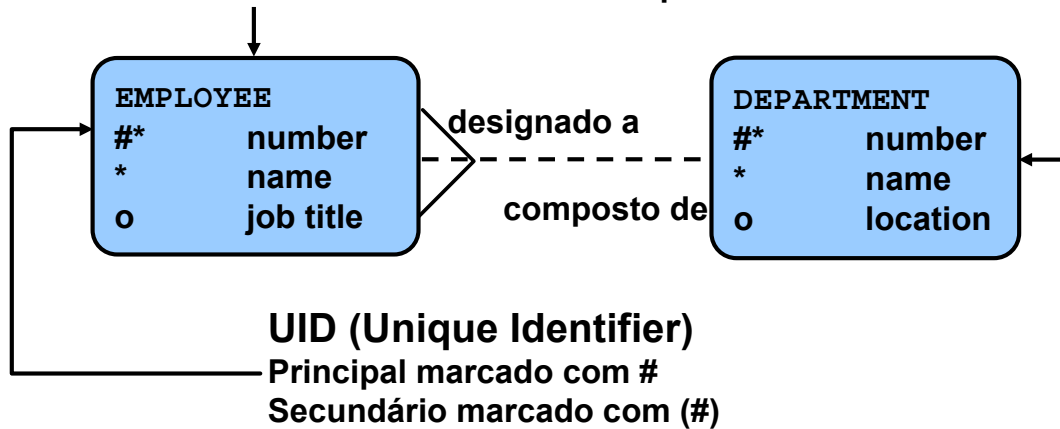
# Convenções de Modelagem de Relacionamento entre Entidades

## Entidade

- Nome exclusivo, singular
- Letras maiúsculas
- Caixa editável
- Sinônimo entre parênteses

## Atributo

- Nome singular
- Letras minúsculas
- Obrigatório marcado com \*
- Opcional marcado com "o"



ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

## Convenções de Modelagem de ER (continuação)

### Relacionamentos

Cada direção do relacionamento contém:

- Um label: por exemplo, *taught by* ou *assigned to*
- Uma opcionalidade: *must be* ou *may be*
- Um grau: *one and only one* ou *one or more*

**Observação:** O termo *cardinalidade* é sinônimo do termo *grau*.

Cada entidade de origem {may be | must be} nome de relacionamento {one and only one | one or more} entidade de destino.

**Observação:** A convenção é a leitura no sentido horário.

### Identificadores Exclusivos

Um UID (Unique Identifier, Identificador Exclusivo) é uma combinação de atributos e/ou relacionamentos utilizada para distinguir ocorrências de uma entidade. Cada ocorrência de entidade deve ser identificada com exclusividade.

- Marque cada atributo que compõe o UID com um símbolo de número: #
- Marque os UIDs secundários com um símbolo de número entre parênteses: (#)

## Relacionando Várias Tabelas

- Cada linha de dados de uma tabela é identificada com exclusividade por uma PK (Primary Key, Chave Primária).
- É possível relacionar logicamente dados de várias tabelas por meio de FKs (Foreign Keys, Chaves Estrangeiras).

Nome da tabela: EMPLOYEES

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	DEPARTMENT_ID
174	Ellen	Abel	80
142	Curtis	Davies	50
102	Lex	De Haan	90
104	Bruce	Ernst	60
202	Pat	Fay	20
206	William	Gietz	110

...

Chave primária

Chave estrangeira

Chave primária

Nome da tabela: DEPARTMENTS

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
50	Shipping	124	1500
60	IT	103	1400
80	Sales	149	2500
90	Executive	100	1700
110	Accounting	205	1700
190	Contracting		1700

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Relacionando Várias Tabelas

Cada tabela contém dados que descrevem exatamente uma entidade. Por exemplo, a tabela EMPLOYEES contém informações sobre funcionários. As categorias de dados são listadas na parte superior de cada tabela e, em casos específicos, na parte inferior. Com um formato de tabela, você pode visualizar imediatamente, compreender e usar as informações.

Como os dados sobre entidades distintas são armazenados em tabelas diferentes, talvez seja necessário combinar duas ou mais tabelas para responder a determinada pergunta. Por exemplo, talvez você queira saber a localização do departamento onde um funcionário trabalha. Nesse caso, você precisará de informações da tabela EMPLOYEES (que contém dados sobre funcionários) e da tabela DEPARTMENTS (que contém informações sobre departamentos). Um RDBMS permite relacionar os dados de uma tabela aos de outra por meio das chaves estrangeiras. Uma chave estrangeira é uma coluna (ou um conjunto de colunas) que faz referência a uma chave primária na mesma tabela ou em outra tabela.

Você pode usar a capacidade de relacionar dados de uma tabela a dados de outra para organizar informações em unidades gerenciáveis separadas. É possível manter os dados sobre funcionários logicamente separados dos dados sobre departamentos armazenando estes últimos em outra tabela.

## Relacionando Várias Tabelas (continuação)

### Diretrizes de Chaves Primárias e Chaves Estrangeiras

- Não é possível usar valores duplicados em uma chave primária.
- Em geral, não é possível alterar chaves primárias.
- As chaves estrangeiras baseiam-se em valores de dados e são ponteiros lógicos (e não físicos).
- Um valor de chave estrangeira deve corresponder a um valor de chave primária existente ou a um valor de chave exclusiva; ou então, deve ser nulo.
- Uma chave estrangeira deve fazer referência a uma coluna de chave primária ou exclusiva.

# Terminologia do Banco de Dados Relacional

2	3	4		5		
	EMPLOYEE_ID	LAST_NAME	FIRST_NAME	SALARY	COMMISSION_PCT	DEPARTMENT_ID
	100	King	Steven	24000		90
	101	Kochhar	Neena	17000		90
	102	De Haan	Lex	17000		90
	103	Hunold	Alexander	9000		60
	104	Ernst	Bruce	6000		60
	107	Lorentz	Diana	4200	6	60
	124	Mourgos	Kevin	5800		50
	141	Rajs	Trenna	3500		50
	142	Davies	Curtis	3100		50
	143	Matos	Randall	2600		50
	144	Vargas	Peter	2500		50
	149	Zlotkey	Eleni	10500	.2	80
	174	Abel	Ellen	11000	.3	80
	176	Taylor	Jonathon	8600	.2	80
	178	Grant	Kimberely	7000	.15	
	200	Whalen	Jennifer	4400		10
1	201	Hartstein	Michael	13000		20
	202	Fay	Pat	6000		20
	205	Higgins	Shelley	12000		110
	206	Gietz	William	8300		110

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Terminologia Usada em um Banco de Dados Relacional

Um banco de dados relacional pode conter uma ou muitas tabelas. A *tabela* é a estrutura básica de armazenamento de um RDBMS. Ela pode conter todos os dados necessários sobre algo relativo ao mundo real, como funcionários, NFFs ou clientes.

O slide mostra o conteúdo da *relação* ou da *tabela* EMPLOYEES. Os números indicam:

1. Uma única *linha* (ou *tupla*) que representa todos os dados necessários a um funcionário específico. Cada linha de uma tabela deve ser identificada por uma chave primária, que impede linhas duplicadas. A ordem das linhas não é importante; especifique essa ordem quando os dados forem recuperados.
2. Uma *coluna* ou um atributo que contém o número do funcionário. O número do funcionário identifica um funcionário com *exclusividade* na tabela EMPLOYEES. Neste exemplo, a coluna contendo o número do funcionário é designada como a *chave primária*. Uma chave primária deve conter um valor, que deve ser exclusivo.
3. Uma coluna que não é um valor-chave. Uma coluna representa um tipo de dados em uma tabela; no exemplo, os dados representam os salários de todos os funcionários. A ordem das colunas não é importante durante o armazenamento de dados; especifique essa ordem quando os dados forem recuperados.

## Terminologia Usada em um Banco de Dados Relacional (continuação)

4. Uma coluna com o número dos departamentos, que também é uma *chave estrangeira*. Uma chave estrangeira é uma coluna que define o relacionamento entre tabelas. Uma chave estrangeira faz referência a uma chave primária ou exclusiva na mesma tabela ou em outra. No exemplo, DEPARTMENT\_ID identifica com *exclusividade* um departamento na tabela DEPARTMENTS.
5. A localização de um *campo* é a interseção entre uma linha e uma coluna. Um campo só pode conter um valor.
6. Um campo pode não conter um valor. Nesse caso, seu valor é nulo. Na tabela EMPLOYEES, somente os funcionários com a atribuição de representante de vendas possuem um valor no campo COMMISSION\_PCT (comissão).



# Propriedades do Banco de Dados Relacional

**Um banco de dados relacional:**

- **Pode ser acessado e modificado com a execução de instruções SQL (Structured Query Language)**
- **Contém um conjunto de tabelas sem ponteiros físicos**
- **Usa um conjunto de operadores**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Propriedades de um 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 estão organizados fisicamente.

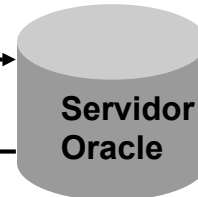
Para acessar o banco de dados, execute uma instrução SQL (Structured Query Language), que é a linguagem padrão ANSI (American National Standards Institute) de operação de bancos de dados relacionais. Essa linguagem contém um amplo conjunto de operadores para particionar e combinar relações. É possível usar instruções SQL para modificar o banco de dados.

# Comunicando-se com um RDBMS por Meio de SQL

**A instrução SQL é informada.**

```
SELECT department_name  
FROM departments;
```

**A instrução é enviada  
ao servidor Oracle.**



DEPARTMENT_NAME
Administration
Marketing
Shipping
IT
Sales
Executive
Accounting
Contracting

ORACLE

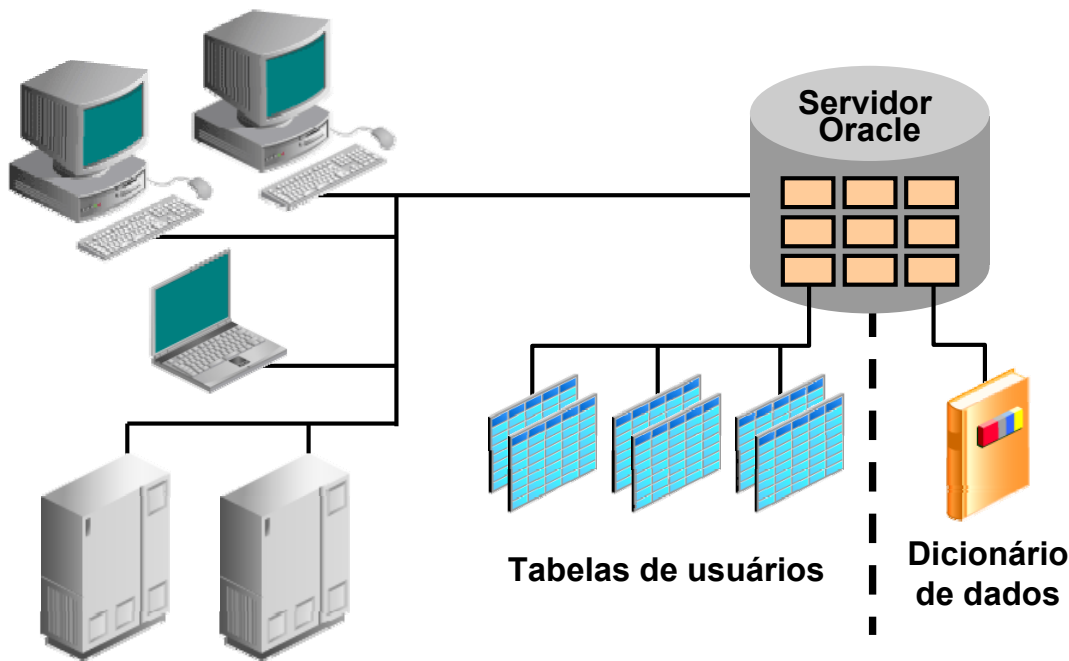
Copyright © 2004, Oracle. Todos os direitos reservados.

## SQL (Structured Query Language)

O uso de SQL permite a comunicação com o servidor Oracle. A linguagem SQL apresenta estas vantagens:

- Eficiente
- Fácil de aprender e usar
- Funcionalmente completa (permite definir, recuperar e manipular os dados das tabelas)

# Sistema de Gerenciamento de Banco de Dados Relacional da Oracle



Copyright © 2004, Oracle. Todos os direitos reservados.

ORACLE

## Sistema de Gerenciamento de Banco de Dados Relacional da Oracle

A Oracle fornece um RDBMS flexível denominado Banco de Dados Oracle 10g. Com os recursos desse produto, você pode armazenar e gerenciar dados com todas as vantagens de uma estrutura relacional e de PL/SQL, um mecanismo que possibilita o armazenamento e a execução de unidades de programa. O Banco de Dados Oracle 10g também suporta Java e XML. O servidor Oracle oferece opções de recuperação de dados com base em técnicas de otimização. Ele contém recursos de segurança que controlam a forma como um banco de dados é acessado e usado. Outros recursos incluem a consistência e a proteção de dados por meio de mecanismos de bloqueio.

A release Oracle10g apresenta um método aberto, completo e integrado para o gerenciamento de informações. Um servidor Oracle consiste em um banco de dados Oracle e uma instância de servidor Oracle. Sempre que um banco de dados é iniciado, uma SGA (System Global Area) é alocada e os processos de background do Oracle são iniciados. A SGA é uma área da memória usada para as informações de banco de dados compartilhadas pelos usuários de banco de dados. A combinação dos processos de background e dos buffers de memória é chamada de *instância* Oracle.

# Instruções SQL

SELECT INSERT UPDATE DELETE MERGE	<b>DML (Data Manipulation Language)</b>
CREATE ALTER DROP RENAME TRUNCATE COMMENT	<b>DDL (Data Definition Language)</b>
COMMIT ROLLBACK SAVEPOINT	<b>Controle de transações</b>
GRANT REVOKE	<b>DCL (Data Control Language)</b>

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

## Instruções SQL

O Oracle SQL adota os padrões aceitos pelo setor. Para assegurar a compatibilidade futura com os padrões em desenvolvimento, a Oracle Corporation mantém ativamente uma equipe especializada nos comitês de padrões SQL. Os comitês aceitos pelo setor são o ANSI (American National Standards Institute) e o ISO (International Standards Organization). Os dois comitês aceitam SQL como a linguagem padrão para os bancos de dados relacionais.

Instrução	Descrição
SELECT INSERT UPDATE DELETE MERGE	Recupera dados do banco de dados, informa novas linhas, altera linhas existentes e remove linhas indesejadas das tabelas do banco de dados, respectivamente. Conhecida coletivamente como DML ( <i>Data Manipulation Language</i> ).
CREATE ALTER DROP RENAME TRUNCATE COMMENT	Configura, altera e remove estruturas de dados das tabelas. Conhecida coletivamente como DDL ( <i>Data Definition Language</i> ).
COMMIT ROLLBACK SAVEPOINT	Gerencia as alterações feitas por instruções DML. As alterações nos dados podem ser agrupadas em transações lógicas.
GRANT REVOKE	Concede ou revoga direitos de acesso ao banco de dados Oracle e às estruturas contidas nele. Conhecida coletivamente como DCL ( <i>Data Control Language</i> ).

# Tabelas Usadas no Curso

## EMPLOYEES

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SAL
100	Steven	King	SKING	515.123.4567	17-JUN-87	AD_PRES	240
101	Neena	Kochhar	NKOCHHAR	515.123.4568	21-SEP-89	AD_VP	170
102	Lex	De Haan	LDEHAAN	515.123.4569	13-JAN-93	AD_VP	170
103	Alexander	Hunold	AHUNOLD	590.423.4567	03-JAN-90	IT_PROG	90
104	Bruce	Ernst	BERNST	590.423.4568	21-MAY-91	IT_PROG	60
107	Diana	Lorentz	DLORENTZ	590.423.5567	07-FEB-99	IT_PROG	42
124	Kevin	Mourgos	KMOURGOS	650.123.5234	16-NOV-99	ST_MAN	58
141	Trenna	Rajs	TRAJS	650.121.8009	17-OCT-95	ST_CLERK	35
142	Curtis	Davies	CDAVIES	650.121.2994	29-JAN-97	ST_CLERK	31

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
50	Shipping	124	1500
60	IT	103	1400
80	Sales	149	2500
90	Executive	100	1700
110	Accounting	205	1700
190	Contracting		1700

## DEPARTMENTS

1.2874	15-MAR-98	ST_CLERK	26
1.2004	09-JUL-98	ST_CLERK	25

GRA	LOWEST_SAL	HIGHEST_SAL
A	1000	2999
B	3000	5999
C	6000	9999
D	10000	14999
E	15000	24999
F	25000	40000

## JOB\_GRADES

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

## Tabelas Usadas no Curso

Estas são as principais tabelas usadas neste curso:

- Tabela EMPLOYEES: Fornece detalhes de todos os funcionários
- Tabela DEPARTMENTS: Fornece detalhes de todos os departamentos
- Tabela JOB\_GRADES: Fornece detalhes de salários para diversos níveis

**Observação:** A estrutura e os dados de todas as tabelas são fornecidos no Apêndice B.0

# Sumário

- **O Banco de Dados Oracle 10g é o banco de dados utilizado para a computação em grade.**
- **Ele se baseia no sistema de gerenciamento de banco de dados relacional de objeto.**
- **Os bancos de dados relacionais são compostos de relações, gerenciados por operações relacionais e regidos por constraints de integridade de dados.**
- **Com o servidor Oracle, você pode armazenar e gerenciar informações com a linguagem SQL e o mecanismo PL/SQL.**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Sumário

Os sistemas de gerenciamento de banco de dados relacional são compostos de objetos ou relações. Eles são gerenciados por operações e regidos por constraints de integridade de dados.

A Oracle Corporation cria produtos e serviços para atender às suas necessidades de RDBMS. Os principais produtos são:

- Oracle Database 10g, com o qual você armazena e gerencia informações por meio de SQL
- Oracle Application Server 10g, com o qual você executa todas as aplicações
- Oracle Enterprise Manager 10g Grid Control, usado para gerenciar e automatizar tarefas administrativas nos conjuntos de sistemas de um ambiente de grade.

## SQL

O servidor Oracle suporta instruções SQL com o padrão ANSI e contém extensões. Com a linguagem SQL, é possível estabelecer comunicação com o servidor para acessar, manipular e controlar dados.

# 1

## Recuperando Dados com a Instrução SQL SELECT

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

# Objetivos

**Ao concluir esta lição, você será capaz de:**

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

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Objetivos

Para extrair dados do banco de dados, é necessário usar a instrução SQL (Structured Query Language) `SELECT`. Talvez você precise restringir as colunas exibidas. Esta lição descreve todas as instruções SQL necessárias para executar essas ações. Você pode criar instruções `SELECT` para usar mais de uma vez.

Esta lição também aborda o ambiente `iSQL*Plus` no qual as instruções SQL são executadas.



# Recursos de Instruções SQL SELECT

## Projeção

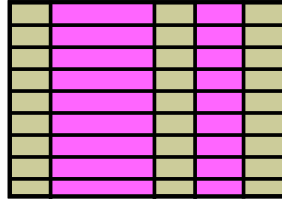


Tabela 1

## Seleção

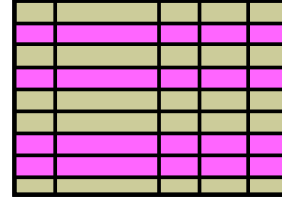


Tabela 1

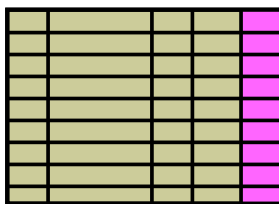


Tabela 1

Join

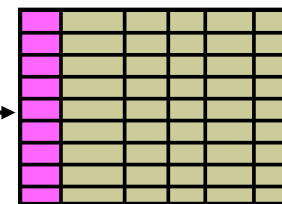


Tabela 2

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Recursos de Instruções SQL SELECT

Uma instrução SELECT recupera informações do banco de dados. Com essa instrução, é possível usar os seguintes recursos:

- **Projeção:** Escolha as colunas de uma tabela a serem retornadas por uma consulta. Escolha quantas colunas forem necessárias
- **Seleção:** Escolha as linhas de uma tabela a serem retornadas por uma consulta. É possível usar vários critérios para restringir as linhas recuperadas.
- **Join:** Una os dados armazenados em diferentes tabelas especificando o vínculo entre elas. As joins SQL são abordadas com mais detalhes em uma lição posterior.

# Instrução SELECT Básica

```
SELECT * | { [DISTINCT] column | expression [alias] , ... }  
FROM      table;
```

- **SELECT** identifica as colunas a serem exibidas
- **FROM** identifica a tabela contendo essas colunas

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

## Instrução SELECT Básica

Em sua forma mais simples, uma instrução SELECT deve incluir:

- Uma cláusula SELECT, que especifica as colunas a serem exibidas
- Uma cláusula FROM, que identifica a tabela com as colunas listadas na cláusula SELECT

Na sintaxe:

SELECT	é uma lista de uma ou mais colunas
*	seleciona todas as colunas
DISTINCT	suprime as colunas duplicadas
column   expression	seleciona a coluna nomeada ou a expressão
alias	fornece cabeçalhos distintos às colunas selecionadas
FROM table	especifica a tabela que contém as colunas

**Observação:** Neste curso, os termos *palavra-chave*, cláusula e instrução são usados da seguinte maneira:

- Uma *palavra-chave* é um elemento individual de SQL.  
Por exemplo, SELECT e FROM são palavras-chave.
- Uma *cláusula* é uma parte de uma instrução SQL.  
Por exemplo, SELECT employee\_id, last\_name, ... é uma cláusula.
- Uma *instrução* é uma combinação de duas ou mais cláusulas.

Por exemplo, SELECT \* FROM employees é uma instrução SQL.

# Selecionando Todas as Colunas

```
SELECT *  
FROM departments;
```

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
50	Shipping	124	1500
60	IT	103	1400
80	Sales	149	2500
90	Executive	100	1700
110	Accounting	205	1700
190	Contracting		1700

8 rows selected.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Selecionando Todas as Colunas de Todas as Linhas

Você pode exibir todas as colunas de dados de uma tabela inserindo um asterisco (\*) após a palavra-chave SELECT. No exemplo do slide, a tabela departments contém quatro colunas: DEPARTMENT\_ID, DEPARTMENT\_NAME, MANAGER\_ID e LOCATION\_ID. A tabela contém sete linhas, uma para cada departamento.

Você também pode exibir todas as colunas da tabela listando-as após a palavra-chave SELECT. Por exemplo, a instrução SQL a seguir (como o exemplo do slide) exibe todas as colunas e linhas da tabela DEPARTMENTS:

```
SELECT department_id, department_name, manager_id, location_id  
FROM departments;
```

# Selecionando Colunas Específicas

```
SELECT department_id, location_id
FROM departments;
```

DEPARTMENT_ID	LOCATION_ID
10	1700
20	1800
50	1500
60	1400
80	2500
90	1700
110	1700
190	1700

8 rows selected.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Selecionando Colunas Específicas de Todas as Linhas

Para exibir colunas específicas de uma tabela, você pode usar a instrução SELECT com os nomes das colunas separados por vírgulas. O exemplo do slide exibe todos os números de departamentos e locais da tabela DEPARTMENTS.

Na cláusula SELECT, especifique as colunas desejadas na ordem em que deverão aparecer na saída. Por exemplo, para exibir o local antes do número do departamento da esquerda para a direita, use a seguinte instrução:

```
SELECT location_id, department_id
FROM departments;
```

LOCATION_ID	DEPARTMENT_ID
1700	10
1800	20
1500	50

...

8 rows selected.

## Criando Instruções SQL

- As instruções SQL não fazem distinção entre maiúsculas e minúsculas.
- As instruções SQL podem ocupar uma ou mais linhas.
- Não é possível abreviar palavras-chave ou dividi-las em duas linhas.
- As cláusulas geralmente são colocadas em linhas separadas.
- Os recuos são usados para melhorar a legibilidade.
- No *iSQL\*Plus*, o encerramento das instruções SQL com ponto-e-vírgula (;) é opcional. A utilização de ponto-e-vírgula será obrigatória se você executar várias instruções SQL.
- No *SQL\*Plus*, você deverá encerrar cada instrução SQL com ponto-e-vírgula (;).

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Criando Instruções SQL

Com estas diretrizes e regras simples, você pode criar instruções válidas de fácil leitura e edição:

- As instruções SQL não fazem distinção entre maiúsculas e minúsculas (a menos que essa distinção seja indicada).
- É possível informar instruções SQL em uma ou várias linhas.
- Não é possível dividir palavras-chave em duas linhas ou abreviá-las.
- As cláusulas normalmente são colocadas em linhas separadas por questões de legibilidade e facilidade de edição.
- Devem ser usados recuos para tornar o código mais legível.
- Em geral, as palavras-chave são informadas em maiúsculas; todas as outras palavras, como nomes de tabelas e colunas, são informadas em minúsculas.

## Executando Instruções SQL

No *iSQL\*Plus*, clique no botão Execute para executar os comandos na janela de edição.

No *SQL\*Plus*, encerre a instrução SQL com ponto-e-vírgula e pressione a tecla Enter para executar o comando.

# Defaults de Cabeçalhos de Colunas

- **iSQL\*Plus:**
  - Alinhamento de cabeçalho default: centralizado
  - Exibição de cabeçalho default: letras maiúsculas
- **SQL\*Plus:**
  - Os cabeçalhos das colunas de caractere e data são alinhados à esquerda
  - Os cabeçalhos das colunas de número são alinhados à direita
  - Exibição de cabeçalho default: letras maiúsculas

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Defaults de Cabeçalhos de Colunas

No iSQL\*Plus, os cabeçalhos das colunas são exibidos em maiúsculas e centralizados.

```
SELECT last_name, hire_date, salary
FROM employees;
```

LAST_NAME	HIRE_DATE	SALARY
King	17-JUN-87	24000
Kochhar	21-SEP-89	17000
De Haan	13-JAN-93	17000
Hunold	03-JAN-90	9000
Ernst	21-MAY-91	6000
...		
Higgins	07-JUN-94	12000
Gietz	07-JUN-94	8300

20 rows selected.

Você pode substituir o cabeçalho da coluna por um apelido. Os apelidos de colunas são abordados posteriormente nesta lição.

# Expressões Aritméticas

**Crie expressões com dados de número e data usando operadores aritméticos.**

Operador	Descrição
+	Somar
-	Subtrair
*	Multiplicar
/	Dividir

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

## Expressões Aritméticas

Talvez você precise modificar a forma como os dados são exibidos, realizar cálculos ou examinar cenários hipotéticos. Todas essas ações são possíveis com o uso de expressões aritméticas. Uma expressão aritmética pode conter nomes de colunas, valores numéricos constantes e operadores aritméticos.

### Operadores Aritméticos

O slide lista os operadores aritméticos disponíveis em SQL. Você pode usar operadores aritméticos em qualquer cláusula de uma instrução SQL (exceto na cláusula FROM).

**Observação:** Nos tipos de dados DATE e TIMESTAMP, só é possível usar os operadores de adição e subtração.

# Usando Operadores Aritméticos

```
SELECT last_name, salary, salary + 300
FROM employees;
```

LAST_NAME	SALARY	SALARY+300
King	24000	24300
Kochhar	17000	17300
De Haan	17000	17300
Hunold	9000	9300
Ernst	6000	6300

20 rows selected.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Usando Operadores Aritméticos

O exemplo do slide usa o operador de adição para calcular um aumento de salário de US\$ 300 para todos os funcionários. O slide também exibe uma coluna `SALARY+300` na saída.

Observe que a coluna `SALARY+300` calculada resultante não é uma nova coluna da tabela `EMPLOYEES`; ela existe apenas para fins de exibição. Por default, o nome de uma nova coluna origina-se do cálculo que a gerou – neste caso, `salary+300`.

**Observação:** O servidor Oracle ignora os espaços em branco antes e depois do operador aritmético.

## Precedência de Operadores

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

Você pode usar parênteses para impor a avaliação da expressão entre parênteses primeiro.

### Regras de Precedência:

- A multiplicação e a divisão ocorrem antes da adição e da subtração.
- Os operadores com a mesma prioridade são avaliados da esquerda para a direita.
- São usados parênteses para sobrepor a precedência default ou tornar a instrução mais clara.



# Precedência de Operadores

```
SELECT last_name, salary, 12*salary+100
FROM employees;
```

1

LAST_NAME	SALARY	12*SALARY+100
King	24000	288100
Kochhar	17000	204100
De Haan	17000	204100

20 rows selected.

```
SELECT last_name, salary, 12*(salary+100)
FROM employees;
```

2

LAST_NAME	SALARY	12*(SALARY+100)
King	24000	289200
Kochhar	17000	205200
De Haan	17000	205200

20 rows selected.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Precedência de Operadores (continuação)

O primeiro exemplo do slide exibe o sobrenome, o salário e a remuneração anual dos funcionários. Para calcular a remuneração anual, ele multiplica o salário mensal por 12 e acrescenta um bônus único de US\$ 100. Observe que a multiplicação é executada antes da adição.

**Observação:** Use parênteses para reforçar a ordem padrão de precedência e para tornar a expressão mais clara. Por exemplo, a expressão do slide pode ter a forma  $(12 * \text{salary}) + 100$  sem alterações no resultado.

## Usando Parênteses

Você pode sobrepor as regras de precedência usando parênteses para especificar a ordem na qual os operadores devem ser executados.

O segundo exemplo do slide exibe o sobrenome, o salário e a remuneração anual dos funcionários. Ele calcula a remuneração anual da seguinte forma: adiciona um bônus mensal de US\$ 100 ao salário mensal e multiplica esse subtotal por 12. Em função dos parênteses, a adição tem prioridade sobre a multiplicação.

## Definindo um Valor Nulo

- Um valor nulo não está disponível nem designado e não é conhecido ou aplicável.
- Um valor nulo é diferente de zero ou de um espaço em branco.

```
SELECT last_name, job_id, salary, commission_pct  
FROM employees;
```

LAST_NAME	JOB_ID	SALARY	COMMISSION_PCT
King	AD_PRES	24000	
Kochhar	AD_VP	17000	
...			
Zlotkey	SA_MAN	10500	.2
Abel	SA_REP	11000	.3
Taylor	SA_REP	8600	.2
...			
Gietz	AC_ACCOUNT	8300	

20 rows selected.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Valores Nulos

Se, em uma coluna específica, uma linha não contiver um valor de dados, o valor será *nulo* ou conterá um valor nulo.

Um valor nulo não está disponível nem designado e não é conhecido ou aplicável. Um valor nulo é diferente de zero ou de um espaço. Zero é um número e um espaço é um caractere.

As colunas de qualquer tipo de dados podem conter valores nulos. No entanto, algumas constraints (NOT NULL e PRIMARY KEY) impedem o uso de valores nulos em colunas.

Na coluna COMMISSION\_PCT da tabela EMPLOYEES, observe que apenas um gerente de vendas ou um representante de vendas pode receber comissão. Os outros funcionários não têm direito a comissões. Um valor nulo representa essa situação.

# Valores Nulos em Expressões Aritméticas

As expressões aritméticas que contêm um valor nulo são avaliadas como nulas.

```
SELECT last_name, 12*salary*commission_pct  
FROM employees;
```

LAST_NAME	12*SALARY*COMMISSION_PCT
King	
Kochhar	
...	
Zlotkey	25200
Abel	39600
Taylor	20640
...	
Gietz	

20 rows selected.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Valores Nulos em Expressões Aritméticas

Se o valor de uma coluna na expressão aritmética for nulo, o resultado será nulo. Por exemplo, se você tentar realizar uma divisão por zero, será gerado um erro. Entretanto, se você dividir um número por um valor nulo, o resultado será nulo ou desconhecido.

No exemplo do slide, o funcionário King não recebe comissão. Como a coluna COMMISSION\_PCT na expressão aritmética é nula, o resultado será nulo.

Para obter mais informações, consulte "Basic Elements of SQL" no manual *SQL Reference*.

# Definindo um Apelido de Coluna

**Um apelido de coluna:**

- **Renomeia um cabeçalho de coluna**
- **É útil em cálculos**
- **Aparece imediatamente após o nome da coluna (Também é possível incluir a palavra-chave opcional AS entre o nome e o apelido da coluna.)**
- **Requer aspas duplas quando contém espaços ou caracteres especiais, ou quando faz distinção entre maiúsculas e minúsculas**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Apelidos de Colunas

Quando exibe o resultado de uma consulta, o *iSQL\*Plus* geralmente usa o nome da coluna selecionada como seu cabeçalho. Como esse cabeçalho pode não ser descritivo, talvez seja difícil compreendê-lo. Para alterar um cabeçalho de coluna, use um apelido.

Especifique o apelido após a coluna na lista **SELECT** usando um espaço como separador. Por default, os cabeçalhos de apelidos aparecem em maiúsculas. Se o apelido contiver espaços ou caracteres especiais (como # ou \$), ou se fizer distinção entre maiúsculas e minúsculas, delimite-o por aspas duplas (" ").

# Usando Apelidos de Colunas

```
SELECT last_name AS name, commission_pct comm
FROM employees;
```

NAME	COMM
King	
Kochhar	
De Haan	

...

20 rows selected.

```
SELECT last_name "Name", salary*12 "Annual Salary"
FROM employees;
```

Name	Annual Salary
King	288000
Kochhar	204000
De Haan	204000

...

20 rows selected.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Apelidos de Colunas (continuação)

O primeiro exemplo exibe os nomes e os percentuais de comissão de todos os funcionários. Observe que a palavra-chave opcional AS foi usada antes do apelido de coluna name. O resultado da consulta será o mesmo com ou sem a inclusão da palavra-chave AS. Observe também que, na instrução SQL, os apelidos das colunas, name e comm, estão em minúsculas, enquanto o resultado da consulta exibe os cabeçalhos das colunas em maiúsculas. Como mencionado em um slide anterior, os cabeçalhos das colunas aparecem em maiúsculas por default.

O segundo exemplo exibe os sobrenomes e os salários anuais de todos os funcionários. Como Annual Salary contém um espaço, ele foi delimitado por aspas duplas. Observe que o cabeçalho da coluna na saída é exatamente igual ao apelido da coluna.

# Operador de Concatenação

Um operador de concatenação:

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

```
SELECT last_name || job_id AS "Employees"
FROM employees;
```

Employees
KingAD_PRES
KochharAD_VP
De HaanAD_VP
...

20 rows selected.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Operador de Concatenação

Você pode vincular uma coluna a outras colunas, expressões aritméticas ou valores de constantes para criar uma expressão de caracteres usando o *operador de concatenação* (||). As colunas nos dois lados do operador são combinadas para formar uma única coluna de saída.

No exemplo, LAST\_NAME e JOB\_ID são concatenadas e recebem o apelido Employees. Observe que o sobrenome do funcionário e o código do cargo são combinados para formar uma única coluna de saída.

A palavra-chave AS antes do apelido facilita a leitura da cláusula SELECT.

### Valores Nulos com o Operador de Concatenação

Se você concatenar um valor nulo com uma string de caracteres, o resultado será uma string de caracteres. LAST\_NAME || NULL resulta em LAST\_NAME.

## Strings de Caracteres Literais

- Um literal é um caractere, uma data ou um número incluído na instrução `SELECT`.
- É necessário delimitar os valores dos literais de caractere e data por aspas simples.
- Cada string de caracteres da saída corresponde a apenas uma linha retornada.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

### Strings de Caracteres Literais

Um literal é um caractere, uma data ou um número incluído na lista `SELECT` que não constitui um nome ou um apelido de coluna. Ele é impresso para cada linha retornada. É possível incluir strings de literais de texto em formato livre no resultado da consulta. Essas strings são tratadas como uma coluna na lista `SELECT`.

Os literais de caractere e data *devem* ser delimitados por aspas simples ( ' '); em literais de número, as aspas não são necessárias.

# Usando Strings de Caracteres Literais

```
SELECT last_name || ' is a ' || job_id  
       AS "Employee Details"  
FROM   employees;
```

Employee Details
King is a AD_PRES
Kochhar is a AD_VP
De Haan is a AD_VP
Hunold is a IT_PROG
Ernst is a IT_PROG
Lorentz is a IT_PROG
Mourgos is a ST_MAN
Rajs is a ST_CLERK

...  
20 rows selected.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Strings de Caracteres Literais (continuação)

O exemplo do slide exibe os sobrenomes e os códigos dos cargos de todos os funcionários. O cabeçalho da coluna é Employee Details. Observe os espaços entre as aspas simples na instrução SELECT. Esses espaços melhoram a legibilidade da saída.

No exemplo a seguir, o sobrenome e o salário de cada funcionário são concatenados com um literal para que as linhas retornadas sejam mais significativas:

```
SELECT last_name || ': 1 Month salary = ' || salary Monthly  
FROM   employees;
```

MONTHLY
King: 1 Month salary = 24000
Kochhar: 1 Month salary = 17000
De Haan: 1 Month salary = 17000
Hunold: 1 Month salary = 9000
Ernst: 1 Month salary = 6000
Lorentz: 1 Month salary = 4200
Mourgos: 1 Month salary = 5800
Rajs: 1 Month salary = 3500

...  
20 rows selected.



## Operador de Aspas (q) Alternativo

- Especifique seu próprio delimitador de aspas
- Escolha qualquer delimitador
- Melhore a legibilidade e a utilização

```
SELECT department name ||  
       q'[, it's assigned Manager Id: ]'  
       || manager_id  
       AS "Department and Manager"  
FROM departments;
```

Department and Manager
Administration, it's assigned manager ID: 200
Marketing, it's assigned manager ID: 201
Shipping, it's assigned manager ID: 124
...

8 rows selected.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Operador de Aspas (q) Alternativo

Várias instruções SQL usam literais de caractere em expressões ou condições. Se o próprio literal contiver aspas simples, você poderá usar o operador de aspas (q) e escolher o delimitador de aspas que desejar.

Você pode escolher qualquer delimitador conveniente, single-byte ou multi-byte, ou um destes pares de caracteres: [ ], { }, ( ) ou < >.

No exemplo mostrado, a string contém aspas simples, que normalmente é interpretada como um delimitador de uma string de caracteres. Entretanto, o uso do operador q requer a utilização de colchetes [] como o delimitador de aspas. A string entre os delimitadores de colchetes é interpretada como uma string de caracteres literais.

# Linhas Duplicadas

A exibição default de consultas mostra todas as linhas, inclusive as linhas duplicadas.

```
SELECT department_id  
FROM employees;
```

1

DEPARTMENT_ID
90
90
90

...

20 rows selected.

```
SELECT DISTINCT department_id  
FROM employees;
```

2

DEPARTMENT_ID
10
20
50

...

8 rows selected.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Linhas Duplicadas

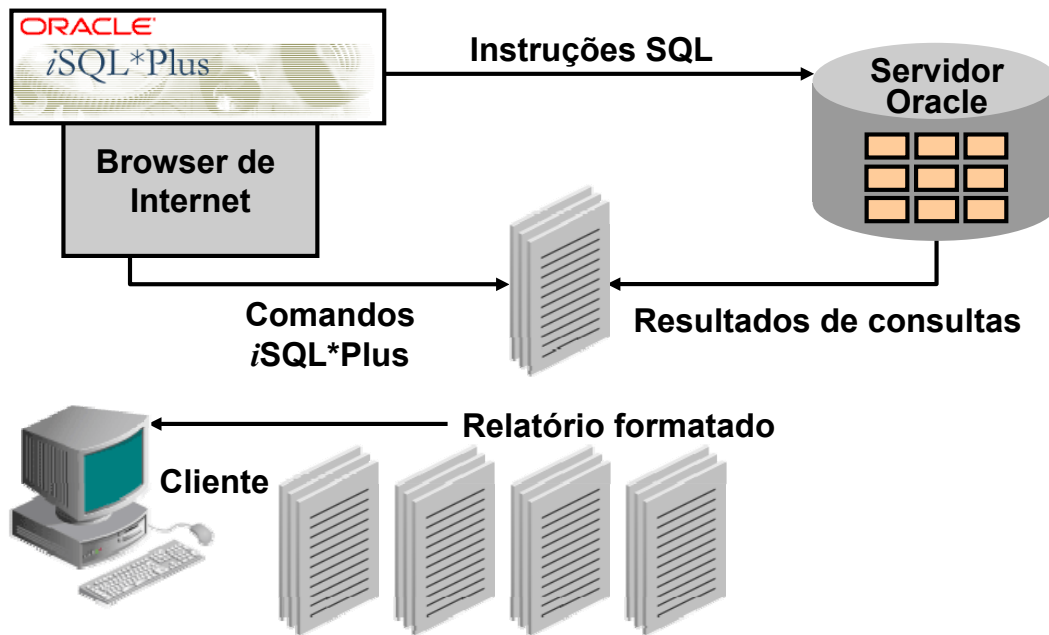
A menos que seja feita outra especificação, o iSQL\*Plus exibe os resultados de uma consulta sem eliminar as linhas duplicadas. O primeiro exemplo do slide exibe todos os números de departamentos da tabela EMPLOYEES. Observe que os números de departamentos são repetidos.

Para eliminar linhas duplicadas do resultado, inclua a palavra-chave DISTINCT na cláusula SELECT logo após a palavra-chave SELECT. No segundo exemplo do slide, a tabela EMPLOYEES contém, na verdade, 20 linhas, mas existem apenas sete números de departamentos exclusivos na tabela.

Você pode especificar várias colunas após o qualificador DISTINCT. Esse qualificador afeta todas as colunas selecionadas e o resultado são todas as combinações distintas das colunas.

```
SELECT DISTINCT department_id, job_id  
FROM employees;
```

# Interação entre SQL e *iSQL\*Plus*



Copyright © 2004, Oracle. Todos os direitos reservados.

ORACLE

## SQL e *iSQL\*Plus*

*SQL* é uma linguagem de comandos para a comunicação com o servidor Oracle a partir de qualquer ferramenta ou aplicação. O Oracle SQL contém várias extensões.

*iSQL\*Plus* é uma ferramenta Oracle que reconhece e submete instruções SQL ao servidor Oracle para execução e contém sua própria linguagem de comandos.

### Recursos de SQL

- Pode ser usada por vários usuários, incluindo aqueles com pouca ou nenhuma experiência em programação
- É uma linguagem não procedural
- É uma linguagem semelhante ao idioma inglês

### Recursos de *iSQL\*Plus*

- É acessado de um browser
- Aceita instruções SQL
- Possibilita a edição on-line para modificar instruções SQL
- Controla definições de ambiente
- Formata resultados de consultas em um relatório básico
- Acessa bancos de dados locais e remotos

# Instruções SQL e Comandos iSQL\*Plus

## SQL

- Uma linguagem
- Padrão ANSI
- Não é possível abreviar palavras-chave.
- As instruções manipulam dados e definições de tabelas no banco de dados.

Instruções  
SQL

## iSQL\*Plus

- Um ambiente
- Propriedade Oracle
- É possível abreviar palavras-chave.
- Os comandos não permitem a manipulação de valores no banco de dados.
- Executado em um browser
- Carregado centralmente; não precisa ser implementado em cada máquina

Comandos  
iSQL\*Plus

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## SQL e iSQL\*Plus (continuação)

A seguinte tabela compara SQL e iSQL\*Plus:

SQL	iSQL*Plus
É uma linguagem para a comunicação com o servidor Oracle a fim de acessar os dados	Reconhece instruções SQL e as envia ao servidor
É baseada no padrão ANSI (American National Standards Institute) SQL	É a interface proprietária da Oracle para a execução de instruções SQL
Recupera dados; manipula dados e definições de tabelas no banco de dados	Não permite a manipulação de valores no banco de dados
Não tem um caractere de continuação	Utiliza um traço (–) como caractere de continuação se o comando ultrapassa uma linha
Não pode ser abreviada	Pode ser abreviado
Utiliza functions para aplicar formatação	Utiliza comandos para formatar dados

# Visão Geral do iSQL\*Plus

Depois de efetuar login no iSQL\*Plus, você pode:

- **Descrever estruturas de tabelas**
- **Informar, executar e editar instruções SQL**
- **Salvar ou incluir instruções SQL em arquivos**
- **Executar ou editar instruções armazenadas em arquivos de script salvos**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## iSQL\*Plus

O iSQL\*Plus é um ambiente no qual você pode:

- Executar instruções SQL para recuperar, modificar, adicionar e remover dados no banco de dados
- Formatar, armazenar e imprimir resultados de consultas na forma de relatórios, bem como realizar cálculos com base nesses resultados
- Criar arquivos de script a fim de armazenar instruções SQL para uso futuro

É possível dividir os comandos iSQL\*Plus nestas principais categorias:

Categoria	Objetivo
Ambiente	Afeta o comportamento geral das instruções SQL na sessão
Formato	Formata os resultados da consulta
Manipulação de arquivos	Salva instruções em arquivos de script de texto e executa instruções a partir desses arquivos
Execução	Envia instruções SQL do browser para o servidor Oracle
Edição	Modifica instruções SQL na janela Edit
Interação	Permite criar e especificar variáveis para instruções SQL, imprimir valores de variáveis e imprimir mensagens na tela
Diversos	Tem vários comandos para estabelecer conexão com o banco de dados, manipular o ambiente iSQL*Plus e exibir definições de colunas

# Efetuando Login no iSQL\*Plus

## No ambiente de browser:

Address <http://esslin05:5560/isqlplus/> Go

Links [Class Accounts!](#) [Classroom Support Links](#) [Global Education](#) [Oracle Online Evaluations](#)

**ORACLE**  
*iSQL\*Plus*

[Help](#)

**Login**

\* Indicates required field

\* Username

\* Password

Connect Identifier

Login

**ORACLE**

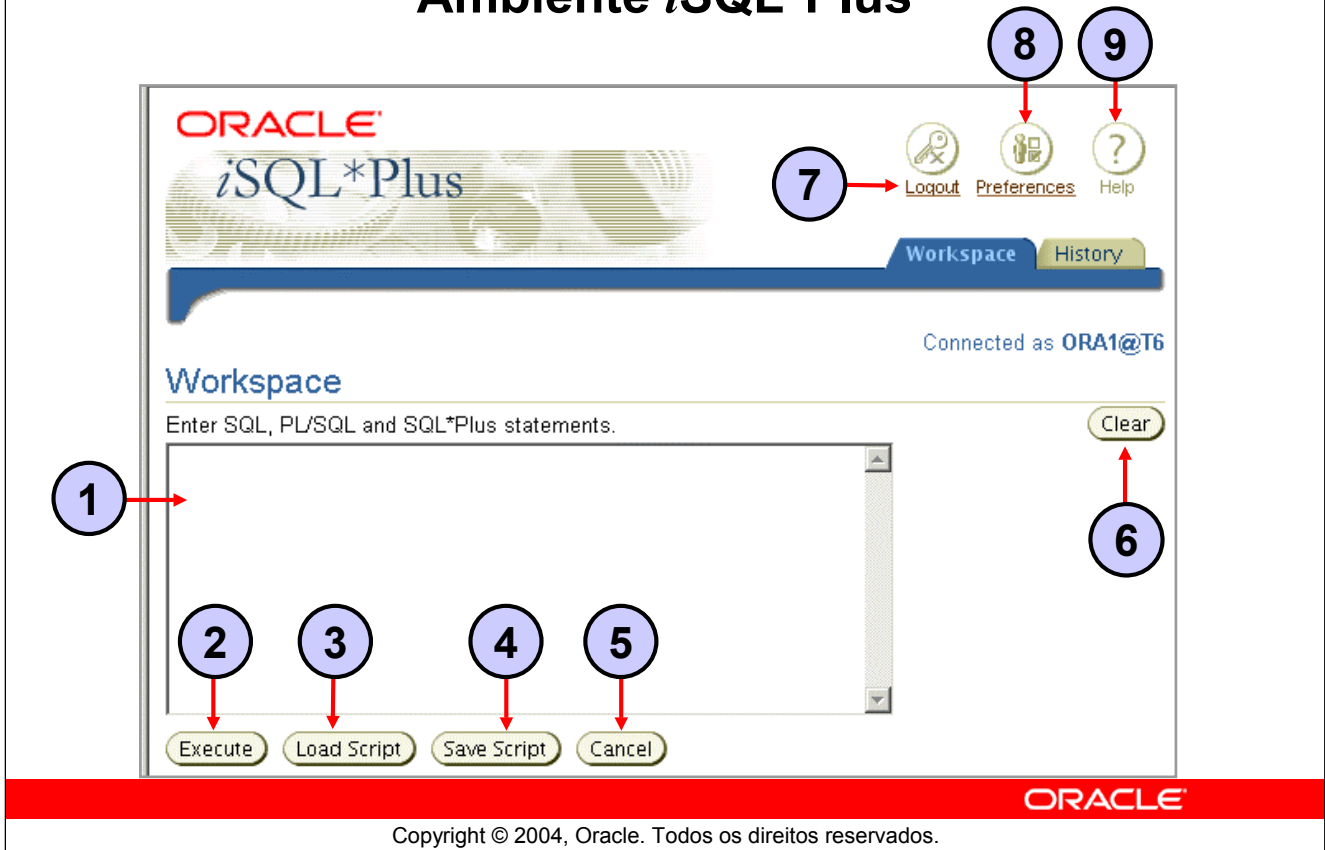
Copyright © 2004, Oracle. Todos os direitos reservados.

## Efetuando Login no iSQL\*Plus

Para efetuar login em um ambiente de browser:

1. Inicie o browser.
2. Informe o endereço URL do ambiente iSQL\*Plus.
3. Na página Login, informe os valores apropriados nos campos Username, Password e Connect Identifier.

## Ambiente iSQL\*Plus



### Ambiente iSQL\*Plus

No browser, a página Workspace do iSQL\*Plus tem as seguintes áreas principais:

1. **Caixa de texto:** Área em que você digita os comandos iSQL\*Plus e as instruções SQL
2. **Botão Execute:** Clique neste botão para executar os comandos e as instruções da caixa de texto
3. **Botão Load Script:** Exibe um form que permite identificar um caminho e um nome de arquivo, ou um URL, com os comandos SQL, PL/SQL ou SQL\*Plus a serem inseridos na caixa de texto
4. **Botão Save Script:** Salva o conteúdo da caixa de texto em um arquivo
5. **Botão Cancel:** Interrompe a execução do comando na caixa de texto
6. **Botão Clear Screen:** Clique neste botão para remover o texto contido na caixa
7. **Ícone Logout:** Clique nele para encerrar a sessão do iSQL\*Plus e retornar à página Login do iSQL\*Plus
8. **Ícone Preferences:** Clique nele para alterar a configuração da interface, a configuração do sistema ou a senha
9. **Ícone Help:** Permite acesso à documentação de ajuda do iSQL\*Plus

## Exibindo a Estrutura de Tabelas

Use o comando *iSQL\*Plus* **DESCRIBE** para exibir a estrutura de uma tabela:

```
DESC[RIBE] tablename
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

### Exibindo a Estrutura de Tabelas

No *iSQL\*Plus*, é possível exibir a estrutura de uma tabela com o comando **DESCRIBE**. O comando exibe os nomes de colunas e os tipos de dados, além de mostrar se uma coluna *deve* conter dados (ou seja, se apresenta uma constraint **NOT NULL**).

Na sintaxe, *tablename* é o nome de uma tabela, view ou sinônimo existente que esteja acessível ao usuário.



# Exibindo a Estrutura de Tabelas

```
DESCRIBE employees
```

Name	Null?	Type
EMPLOYEE_ID	NOT NULL	NUMBER(6)
FIRST_NAME		VARCHAR2(20)
LAST_NAME	NOT NULL	VARCHAR2(25)
EMAIL	NOT NULL	VARCHAR2(25)
PHONE_NUMBER		VARCHAR2(20)
HIRE_DATE	NOT NULL	DATE
JOB_ID	NOT NULL	VARCHAR2(10)
SALARY		NUMBER(8,2)
COMMISSION_PCT		NUMBER(2,2)
MANAGER_ID		NUMBER(6)
DEPARTMENT_ID		NUMBER(4)

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Exibindo a Estrutura de Tabelas (continuação)

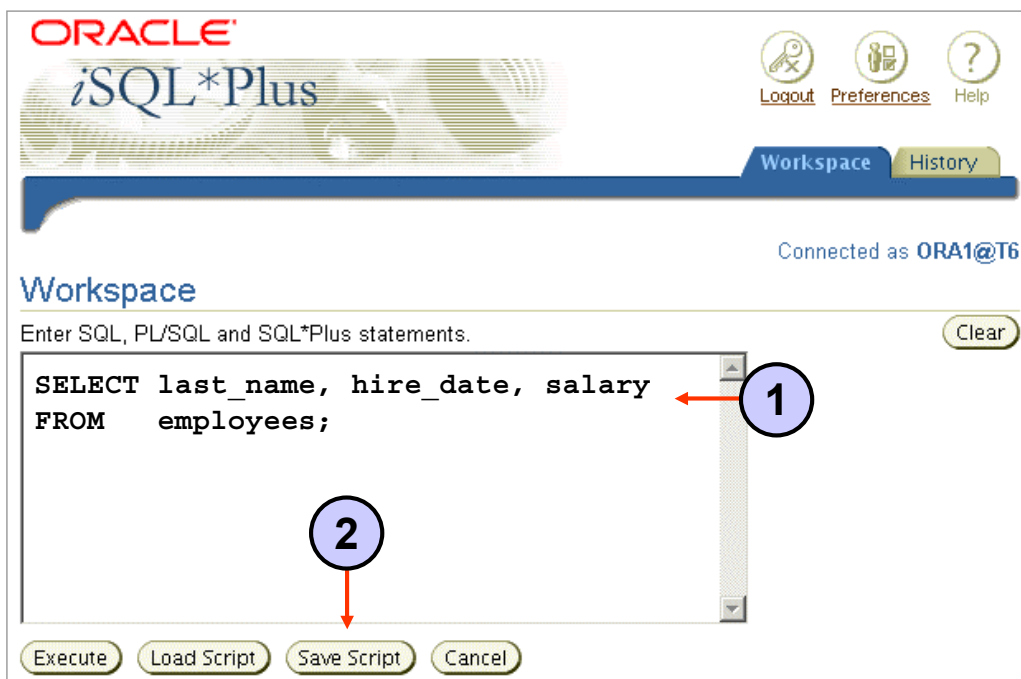
O exemplo do slide exibe as informações sobre a estrutura da tabela DEPARTMENTS.

Na exibição resultante, *Null?* indica que os valores dessa coluna podem ser desconhecidos. NOT NULL indica que uma coluna deve conter dados. *Type* exibe o tipo de dados de uma coluna.

Os tipos de dados são descritos nesta tabela:

Tipo de Dados	Descrição
NUMBER ( <i>p</i> , <i>s</i> )	Valor numérico com um número máximo de <i>p</i> dígitos e <i>s</i> dígitos à direita da vírgula decimal
VARCHAR2 ( <i>s</i> )	Valor de caractere de tamanho variável cujo tamanho máximo é igual a <i>s</i>
DATE	Valor de data e horário entre 1º de janeiro de 4712 A.C. e 31 de dezembro de 9999 D.C.
CHAR ( <i>s</i> )	Valor de caractere de tamanho fixo cujo tamanho é igual a <i>s</i>

## Interagindo com Arquivos de Script



Copyright © 2004, Oracle. Todos os direitos reservados.

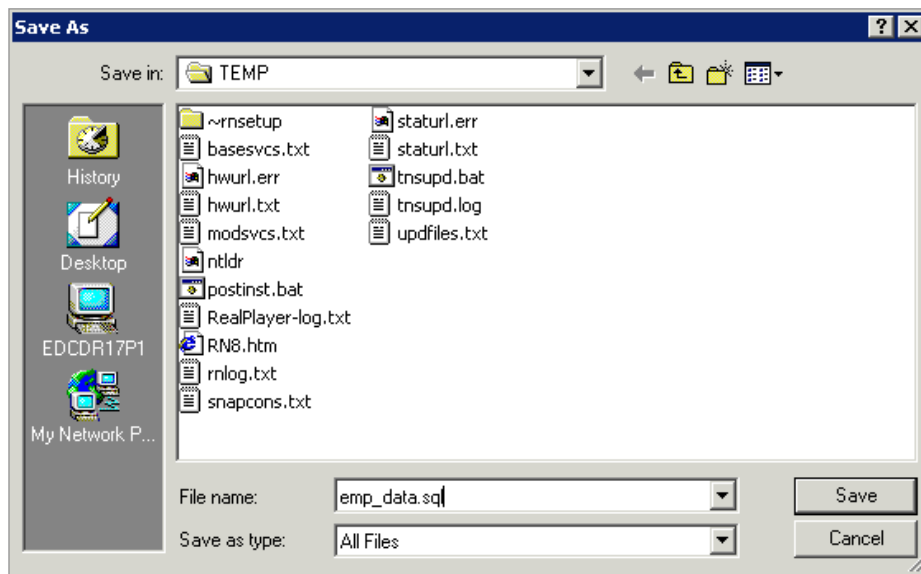
## Interagindo com Arquivos de Script

### Incluindo Instruções e Comandos em um Arquivo de Script de Texto

Você pode salvar comandos e instruções da caixa de texto do iSQL\*Plus em um arquivo de script de texto da seguinte forma:

1. Digite as instruções SQL na caixa de texto do iSQL\*Plus.
2. Clique no botão Save Script. Essa ação abre a caixa de diálogo File Save do Windows. Identifique o nome do arquivo. A extensão `.uix` é usada como default. É possível alterar o tipo de arquivo para um arquivo de texto ou salvá-lo como um arquivo `.sql`.

# Interagindo com Arquivos de Script



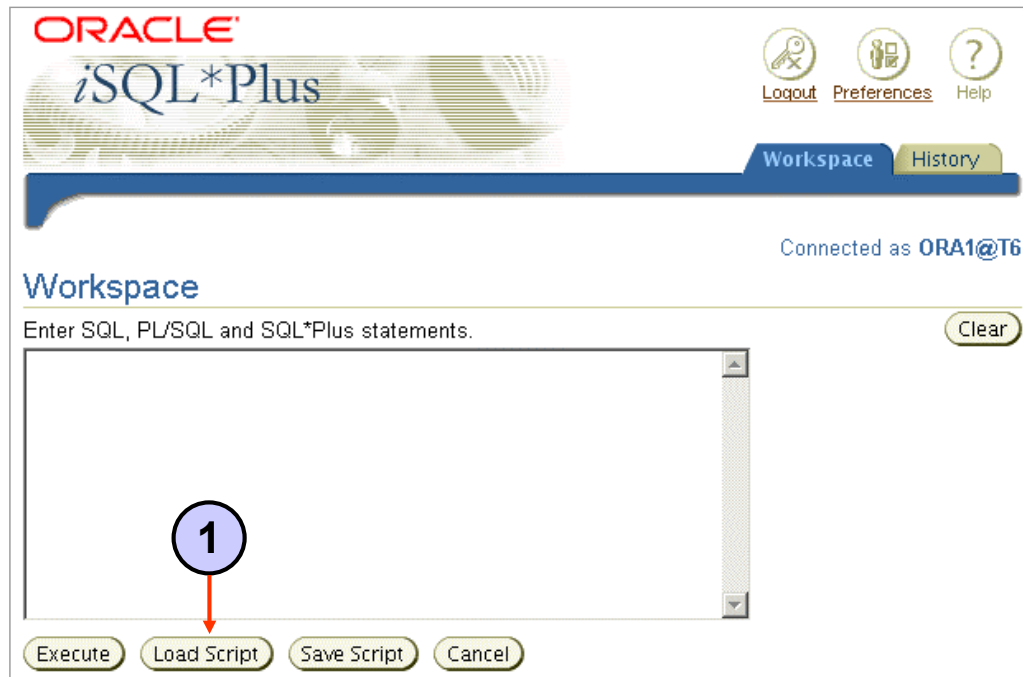
ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Interagindo com Arquivos de Script (continuação)

No exemplo mostrado, a instrução SQL `SELECT` especificada na caixa de texto é salva no arquivo `emp_data.sql`. Você pode escolher o tipo do arquivo, o nome do arquivo e o local em que deseja salvar o arquivo de script.

# Interagindo com Arquivos de Script



## Interagindo com Arquivos de Script (continuação)

### Usando Instruções e Comandos de um Arquivo de Script no iSQL\*Plus

Você pode usar os comandos e as instruções salvos anteriormente em um arquivo de script do iSQL\*Plus da seguinte forma:

1. Clique no botão Load Script. Essa ação abre um form que permite digitar o nome do arquivo ou um URL com os comandos SQL, PL/SQL ou SQL\*Plus a serem informados na caixa de texto.

## Interagindo com Arquivos de Script

ORACLE<sup>®</sup>  
iSQL\*Plus

Logout Preferences Help

Workspace History

Connected as ORA1@T6

### Load Script

Enter a URL, or a path and file name of the script to load.

Cancel Load

URL

File  Browse...

Cancel Load

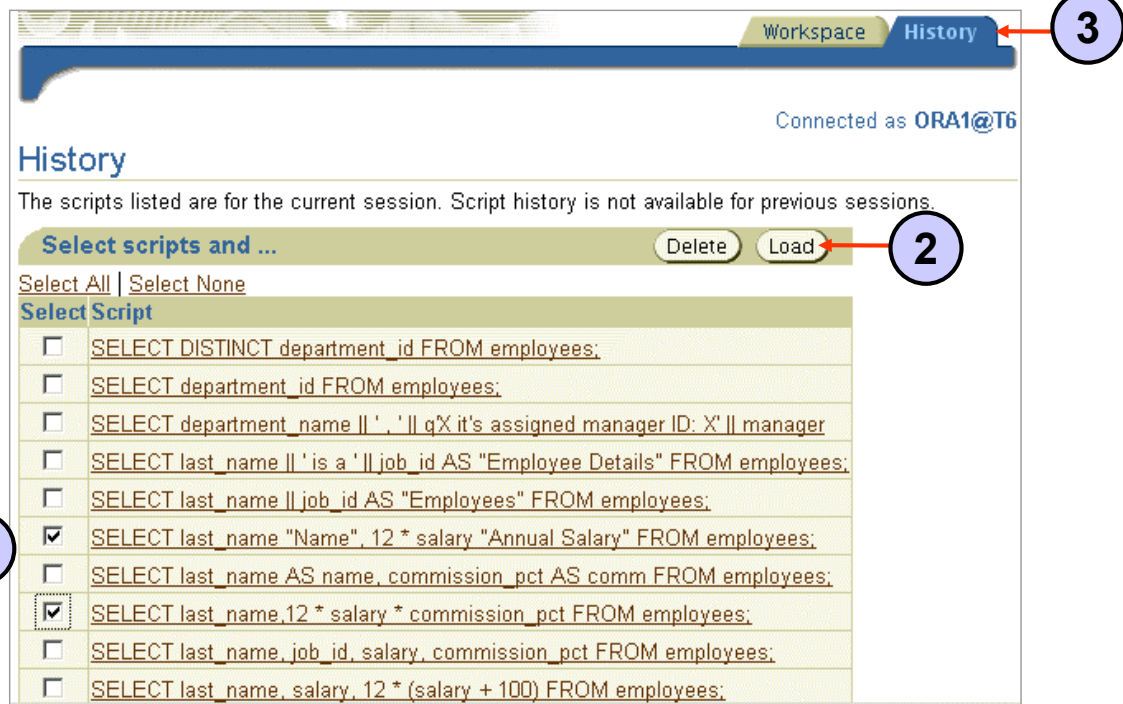
Workspace | History | Logout | Preferences | Help

Copyright © 2004, Oracle. All rights reserved.

### Interagindo com Arquivos de Script (continuação)

2. Informe o nome e o caminho do script ou a localização do URL. Se preferir, você poderá clicar no botão Browse para procurar o nome e a localização do script.
3. Clique no botão Load para incluir o conteúdo do arquivo ou a localização do URL na caixa de texto.

## Página History do iSQL\*Plus



ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

### Executando Instruções Anteriores

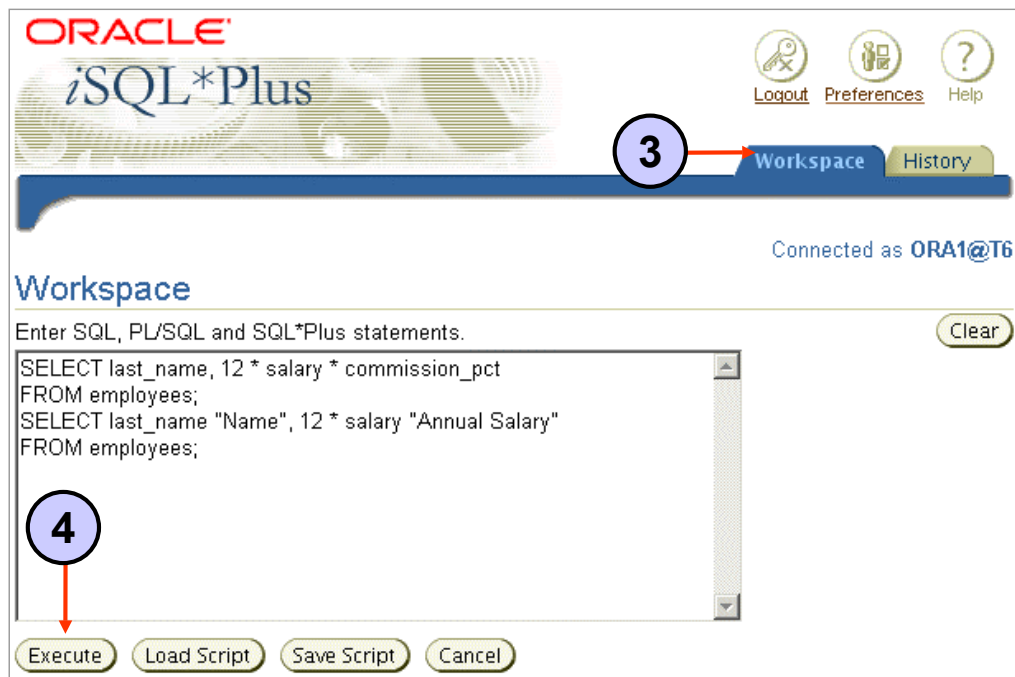
A página History do iSQL\*Plus permite executar instruções anteriores na sessão. Essa página mostra as instruções SQL e os comandos iSQL\*Plus executados mais recentemente. Para reexecutar as instruções:

1. Selecione a instrução que deseja executar.
2. Clique no botão Load.

### Observação

- É possível controlar o número de instruções mostradas na página History com as configurações de Preferences.
- Você pode optar por deletar as instruções selecionadas clicando no botão Delete.

## Página History do iSQL\*Plus



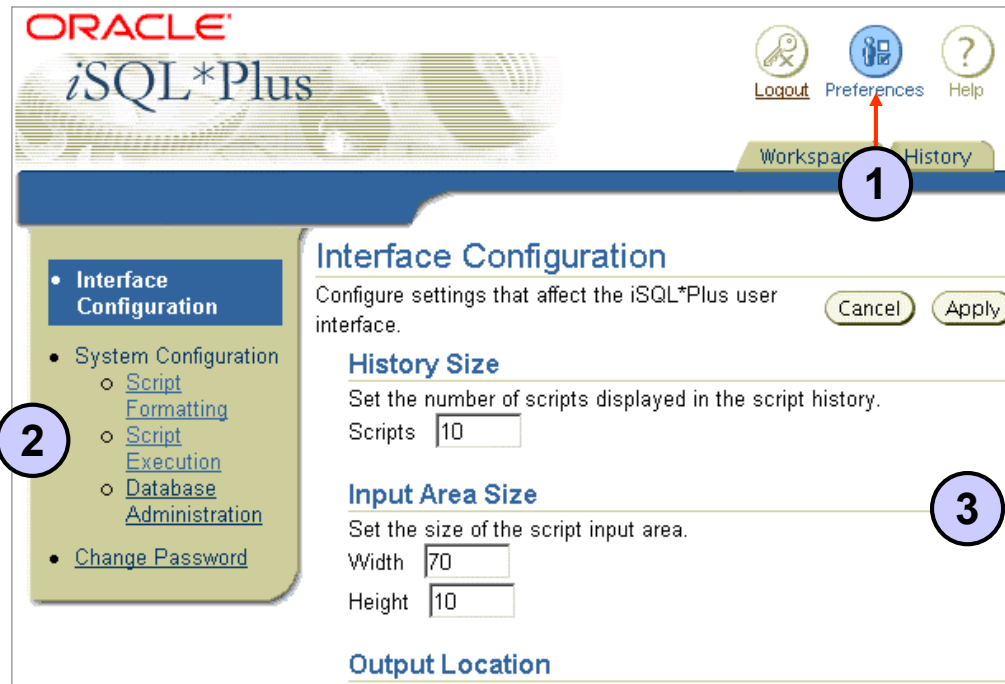
ORACLE<sup>®</sup>

Copyright © 2004, Oracle. Todos os direitos reservados.

### Executando Instruções Anteriores (continuação)

3. Retorne à página Workspace.
4. Clique no botão Execute para executar os comandos especificados na caixa de texto.

# Definindo Preferências do iSQL\*Plus



Copyright © 2004, Oracle. Todos os direitos reservados.

ORACLE

## Preferências do iSQL\*Plus

- Para definir preferências relativas à sessão do iSQL\*Plus, clique no ícone Preferences.
- As preferências são divididas em categorias. É possível definir preferências relacionadas à formatação e à execução de scripts, bem como à administração de banco de dados. Você também pode alterar sua senha.
- Quando você escolhe uma categoria de preferência, um form é exibido e permite definir as preferências dessa categoria.



## Definindo a Preferência de Localização da Saída

**Interface Configuration**

Configure settings that affect the iSQL\*Plus user interface. Cancel Apply

**History Size**  
Set the number of scripts displayed in the script history.  
Scripts

**Input Area Size**  
Set the size of the script input area.  
Width   
Height

**Output Location**  
Set where script output is displayed.

- ☐ Below Input Area
- ☒ Save to HTML File
- ☐ Printable output in new browser window
- ☐ Printable output in same browser window

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

### Alterando a Localização da Saída

É possível enviar os resultados gerados por uma instrução SQL ou um comando iSQL\*Plus para a tela (default), um arquivo ou outra janela do browser.

Na página Preferences:

1. Selecione uma opção em Output Location.
2. Clique no botão Apply.

# Sumário

Nesta lição, você aprendeu a:

- Criar uma instrução **SELECT** que:
  - Retorna todas as linhas e colunas de uma tabela
  - Retorna as colunas especificadas de uma tabela
  - Usa apelidos para exibir cabeçalhos de colunas mais descritivos
- Usar o ambiente **iSQL\*Plus** para criar, salvar e executar instruções SQL e comandos **iSQL\*Plus**

```
SELECT * | { [DISTINCT] column / expression [alias], ... }  
FROM table;
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Instrução SELECT

Nesta lição, você aprendeu a recuperar dados de uma tabela do banco de dados com a instrução **SELECT**.

```
SELECT * | { [DISTINCT] column [alias], ... }  
FROM table;
```

Na sintaxe:

<b>SELECT</b>	é uma lista de uma ou mais colunas
<b>*</b>	seleciona todas as colunas
<b>DISTINCT</b>	suprime as colunas duplicadas
<i>column   expression</i>	seleciona a coluna nomeada ou a expressão
<i>alias</i>	fornece cabeçalhos distintos às colunas selecionadas
<b>FROM table</b>	especifica a tabela que contém as colunas

## iSQL\*Plus

**iSQL\*Plus** é um ambiente de execução que permite enviar instruções SQL ao servidor de banco de dados, bem como editar e salvar essas instruções. É possível executar as instruções no prompt SQL ou em um arquivo de script.

# Exercício 1: Visão Geral

**Este exercício aborda os seguintes tópicos:**

- **Selecionando todos os dados de tabelas diferentes**
- **Descrivendo a estrutura de tabelas**
- **Realizando cálculos aritméticos e especificando nomes de colunas**
- **Usando o *iSQL\*Plus***

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

## Exercício 1: Visão Geral

Este é o primeiro de vários exercícios deste curso. As soluções (se forem necessárias) estão no Apêndice A. Os exercícios têm como objetivo abordar todos os tópicos apresentados na lição correspondente.

*Observe o seguinte local dos arquivos de laboratório:*

*E:\labs\SQL1\labs*

*Se você receber uma solicitação para salvar esses arquivos, salve-os nesse local.*

Para iniciar o *iSQL\*Plus*, inicie o browser. É necessário informar um URL para acessar o *iSQL\*Plus*. O URL requer o nome do host, que será fornecido pelo instrutor. Informe o comando a seguir, substituindo o nome do host pelo valor fornecido pelo instrutor:

`http://<HOSTNAME:5561>/isqlplus`

Alguns exercícios podem ser antecidos pelas expressões "Se tiver tempo" ou "Se quiser tomar parte em mais um desafio". Faça esses exercícios somente quando terminar todos os outros no tempo designado e se desejar mais um desafio para testar suas habilidades.

Desenvolva os exercícios devagar e com precisão. Como teste, você pode salvar e executar arquivos de comandos. Em caso de dúvidas, consulte o instrutor.

## Exercício 1

### Parte 1

Teste seu conhecimento:

1. Inicie uma sessão do *iSQL\*Plus* com o ID de usuário e a senha fornecidos pelo instrutor.
2. Os comandos *iSQL\*Plus* acessam o banco de dados.  
Verdadeiro/Falso
3. Esta instrução *SELECT* é executada com êxito:  

```
SELECT last_name, job_id, salary AS Sal
FROM employees;
```

Verdadeiro/Falso

4. Esta instrução *SELECT* é executada com êxito:  

```
SELECT *
FROM job_grades;
```

Verdadeiro/Falso

5. Há quatro erros de codificação na instrução a seguir. Você consegue identificá-los?

```
SELECT      employee_id, last_name
sal x 12    ANNUAL SALARY
FROM        employees;
```

### Parte 2

Observe o seguinte local dos arquivos de laboratório:

*E:\labs\SQL1\labs*

Se você receber uma solicitação para salvar esses arquivos, salve-os nesse local

Para iniciar o *iSQL\*Plus*, inicie o browser. É necessário informar um URL para acessar o *iSQL\*Plus*. O URL requer o nome do host, que será fornecido pelo instrutor. Informe o comando a seguir, substituindo o nome do host pelo valor fornecido pelo instrutor:  
<http://<HOSTNAME:5561>/isqlplus>

Você foi admitido como programador SQL da Acme Corporation. Sua primeira tarefa é criar alguns relatórios com base nos dados das tabelas de recursos humanos.

6. Sua primeira tarefa é determinar a estrutura da tabela *DEPARTMENTS* e seu conteúdo.

Name	Null?	Type
DEPARTMENT_ID	NOT NULL	NUMBER(4)
DEPARTMENT_NAME	NOT NULL	VARCHAR2(30)
MANAGER_ID		NUMBER(6)
LOCATION_ID		NUMBER(4)

## Exercício 1 (continuação)

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
50	Shipping	124	1500
60	IT	103	1400
80	Sales	149	2500
90	Executive	100	1700
110	Accounting	205	1700
190	Contracting		1700

8 rows selected.

7. Você precisa determinar a estrutura da tabela EMPLOYEES.

Name	Null?	Type
EMPLOYEE_ID	NOT NULL	NUMBER(6)
FIRST_NAME		VARCHAR2(20)
LAST_NAME	NOT NULL	VARCHAR2(25)
EMAIL	NOT NULL	VARCHAR2(25)
PHONE_NUMBER		VARCHAR2(20)
HIRE_DATE	NOT NULL	DATE
JOB_ID	NOT NULL	VARCHAR2(10)
SALARY		NUMBER(8,2)
COMMISSION_PCT		NUMBER(2,2)
MANAGER_ID		NUMBER(6)
DEPARTMENT_ID		NUMBER(4)

O departamento de recursos humanos deseja executar uma consulta para exibir o sobrenome, o código do cargo, a data de admissão e o telefone de cada funcionário, com o número do funcionário exibido primeiro. Forneça o apelido STARTDATE para a coluna HIRE\_DATE. Salve a instrução SQL no arquivo lab\_01\_07.sql para encaminhar esse arquivo ao departamento de recursos humanos.

**Exercício 1 (continuação)**

8. Teste a consulta no arquivo lab\_01\_07.sql para verificar se é executada corretamente.

EMPLOYEE_ID	LAST_NAME	JOB_ID	STARTDATE
100	King	AD_PRES	17-JUN-87
101	Kochhar	AD_VP	21-SEP-89
102	De Haan	AD_VP	13-JAN-93
103	Hunold	IT_PROG	03-JAN-90
104	Ernst	IT_PROG	21-MAY-91
107	Lorentz	IT_PROG	07-FEB-99
124	Mourgos	ST_MAN	16-NOV-99
141	Rajs	ST_CLERK	17-OCT-95
142	Davies	ST_CLERK	29-JAN-97
143	Matos	ST_CLERK	15-MAR-98
144	Vargas	ST_CLERK	09-JUL-98
149	Zlotkey	SA_MAN	29-JAN-00
174	Abel	SA_REP	11-MAY-96
176	Taylor	SA_REP	24-MAR-98
...			
206	Gietz	AC_ACCOUNT	07-JUN-94

20 rows selected.

9. O departamento de recursos humanos precisa de uma consulta para exibir todos os códigos de cargo exclusivos da tabela EMPLOYEES.

JOB_ID
AC_ACCOUNT
AC_MGR
AD_ASST
AD_PRES
AD_VP
IT_PROG
MK_MAN
MK_REP
SA_MAN
SA_REP
ST_CLERK
ST_MAN

12 rows selected.

## Exercício 1 (continuação)

### Parte 3

Se tiver tempo, faça os seguintes exercícios:

10. O departamento de recursos humanos deseja cabeçalhos de coluna mais descritivos em seu relatório sobre funcionários. Copie a instrução de `lab_01_07.sql` para a caixa de texto do *iSQL\*Plus*. Nomeie os cabeçalhos de coluna como `Emp #`, `Employee`, `Job` e `Hire Date`, respectivamente. Execute a consulta novamente.

Emp #	Employee	Job	Hire Date
100	King	AD_PRES	17-JUN-87
101	Kochhar	AD_VP	21-SEP-89
102	De Haan	AD_VP	13-JAN-93
103	Hunold	IT_PROG	03-JAN-90
104	Ernst	IT_PROG	21-MAY-91
107	Lorentz	IT_PROG	07-FEB-99
124	Mourgos	ST_MAN	16-NOV-99
141	Rajs	ST_CLERK	17-OCT-95
142	Davies	ST_CLERK	29-JAN-97
143	Matos	ST_CLERK	15-MAR-98
144	Vargas	ST_CLERK	09-JUL-98
...			
206	Gietz	AC_ACCOUNT	07-JUN-94

20 rows selected.

11. O departamento de recursos humanos solicitou um relatório de todos os funcionários e os respectivos IDs de cargo. Exiba o sobrenome concatenado com o ID do cargo (separado por uma vírgula e um espaço) e nomeie a coluna como `Employee and Title`.

Employee and Title
King, AD_PRES
Kochhar, AD_VP
De Haan, AD_VP
Hunold, IT_PROG
Ernst, IT_PROG
Lorentz, IT_PROG
Mourgos, ST_MAN
Rajs, ST_CLERK
Davies, ST_CLERK
...
Gietz, AC_ACCOUNT

20 rows selected.

## Exercício 1 (continuação)

Se quiser tomar parte em mais um desafio, faça este exercício:

- Para se familiarizar com os dados da tabela EMPLOYEES, crie uma consulta para exibir todos os dados dessa tabela. Separe cada saída de coluna com uma vírgula. Nomeie o título da coluna como THE\_OUTPUT.

THE_OUTPUT										
100	Steven	King	SKING	515.123.4567	AD_PRES	,	17-JUN-87	24000	,	90
101	Neena	Kochhar	NKOCHHAR	515.123.4568	AD_VP	,	100,21-SEP-89	17000	,	90
102	Lex	De Haan	LDEHAAN	515.123.4569	AD_VP	,	100,13-JAN-93	17000	,	90
103	Alexander	Hunold	AHUNOLD	590.423.4567	IT_PROG	,	102,03-JAN-90	9000	,	60
104	Bruce	Ernst	BERNST	590.423.4568	IT_PROG	,	103,21-MAY-91	6000	,	60
107	Diana	Lorentz	DLORENTZ	590.423.5567	IT_PROG	,	103,07-FEB-99	4200	,	60
124	Kevin	Mourgos	KMOURGOS	650.123.5234	ST_MAN	,	100,16-NOV-99	5800	,	50
141	Trenna	Rajs	TRAJS	650.121.8009	ST_CLERK	,	124,17-OCT-95	3500	,	50
142	Curtis	Davies	CDAVIES	650.121.2994	ST_CLERK	,	124,29-JAN-97	3100	,	50
143	Randall	Matos	RMATOS	650.121.2874	ST_CLERK	,	124,15-MAR-98	2600	,	50
144	Peter	Vargas	PVARGAS	650.121.2004	ST_CLERK	,	124,09-JUL-98	2500	,	50
■ ■ ■										
206	William	Gietz	WGIETZ	515.123.8181	AC_ACCOUNT	,	205,07-JUN-94	8300	,	110

20 rows selected.



# 2

## Restringindo e Classificando Dados

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

# Objetivos

**Ao concluir esta lição, você será capaz de:**

- **Limitar as linhas recuperadas por uma consulta**
- **Classificar as linhas recuperadas por uma consulta**
- **Usar a substituição com E comercial no iSQL\*Plus para restringir e classificar a saída em runtime**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Objetivos

Durante a recuperação de dados do banco de dados, talvez seja necessário:

- Restringir as linhas de dados a serem exibidas
- Especificar a ordem de exibição das linhas

Esta lição explica as instruções SQL usadas para executar essas ações.

# Limitando Linhas por Seleção

## EMPLOYEES

EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
100	King	AD_PRES	90
101	Kochhar	AD_VP	90
102	De Haan	AD_VP	90
103	Hunold	IT_PROG	60
104	Ernst	IT_PROG	60
107	Lorentz	IT_PROG	60
124	Mourgos	ST_MAN	50

...

20 rows selected.

**"recuperar todos  
os funcionários do  
departamento 90"**



EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
100	King	AD_PRES	90
101	Kochhar	AD_VP	90
102	De Haan	AD_VP	90

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Limitando Linhas por Seleção

No exemplo do slide, suponha que você queira exibir todos os funcionários do departamento 90. As linhas com o valor 90 na coluna `DEPARTMENT_ID` são as únicas retornadas. Esse método de restrição é a base da cláusula `WHERE` em `SQL`.

# Limitando as Linhas Seleccionadas

- Restrinja as linhas retornadas com a cláusula WHERE:

```
SELECT * | { [DISTINCT] column/expression [alias], ... }  
FROM table  
[WHERE condition(s)];
```

- A cláusula WHERE é especificada após a cláusula FROM.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Limitando as Linhas Seleccionadas

É possível restringir as linhas retornadas por uma consulta com a cláusula WHERE. Essa cláusula contém uma condição a ser atendida e é inserida imediatamente após a cláusula FROM. Se a condição for verdadeira, a linha que atender a essa condição será retornada.

Na sintaxe:

WHERE	restringe a consulta às linhas que atendem a uma condição
<i>condition</i>	é composta de nomes de colunas, expressões, constantes e um operador de comparação

A cláusula WHERE pode comparar valores em colunas, valores literais, expressões aritméticas ou functions. Ela consiste em três elementos:

- Nome de coluna
- Condição de comparação
- Nome de coluna, constante ou lista de valores

## Usando a Cláusula WHERE

```
SELECT employee_id, last_name, job_id, department_id
FROM   employees
WHERE  department_id = 90 ;
```

EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
100	King	AD_PRES	90
101	Kochhar	AD_VP	90
102	De Haan	AD_VP	90

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

### Usando a Cláusula WHERE

No exemplo, a instrução SELECT recupera o ID, o nome, o ID do cargo e o número do departamento de todos os funcionários do departamento 90.

## Strings de Caracteres e Datas

- As strings de caracteres e os valores de data são delimitados por aspas simples.
- Os valores de caractere fazem distinção entre maiúsculas e minúsculas, e os valores de data fazem distinção de formato.
- O formato default de data é DD-MON-RR.

```
SELECT last_name, job_id, department_id
FROM employees
WHERE last_name = 'Whalen' ;
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Strings de Caracteres e Datas

Na cláusula WHERE, as strings de caracteres e as datas devem ser delimitadas por aspas simples ( ' '), mas não as constantes numéricas.

Todas as pesquisas de caracteres fazem distinção entre maiúsculas e minúsculas. No exemplo a seguir, não são retornadas linhas, pois a tabela EMPLOYEES armazena todos os sobrenomes em maiúsculas e minúsculas:

```
SELECT last_name, job_id, department_id
FROM employees
WHERE last_name = 'WHALEN' ;
```

Os bancos de dados Oracle armazenam datas em um formato numérico interno, que representa o século, o ano, o mês, o dia, as horas, os minutos e os segundos. A exibição default de data é DD-MON-RR.

**Observação:** Para obter detalhes sobre o formato RR e como alterar o formato de data default, consulte a próxima lição.

## Condições de Comparação

Operador	Significado
=	Igual a
>	Maior que
>=	Maior que ou igual a
<	Menor que
<=	Menor que ou igual a
<>	Diferente de
BETWEEN ...AND...	Entre dois valores (inclusivo)
IN(set)	Corresponde a qualquer
LIKE	Corresponde a um padrão de
IS NULL	É um valor nulo

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

### Condições de Comparação

As condições de comparação são usadas em condições que comparam uma expressão a outro valor ou expressão. Elas são usadas na cláusula WHERE no seguinte formato:

#### Sintaxe

```
... WHERE expr operator value
```

#### Exemplo

```
... WHERE hire_date='01-JAN-95'
... WHERE salary >=6000
... WHERE last_name='Smith'
```

Não é possível usar um apelido na cláusula WHERE.

**Observação:** Os símbolos != e ^= também podem representar a condição *diferente de*.

## Usando Condições de Comparação

```
SELECT last_name, salary
FROM   employees
WHERE  salary <= 3000 ;
```

LAST_NAME	SALARY
Matos	2600
Vargas	2500

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

### Usando Condições de Comparação

No exemplo, a instrução SELECT recupera, com base na tabela EMPLOYEES, o sobrenome e o salário de todos os funcionários cujo salário é menor que US\$ 3.000 ou igual a esse valor. Observe que foi fornecido um valor explícito na cláusula WHERE. O valor explícito 3000 é comparado ao valor do salário na coluna SALARY da tabela EMPLOYEES.



## Usando a Condição BETWEEN

Use a condição **BETWEEN** para exibir linhas com base em uma faixa de valores:

```
SELECT last_name, salary
FROM   employees
WHERE  salary BETWEEN 2500 AND 3500 ;
```

Limite inferior    Limite superior

LAST_NAME	SALARY
Rajs	3500
Davies	3100
Matos	2600
Vargas	2500

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Usando a Condição BETWEEN

É possível exibir linhas com base em uma faixa de valores usando a condição de faixa **BETWEEN**. A faixa especificada contém os limites inferior e superior.

A instrução **SELECT** do slide retorna as linhas da tabela **EMPLOYEES** relativas aos funcionários cujo salário está contido na faixa entre US\$ 2.500 e US\$ 3.500.

Os valores especificados com a condição **BETWEEN** são inclusivos. Especifique o limite inferior primeiro.

Também é possível usar a condição **BETWEEN** em valores de caractere:

```
SELECT last_name
FROM   employees
WHERE  last_name BETWEEN 'King' AND 'Smith';
```

# Usando a Condição IN

Use a condição de associação IN para testar os valores de uma lista:

```
SELECT employee_id, last_name, salary, manager_id
FROM employees
WHERE manager_id IN (100, 101, 201) ;
```

EMPLOYEE_ID	LAST_NAME	SALARY	MANAGER_ID
202	Fay	6000	201
200	Whalen	4400	101
205	Higgins	12000	101
101	Kochhar	17000	100
102	De Haan	17000	100
124	Mourgos	5800	100
149	Zlotkey	10500	100
201	Hartstein	13000	100

8 rows selected.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

## Usando a Condição IN

Para testar os valores de um conjunto de valores especificado, use a condição IN. Essa condição também é conhecida como *condição de associação*.

O exemplo do slide exibe números de funcionários, sobrenomes, salários e números de funcionário de gerentes relativos a todos os funcionários cujo número de funcionário do gerente seja 100, 101 ou 201.

É possível usar a condição IN com qualquer tipo de dados. O exemplo a seguir retorna uma linha da tabela EMPLOYEES para cada funcionário cujo sobrenome está incluído na lista de nomes da cláusula WHERE:

```
SELECT employee_id, manager_id, department_id
FROM employees
WHERE last_name IN ('Hartstein', 'Vargas');
```

Se forem usados caracteres ou datas na lista, eles deverão ser delimitados por aspas simples (' ').

## Usando a Condição LIKE

- Use a condição **LIKE** para executar pesquisas com curinga de valores válidos de strings de pesquisa.
- As condições de pesquisa podem conter números ou caracteres literais:
  - % indica zero ou vários caracteres.
  - \_ indica um caractere.

```
SELECT first_name
FROM employees
WHERE first_name LIKE 'S%';
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Usando a Condição LIKE

Nem sempre você sabe o valor exato a ser pesquisado. É possível selecionar linhas que correspondam a um padrão de caracteres usando a condição **LIKE**. A operação de correspondência a um padrão de caracteres é conhecida como pesquisa com *curinga*. É possível usar dois símbolos para criar a string de pesquisa.

Símbolo	Descrição
%	Representa qualquer seqüência de zero ou mais caracteres
_	Representa qualquer caractere simples

A instrução **SELECT** do slide retorna, com base na tabela **EMPLOYEES**, os nomes de todos os funcionários que começam com a letra *S*. Observe o *S* maiúsculo. Os nomes que começam com *s* não são retornados.

É possível usar a condição **LIKE** como um atalho para algumas comparações **BETWEEN**. O exemplo a seguir exibe os sobrenomes e as datas de admissão de todos os funcionários admitidos entre janeiro de 1995 e dezembro de 1995:

```
SELECT last_name, hire_date
FROM employees
WHERE hire_date LIKE '%95';
```

## Usando a Condição LIKE

- Você pode combinar caracteres com padrões correspondentes:

```
SELECT last_name  
FROM employees  
WHERE last_name LIKE '_o%' ;
```

LAST_NAME
Kochhar
Lorentz
Mourgos

- Você pode usar o identificador **ESCAPE** para pesquisar os símbolos % e \_ verdadeiros.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

### Combinando Caracteres Curinga

É possível usar os símbolos % e \_ em qualquer combinação com caracteres literais. O exemplo do slide exhibe os nomes de todos os funcionários cujo sobrenome inclui a letra o como o segundo caractere.

#### Opção ESCAPE

Quando você precisar de uma correspondência exata para os caracteres % e \_ verdadeiros, use a opção ESCAPE. Essa opção especifica o que é o caractere de escape. Para procurar strings contendo 'SA\_', você poderá usar a seguinte instrução SQL:

```
SELECT employee_id, last_name, job_id  
FROM employees WHERE job_id LIKE '%SA\_%' ESCAPE '\\';
```

EMPLOYEE_ID	LAST_NAME	JOB_ID
149	Zlotkey	SA_MAN
174	Abel	SA_REP
176	Taylor	SA_REP
178	Grant	SA_REP

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

# Usando as Condições NULL

Teste valores nulos com o operador IS NULL.

```
SELECT last_name, manager_id
FROM   employees
WHERE  manager_id IS NULL ;
```

LAST_NAME	MANAGER_ID
King	

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Usando as Condições NULL

As condições NULL incluem IS NULL e IS NOT NULL.

A condição IS NULL testa valores nulos. Um valor nulo é aquele que não está disponível nem designado e não é conhecido ou aplicável. Portanto, não é possível testá-lo com =, pois não pode ser igual a um valor ou diferente dele. O exemplo do slide recupera os sobrenomes e os gerentes de todos os funcionários que não estão subordinados a um gerente.

Veja outro exemplo: Para exibir o sobrenome, o ID do cargo e a comissão de todos os funcionários *sem* direito a comissão, use a seguinte instrução SQL:

```
SELECT last_name, job_id, commission_pct
FROM   employees
WHERE  commission_pct IS NULL;
```

LAST_NAME	JOB_ID	COMMISSION_PCT
King	AD_PRES	
Kochhar	AD_VP	
...		
Higgins	AC_MGR	
Gietz	AC_ACCOUNT	

16 rows selected.

# Condições Lógicas

Operador	Significado
AND	Retornará TRUE se as duas condições componentes forem verdadeiras
OR	Retornará TRUE se uma das condições componentes for verdadeira
NOT	Retornará TRUE se a condição seguinte for falsa

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

## Condições Lógicas

Uma condição lógica combina o resultado de duas condições componentes para produzir um único resultado com base nessas condições, ou inverte o resultado de uma única condição. Só será retornada uma linha se o resultado geral da condição for verdadeiro.

Três operadores lógicos estão disponíveis em SQL:

- AND
- OR
- NOT

Todos os exemplos até então especificaram apenas uma condição na cláusula WHERE. Você pode usar várias condições em uma cláusula WHERE com os operadores AND e OR.

# Usando o Operador AND

**AND exige que as duas condições sejam verdadeiras:**

```
SELECT employee_id, last_name, job_id, salary
FROM   employees
WHERE  salary >=10000
AND    job_id LIKE '%MAN%' ;
```

EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
149	Zlotkey	SA_MAN	10500
201	Hartstein	MK_MAN	13000

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

## Usando o Operador AND

No exemplo, as duas condições deverão ser verdadeiras para que sejam selecionados registros. Portanto, somente os funcionários cujos cargos contiverem a string 'MAN' e que receberem US\$ 10.000 ou mais serão selecionados.

Todas as pesquisas de caracteres fazem distinção entre maiúsculas e minúsculas. Não serão retornadas linhas se a string 'MAN' não estiver em maiúsculas. As strings de caracteres devem ser delimitadas por aspas.

### Tabela de Valores Verdadeiros com 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 uma das condições seja verdadeira:**

```
SELECT employee_id, last_name, job_id, salary
FROM   employees
WHERE  salary >= 10000
OR     job_id LIKE '%MAN%' ;
```

EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
100	King	AD_PRES	24000
101	Kochhar	AD_VP	17000
102	De Haan	AD_VP	17000
124	Mourgos	ST_MAN	5800
149	Zlotkey	SA_MAN	10500
174	Abel	SA_REP	11000
201	Hartstein	MK_MAN	13000
205	Higgins	AC_MGR	12000

8 rows selected.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

## Usando o Operador OR

No exemplo, uma das duas condições poderá ser verdadeira para que sejam selecionados registros. Portanto, os funcionários cujos IDs de cargo contiverem a string 'MAN' *ou* que receberem US\$ 10.000 ou mais serão selecionados.

### Tabela de Valores Verdadeiros com 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

```
SELECT last_name, job_id
FROM   employees
WHERE  job_id
      NOT IN ('IT_PROG', 'ST_CLERK', 'SA_REP') ;
```

LAST_NAME	JOB_ID
King	AD_PRES
Kochhar	AD_VP
De Haan	AD_VP
Mourgos	ST_MAN
Zlotkey	SA_MAN
Whalen	AD_ASST
Hartstein	MK_MAN
Fay	MK_REP
Higgins	AC_MGR
Gietz	AC_ACCOUNT

10 rows selected.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Usando o Operador NOT

O exemplo do slide exibe o sobrenome e o ID do cargo de todos os funcionários cujo ID de cargo é *diferente* de IT\_PROG, ST\_CLERK ou SA\_REP.

### Tabela de Valores Verdadeiros com NOT

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

NOT	TRUE	FALSE	NULL
	FALSE	TRUE	NULL

**Observação:** Também é possível usar o operador NOT com outros operadores SQL, como BETWEEN, LIKE e NULL.

```
... WHERE job_id NOT IN ('AC_ACCOUNT', 'AD_VP')
... WHERE salary NOT BETWEEN 10000 AND 15000
... WHERE last_name NOT LIKE '%A%'
... WHERE commission_pct IS NOT NULL
```

# Regras de Precedência

Operador	Significado
1	Operadores aritméticos
2	Operador de concatenação
3	Condições de comparação
4	IS [NOT] NULL, LIKE, [NOT] IN
5	[NOT] BETWEEN
6	Diferente de
7	Condição lógica NOT
8	Condição lógica AND
9	Condição lógica OR

**Você pode usar parênteses para sobrepor as regras de precedência.**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Regras de Precedência

As regras de precedência determinam a ordem de avaliação e cálculo das expressões. A tabela relaciona a ordem de precedência default. Você pode sobrepor a ordem default usando parênteses para delimitar as expressões a serem calculadas primeiro.

# Regras de Precedência

```
SELECT last_name, job_id, salary
FROM employees
WHERE job_id = 'SA_REP'
OR job_id = 'AD_PRES'
AND salary > 15000;
```

1

LAST_NAME	JOB_ID	SALARY
King	AD_PRES	24000
Abel	SA_REP	11000
Taylor	SA_REP	8600
Grant	SA_REP	7000

```
SELECT last_name, job_id, salary
FROM employees
WHERE (job_id = 'SA_REP'
OR job_id = 'AD_PRES')
AND salary > 15000;
```

2

LAST_NAME	JOB_ID	SALARY
King	AD_PRES	24000

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## 1. Exemplo da Precedência do Operador AND

Neste exemplo, existem duas condições:

- A primeira condição é que o ID do cargo seja AD\_PRES e o salário seja maior que US\$ 15.000.
- A segunda condição é que o ID do cargo seja SA\_REP.

Portanto, a leitura da instrução SELECT será esta:

"Selecione a linha se o funcionário for presidente e receber mais de US\$15.000, ou se ele for representante de vendas".

## 2. Exemplo da Utilização de Parênteses

Neste exemplo, existem duas condições:

- A primeira condição é que o ID do cargo seja AD\_PRES ou SA\_REP.
- A segunda condição é que o salário seja maior que US\$ 15.000.

Portanto, a leitura da instrução SELECT será esta:

"Selecione a linha se o funcionário for presidente ou representante de vendas e receber mais de US\$15.000".

## Usando a Cláusula ORDER BY

- **Classifique as linhas recuperadas com a cláusula ORDER BY:**
  - **ASC:** ordem crescente, default
  - **DESC:** ordem decrescente
- **A cláusula ORDER BY é inserida por último na instrução SELECT:**

```
SELECT last_name, job_id, department_id, hire_date
FROM employees
ORDER BY hire_date;
```

LAST_NAME	JOB_ID	DEPARTMENT_ID	HIRE_DATE
King	AD_PRE	90	17-JUN-87
Whalen	AD_ASST	10	17-SEP-87
Kochhar	AD_VP	90	21-SEP-89
Hunold	IT_PROG	60	03-JAN-90
Ernst	IT_PROG	60	21-MAY-91

...  
20 rows selected.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Usando a Cláusula ORDER BY

A ordem das linhas retornadas no resultado de uma consulta é indefinida. É possível usar a cláusula ORDER BY para classificar as linhas. Se você usar essa cláusula, ela deverá ser a última da instrução SQL. Você pode especificar uma expressão, um apelido ou uma posição de coluna como a condição de classificação.

### Sintaxe

```
SELECT      expr
FROM        table
[WHERE      condition(s)]
[ORDER BY   {column, expr, numeric_position} [ASC|DESC]];
```

Na sintaxe:

ORDER BY	especifica a ordem na qual as linhas recuperadas são exibidas
ASC	ordena as linhas em ordem crescente (essa é a ordem default)
DESC	ordena as linhas em ordem decrescente

Se a cláusula ORDER BY não for usada, a ordem de classificação será indefinida e o servidor Oracle poderá não extrair (fetch) as linhas na mesma ordem para consultas idênticas. Use a cláusula ORDER BY para exibir as linhas em uma ordem específica.

# Classificação

- **Classificação em ordem decrescente:**

```
SELECT last_name, job_id, department_id, hire_date
FROM employees
ORDER BY hire_date DESC;
```

1

- **Classificação por apelido de coluna:**

```
SELECT employee_id, last_name, salary*12 annsal
FROM employees
ORDER BY annsal ;
```

2

- **Classificação por várias colunas:**

```
SELECT last_name, department_id, salary
FROM employees
ORDER BY department_id, salary DESC;
```

3

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

## Ordenação de Dados Default

A ordem de classificação default é crescente:

- Os valores numéricos são exibidos com os menores valores primeiro (por exemplo, 1 a 999).
- Os valores de data são exibidos em ordem cronológica (por exemplo, 01-JAN-92 antes de 01-JAN-95).
- Os valores de caractere são exibidos em ordem alfabética (por exemplo, de A a Z).
- Os valores nulos são exibidos por último em seqüências crescentes e no início em seqüências decrescentes.
- Você pode classificar por uma coluna não incluída na lista SELECT.

## Exemplos

1. 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 de acordo com o funcionário admitido mais recentemente.
2. Você pode usar um apelido de coluna na cláusula ORDER BY. O exemplo do slide classifica os dados por salário anual.
3. Você pode classificar resultados de consultas por mais de uma coluna. O limite de classificação é o número de colunas da tabela. Na cláusula ORDER BY, especifique as colunas e separe os nomes correspondentes por vírgulas. Para inverter a ordem de uma coluna, especifique DESC após seu nome.

# Variáveis de Substituição



ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Variáveis de Substituição

Até o momento, foram apresentados exemplos codificados. Em uma aplicação concluída, o usuário acionaria o relatório, que seria executado sem outros prompts. A faixa de dados seria determinada pela cláusula `WHERE` fixa no arquivo de script do `iSQL*Plus`.

Com o `iSQL*Plus`, é possível criar relatórios que solicitam aos usuários seus próprios valores para restringir a faixa de dados retornados com variáveis de substituição. É possível incorporar *variáveis de substituição* a um arquivo de comandos ou a uma única instrução SQL. Uma variável pode ser considerada como um container no qual os valores são armazenados temporariamente. Quando a instrução é executada, o valor é substituído.

# Variáveis de Substituição

- Use as variáveis de substituição do *iSQL\*Plus* para:
  - Armazenar temporariamente valores com substituição de E comercial único (&) e duplo (&&)
- Use as variáveis de substituição para complementar:
  - Condições WHERE
  - Cláusulas ORDER BY
  - Expressões de coluna
  - Nomes de tabelas
  - Instruções SELECT inteiras

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

## Variáveis de Substituição (continuação)

No *iSQL\*Plus*, é possível usar variáveis de substituição com E comercial único (&) para armazenar valores temporariamente.

Você pode predefinir variáveis no *iSQL\*Plus* com o comando DEFINE. Esse comando cria e designa um valor a uma variável.

### Exemplos de Faixas de Dados Restritas

- Gerar relatório sobre valores relativos apenas ao trimestre atual ou a uma faixa de datas especificada
- Gerar relatório sobre dados relacionados apenas ao usuário que o solicitou
- Exibir pessoal apenas de um departamento específico

### Outros Efeitos Interativos

Os efeitos interativos não se restringem à interação direta do usuário com a cláusula WHERE. Os mesmos princípios podem ser usados para atingir outras metas, como:

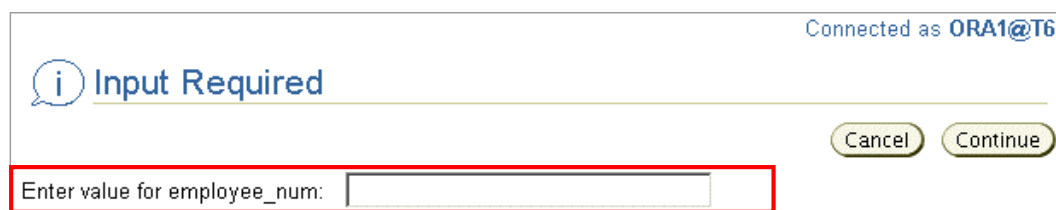
- Obtenção de valores de entrada de um arquivo, e não de uma pessoa
- Transmissão de valores de uma instrução SQL para outra

O *iSQL\*Plus* não suporta verificações de validação (exceto no caso de tipo de dados) na entrada do usuário.

# Usando a Variável de Substituição &

Use uma variável que tenha como prefixo o símbolo E comercial (&) para solicitar um valor ao usuário:

```
SELECT employee_id, last_name, salary, department_id
FROM employees
WHERE employee_id = &employee_num ;
```



ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Variável de Substituição com E Comercial Único

Durante a execução de um relatório, os usuários geralmente desejam restringir os dados retornados de forma dinâmica. O *iSQL\*Plus* permite essa flexibilidade com variáveis de usuário. Use o símbolo E comercial (&) para identificar cada variável na instrução SQL. Não é necessário definir o valor de cada variável.

Notação	Descrição
<i>&amp;user_variable</i>	Indica uma variável em uma instrução SQL; se a variável não existir, o <i>iSQL*Plus</i> solicitará um valor ao usuário (o <i>iSQL*Plus</i> descartará uma nova variável depois que ela for utilizada.)

O exemplo do slide cria uma variável de substituição do *iSQL\*Plus* para um número de funcionário. Quando a instrução é executada, o *iSQL\*Plus* solicita ao usuário um número de funcionário e, em seguida, exibe o número, o sobrenome, o salário e o número do departamento desse funcionário.

Com o símbolo E comercial único, o usuário é solicitado a especificar essa informação a cada execução do comando caso a variável não exista.



## Usando a Variável de Substituição &

ORACLE<sup>®</sup>  
iSQL\*Plus

Logout Preferences Help

Workspace History

Connected as ORA1@T6

**i Input Required**

Enter value for employee\_num:

Cancel Continue

old 3: WHERE employee\_id = &employee\_num  
new 3: WHERE employee\_id = 101

EMPLOYEE_ID	LAST_NAME	SALARY	DEPARTMENT_ID
101	Kochhar	17000	90

ORACLE<sup>®</sup>

Copyright © 2004, Oracle. Todos os direitos reservados.

### Variável de Substituição com E Comercial Único (continuação)


Quando o iSQL\*Plus detecta que a instrução SQL contém o símbolo de E comercial, você é solicitado a informar um valor para a variável de substituição citada na instrução SQL.

Depois que você informa um valor e clica no botão Continue, os resultados são exibidos na área de saída da sessão do iSQL\*Plus.

# Valores de Caractere e Data com Variáveis de Substituição

Use aspas simples para valores de data e caractere:

```
SELECT last_name, department_id, salary*12
FROM employees
WHERE job_id = '&job_title' ;
```

 **Input Required**

Cancel

Continue

Enter value for job\_title:

LAST_NAME	DEPARTMENT_ID	SALARY*12
Hunold	60	108000
Ernst	60	72000
Lorentz	60	50400

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Especificando Valores de Caractere e Data com Variáveis de Substituição


Em uma cláusula WHERE, é necessário delimitar os valores de data e caractere por aspas simples. A mesma regra se aplica às variáveis de substituição.

Delimite a variável por aspas simples na própria instrução SQL.

O slide mostra uma consulta para recuperar os nomes de funcionários, os números de departamentos e os salários anuais de todos os funcionários com base no valor de cargo da variável de substituição do *iSQL*\*Plus.

## Especificando Nomes de Colunas, Expressões e Texto

```
SELECT employee_id, last_name, job_id, &column_name  
FROM employees  
WHERE &condition  
ORDER BY &order_column ;
```

 **Input Required**

Enter value for column\_name:  Cancel Continue

Enter value for condition:  Cancel Continue

Enter value for order\_column:  Cancel Continue

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

### Especificando Nomes de Colunas, Expressões e Texto

Você pode usar as variáveis de substituição não apenas na cláusula WHERE de uma instrução SQL, mas também nos casos em que deseja substituir nomes de colunas, expressões ou texto.

#### Exemplo

O exemplo do slide exibe o número, o nome e o cargo do funcionário, bem como qualquer outra coluna especificada pelo usuário durante o runtime, com base na tabela EMPLOYEES. Para cada variável de substituição na instrução SELECT, um valor será solicitado. Após informar esse valor, clique no botão Continue para prosseguir.


Se você não informar um valor para a variável de substituição, receberá um erro quando executar a instrução anterior.

**Observação:** Uma variável de substituição pode ser usada em qualquer local na instrução SELECT, exceto como a primeira palavra especificada no prompt de comando.

# Usando a Variável de Substituição &&

Use o símbolo de E comercial duplo (&&) para reutilizar o valor da variável sem solicitar sempre um valor ao usuário.

```
SELECT  employee_id, last_name, job_id, &&column_name
FROM    employees
ORDER BY &column_name ;
```

 **Input Required**

Enter value for column\_name:

EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
200	Whalen	AD_ASST	10
201	Hartstein	MK_MAN	20

...  
20 rows selected.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Variável de Substituição com E Comercial Duplo

É possível usar a variável de substituição com E comercial duplo (&&) para reutilizar o valor da variável sem solicitar sempre um valor ao usuário. O valor é solicitado ao usuário apenas uma vez. No exemplo do slide, o valor da variável `column_name` é solicitado apenas uma vez ao usuário. O valor fornecido por ele (`department_id`) é usado para exibir e ordenar os dados.

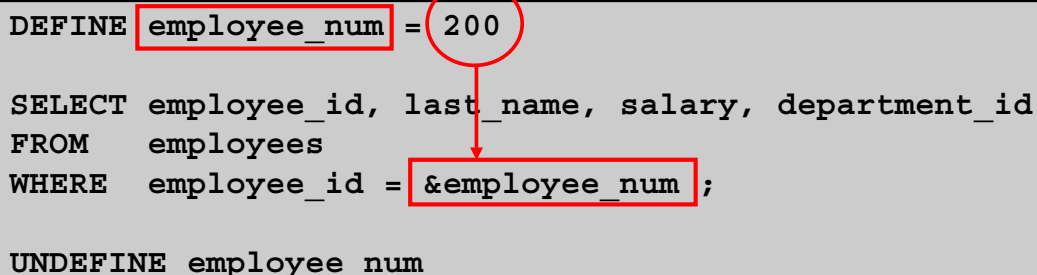
O *iSQL*\*Plus armazena o valor fornecido com o comando `DEFINE` e reutiliza esse valor sempre que é feita referência ao nome da variável. Após a definição de uma variável de usuário, é necessário usar o comando `UNDEFINE` para deletá-la da seguinte forma:

```
UNDEFINE column_name
```

## Usando o Comando DEFINE do iSQL\*Plus

- Use o comando **DEFINE** do **iSQL\*Plus** para criar e designar um valor a uma variável.
- Use o comando **UNDEFINE** do **iSQL\*Plus** para remover uma variável.

```
DEFINE employee_num = 200  
  
SELECT employee_id, last_name, salary, department_id  
FROM employees  
WHERE employee_id = &employee_num;  
  
UNDEFINE employee_num
```



ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

### Usando o Comando DEFINE do iSQL\*Plus

O exemplo mostrado cria uma variável de substituição do **iSQL\*Plus** para um número de funcionário com o comando **DEFINE**. Durante o runtime, são exibidos o número, o nome, o salário e o número do departamento desse funcionário.

Como a variável é criada com o comando **DEFINE** do **iSQL\*Plus**, não é solicitado ao usuário um valor relativo ao número de funcionário. Nesse caso, o valor definido da variável é substituído automaticamente na instrução **SELECT**.

A variável de substituição **EMPLOYEE\_NUM** estará presente na sessão até que o usuário remova sua definição ou saia da sessão do **iSQL\*Plus**.

## Usando o Comando VERIFY

Use o comando VERIFY para alternar a exibição da variável de substituição, antes e depois de o *iSQL\*Plus* substituir as variáveis por valores:

```
SET VERIFY ON  
SELECT employee_id, last_name, salary, department_id  
FROM   employees  
WHERE  employee_id = &employee_num;
```

"employee\_num" 200

```
old   3: WHERE  employee_id = &employee_num  
new   3: WHERE  employee_id = 200
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

### Usando o Comando VERIFY

Para confirmar as alterações na instrução SQL, use o comando VERIFY do *iSQL\*Plus*. A definição de SET VERIFY ON força o *iSQL\*Plus* a exibir o texto de um comando antes e depois de substituir as variáveis por valores.

O exemplo do slide exibe os valores antigo e novo da coluna EMPLOYEE\_ID.

### Variáveis de Sistema do *iSQL\*Plus*

O *iSQL\*Plus* usa diversas variáveis de sistema que controlam o ambiente de trabalho. Uma delas é VERIFY. Para obter uma lista completa de todas as variáveis de sistema, execute o comando SHOW ALL.

# Sumário

Nesta lição, você aprendeu a:

- Usar a cláusula **WHERE** para restringir as linhas da saída:
  - Usar as condições de comparação
  - Usar as condições **BETWEEN**, **IN**, **LIKE** e **NULL**
  - Aplicar os operadores lógicos **AND**, **OR** e **NOT**
- Usar a cláusula **ORDER BY** para classificar as linhas da saída:

```
SELECT * | { [DISTINCT] column/expression [alias],... }  
FROM table  
[WHERE condition(s)]  
[ORDER BY {column, expr, alias} [ASC|DESC]] ;
```

- Usar a substituição com **E comercial** no **iSQL\*Plus** para restringir e classificar a saída durante o runtime

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

## Sumário

Nesta lição, você aprendeu a restringir e classificar as linhas retornadas pela instrução **SELECT**. Você também aprendeu a implementar diversos operadores e condições.

Com as variáveis de substituição do **iSQL\*Plus**, é possível obter mais flexibilidade nas instruções **SQL**. Você pode consultar usuários durante o runtime e permitir que especifiquem critérios.

## Exercício 2: Visão Geral

**Este exercício aborda os seguintes tópicos:**

- **Selecionando dados e alterando a ordem de exibição das linhas**
- **Restringindo linhas com a cláusula WHERE**
- **Classificando linhas com a cláusula ORDER BY**
- **Usando variáveis de substituição para obter mais flexibilidade nas instruções SQL SELECT**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

### Exercício 2: Visão Geral

Neste exercício, você criará mais relatórios, inclusive instruções que utilizam as cláusulas WHERE e ORDER BY. Para tornar as instruções SQL mais reutilizáveis e genéricas, use a substituição com o símbolo de E comercial.



## Exercício 2

O departamento de recursos humanos precisa da sua ajuda para criar algumas consultas.

1. Em função de questões orçamentárias, o departamento precisa de um relatório com o sobrenome e o salário dos funcionários que ganham mais de US\$ 12.000. Inclua a instrução SQL no arquivo de texto `lab_02_01.sql`. Execute a consulta.

LAST_NAME	SALARY
King	24000
Kochhar	17000
De Haan	17000
Hartstein	13000

2. Crie um relatório que exiba o sobrenome e o número do departamento do funcionário 176.

LAST_NAME	DEPARTMENT_ID
Taylor	80

3. O departamento de recursos humanos precisa localizar funcionários com altos e baixos salários. Modifique `lab_02_01.sql` para exibir o sobrenome e o salário de todos os funcionários cuja faixa salarial não esteja entre US\$ 5.000 e US\$ 12.000. Inclua a instrução SQL no arquivo de texto `lab_02_03.sql`.

LAST_NAME	SALARY
King	24000
Kochhar	17000
De Haan	17000
Lorentz	4200
Rajs	3500
Davies	3100
Matos	2600
Vargas	2500
Whalen	4400
Hartstein	13000

10 rows selected.

## Exercício 2 (continuação)

4. Crie um relatório para exibir o sobrenome, o ID do cargo e a data de admissão dos funcionários cujos sobrenomes sejam Matos e Taylor. Organize a consulta em ordem crescente por data de admissão.

LAST_NAME	JOB_ID	HIRE_DATE
Matos	ST_CLERK	15-MAR-98
Taylor	SA_REP	24-MAR-98

5. Exiba o sobrenome e o número do departamento de todos os funcionários nos departamentos 20 e 50 em ordem alfabética crescente por nome.

LAST_NAME	DEPARTMENT_ID
Davies	50
Fay	20
Hartstein	20
Matos	50
Mourgos	50
Rajs	50
Vargas	50

7 rows selected.

6. Modifique lab\_02\_03.sql para exibir o sobrenome e o salário dos funcionários que ganham entre US\$ 5.000 e US\$ 12.000 e estão no departamento 20 ou 50. Atribua às colunas os labels Employee e Monthly Salary, respectivamente. Salve novamente lab\_02\_03.sql como lab\_02\_06.sql. Execute a instrução em lab\_02\_06.sql.

Employee	Monthly Salary
Fay	6000
Mourgos	5800

## Exercício 2 (continuação)

7. O departamento de recursos humanos precisa de um relatório que exiba o sobrenome e a data de admissão de todos os funcionários admitidos em 1994.

LAST_NAME	HIRE_DATE
Higgins	07-JUN-94
Gietz	07-JUN-94

8. Crie um relatório que exiba o sobrenome e o cargo de todos os funcionários não subordinados a um gerente.

LAST_NAME	JOB_ID
King	AD_PRES

9. Crie um relatório para exibir o sobrenome, o salário e a comissão de todos os funcionários que ganham comissão. Classifique os dados em ordem decrescente de salário e comissões.

LAST_NAME	SALARY	COMMISSION_PCT
Abel	11000	.3
Zlotkey	10500	.2
Taylor	8600	.2
Grant	7000	.15

10. Os membros do departamento de recursos humanos desejam ter mais flexibilidade em relação às consultas criadas. Eles desejam um relatório que exiba o sobrenome e o salário dos funcionários que ganham mais do que uma quantia especificada pelo usuário após o prompt. (Você pode usar a consulta criada no exercício 1 e modificá-la.) Salve essa consulta no arquivo `lab_02_10.sql`. Se você informar 12000 quando a quantia for solicitada, o relatório exibirá estes resultados:

LAST_NAME	SALARY
King	24000
Kochhar	17000
De Haan	17000
Hartstein	13000

## Exercício 2 (continuação)

11. O departamento de recursos humanos deseja executar relatórios baseados em um gerente. Crie uma consulta que solicite um ID de gerente ao usuário e gere o ID de funcionário, o sobrenome, o salário e o departamento dos funcionários desse gerente. O departamento de recursos humanos deseja ter permissão para classificar o relatório em uma coluna selecionada. Você pode testar os dados com os seguintes valores:

ID do gerente = 103, classificado pelo sobrenome do funcionário:

EMPLOYEE_ID	LAST_NAME	SALARY	DEPARTMENT_ID
104	Ernst	6000	60
107	Lorentz	4200	60

ID do gerente = 201, classificado pelo salário:

EMPLOYEE_ID	LAST_NAME	SALARY	DEPARTMENT_ID
202	Fay	6000	20

ID do gerente = 124, classificado pelo ID do funcionário:

EMPLOYEE_ID	LAST_NAME	SALARY	DEPARTMENT_ID
141	Rajs	3500	50
142	Davies	3100	50
143	Matos	2600	50
144	Vargas	2500	50

## Exercício 2 (continuação)

Se tiver tempo, faça os seguintes exercícios:

12. Exiba todos os sobrenomes dos funcionários cuja terceira letra do nome seja *a*.

LAST_NAME
Grant
Whalen

13. Exiba o sobrenome de todos os funcionários que contenha *a* e *e*.

LAST_NAME
Davies
De Haan
Hartstein
Whalen

Se quiser tomar parte em mais um desafio, faça estes exercícios:

14. Exiba o sobrenome, o cargo e o salário de todos os funcionários cujo cargo seja representante de vendas ou estoquista e cujo salário seja diferente de US\$ 2.500, US\$ 3.500 ou US\$ 7.000.

LAST_NAME	JOB_ID	SALARY
Abel	SA_REP	11000
Taylor	SA_REP	8600
Davies	ST_CLERK	3100
Matos	ST_CLERK	2600

15. Modifique `lab_02_06.sql` para exibir o sobrenome, o salário e a comissão de todos os funcionários cuja comissão seja de 20%. Salve novamente `lab_02_06.sql` como `lab_02_15.sql`. Reexecute a instrução em `lab_02_15.sql`.

Employee	Monthly Salary	COMMISSION_PCT
Zlotkey	10500	.2
Taylor	8600	.2



# 3

## Usando Functions de uma Única Linha para Personalizar a Saída

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

# Objetivos

**Ao concluir esta lição, você será capaz de:**

- **Descrever vários tipos de functions disponíveis em SQL**
- **Usar functions de caractere, número e data em instruções `SELECT`**
- **Descrever o uso de functions de conversão**

ORACLE

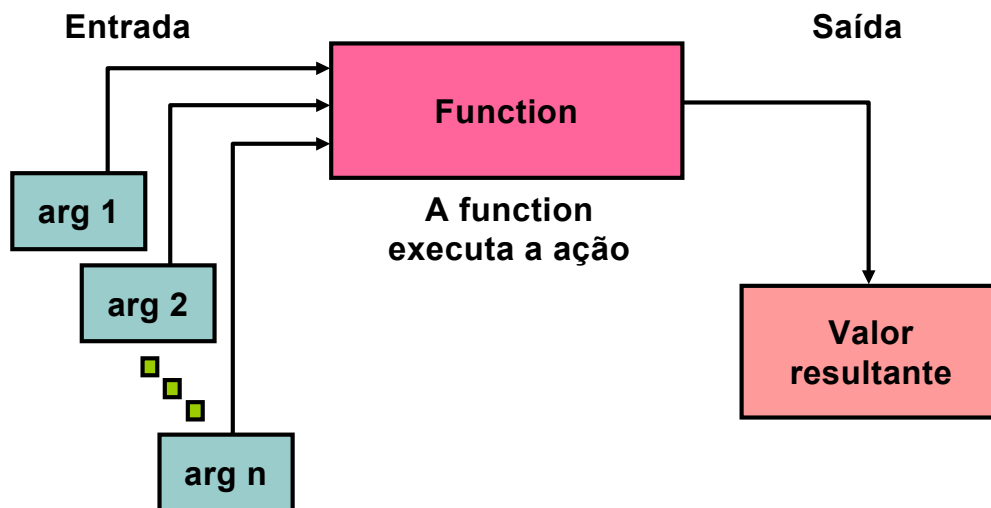
Copyright © 2004, Oracle. Todos os direitos reservados.

## Objetivos

As functions tornam o bloco básico da consulta mais eficiente e são usadas para manipular valores de dados. Esta é a primeira das duas lições que descrevem o uso de functions. Ela concentra-se em functions de caractere, número e data de uma única linha, bem como em functions que convertem os dados de um tipo em outro (por exemplo, dados de caractere em dados numéricos).



# Functions SQL



ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Functions SQL

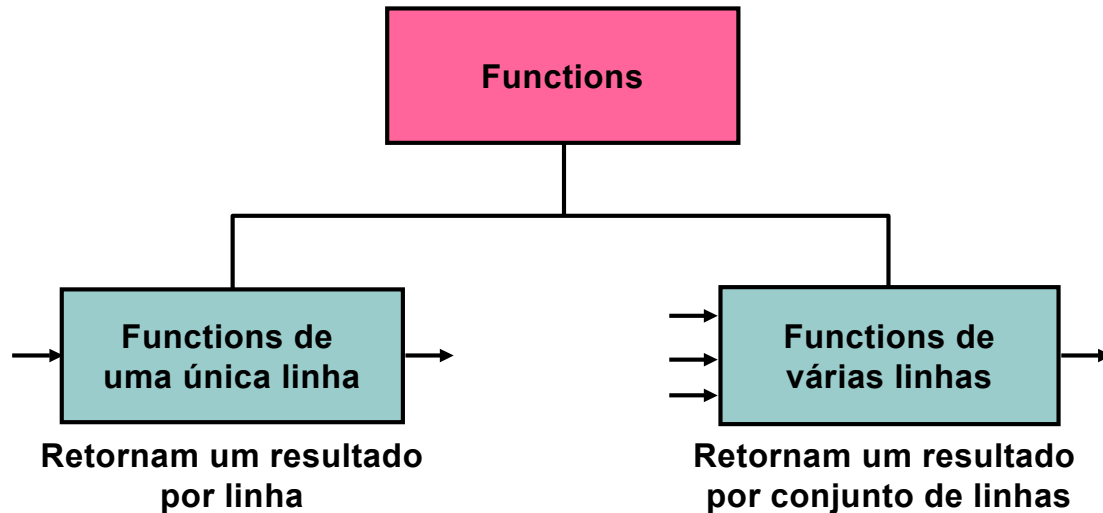
As functions constituem um recurso eficiente de SQL. É possível usá-las com os seguintes objetivos:

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

As functions SQL às vezes aceitam argumentos e sempre retornam um valor.

**Observação:** A maioria das functions descritas nesta lição são específicas da versão de SQL do Oracle.

# Dois Tipos de Functions SQL



ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Functions SQL (continuação)

Há dois tipos de functions:

- Functions de uma única linha
- Functions de várias linhas

### Functions de uma Única Linha

Estas functions só operam em linhas isoladas e retornam um resultado por linha. Existem tipos diferentes de functions de uma única linha. Esta lição aborda as seguintes functions:

- Caractere
- Número
- Data
- Conversão
- Geral

### Functions de Várias Linhas

As functions podem manipular grupos de linhas para fornecer um resultado por grupo. Essas functions também são conhecidas como *functions de grupo* (e serão abordadas em uma lição posterior).

**Observação:** Para obter mais informações e uma lista completa das functions disponíveis e a sintaxe correspondente, consulte o manual *Oracle SQL Reference*.

# Functions de uma Única Linha

## Functions de uma única linha:

- Manipulam itens de dados
- Aceitam argumentos e retornam um valor
- Agem sobre cada linha retornada
- Retornam um resultado por linha
- Podem modificar o tipo de dados
- Podem ser aninhadas
- Aceitam argumentos, como uma coluna ou uma expressão

```
function_name [(arg1, arg2,...)]
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

## Functions de uma Única Linha

As functions 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:

- Uma constante fornecida pelo usuário
- Um valor variável
- Um nome de coluna
- Uma expressão

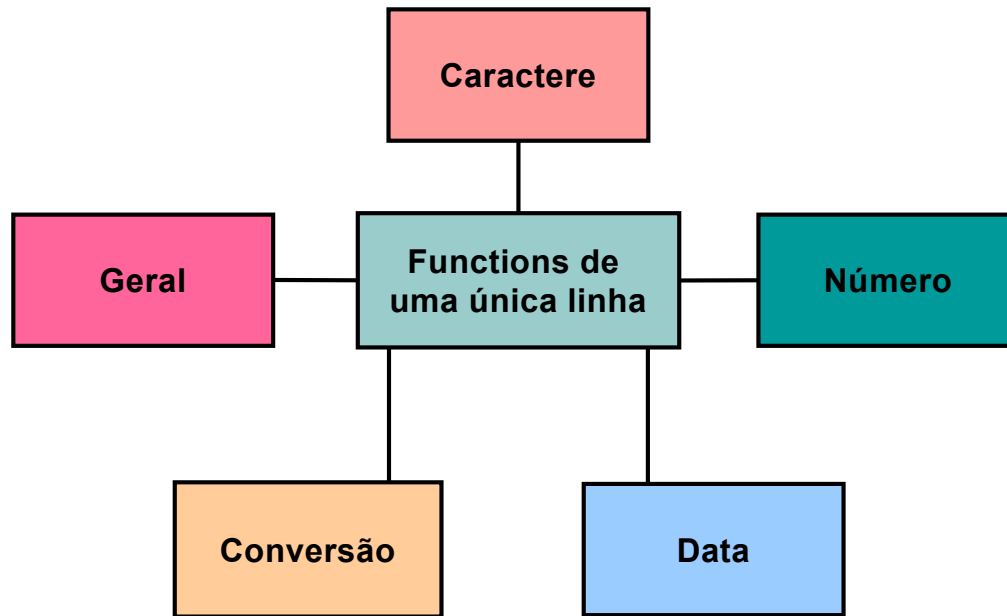
Os recursos de functions de uma única linha são:

- Agir sobre cada linha retornada pela consulta
- Retornar um resultado por linha
- Possibilitar o retorno de um valor de dados de um tipo diferente do referenciado
- Possibilitar a inclusão de um ou mais argumentos
- Ser usada em cláusulas SELECT, WHERE e ORDER BY; ser aninhada

Na sintaxe:

<i>function_name</i>	é o nome da function
<i>arg1, arg2</i>	é um argumento a ser usado pela function. Pode ser representado por um nome de coluna ou uma expressão.

# Functions de uma Única Linha



ORACLE

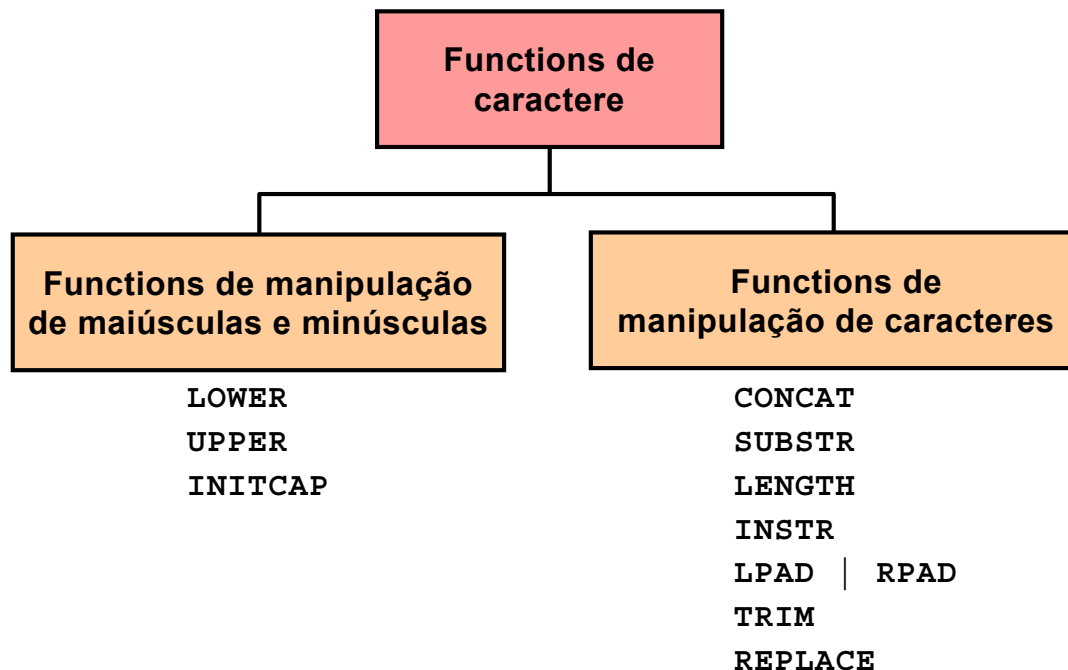
Copyright © 2004, Oracle. Todos os direitos reservados.

## Functions de uma Única Linha (continuação)

Esta lição aborda as seguintes functions de uma única linha:

- **Functions de caractere:** Aceitam a entrada de caracteres e podem retornar valores numéricos ou de caractere
- **Functions numéricas:** Aceitam a entrada numérica e retornam valores numéricos
- **Functions de data:** Operam em valores do tipo de dados DATE (Todas as functions de data retornam um valor de tipo de dados DATE, com exceção da function MONTHS\_BETWEEN, que retorna um número.)
- **Functions de conversão:** Convertem um valor de um tipo de dados em outro
- **Functions gerais:**
  - NVL
  - NVL2
  - NULLIF
  - COALESCE
  - CASE
  - DECODE

# Functions de Caractere



ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

## Functions de Caractere

As functions de caractere de uma única linha aceitam dados de caractere como entrada e podem retornar valores numéricos ou de caractere. É possível dividir as functions de caractere em:

- Functions de manipulação de maiúsculas e minúsculas
- Functions de manipulação de caracteres

Function	Objetivo
LOWER( <i>column</i> / <i>expression</i> )	Converte valores de caractere alfa em minúsculas
UPPER( <i>column</i> / <i>expression</i> )	Converte valores de caractere alfa em maiúsculas
INITCAP( <i>column</i> / <i>expression</i> )	Converte valores de caractere alfa em maiúsculas (apenas a primeira letra de cada palavra; todas as outras letras permanecem minúsculas)
CONCAT( <i>column1</i> / <i>expression1</i> , <i>column2</i> / <i>expression2</i> )	Concatena o primeiro valor de caractere para o segundo valor de caractere; equivalente ao operador de concatenação (  )
SUBSTR( <i>column</i> / <i>expression</i> , <i>m</i> [ <i>,n</i> ])	Retorna caracteres especificados a partir do valor de caractere que inicia na posição <i>m</i> , <i>n</i> caracteres (Se <i>m</i> for negativo, a contagem iniciará a partir do final do valor do caractere. Se <i>n</i> for omitido, todos os caracteres do final da string serão retornados.)

**Observação:** Nesta lição, são abordadas apenas algumas functions disponíveis.

## Functions de Caractere (continuação)

Function	Objetivo
LENGTH( <i>column</i> / <i>expression</i> )	Retorna o número de caracteres na expressão
INSTR( <i>column</i> / <i>expression</i> , ' <i>string</i> ', [ <i>m</i> ], [ <i>n</i> ] )	Retorna a posição numérica de uma string nomeada. Como opção, você pode fornecer uma posição <i>m</i> para iniciar a pesquisa e a ocorrência <i>n</i> da string. O default de <i>m</i> e <i>n</i> é 1, ou seja, começar a pesquisa desde o início e reportar a primeira ocorrência.
LPAD( <i>column</i> / <i>expression</i> , <i>n</i> , ' <i>string</i> ') RPAD( <i>column</i> / <i>expression</i> , <i>n</i> , ' <i>string</i> ')	Insere o valor de caractere justificado à direita com uma largura total de <i>n</i> posições de caractere Insere o valor do caractere justificado à esquerda com uma largura total de <i>n</i> posições de caractere
TRIM( <i>leading</i> / <i>trailing</i> / <i>both</i> , <i>trim_character</i> FROM <i>trim_source</i> )	Permite reduzir os caracteres à direita ou à esquerda (ou nas duas direções) de uma string de caracteres. Se <i>trim_character</i> ou <i>trim_source</i> for um literal de caractere, delimite-o com aspas simples. Esse recurso está disponível no Oracle8i e em versões posteriores.
REPLACE( <i>text</i> , <i>search_string</i> , <i>replacement_string</i> )	Procura uma string de caracteres em uma expressão de texto e substitui essa string por uma string de substituição especificada quando a encontra

# Functions de Manipulação de Maiúsculas e Minúsculas

Estas functions convertem letras maiúsculas em minúsculas e vice-versa em strings de caracteres:

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

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

## Functions de Manipulação de Maiúsculas e Minúsculas

LOWER, UPPER e INITCAP são as três functions de conversão entre maiúsculas e minúsculas.

- **LOWER**: Converte strings de caracteres em maiúsculas ou em maiúsculas e minúsculas em strings de caracteres em minúsculas
- **UPPER**: Converte strings de caracteres em minúsculas ou em maiúsculas e minúsculas em strings de caracteres em maiúsculas
- **INITCAP**: Converte a primeira letra de cada palavra em maiúscula e as letras restantes em minúsculas

```
SELECT 'The job id for '||UPPER(last_name)||' is '  
      ||LOWER(job_id) AS "EMPLOYEE DETAILS"  
FROM employees;
```

EMPLOYEE DETAILS
The job id for KING is ad_pres
The job id for KOCHHAR is ad_vp
The job id for DE HAAN is ad_vp
■ ■ ■
The job id for HIGGINS is ac_mgr
The job id for GIETZ is ac_account

20 rows selected

# Usando Functions de Manipulação de Maiúsculas e Minúsculas

Exiba o número, o nome e o número de departamento do funcionário Higgins:

```
SELECT employee_id, last_name, department_id
FROM   employees
WHERE  last_name = 'higgins';
no rows selected
```

```
SELECT employee_id, last_name, department_id
FROM   employees
WHERE  LOWER(last_name) = 'higgins';
```

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
205	Higgins	110

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Usando Functions de Manipulação de Maiúsculas e Minúsculas

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

A cláusula WHERE da primeira instrução SQL especifica o nome do funcionário como higgins. Como todos os dados da tabela EMPLOYEES estão armazenados com as iniciais das palavras em letras maiúsculas, o nome higgins não detecta uma correspondência na tabela e, como resultado, nenhuma linha é selecionada.

A cláusula WHERE da segunda instrução SQL especifica que o nome do funcionário na tabela EMPLOYEES seja comparado a higgins, convertendo a coluna LAST\_NAME em letras minúsculas para fins de comparação. Como agora os dois nomes estão em letras minúsculas, uma correspondência é detectada e uma linha é selecionada. É possível recriar a cláusula WHERE da seguinte maneira para produzir o mesmo resultado:

```
...WHERE last_name = 'Higgins'
```

O nome na saída aparece como foi armazenado no banco de dados. Para exibir o nome apenas com a primeira letra maiúscula, use a function UPPER na instrução SELECT.

```
SELECT employee_id, UPPER(last_name), department_id
FROM employees
WHERE INITCAP(last_name) = 'Higgins';
```



# Functions de Manipulação de Caracteres

Estas functions manipulam strings de caracteres:

Function	Resultado
CONCAT('Hello', 'World')	HelloWorld
SUBSTR('HelloWorld',1,5)	Hello
LENGTH('HelloWorld')	10
INSTR('HelloWorld', 'W')	6
LPAD(salary,10,'*')	*****24000
RPAD(salary, 10, '*')	24000*****
REPLACE ( 'JACK and JUE', 'J', 'BL' )	BLACK and BLUE
TRIM('H' FROM 'HelloWorld')	elloWorld

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

## Functions de Manipulação de Caracteres

CONCAT, SUBSTR, LENGTH, INSTR, LPAD, RPAD e TRIM são as functions de manipulação de caracteres abordadas nesta lição.

- **CONCAT**: Une valores (Você está limitado a usar dois parâmetros com CONCAT.)
- **SUBSTR**: Extrai uma string de tamanho determinado
- **LENGTH**: Mostra o tamanho de uma string como um valor numérico
- **INSTR**: Localiza a posição numérica de um caractere nomeado
- **LPAD**: Preenche o valor do caractere à direita
- **RPAD**: Preenche o valor do caractere à esquerda
- **TRIM**: Reduz os caracteres à direita ou à esquerda (ou nas duas direções) de uma string de caracteres (Se *trim\_character* ou *trim\_source* for um literal de caractere, delimite-o por aspas simples.)

**Observação:** É possível usar functions como UPPER e LOWER com a substituição de E comercial. Por exemplo, use UPPER(' &job\_title') para que o usuário não precise especificar o cargo em letras maiúsculas ou minúsculas.

# Usando as Functions de Manipulação de Caracteres

```

SELECT employee_id, CONCAT(first_name, last_name) NAME,
       job_id, LENGTH (last_name),
       INSTR(last_name, 'a') "Contains 'a'?"
FROM   employees
WHERE  SUBSTR(job_id, 4) = 'REP';
    
```

EMPLOYEE_ID	NAME	JOB_ID	LENGTH(LAST_NAME)	Contains 'a'?
174	EllenAbel	SA_REP	4	0
176	JonathonTaylor	SA_REP	6	2
178	KimberelyGrant	SA_REP	5	3
202	PatFay	MK_REP	3	2

1

2

3

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Usando as Functions de Manipulação de Caracteres

O exemplo do slide exibe os nomes e os sobrenomes dos funcionários unidos, o tamanho do sobrenome do funcionário e a posição numérica da letra *a* no sobrenome de todos os funcionários cujo ID de cargo contém a string REP a partir da quarta posição.

### Exemplo

Modifique a instrução SQL no slide para exibir os dados relativos aos funcionários cujos sobrenomes terminam com a letra *n*.

```

SELECT employee_id, CONCAT(first_name, last_name) NAME,
       LENGTH (last_name), INSTR(last_name, 'a') "Contains 'a'?"
FROM   employees
WHERE  SUBSTR(last_name, -1, 1) = 'n';
    
```

EMPLOYEE_ID	NAME	LENGTH(LAST_NAME)	Contains 'a'?
102	LexDe Haan	7	5
200	JenniferWhalen	6	3
201	MichaelHartstein	9	2

## Functions de Número

- **ROUND:** Arredonda o valor até o decimal especificado
- **TRUNC:** Trunca o valor até o decimal especificado
- **MOD:** Retorna o resto da divisão

Function	Resultado
ROUND (45.926, 2)	45.93
TRUNC (45.926, 2)	45.92
MOD (1600, 300)	100

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

## Functions de Número

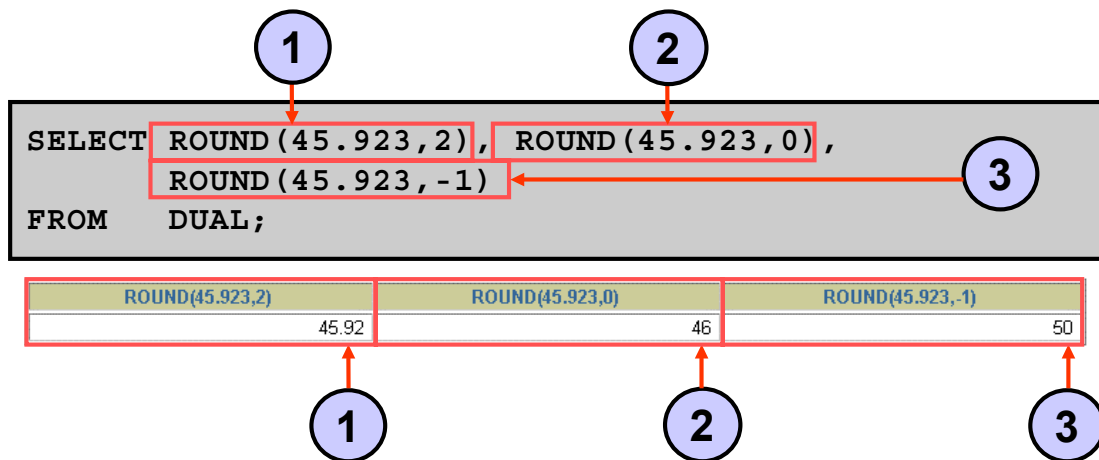
As functions de número aceitam a entrada numérica e retornam valores numéricos. Esta seção descreve algumas functions de número.

Function	Objetivo
ROUND ( <i>column</i>   <i>expression</i> , <i>n</i> )	Arredonda a coluna, a expressão ou o valor para <i>n</i> casas decimais; se <i>n</i> for omitido, não haverá casas decimais (Se <i>n</i> for negativo, os números à esquerda da vírgula decimal serão arredondados.)
TRUNC ( <i>column</i>   <i>expression</i> , <i>n</i> )	Trunca a coluna, a expressão ou o valor para <i>n</i> casas decimais; se <i>n</i> for omitido, <i>n</i> assumirá o default zero
MOD ( <i>m</i> , <i>n</i> )	Retorna o restante de <i>m</i> dividido por <i>n</i>

**Observação:** Esta lista contém somente algumas das functions de número disponíveis.

Para obter mais informações, consulte "Number Functions" no manual *Oracle SQL Reference*.

# Usando a Function ROUND



**DUAL é uma tabela fictícia que pode ser usada para exibir resultados de functions e cálculos.**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Function ROUND

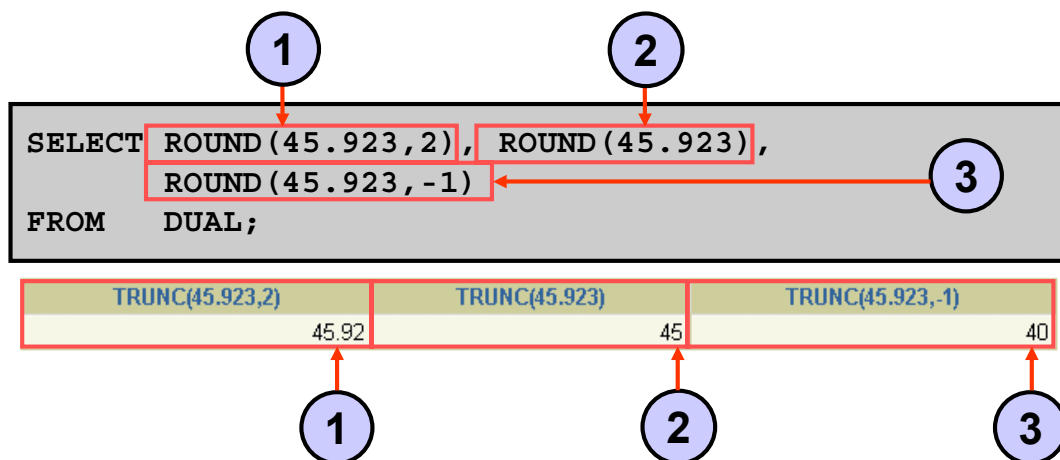
A function ROUND arredonda a coluna, a expressão ou o valor até *n* casas decimais. Se o segundo argumento for 0 ou não estiver presente, o valor será arredondado até zero casas decimais. Se o segundo argumento for 2, o valor será arredondado até duas casas decimais. Por outro lado, se o segundo argumento for -2, o valor será arredondado até duas casas decimais à esquerda (arredondado para a unidade mais próxima de 10).

Também é possível usar a function ROUND com functions de data. São fornecidos exemplos posteriormente nesta lição.

### Tabela 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 você deseja retornar um valor apenas uma vez (por exemplo, o valor de uma constante, de uma pseudocoluna ou de uma expressão não derivado de uma tabela com dados do usuário). A tabela DUAL é usada geralmente para completar a sintaxe da cláusula SELECT, pois as cláusulas SELECT e FROM são obrigatórias e diversos cálculos não precisam de seleções das tabelas reais.

# Usando a Function TRUNC



ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Function TRUNC

A function TRUNC trunca a coluna, a expressão ou o valor até n casas decimais.

Essa function opera com argumentos semelhantes aos da function ROUND. Se o segundo argumento for 0 ou não estiver presente, o valor será truncado até zero casas decimais. Se o segundo argumento for 2, o valor será truncado até duas casas decimais. Por outro lado, se o segundo argumento for -2, o valor será truncado até duas casas decimais à esquerda. Se o segundo argumento for -1, o valor será truncado até uma casa decimal à esquerda.

Assim como a function ROUND, é possível usar a function TRUNC com functions de data.

## Usando a Function MOD

Para todos os funcionários com o cargo de representante de vendas, calcule o resto do salário após dividi-lo por 5.000.

```
SELECT last_name, salary, MOD(salary, 5000)
FROM   employees
WHERE  job_id = 'SA_REP';
```

LAST_NAME	SALARY	MOD(SALARY,5000)
Abel	11000	1000
Taylor	8600	3600
Grant	7000	2000

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

### Function MOD

A function MOD localiza o resto do primeiro argumento dividido pelo segundo argumento. O exemplo do slide calcula o resto do salário após a divisão por 5.000 para todos os funcionários cujo ID de cargo é SA\_REP.

**Observação:** A function MOD é usada com frequência para determinar se o valor é par ou ímpar.

# Trabalhando com Datas

- O banco de dados Oracle armazena datas em um formato numérico interno: século, ano, mês, dia, horas, minutos e segundos.
- O formato default de exibição de data é DD-MON-RR.
  - Permite armazenar as datas do século XXI no século XX especificando apenas os últimos dois dígitos do ano
  - Permite armazenar as datas do século XX no século XXI da mesma forma

```
SELECT last_name, hire_date
FROM employees
WHERE hire_date < '01-FEB-88';
```

LAST_NAME	HIRE_DATE
King	17-JUN-87
Whalen	17-SEP-87

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Formato de Datas do Oracle

O banco de dados Oracle armazena datas em um formato numérico interno, que representa o século, o ano, o mês, o dia, as horas, os minutos e os segundos.

A exibição e o formato de entrada default de qualquer data correspondem a DD-MON-RR. As datas válidas no Oracle são de 1 de janeiro de 4712 A.C. a 31 de dezembro 9999 D.C.

No exemplo do slide, a saída da coluna HIRE\_DATE é exibida no formato default DD-MON-RR. No entanto, as datas não são armazenadas no banco de dados nesse formato. Todos os componentes da data e do horário são armazenados. Portanto, embora um valor de HIRE\_DATE como 17-JUN-87 seja exibido no formato dia, mês e ano, também há informações sobre *horário* e *século* associadas à data. Os dados completos são 17 de junho de 1987, 5:10:43 p.m.

## Formato de Datas do Oracle (continuação)

Estes dados são armazenados internamente da seguinte maneira:

CENTURY SECOND	YEAR	MONTH	DAY	HOUR	MINUTE
19 43	87	06	17	17	10

### Séculos e o Ano 2000

Quando um registro com uma coluna de data é inserido em uma tabela, as informações sobre o *século* são obtidas da function SYSDATE. No entanto, quando a coluna de data é exibida na tela, o componente de século não é mostrado (por default).

O tipo de dados DATE sempre armazena internamente as informações sobre o ano como um número de quatro dígitos: dois dígitos para o século e dois para o ano. Por exemplo, o banco de dados Oracle armazena o ano como 1987 ou 2004, e não apenas como 87 ou 04.



# Trabalhando com Datas

**SYSDATE é uma function que retorna:**

- **Data**
- **Horário**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

## Function SYSDATE

SYSDATE é uma function de data que retorna a data e o horário atuais do servidor de banco de dados. Você pode usar SYSDATE como qualquer outro nome de coluna. Por exemplo, para exibir a data atual, selecione SYSDATE em uma tabela. É comum selecionar SYSDATE em uma tabela fictícia denominada DUAL.

### Exemplo

Exiba a data atual usando a tabela DUAL.

```
SELECT SYSDATE  
FROM DUAL;
```

SYSDATE
28-SEP-01

# Aritmética com Datas

- Some um número a uma data ou subtraia-o dessa data para obter um valor de data resultante.
- Subtraia uma data de outra para descobrir o número de dias entre elas.
- Some horas a uma data dividindo o número de horas por 24.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

## Aritmética com Datas

Como o banco de dados armazena datas como números, você pode realizar cálculos usando operadores aritméticos como os de adição e subtração. É possível somar e subtrair constantes de números e datas.

Você pode executar as seguintes operações:

Operação	Resultado	Descrição
data + número	Data	Adiciona um número de dias a 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 a uma data

# Usando Operadores Aritméticos com Datas

```
SELECT last_name, (SYSDATE-hire_date)/7 AS WEEKS  
FROM employees  
WHERE department_id = 90;
```

LAST_NAME	WEEKS
King	744.245395
Kochhar	626.102538
De Haan	453.245395

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Aritmética com Datas (continuação)

O exemplo do slide exibe o sobrenome e o número de semanas desde a admissão de todos os funcionários do departamento 90. Ele subtrai a data na qual o funcionário foi admitido da data atual (SYSDATE) e divide o resultado por 7 para calcular há quantas semanas a pessoa está admitida.

**Observação:** SYSDATE é uma function SQL que retorna a data e o horário atuais. Os resultados podem diferir do exemplo.

Se uma data mais recente for subtraída de uma data mais antiga, a diferença será um número negativo.

# Functions de Data

Function	Resultado
<b>MONTHS_BETWEEN</b>	<b>Número de meses entre duas datas</b>
<b>ADD_MONTHS</b>	<b>Adiciona meses do calendário à data</b>
<b>NEXT_DAY</b>	<b>Dia seguinte ao da data especificada</b>
<b>LAST_DAY</b>	<b>Último dia do mês</b>
<b>ROUND</b>	<b>Arredonda a data</b>
<b>TRUNC</b>	<b>Trunca a data</b>

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Functions de Data

As functions de data operam em datas Oracle. Todas as functions de data retornam um valor do tipo de dados DATE, com exceção de MONTHS\_BETWEEN, que retorna um valor numérico.

- **MONTHS\_BETWEEN(*date1*, *date2*)**: Obtém o número de meses entre *date1* e *date2*. O resultado pode ser positivo ou negativo. Se *date1* for posterior a *date2*, o resultado será positivo; caso contrário, o resultado será negativo. A parte decimal do resultado representa uma parte do mês.
- **ADD\_MONTHS(*date*, *n*)**: Adiciona *n* meses do calendário à *date*. O valor *n* deve ser inteiro e pode ser negativo.
- **NEXT\_DAY(*date*, '*char*')**: Obtém a data do próximo dia especificado da semana ('*char*') após a *date* em questão. O valor de *char* pode ser um número que represente um dia ou uma string de caracteres.
- **LAST\_DAY(*date*)**: Obtém a data do último dia do mês que contém a *date* em questão.
- **ROUND(*date* [, '*fmt*'])**: Retorna a *date* arredondada até a unidade especificada pelo modelo de formato *fmt*. Se o modelo de formato *fmt* for omitido, a *date* será arredondada até o dia mais próximo.
- **TRUNC(*date* [, '*fmt*'])**: Retorna a *date* com a parte do horário do dia truncada até a unidade especificada pelo modelo de formato *fmt*. Se o modelo de formato *fmt* for omitido, a data será truncada até o dia mais próximo.

A lista é um subconjunto de functions de data disponíveis. Os modelos de formato são abordados posteriormente nesta lição. Os exemplos de modelos de formato são month e year.

## Usando Functions de Data

Function	Resultado
<b>MONTHS_BETWEEN</b> ( '01-SEP-95' , '11-JAN-94' )	19.6774194
<b>ADD_MONTHS</b> ( '11-JAN-94' , 6 )	'11-JUL-94 '
<b>NEXT_DAY</b> ( '01-SEP-95' , 'FRIDAY' )	'08-SEP-95 '
<b>LAST_DAY</b> ( '01-FEB-95' )	'28-FEB-95 '

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

### Functions de Data (continuação)

Por exemplo, exiba o número de funcionário, a data de admissão, o número de meses desde a admissão, a data da revisão semestral, a primeira sexta-feira após a data de admissão e o último dia do mês da admissão relativos a todos os funcionários admitidos há menos de 36 meses.

```
SELECT employee_id, hire_date,
       MONTHS_BETWEEN (SYSDATE, hire_date) TENURE,
       ADD_MONTHS (hire_date, 6) REVIEW,
       NEXT_DAY (hire_date, 'FRIDAY'), LAST_DAY(hire_date)
FROM employees
WHERE MONTHS_BETWEEN (SYSDATE, hire_date) < 36;
```

EMPLOYEE_ID	HIRE_DATE	TENURE	REVIEW	NEXT_DAY(	LAST_DAY(
107	07-FEB-99	31.6982407	07-AUG-99	12-FEB-99	28-FEB-99
124	16-NOV-99	22.4079182	16-MAY-00	19-NOV-99	30-NOV-99
149	29-JAN-00	19.9885633	29-JUL-00	04-FEB-00	31-JAN-00
178	24-MAY-99	28.1498536	24-NOV-99	28-MAY-99	31-MAY-99

# Usando Functions de Data

Suponha que SYSDATE = '25-JUL-03':

Function	Resultado
ROUND (SYSDATE, 'MONTH')	01-AUG-03
ROUND (SYSDATE , 'YEAR')	01-JAN-04
TRUNC (SYSDATE , 'MONTH')	01-JUL-03
TRUNC (SYSDATE , 'YEAR')	01-JAN-03

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Functions de Data (continuação)

É possível usar as functions ROUND e TRUNC para valores de número e data. Quando usadas com datas, essas functions arredondam ou truncam o valor até o modelo de formato especificado. Portanto, você pode arredondar as datas até o mês ou o ano mais próximo.

### Exemplo

Compare as datas de admissão de todos os funcionários admitidos em 1997. Exiba o número do funcionário, a data de admissão e o mês da admissão usando as functions ROUND e TRUNC.

```
SELECT employee_id, hire_date,  
       ROUND(hire_date, 'MONTH'), TRUNC(hire_date, 'MONTH')  
FROM employees  
WHERE hire_date LIKE '%97';
```

EMPLOYEE_ID	HIRE_DATE	ROUND(HIR	TRUNC(HIR
142	29-JAN-97	01-FEB-97	01-JAN-97
202	17-AUG-97	01-SEP-97	01-AUG-97

## Exercício 3: Visão Geral da Parte 1

**Este exercício aborda os seguintes tópicos:**

- Criação de uma consulta para exibir a data atual
- Criação de consultas que exigem o uso de functions numéricas, de caractere e de data
- Cálculo do número de anos e meses de serviço de um funcionário

ORACLE

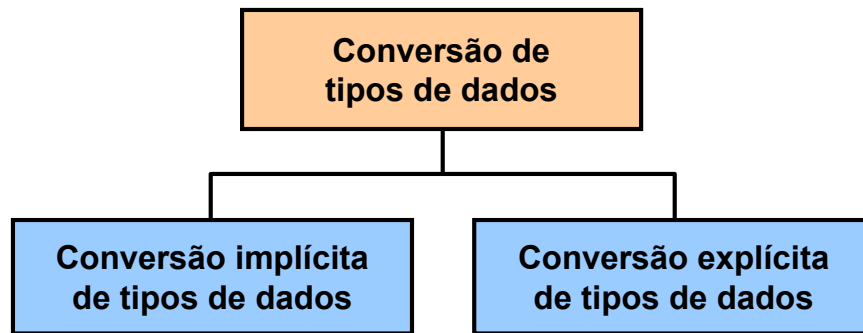
Copyright © 2004, Oracle. Todos os direitos reservados.

### Exercício 3: Visão Geral da Parte 1

A Parte 1 do exercício desta lição contém várias atividades que utilizam diferentes functions disponíveis para os tipos de dados de caractere, número e data.

Para a Parte 1, faça as questões de 1 a 6 no final desta lição.

# Functions de Conversão



ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

## Functions de Conversão

Além dos tipos de dados do Oracle, é possível definir as colunas das tabelas de um banco de dados Oracle usando os tipos de dados ANSI, DB2 e SQL/DS. No entanto, o servidor Oracle converte internamente esses tipos de dados nos tipos de dados do Oracle.

Em alguns casos, ele usa dados de um tipo diferente do esperado. Quando isso acontece, o servidor Oracle pode convertê-los automaticamente no tipo de dados esperado. Essa conversão de tipo de dados pode ser efetuada *implicitamente* pelo servidor Oracle ou *explicitamente* pelo usuário.

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

As conversões explícitas de tipos de dados são feitas por meio das functions de conversão. As functions de conversão convertem um valor de um tipo de dados em outro. Em geral, a forma dos nomes das functions segue a convenção *tipo de dados TO tipo de dados*. O primeiro tipo de dados é o da entrada, e o segundo, o da saída.

**Observação:** Embora a conversão implícita de tipos de dados esteja disponível, é recomendável usar a conversão explícita para garantir a confiabilidade das instruções SQL.



# Conversão Implícita de Tipos de Dados

No caso de designações, o servidor Oracle pode converter automaticamente:

De	Em
VARCHAR2 ou CHAR	NUMBER
VARCHAR2 ou CHAR	DATE
NUMBER	VARCHAR2
DATE	VARCHAR2

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Conversão Implícita de Tipos de Dados

A designação será bem-sucedida se o servidor Oracle conseguir converter o tipo de dados do valor usado na designação no tipo de dados de destino da designação.

Por exemplo, a expressão `hire_date > '01-JAN-90'` resulta na conversão implícita da string `'01-JAN-90'` em uma data.

# Conversão Implícita de Tipos de Dados

No caso de avaliação de expressões, o Oracle Server pode converter automaticamente:

De	Em
VARCHAR2 ou CHAR	NUMBER
VARCHAR2 ou CHAR	DATE

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

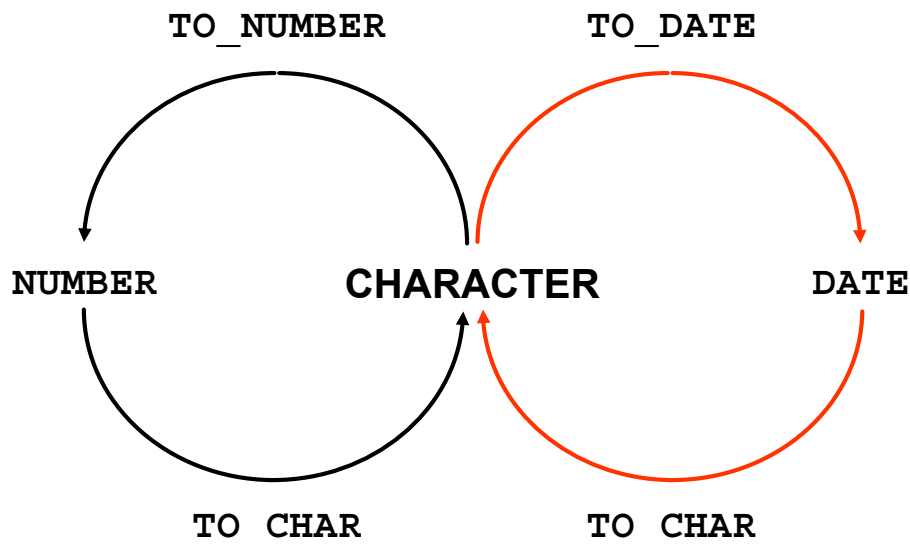
## Conversão Implícita de Tipos de Dados (continuação)

Em geral, o servidor Oracle usa a regra para expressões quando é necessária uma conversão de tipos de dados em casos não previstos por uma regra para conversões de designações.

Por exemplo, a expressão `salary = '20000'` resulta na conversão implícita da string `'20000'` no número 20000.

**Observação:** As conversões de CHAR em NUMBER só serão bem-sucedidas se a string de caracteres representar um número válido.

# Conversão Explícita de Tipos de Dados



ORACLE

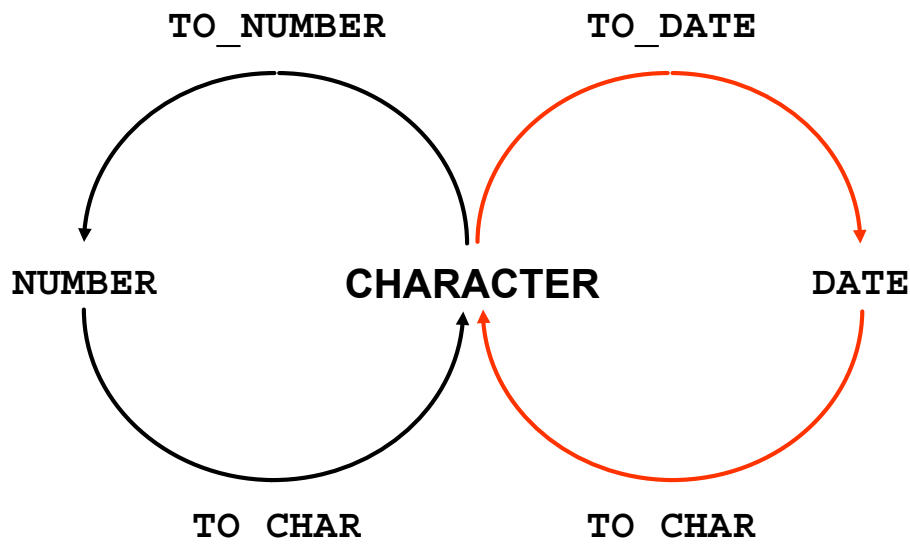
Copyright © 2004, Oracle. Todos os direitos reservados.

## Conversão Explícita de Tipos de Dados

O SQL fornece três functions para converter um valor de um tipo de dados em outro:

Function	Objetivo
<code>TO_CHAR(<i>number</i>   <i>date</i>, [ <i>fmt</i> ], [ <i>nlsparms</i> ] )</code>	<p>Converte um valor de número ou data em uma string de caracteres VARCHAR2 com o modelo de formato <i>fmt</i></p> <p><b>Conversão de número:</b> O parâmetro <i>nlsparms</i> especifica os seguintes caracteres, que são retornados pelos elementos de formato numérico:</p> <ul style="list-style-type: none"> <li>• Caractere decimal</li> <li>• Separador de grupos</li> <li>• Símbolo de moeda local</li> <li>• Símbolo de moeda internacional</li> </ul> <p>Se <i>nlsparms</i> ou qualquer outro parâmetro for omitido, esta function utilizará os valores de parâmetro default para a sessão.</p>

# Conversão Explícita de Tipos de Dados



ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

## Conversão Explícita de Tipos de Dados (continuação)

Function	Objetivo
<code>TO_CHAR(number date, [fmt], [nlsparams])</code>	<b>Conversão de data:</b> O parâmetro <code>nlsparams</code> especifica o idioma em que são retornados as abreviações e os nomes de dias e meses. Se esse parâmetro for omitido, esta function usará os idiomas de data default para a sessão.
<code>TO_NUMBER(char, [fmt], [nlsparams])</code>	Converte uma string de caracteres com dígitos em um número no formato especificado pelo modelo de formato opcional <code>fmt</code> . Nesta function, o parâmetro <code>nlsparams</code> desempenha o mesmo papel que na function <code>TO_CHAR</code> no que diz respeito à conversão de número.
<code>TO_DATE(char, [fmt], [nlsparams])</code>	Converte uma string de caracteres que representa uma data em um valor de data de acordo com o <code>fmt</code> especificado. Se o <code>fmt</code> for omitido, o formato será DD-MON-YY. Nesta function, o parâmetro <code>nlsparams</code> desempenha o mesmo papel que na function <code>TO_CHAR</code> no que diz respeito à conversão de data.

## Conversão Explícita de Tipos de Dados (continuação)

**Observação:** A lista de functions mencionada nesta lição contém apenas algumas das functions de conversão disponíveis.

Para obter mais informações, consulte "Conversion Functions" no manual *Oracle SQL Reference*.

# Usando a Function TO\_CHAR com Datas

```
TO_CHAR(date, 'format_model')
```

## O modelo de formato:

- Deve ser delimitado por aspas simples
- Faz distinção entre maiúsculas e minúsculas
- Pode incluir qualquer elemento de formato de data válido
- Tem um elemento `fm` para remover os espaços vazios preenchidos ou para suprimir os zeros à esquerda
- É separado do valor da data por vírgula

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Exibindo uma Data em um Formato Específico

Anteriormente, todos os valores de data do Oracle eram exibidos no formato DD-MÊS-AA. Você pode usar a function TO\_CHAR para converter uma data desse formato default em um formato especificado.

### Diretrizes

- O modelo de formato deve ser delimitado por aspas simples e fazer distinção entre maiúsculas e minúsculas.
- O modelo de formato pode incluir qualquer elemento de formato de data válido. Separe o valor da data do modelo de formato por vírgula.
- Os espaços vazios em nomes de dias e meses na saída serão preenchidos automaticamente.
- Para remover os espaços vazios preenchidos ou suprimir os zeros à esquerda, use o elemento `fm` do modo de preenchimento.
- Você pode formatar o campo de caracteres resultante com o comando `iSQL*Plus COLUMN` (abordado em uma lição posterior).

```
SELECT employee_id, TO_CHAR(hire_date, 'MM/YY') Month_Hired
FROM employees
WHERE last_name = 'Higgins';
```

EMPLOYEE_ID	MONTH
205	06/94

## Elementos do Modelo de Formato de Data

Elemento	Resultado
YYYY	Ano completo em números
YEAR	Ano por extenso (em inglês)
MM	Valor de dois dígitos para o mês
MONTH	Nome completo do mês
MON	Abreviação de três letras do mês
DY	Abreviação de três letras do dia da semana
DAY	Nome completo do dia da semana
DD	Dia numérico do mês

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Exemplo de Elementos de Formatos de Data Válidos

Elemento	Descrição
SCC ou CC	Século; o servidor prefixa a data A.C. com -
Anos em datas YYYY ou SYYYY	Ano; o servidor prefixa a data A.C. com -
YYY ou YY ou Y	Últimos três ou dois dígitos do ano, ou apenas o último dígito
Y.YYY	Ano com ponto nesta posição
IYYY, IYY, IY, I	Ano de quatro, três, dois ou um dígito baseado no padrão ISO
SYEAR ou YEAR	Ano por extenso; o servidor prefixa a data A.C. com -
BC ou AD	Indica o ano A.C. ou D.C.
B.C. ou A.D.	Indica o ano A.C. ou D.C. usando pontos
Q	Trimestre do ano
MM	Mês: valor de dois dígitos
MONTH	Nome do mês preenchido com espaços em branco (até nove caracteres)
MON	Nome do mês; abreviação de três letras
RM	Mês em numeral romano
WW ou W	Semana do ano ou mês
DDD ou DD ou D	Dia do ano, do mês ou da semana
DAY	Nome do dia preenchido com espaços em branco (até nove caracteres)
DY	Nome do dia; abreviação de três letras
J	Dia juliano; o número de dias desde 31 de dezembro de 4713 A.C.



## Elementos do Modelo de Formato de Data

- Os elementos de horário formatam a parte relativa ao horário da data:

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

- Adicione strings de caracteres delimitando-as por aspas duplas:

DD "of" MONTH	12 of OCTOBER
---------------	---------------

- Os sufixos de números exibem os números por extenso:

ddspth	fourteenth
--------	------------

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

### Elementos de Formato de Data: Formatos de Horário

Use os formatos listados nas tabelas a seguir para exibir informações sobre horário e literais, bem como alterar numerais para números por extenso.

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

## Outros Formatos

Elemento	Descrição
/ . ,	Pontuação é reproduzida no resultado.
“of the”	String entre aspas é reproduzida no resultado.

## Especificando Sufixos para Influenciar a Exibição de Números

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

# Usando a Function TO\_CHAR com Datas

```
SELECT last_name,  
       TO_CHAR(hire_date, 'fmDD Month YYYY')  
       AS HIREDATE  
FROM   employees;
```

LAST_NAME	HIREDATE
King	17 June 1987
Kochhar	21 September 1989
De Haan	13 January 1993
Hunold	3 January 1990
Ernst	21 May 1991
Lorentz	7 February 1999
Mourgos	16 November 1999

...  
20 rows selected.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Usando a Function TO\_CHAR com Datas

A instrução SQL do slide exibe os sobrenomes e as datas de admissão de todos os funcionários. A data de admissão aparece como 17 June 1987.

### Exemplo

Modifique o exemplo do slide para exibir as datas em um formato que apareça como "Seventeenth of June 1987 12:00:00 AM".

```
SELECT last_name,  
       TO_CHAR(hire_date,  
               'fmDdsph "of" Month YYYY fmHH:MI:SS AM')  
       HIREDATE  
FROM   employees;
```

LAST_NAME	HIREDATE
King	Seventeenth of June 1987 12:00:00 AM
Kochhar	Twenty-First of September 1989 12:00:00 AM

...

Observe que o mês segue o modelo de formato especificado, ou seja, a primeira letra é maiúscula e as outras são minúsculas.

## Usando a Function TO\_CHAR com Números

```
TO_CHAR(number, 'format_model') ddspth
```

Estes são alguns dos elementos de formato que você pode usar com a function TO\_CHAR para exibir um valor de número como um caractere:

Elemento	Resultado
9	Representa um número
0	Impõe a exibição de um zero
\$	Insere um sinal de dólar flutuante
L	Usa o símbolo da moeda local flutuante
.	Imprime uma casa decimal
,	Imprime uma vírgula como indicador de milhar

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Usando a Function TO\_CHAR com Números

Quando estiver trabalhando com valores de números como strings de caracteres, converta esses números no tipo de dados de caractere usando a function TO\_CHAR, que converte um valor do tipo de dados NUMBER no tipo de dados VARCHAR2. Essa técnica é especialmente útil com a concatenação.

## Usando a Function TO\_CHAR com Números (continuação)

### Elementos de Formato de Número

Se você estiver convertendo um número no tipo de dados de caractere, poderá usar os seguintes elementos de formato:

Elemento	Descrição	Exemplo	Resultado
9	Posição numérica (o número de 9s determina a largura da exibição)	999999	1234
0	Exibe zeros à esquerda	099999	001234
\$	Símbolo de dólar flutuante	\$999999	\$1234
L	Símbolo de moeda local flutuante	L999999	FF1234
D	Retorna o caractere decimal na posição especificada. O default é uma vírgula (.).	99,99	99D99
.	Vírgula decimal na posição especificada	999999,99	1234,00
G	Retorna o separador de grupos na posição especificada. Você pode especificar vários separadores de grupos em um modelo de formato numérico.	9.999	9G999
,	Ponto na posição especificada	999.999	1.234
MI	Sinal de subtração à direita (valores negativos)	999999MI	1234-
PR	Números negativos entre colchetes	999999PR	<1234>
EEEE	Notação científica (o formato deve especificar quatro Es)	99,999EEEE	1,234E+03
U	Retorna o símbolo monetário dual "Euro" (ou outro) na posição especificada	U9999	€1234
V	Multiplica por 10 <i>n</i> vezes ( <i>n</i> = número de 9s após V)	9999V99	123400
S	Retorna o valor negativo ou positivo	S9999	-1234 ou +1234
B	Exibe valores zero em branco, e não 0	B9999,99	1234,00

## Usando a Function TO\_CHAR com Números

```
SELECT TO_CHAR(salary, '$99,999.00') SALARY  
FROM   employees  
WHERE  last_name = 'Ernst';
```

SALARY
\$6,000.00

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

### Diretrizes

- O servidor Oracle exibe uma string de símbolos de número (#) no lugar de um número inteiro cujos dígitos ultrapassem o número de dígitos fornecido no modelo de formato.
- O servidor Oracle arredonda o valor decimal armazenado até o número de casas decimais do modelo de formato.

## Usando as Functions TO\_NUMBER e TO\_DATE

- **Converta uma string de caracteres em um formato de número usando a function TO\_NUMBER:**

```
TO_NUMBER(char[, 'format_model'])
```

- **Converta uma string de caracteres em um formato de data usando a function TO\_DATE:**

```
TO_DATE(char[, 'format_model'])
```

- **Estas functions têm um modificador fx. Esse modificador especifica a correspondência exata para o argumento de caractere e o modelo de formato de data de uma function TO\_DATE.**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

### Usando as Functions TO\_NUMBER e TO\_DATE

É possível converter uma string de caracteres em um número ou uma data. Para realizar essa tarefa, use a function TO\_NUMBER ou TO\_DATE. O modelo de formato escolhido baseia-se nos elementos de formato demonstrados anteriormente.

O modificador fx especifica a correspondência exata para o argumento de caractere e o modelo de formato de data de uma function TO\_DATE:

- A pontuação e o texto no argumento de caractere devem corresponder exatamente (exceto em relação a maiúsculas e minúsculas) às partes associadas do modelo de formato.
- O argumento de caractere não pode conter espaços em branco adicionais. Sem fx, o Oracle ignora os espaços em branco adicionais.
- Os dados numéricos no argumento de caractere devem ter o mesmo número de dígitos do elemento correspondente no modelo de formato. Sem fx, os números no argumento de caractere podem omitir os zeros à esquerda.

## Usando as Functions TO\_NUMBER e TO\_DATE (continuação)

### Exemplo

Exiba o nome e a data de admissão de todos os funcionários admitidos em 24 de maio de 1999. Como o modificador *fx* é usado, uma correspondência exata é necessária e os espaços após a palavra *May* não são reconhecidos:

```
SELECT last_name, hire_date
FROM employees
WHERE hire_date = TO_DATE('May 24, 1999', 'fxMonth DD, YYYY');

WHERE hire_date = TO_DATE('May 24, 1999', 'fxMonth DD, YYYY')
*
```

ERROR at line 3:

ORA-01858: a non-numeric character was found where a numeric was expected



## Formato de Data RR

Ano Atual	Data Especificada	Formato RR	Formato YY
1995	27.10.95	1995	1995
1995	27.10.17	2017	1917
2001	27.10.17	2017	2017
2001	27.10.95	1995	2095

		Se o ano de dois dígitos especificado for:	
		0–49	50–99
Se os dois dígitos do ano atual forem:	0–49	A data retornada estará contida no século atual	A data retornada estará contida no século imediatamente anterior ao atual
	50–99	A data retornada estará contida no século imediatamente posterior ao atual	A data retornada estará contida no século atual

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

### Elemento de Formato de Data RR

O formato de data RR é semelhante ao elemento YY, mas pode ser usado para especificar séculos diferentes. Use o elemento de formato de data RR, em vez de YY, para que o século do valor retornado 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 Especificada	Interpretação (RR)	Interpretação (YY)
1994	27-OUT-95	1995	1995
1994	27-OUT-17	2017	1917
2001	27-OUT-17	2017	2017

## Exemplo do Formato de Data RR

Para localizar os funcionários admitidos antes de 1990, use o formato de data RR, que produz os mesmos resultados quer o comando seja executado em 1999 ou agora:

```
SELECT last_name, TO_CHAR(hire_date, 'DD-Mon-YYYY')
FROM employees
WHERE hire_date < TO_DATE('01-Jan-90', 'DD-Mon-RR');
```

LAST_NAME	TO_CHAR(HIR
King	17-Jun-1987
Kochhar	21-Sep-1989
Whalen	17-Sep-1987

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Exemplo do Formato de Data RR

Para localizar os funcionários admitidos antes de 1990, você poderá usar o formato RR. Como o ano atual é maior que 1999, o formato RR interpretará a parte da data relativa ao ano de 1950 a 1999.

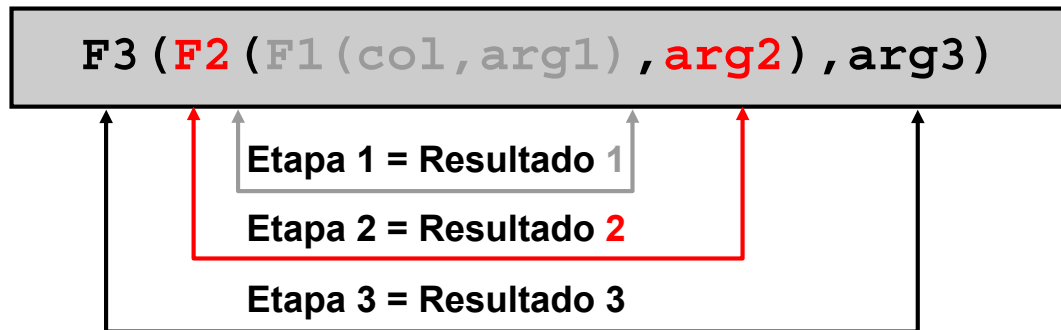
O comando a seguir, por outro lado, não resultará na seleção de linhas porque o formato YY interpretará a parte da data relativa ao ano de acordo com o século atual (2090).

```
SELECT last_name, TO_CHAR(hire_date, 'DD-Mon-yyyy')
FROM employees
WHERE TO_DATE(hire_date, 'DD-Mon-yy') < '01-Jan-1990';
```

no rows selected

## Aninhando Functions

- É possível aninhar as functions de uma única linha em quantos níveis forem necessários.
- As functions aninhadas são avaliadas do nível mais profundo para o nível mais superficial.



ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Aninhando Functions

É possível aninhar as functions de uma única linha em quantos níveis forem necessários. As functions aninhadas são avaliadas do nível mais interno para o nível mais externo. Veja a seguir alguns exemplos que mostram a flexibilidade dessas functions.

# Aninhando Functions

```
SELECT last name,  
       UPPER(CONCAT(SUBSTR (LAST_NAME, 1, 8), '_US'))  
FROM   employees  
WHERE  department_id = 60;
```

LAST_NAME	UPPER(CONCAT(SUBSTR(LAST_NAME,1,8
Hunold	HUNOLD_US
Ernst	ERNST_US
Lorentz	LORENTZ_US

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Aninhando Functions (continuação)

O exemplo do slide exibe os sobrenomes dos funcionários do departamento 60. A avaliação da instrução SQL abrange três etapas:

1. A function interna recupera os oito primeiros caracteres do sobrenome.  
Result1 = SUBSTR (LAST\_NAME, 1, 8)
2. A function externa concatena o resultado com \_US.  
Result2 = CONCAT(Result1, '\_US')
3. A function mais externa converte os resultados em letras maiúsculas.

A expressão inteira torna-se o cabeçalho da coluna, pois não foi fornecido um apelido para essa coluna.

## Exemplo

Exiba a data da próxima sexta-feira que está a seis meses da data de admissão. A data resultante deverá aparecer como Friday, August 13th, 1999. Ordene os resultados por data de admissão.

```
SELECT    TO_CHAR(NEXT_DAY(ADD_MONTHS  
                    (hire_date, 6), 'FRIDAY'),  
          'fmDay, Month DDth, YYYY')  
          "Next 6 Month Review"  
FROM      employees  
ORDER BY  hire_date;
```

# Functions Gerais

**As functions a seguir agem com qualquer tipo de dados e estão relacionadas ao uso de valores nulos:**

- NVL (expr1, expr2)
- NVL2 (expr1, expr2, expr3)
- NULLIF (expr1, expr2)
- COALESCE (expr1, expr2, ..., exprn)

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Functions Gerais

Essas functions agem com qualquer tipo de dados e estão relacionadas ao uso de valores nulos na lista de expressões.

Function	Descrição
NVL	Converte um valor nulo em um valor real
NVL2	Se expr1 não for nulo, NVL2 retornará expr2. Se expr1 for nulo, NVL2 retornará expr3. É possível expressar o argumento expr1 em qualquer tipo de dados.
NULLIF	Compara duas expressões; se elas forem iguais, retornará um valor nulo e, se forem diferentes, retornará a primeira expressão
COALESCE	Retorna a primeira expressão não nula da lista de expressões

**Observação:** Para obter mais informações sobre as centenas de functions disponíveis, consulte "Functions" no manual *Oracle SQL Reference*.

# Function NVL

**Converte um valor nulo em um valor real:**

- É possível usar os tipos de dados de data, caractere e número.
- A correspondência entre os tipos de dados é necessária:
  - `NVL(commission_pct,0)`
  - `NVL(hire_date, '01-JAN-97')`
  - `NVL(job_id, 'No Job Yet')`

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

## Function NVL

Para converter um valor nulo em um valor real, use a function NVL.

### Sintaxe

`NVL (expr1, expr2)`

Na sintaxe:

- *expr1* é o valor de origem ou a expressão que poderá conter um valor nulo
- *expr2* é o valor de destino para a conversão do valor nulo

Você pode usar a function NVL para converter qualquer tipo de dados, mas o valor retornado é sempre igual ao do tipo de dados de *expr1*.

### Conversões NVL de Vários Tipos de Dados

Tipo de Dados	Exemplo de Conversão
NUMBER	<code>NVL(number_column, 9)</code>
DATE	<code>NVL(date_column, '01-JAN-95')</code>
CHAR or VARCHAR2	<code>NVL(character_column, 'Unavailable')</code>

## Usando a Function NVL

```
SELECT last_name, salary, NVL(commission_pct, 0)
      (salary*12) + (salary*12*NVL(commission_pct, 0)) AN_SAL
FROM employees;
```

LAST_NAME	SALARY	NVL(COMMISSION_PCT,0)	AN_SAL
King	24000	0	288000
Kochhar	17000	0	204000
De Haan	17000	0	204000
Hunold	9000	0	108000
Ernst	6000	0	72000
Lorentz	4200	0	50400
Mourgos	5800	0	69600
Rajs	3500	0	42000

...  
20 rows selected.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Usando a Function NVL

Para calcular a remuneração anual de todos os funcionários, você precisa multiplicar o salário mensal por 12 e adicionar o percentual de comissão ao resultado:

```
SELECT last_name, salary, commission_pct,
      (salary*12) + (salary*12*commission_pct) AN_SAL
FROM employees;
```

LAST_NAME	SALARY	COMMISSION_PCT	AN_SAL
Vargas	2500		
Zlotkey	10500	.2	151200
Abel	11000	.3	171600
Taylor	8600	.2	123840

Observe que a remuneração anual é calculada apenas para os funcionários que recebem comissão. Se o valor de uma coluna da expressão for nulo, o resultado será nulo. Para calcular valores relativos a todos os funcionários, converta o valor nulo em um número antes de aplicar o operador aritmético. No exemplo do slide, a function NVL é usada para converter valores nulos em zero.

# Usando a Function NVL2

```
SELECT last name, salary, commission_pct  
      NVL2(commission_pct,  
            'SAL+COMM', 'SAL') income  
FROM   employees WHERE department_id IN (50, 80);
```

LAST_NAME	SALARY	COMMISSION_PCT	INCOME
Zlotkey	10500	.2	SAL+COMM
Abel	11000	.3	SAL+COMM
Taylor	8600	.2	SAL+COMM
Mourgos	5800		SAL
Rajs	3500		SAL
Davies	3100		SAL
Matos	2600		SAL
Vargas	2500		SAL

8 rows selected.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Usando a Function NVL2

A function NVL2 examina a primeira expressão. Se a primeira expressão não for nula, a function NVL2 retornará a segunda expressão. Se a primeira expressão for nula, a terceira expressão será retornada.

### Sintaxe

NVL2 (*expr1*, *expr2*, *expr3*)

Na sintaxe:

- *expr1* é o valor de origem ou a expressão que poderá conter um valor nulo
- *expr2* é o valor retornado quando *expr1* não é nulo
- *expr3* é o valor retornado quando *expr2* é nulo

No exemplo mostrado no slide, a coluna COMMISSION\_PCT é examinada. Se for detectado um valor, a segunda expressão SAL+COMM será retornada. Se a coluna COMMISSION\_PCT contiver um valor nulo, a terceira expressão SAL será retornada.

É possível expressar o argumento *expr1* em qualquer tipo de dados. Os argumentos *expr2* e *expr3* podem ser expressos em qualquer tipo de dados, exceto LONG. Se os tipos de dados de *expr2* e *expr3* forem diferentes, o servidor Oracle converterá *expr3* no tipo de dados de *expr2* antes de compará-los, a menos que *expr3* seja uma constante nula. No último caso, não será necessária uma conversão de tipo de dados. O tipo de dados do valor retornado será sempre igual ao tipo de dados de *expr2*, a menos que *expr2* seja expresso em dados de caractere; nesse caso, o tipo de dados do valor retornado será VARCHAR2.



# Usando a Function NULLIF

1

```
SELECT first_name, LENGTH(first_name) "expr1",
       last_name, LENGTH(last_name) "expr2",
       NULLIF(LENGTH(first_name), LENGTH(last_name)) result
FROM employees;
```

2

3

FIRST_NAME	expr1	LAST_NAME	expr2	RESULT
Steven	6	King	4	6
Neena	5	Kochhar	7	5
Lex	3	De Haan	7	3
Alexander	9	Hunold	6	9
Bruce	5	Ernst	5	
Diana	5	Lorentz	7	5
Kevin	5	Mourgos	7	5
Trenna	6	Rajs	4	6
Curtis	6	Davies	6	

...  
20 rows selected.

1

2

3

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Usando a Function NULLIF

A function NULLIF compara duas expressões. Se elas forem iguais, a function retornará um valor nulo. Se elas forem diferentes, a function retornará a primeira expressão. Não é possível especificar o literal NULL para a primeira expressão.

### Sintaxe

NULLIF (*expr1*, *expr2*)

Na sintaxe:

- *expr1* é o valor de origem comparado a *expr2*
- *expr2* é o valor de origem comparado a *expr1* (Se ele não for igual a *expr1*, *expr1* será retornado.)

No exemplo do slide, o tamanho do nome na tabela EMPLOYEES é comparado ao tamanho do sobrenome nessa mesma tabela. Quando os tamanhos dos nomes são iguais, um valor nulo é exibido. Quando eles são diferentes, é exibido o tamanho do nome.

**Observação:** A function NULLIF equivale logicamente à expressão CASE a seguir. A expressão CASE será abordada em uma página adiante:

CASE WHEN *expr1* = *expr2* THEN NULL ELSE *expr1* END

## Usando a Function COALESCE

- A vantagem da function COALESCE em relação à function NVL é que a primeira pode assumir diversos valores alternativos.
- Se a primeira expressão não for nula, a function COALESCE retornará essa expressão; caso contrário, será usada a function COALESCE para as expressões restantes.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

### Usando a Function COALESCE

A function COALESCE retorna a primeira expressão não nula da lista.

#### Sintaxe

COALESCE (*expr1*, *expr2*, ... *exprn*)

Na sintaxe:

- *expr1* retornará essa expressão se ela não for nula
- *expr2* retornará essa expressão se ela não for nula e a primeira expressão for nula
- *exprn* retornará essa expressão se as expressões anteriores forem nulas

Todas as expressões devem ter o mesmo tipo de dados.

## Usando a Function COALESCE

```
SELECT last_name,  
       COALESCE(manager_id,commission_pct, -1) comm  
FROM   employees  
ORDER BY commission_pct;
```

LAST_NAME	COMM
Grant	149
Zlotkey	100
Taylor	149
Abel	149
King	-1
Kochhar	100
De Haan	100

...  
20 rows selected.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

### Usando a Function COALESCE (continuação)

No exemplo mostrado no slide, se o valor de `MANAGER_ID` não for nulo, ele será exibido. Se o valor de `MANAGER_ID` for nulo, `COMMISSION_PCT` será exibido. Se os valores de `MANAGER_ID` e `COMMISSION_PCT` forem nulos, será exibido o valor `-1`.

# Expressões Condicionais

- **Permitem usar a lógica IF-THEN-ELSE em uma instrução SQL**
- **Usam dois métodos:**
  - **Expressão CASE**
  - **Function DECODE**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Expressões Condicionais

Os dois métodos usados para implementar o processamento condicional (a lógica IF-THEN-ELSE) em uma instrução SQL são a expressão CASE e a function DECODE.

**Observação:** A expressão CASE está de acordo com o padrão ANSI SQL. A function DECODE é específica da sintaxe do Oracle.

# Expressão CASE

**Facilita consultas condicionais executando o trabalho de uma instrução IF-THEN-ELSE:**

```
CASE expr WHEN comparison_expr1 THEN return_expr1  
      [WHEN comparison_expr2 THEN return_expr2  
      WHEN comparison_exprn THEN return_exprn  
      ELSE else_expr]  
END
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Expressão CASE

As expressões CASE permitem usar a lógica IF-THEN-ELSE em instruções SQL sem acionar procedures.

Em uma expressão CASE simples, o servidor Oracle pesquisa o primeiro par WHEN . . . THEN no qual *expr* é igual a *comparison\_expr* e retorna *return\_expr*. Se nenhum par WHEN . . . THEN atender a essa condição e existir uma cláusula ELSE, o servidor Oracle retornará *else\_expr*. Caso contrário, ele retornará um valor nulo. Não é possível especificar o literal NULL para todos os valores de *return\_exprs* e *else\_expr*.

Todas as expressões (*expr*, *comparison\_expr* e *return\_expr*) devem ter o mesmo tipo de dados, que pode ser CHAR, VARCHAR2, NCHAR ou NVARCHAR2.

# Usando a Expressão CASE

**Facilita consultas condicionais executando o trabalho de uma instrução IF-THEN-ELSE:**

```
SELECT last_name, job_id, salary,  
       CASE job_id WHEN 'IT_PROG' THEN 1.10*salary  
                  WHEN 'ST_CLERK' THEN 1.15*salary  
                  WHEN 'SA_REP' THEN 1.20*salary  
       ELSE salary END "REVISED_SALARY"  
FROM employees;
```

LAST_NAME	JOB_ID	SALARY	REVISED_SALARY
...			
Lorentz	IT_PROG	4200	4620
Mourgos	ST_MAN	5800	5800
Rajs	ST_CLERK	3500	4025
...			
Gietz	AC_ACCOUNT	8300	8300

20 rows selected.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Usando a Expressão CASE

Na instrução SQL do slide, o valor de JOB\_ID é decodificado. Se o valor de JOB\_ID for IT\_PROG, o aumento de salário será de 10%; se esse valor for ST\_CLERK, o aumento de salário será de 15%; se o valor for SA\_REP, o aumento de salário será de 20%. Para todos os outros cargos, não haverá aumento de salário.

É possível criar a mesma instrução com a function DECODE.

Este é um exemplo de uma expressão CASE pesquisada. Nessa expressão, a pesquisa é feita da esquerda para a direita até que seja localizada uma ocorrência da condição listada. Em seguida, obtém-se a expressão de retorno. Se nenhuma condição for considerada verdadeira e existir uma cláusula ELSE, a expressão de retorno nessa cláusula será exibida; caso contrário, será retornado NULL.

```
SELECT last_name, salary,  
       (CASE WHEN salary<5000 THEN 'Low'  
            WHEN salary<10000 THEN 'Medium'  
            WHEN salary<20000 THEN 'Good'  
            ELSE 'Excellent'  
       END) qualified_salary  
FROM employees;
```

# Function DECODE

**Facilita consultas condicionais executando o trabalho de uma expressão CASE ou de uma instrução IF-THEN-ELSE:**

```
DECODE(col/expressão, search1, result1  
        [, search2, result2,...,]  
        [, default])
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Function DECODE

A function DECODE decodifica uma expressão de maneira semelhante à lógica IF-THEN-ELSE usada em várias linguagens. Essa function decodifica a *expressão* depois de compará-la a cada valor da *pesquisa*. Se a expressão for igual à da *pesquisa*, o *resultado* será retornado.

Se o valor default for omitido, um valor nulo será retornado quando um valor da pesquisa não corresponder a nenhum dos valores do resultado.

# Usando a Function DECODE

```
SELECT last name, job id, salary,  
       DECODE(job_id, 'IT_PROG', 1.10*salary,  
                'ST_CLERK', 1.15*salary,  
                'SA_REP', 1.20*salary,  
                salary)  
       REVISED_SALARY  
FROM   employees;
```

LAST_NAME	JOB_ID	SALARY	REVISED_SALARY
...			
Lorentz	IT_PROG	4200	4620
Mourgos	ST_MAN	5800	5800
Rajs	ST_CLERK	3500	4025
...			
Gietz	AC_ACCOUNT	8300	8300

20 rows selected.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Usando a Function DECODE

Na instrução SQL do slide, o valor de JOB\_ID é testado. Se o valor de JOB\_ID for IT\_PROG, o aumento de salário será de 10%; se esse valor for ST\_CLERK, o aumento de salário será de 15%; se o valor for SA\_REP, o aumento de salário será de 20%. Para todos os outros cargos, não haverá aumento de salário.

É possível expressar a mesma instrução em pseudocódigo como uma instrução IF-THEN-ELSE:

```
IF job_id = 'IT_PROG'      THEN salary = salary*1.10  
IF job_id = 'ST_CLERK'    THEN salary = salary*1.15  
IF job_id = 'SA_REP'      THEN salary = salary*1.20  
ELSE salary = salary
```



# Usando a Function DECODE

**Exiba a alíquota de imposto aplicável para cada funcionário do departamento 80:**

```
SELECT last_name, salary,
       DECODE (TRUNC(salary/2000, 0),
               0, 0.00,
               1, 0.09,
               2, 0.20,
               3, 0.30,
               4, 0.40,
               5, 0.42,
               6, 0.44,
               0.45) TAX_RATE
FROM   employees
WHERE  department_id = 80;
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Usando a function DECODE (continuação)

Este slide mostra outro exemplo com a function DECODE. Nesse exemplo, determinamos a alíquota de imposto de cada funcionário do departamento 80 com base no salário mensal. As alíquotas de imposto são as seguintes:

<i>Faixa de Salário Mensal</i>	<i>Alíquota de Imposto</i>
US\$ 0,00 – 1.999,99	00%
US\$ 2.000,00 – 3.999,99	09%
US\$ 4.000,00 – 5.999,99	20%
US\$ 6.000,00 – 7.999,99	30%
US\$ 8.000,00 – 9.999,99	40%
US\$ 10.000,00 – 11.999,99	42%
US\$ 12.200,00 – 13.999,99	44%
US\$14.000,00 ou mais	45%

LAST_NAME	SALARY	TAX_RATE
Zlotkey	10500	.42
Abel	11000	.42
Taylor	8600	.4

# Sumário

**Nesta lição, você aprendeu a:**

- **Executar cálculos em dados usando functions**
- **Modificar itens de dados individuais usando functions**
- **Manipular a saída de grupos de linhas usando functions**
- **Alterar os formatos de data para exibição usando functions**
- **Converter tipos de dados de colunas usando functions**
- **Usar functions NVL**
- **Usar a lógica IF-THEN-ELSE**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

## Sumário

É possível aninhar as functions de uma única linha em quantos níveis forem necessários. As functions de uma única linha podem manipular:

- 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

Lembre-se do seguinte:

- Os valores de data também podem usar operadores aritméticos.
- As functions de conversão podem converter valores numéricos, de caractere e data: TO\_CHAR, TO\_DATE, TO\_NUMBER
- Várias functions estão relacionadas ao uso de valores nulos, incluindo NVL, NVL2, NULLIF e COALESCE.
- É possível aplicar a lógica IF-THEN-ELSE em uma instrução SQL com a expressão CASE ou a function DECODE.

## **SYSDATE e DUAL**

SYSDATE é uma function de data que retorna a data e o horário atuais. É comum selecionar SYSDATE em uma tabela fictícia denominada DUAL.

## Exercício 3: Visão Geral da Parte 2

**Este exercício aborda os seguintes tópicos:**

- **Criação de consultas que exigem o uso de functions numéricas, de caractere e data**
- **Uso de concatenação com functions**
- **Criação de consultas sem distinção entre maiúsculas e minúsculas para testar a utilidade das functions de caractere**
- **Cálculos dos anos e meses de serviço de um funcionário**
- **Obtenção da data de revisão de um funcionário**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

### Exercício 3: Visão Geral da Parte 2

A Parte 2 do exercício desta lição contém várias atividades que utilizam diferentes functions disponíveis para os tipos de dados de caractere, número e data. Faça as questões de 7 a 14 relativas a essa parte.

Lembre-se de que, nas functions aninhadas, os resultados são avaliados da function mais interna para a mais externa.

## Exercício 3

### Parte 1

1. Crie uma consulta para exibir a data atual. Atribua o label `Date` à coluna.

Date
31-DEC-03

2. O departamento de recursos humanos precisa de um relatório para exibir o número do funcionário, o sobrenome, o salário e o salário com 15,5% de aumento (especificado como um número inteiro) de cada funcionário. Atribua o label `New Salary` à coluna. Inclua a instrução SQL no arquivo de texto `lab_03_02.sql`.
3. Execute a consulta no arquivo `lab_03_02.sql`.

EMPLOYEE_ID	LAST_NAME	SALARY	New Salary
100	King	24000	27720
101	Kochhar	17000	19635
...			
202	Fay	6000	6930
205	Higgins	12000	13860
206	Gietz	8300	9587

20 rows selected.

4. Modifique a consulta `lab_03_02.sql` para adicionar uma coluna que subtraia o salário antigo do novo salário. Atribua o label `Increase` à coluna. Salve o conteúdo do arquivo como `lab_03_04.sql`. Execute a consulta revisada.

EMPLOYEE_ID	LAST_NAME	SALARY	New Salary	Increase
100	King	24000	27720	3720
101	Kochhar	17000	19635	2635
102	De Haan	17000	19635	2635
202	Fay	6000	6930	930
205	Higgins	12000	13860	1860
206	Gietz	8300	9587	1287

20 rows selected.

### Exercício 3 (continuação)

5. Crie uma consulta que exiba o sobrenome (com a primeira letra maiúscula e todas as outras minúsculas) e o tamanho do sobrenome de todos os funcionários cujos nomes comecem com a letra *J*, *A* ou *M*. Atribua um label apropriado a cada coluna. Classifique os resultados pelos sobrenomes dos funcionários.

Name	Length
Abel	4
Matos	5
Mourgos	7

Recrie a consulta para que o usuário seja solicitado a informar a letra inicial do sobrenome. Por exemplo, se o usuário informar H quando uma letra for solicitada, a saída deverá mostrar todos os funcionários cujos sobrenomes começam com a letra *H*.

Name	Length
Hartstein	9
Higgins	7
Hunold	6

### Exercício 3 (continuação)

6. O departamento de recursos humanos deseja saber qual é o tempo de emprego de cada funcionário. Para cada funcionário, exiba o sobrenome e calcule o número de meses entre hoje e a data de admissão do funcionário. Atribua o label MONTHS\_WORKED à coluna. Ordene os resultados pelo número de meses em que o funcionário está empregado. Arredonde o número de meses para o número inteiro mais próximo.

**Observação:** Os resultados serão diferentes.

LAST_NAME	MONTHS_WORKED
Zlotkey	47
Mourgos	50
Grant	55
Lorentz	59
Vargas	66
Taylor	69
Matos	70
Fay	76
Davies	83
Abel	92
Hartstein	94
Rajs	98
Higgins	115
Gietz	115
De Haan	132
Ernst	151
Hunold	168
Kochhar	171
Whalen	195
King	198

20 rows selected.

## Exercício 3 (continuação)

### Parte 2

7. Crie um relatório que produza estas informações para cada funcionário:  
<sobrenome do funcionário> recebe <salário> mensalmente,  
mas deseja <3 vezes o salário>. Atribua o label Dream Salaries  
à coluna.

Dream Salaries
King earns \$24,000.00 monthly but wants \$72,000.00.
Kochhar earns \$17,000.00 monthly but wants \$51,000.00.
De Haan earns \$17,000.00 monthly but wants \$51,000.00.
...
Hartstein earns \$13,000.00 monthly but wants \$39,000.00.
Fay earns \$6,000.00 monthly but wants \$18,000.00.
Higgins earns \$12,000.00 monthly but wants \$36,000.00.
Gietz earns \$8,300.00 monthly but wants \$24,900.00.

20 rows selected.

Se tiver tempo, faça os seguintes exercícios:

8. Crie uma consulta que exiba o sobrenome e o salário de todos os funcionários.  
Formate o salário para defini-lo com um tamanho de 15 caracteres e preenchê-lo à  
esquerda com o símbolo \$. Atribua o label SALARY à coluna.

LAST_NAME	SALARY
King	\$\$\$\$\$\$\$\$\$24000
Kochhar	\$\$\$\$\$\$\$\$\$17000
De Haan	\$\$\$\$\$\$\$\$\$17000
Hunold	\$\$\$\$\$\$\$\$\$9000
...	
Fay	\$\$\$\$\$\$\$\$\$6000
Higgins	\$\$\$\$\$\$\$\$\$12000
Gietz	\$\$\$\$\$\$\$\$\$8300

20 rows selected.

**Exercício 3 (continuação)**

9. Exiba o sobrenome, a data de admissão e a data de revisão de salário de cada funcionário, que é a primeira segunda-feira após seis meses de serviço. Atribua o label REVIEW à coluna. Formate as datas para que sejam exibidas no formato semelhante a "Monday, the Thirty-First of July, 2000".

LAST_NAME	HIRE_DATE	REVIEW
King	17-JUN-87	Monday, the Twenty-First of December, 1987
Kochhar	21-SEP-89	Monday, the Twenty-Sixth of March, 1990
De Haan	13-JAN-93	Monday, the Nineteenth of July, 1993
Hunold	03-JAN-90	Monday, the Ninth of July, 1990
Ernst	21-MAY-91	Monday, the Twenty-Fifth of November, 1991
Lorentz	07-FEB-99	Monday, the Ninth of August, 1999
...		
Higgins	07-JUN-94	Monday, the Twelfth of December, 1994
Gietz	07-JUN-94	Monday, the Twelfth of December, 1994

20 rows selected.

10. Exiba o sobrenome, a data de admissão e o dia da semana em que o funcionário começou a trabalhar. Atribua o label DAY à coluna. Ordene os resultados pelo dia da semana, começando por segunda-feira.

LAST_NAME	HIRE_DATE	DAY
Grant	24-MAY-99	MONDAY
Ernst	21-MAY-91	TUESDAY
Mourgos	16-NOV-99	TUESDAY
Taylor	24-MAR-98	TUESDAY
...		
Lorentz	07-FEB-99	SUNDAY
Fay	17-AUG-97	SUNDAY
Matos	15-MAR-98	SUNDAY

20 rows selected.



### Exercício 3 (continuação)

Se quiser tomar parte em mais um desafio, faça estes exercícios:

11. Crie uma consulta que exiba os sobrenomes e as comissões dos funcionários. Se um funcionário não ganhar comissão, a informação "No Commission" deverá ser exibida. Atribua o label COMM à coluna.

LAST_NAME	COMM
King	No Commission
Kochhar	No Commission
...	
Zlotkey	.2
Abel	.3
Taylor	.2
Grant	.15
Whalen	No Commission
Hartstein	No Commission
Fay	No Commission
Higgins	No Commission
Gietz	No Commission

20 rows selected.

12. Crie uma consulta que exiba os oito primeiros caracteres dos sobrenomes dos funcionários e indique os valores dos salários com asteriscos. Cada asterisco representa mil dólares. Classifique os dados em ordem decrescente de salário. Atribua o label EMPLOYEES\_AND\_THEIR\_SALARIES à coluna.

EMPLOYEES_AND_THEIR_SALARIES	
King	*****
Kochhar	*****
De Haan	*****
Hartstei	*****
Higgins	*****
...	
Matos	**
Vargas	**

20 rows selected.

### Exercício 3 (continuação)

13. Com a function DECODE, crie uma consulta que exiba o nível de todos os funcionários com base no valor da coluna JOB\_ID. Use estes dados:

<i>Cargo</i>	<i>Nível</i>
AD_PRES	A
ST_MAN	B
IT_PROG	C
SA_REP	D
ST_CLERK	E
Nenhuma das opções anteriores	0

JOB_ID	GRA
AC_ACCOUNT	0
AC_MGR	0
AD_ASST	0
AD_PRES	A
AD_VP	0
AD_VP	0
IT_PROG	C
IT_PROG	C
IT_PROG	C
MK_MAN	0
MK_REP	0
SA_MAN	0
SA_REP	D
SA_REP	D
SA_REP	D
ST_CLERK	E
ST_CLERK	E
ST_CLERK	E
ST_CLERK	E
ST_MAN	B

20 rows selected.

14. Recrie a instrução no exercício anterior usando a sintaxe CASE.

# 4

## **Gerando Relatórios de Dados Agregados com as Functions de Grupo**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

# Objetivos

**Ao concluir esta lição, você será capaz de:**

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

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Objetivos

Esta lição aborda o uso de functions com mais detalhes. Ela concentra-se na obtenção de informações sumariadas (como médias) relativas a grupos de linhas. A lição mostra como agrupar as linhas de uma tabela em conjuntos menores e como especificar critérios de pesquisa para grupos de linhas.

# O Que São Functions de Grupo?

As functions de grupo operam em conjuntos de linhas para fornecer um resultado por grupo.

EMPLOYEES

DEPARTMENT_ID	SALARY
90	24000
90	17000
90	17000
60	9000
60	6000
60	4200
50	5800
50	3500
50	3100
50	2600
50	2500
80	10500
80	11000
80	8600
	7000
10	4400

...  
20 rows selected.

Salário máximo na  
tabela EMPLOYEES

MAX(SALARY)
24000

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Functions de Grupo

Diferentemente das functions de uma única linha, as functions de grupo operam em conjuntos de linhas para fornecer um resultado por grupo. Esses conjuntos podem abranger a tabela inteira ou a tabela dividida em grupos.

# Tipos de Functions de Grupo

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



ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Tipos de Functions de Grupo

Cada function aceita um argumento. Esta tabela identifica as opções que você pode usar na sintaxe:

Function	Descrição
AVG ( [DISTINCT   <u>ALL</u> ] <i>n</i> )	Valor médio de <i>n</i> ; ignora valores nulos
COUNT ( { *   [DISTINCT   <u>ALL</u> ] <i>expr</i> } )	Número de linhas, em que <i>expr</i> é avaliado como um valor diferente de nulo (conta todas as linhas selecionadas usando *, inclusive valores duplicados e linhas com valores nulos)
MAX ( [DISTINCT   <u>ALL</u> ] <i>expr</i> )	Valor máximo de <i>expr</i> ; ignora valores nulos
MIN ( [DISTINCT   <u>ALL</u> ] <i>expr</i> )	Valor mínimo de <i>expr</i> ; ignora valores nulos
STDDEV ( [DISTINCT   <u>ALL</u> ] <i>x</i> )	Desvio padrão de <i>n</i> ; ignora valores nulos
SUM ( [DISTINCT   <u>ALL</u> ] <i>n</i> )	Valores somados de <i>n</i> ; ignora valores nulos
VARIANCE ( [DISTINCT   <u>ALL</u> ] <i>x</i> )	Variação de <i>n</i> ; ignora valores nulos

# Functions de Grupo: Sintaxe

```
SELECT      [column,] group_function(column), ...  
FROM        table  
[WHERE      condition]  
[GROUP BY   column]  
[ORDER BY   column];
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Diretrizes para a Utilização de Functions de Grupo

- Com DISTINCT, a function considera apenas valores não duplicados; com ALL, ela considera todos os valores, inclusive os duplicados. Como é o default, ALL não precisa ser especificado.
- Os tipos de dados das functions com um argumento *expr* podem ser CHAR, VARCHAR2, NUMBER ou DATE.
- Todas as functions de grupo ignoram valores nulos. Para substituir um valor por valores nulos, use as functions NVL, NVL2 ou COALESCE.

# Usando as Functions AVG e SUM

É possível usar as functions AVG e SUM para dados numéricos.

```
SELECT AVG(salary), MAX(salary),  
       MIN(salary), SUM(salary)  
FROM   employees  
WHERE  job_id LIKE '%REP%';
```

AVG(SALARY)	MAX(SALARY)	MIN(SALARY)	SUM(SALARY)
8150	11000	6000	32600

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Usando as Functions de Grupo

É possível usar as functions AVG, SUM, MIN e MAX em colunas que podem armazenar dados numéricos. O exemplo do slide mostra os salários médio, máximo e mínimo, bem como a soma dos salários mensais, de todos os representantes de vendas.



# Usando as Functions MIN e MAX

É possível usar MIN e MAX para tipos de dados numéricos, de caractere e de data.

```
SELECT MIN(hire date), MAX(hire date)
FROM employees;
```

MIN(HIRE_	MAX(HIRE_
17-JUN-87	29-JAN-00

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Usando as Functions de Grupo (continuação)

É possível usar as functions MAX e MIN para os tipos de dados numéricos, de caractere e de data. O exemplo do slide mostra o funcionário mais recente e o mais antigo.

O exemplo a seguir mostra o primeiro e o último sobrenome de funcionário em uma lista em ordem alfabética de todos os funcionários:

```
SELECT MIN(last_name), MAX(last_name)
FROM employees;
```

MIN(LAST_NAME)	MAX(LAST_NAME)
Abel	Zlotkey

**Observação:** Só é possível usar as functions AVG, SUM, VARIANCE e STDDEV com tipos de dados numéricos. MAX e MIN não podem ser usadas com o tipo de dados LOB ou LONG.

# Usando a Function COUNT

**COUNT (\*) retorna o número de linhas de uma tabela:**

1

```
SELECT COUNT (*)  
FROM employees  
WHERE department_id = 50;
```

COUNT(*)
5

**COUNT (expr) retorna o número de linhas com valores não nulos para expr:**

2

```
SELECT COUNT (commission_pct)  
FROM employees  
WHERE department_id = 80;
```

COUNT(COMMISSION_PCT)
3

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Function COUNT

A function COUNT tem três formatos:

- COUNT (\*)
- COUNT (expr)
- COUNT (DISTINCT expr)

COUNT (\*) retorna o número de linhas de uma tabela que atendem aos critérios da instrução SELECT, incluindo as linhas duplicadas e as linhas com valores nulos de qualquer uma das colunas. Se uma cláusula WHERE estiver incluída na instrução SELECT, COUNT (\*) retornará o número de linhas que atendem à condição especificada nessa cláusula.

Por outro lado, COUNT (expr) retorna o número de valores não nulos na coluna identificada por expr.

COUNT (DISTINCT expr) retorna o número de valores exclusivos e não nulos na coluna identificada por expr.

## Exemplos

1. O exemplo do slide mostra o número de funcionários do departamento 50.
2. O exemplo do slide mostra o número de funcionários do departamento 80 que podem receber comissão.

# Usando a Palavra-Chave DISTINCT

- **COUNT(DISTINCT *expr*)** retorna o número de valores não nulos e distintos de *expr*.
- Para exibir os valores de números de departamentos distintos da tabela **EMPLOYEES**:

```
SELECT COUNT(DISTINCT department_id)
FROM employees;
```

COUNT(DISTINCTDEPARTMENT_ID)
7

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Palavra-chave DISTINCT

Use a palavra-chave **DISTINCT** para suprimir a contagem de valores duplicados em uma coluna.

O exemplo do slide mostra os valores de números de departamentos distintos da tabela **EMPLOYEES**.

# Functions de Grupo e Valores Nulos

As functions de grupo ignoram valores nulos na

1	<pre>SELECT AVG(commission_pct) FROM employees;</pre>	
	AVG(COMMISSION_PCT)	2125

A function NVL determina que as functions de grupo incluam valores nulos:

2	<pre>SELECT AVG(NVL(commission_pct, 0)) FROM employees;</pre>	
	AVG(NVL(COMMISSION_PCT,0))	.0425

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Functions de Grupo e Valores Nulos

Todas as functions de grupo ignoram valores nulos na coluna.

A function NVL determina que as functions de grupo incluam valores nulos.

### Exemplos

1. A média é calculada com base *apenas* nas linhas da tabela com valores válidos armazenados na coluna COMMISSION\_PCT. A média é calculada como a comissão total paga a todos os funcionários dividida pelo número de funcionários que recebem comissão (quatro).
2. A média é calculada com base em *todas* as linhas da tabela, mesmo que valores nulos estejam armazenados na coluna COMMISSION\_PCT. A média é calculada como a comissão total paga a todos os funcionários dividida pelo número total de funcionários da empresa (20).

# Criando Grupos de Dados

## EMPLOYEES

DEPARTMENT_ID	SALARY
10	4400
20	13000
20	6000
50	5800
50	3500
50	3100
50	2500
50	2600
60	9000
60	6000
60	4200
80	10500
80	8600
80	11000
90	24000
90	17000

...  
20 rows selected.

4400

9500

3500

6400

10033

Salário médio  
na tabela  
EMPLOYEES  
para cada  
departamento

DEPARTMENT_ID	AVG(SALARY)
10	4400
20	9500
50	3500
60	6400
80	10033.3333
90	19333.3333
110	10150
	7000

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Criando Grupos de Dados

Até esta etapa do nosso estudo, todas as functions de grupo trataram a tabela como um grande grupo de informações.

Entretanto, às vezes é necessário dividir a tabela de informações em grupos menores. É possível efetuar essa divisão com a cláusula GROUP BY.

## Criando Grupos de Dados: Sintaxe da Cláusula GROUP BY

```
SELECT    column, group_function(column)
FROM      table
[WHERE    condition]
[GROUP BY group_by_expression]
[ORDER BY column];
```

É possível dividir as linhas de uma tabela em grupos menores com a cláusula GROUP BY.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

### Cláusula GROUP BY

Você pode usar a cláusula GROUP BY para dividir as linhas de uma tabela em grupos. Depois, pode usar as functions de grupo para retornar informações sumariadas de cada grupo.

Na sintaxe:

*group\_by\_expression* especifica colunas cujos valores determinam a base do agrupamento de linhas

### Diretrizes

- Se incluir uma function de grupo em uma cláusula SELECT, você não poderá selecionar resultados individuais, *a menos que* a coluna individual apareça na cláusula GROUP BY. Se não conseguir incluir a lista de colunas na cláusula GROUP BY, você receberá uma mensagem de erro.
- Com uma cláusula WHERE, você poderá excluir linhas antes de dividi-las em grupos.
- Inclua as *colunas* na cláusula GROUP BY.
- Não é possível usar um apelido de coluna na cláusula GROUP BY.

# Usando a Cláusula GROUP BY

Todas as colunas da lista SELECT que não são functions de grupo devem estar incluídas na cláusula GROUP BY.

```
SELECT department_id, AVG(salary)
FROM employees
GROUP BY department_id ;
```

DEPARTMENT_ID	AVG(SALARY)
10	4400
20	9500
50	3500
60	6400
80	10033.3333
90	19333.3333
110	10150
	7000

8 rows selected.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Usando a Cláusula GROUP BY

Ao usar a cláusula GROUP BY, verifique se todas as colunas da lista SELECT que não são functions de grupo estão incluídas nessa cláusula. O exemplo do slide mostra o número e o salário médio de cada departamento. A instrução SELECT com uma cláusula GROUP BY é avaliada da seguinte maneira:

- A cláusula SELECT especifica as colunas a serem recuperadas da seguinte forma:
  - A coluna de número de departamento da tabela EMPLOYEES
  - A média de todos os salários no grupo especificado na cláusula GROUP BY
- A cláusula FROM especifica as tabelas que o banco de dados deve acessar: a tabela EMPLOYEES.
- A cláusula WHERE especifica as linhas a serem recuperadas. Como não há cláusula WHERE, todas as linhas são recuperadas por default.
- A cláusula GROUP BY especifica como as linhas devem ser agrupadas. As linhas são agrupadas por número de departamento, portanto, a function AVG aplicada à coluna de salário calculará o *salário médio de cada departamento*.

# Usando a Cláusula GROUP BY

A cláusula GROUP BY seguida pelo nome da coluna não precisa estar na lista SELECT.

```
SELECT    AVG(salary)
FROM      employees
GROUP BY  department_id ;
```

AVG(SALARY)	
	4400
	9500
	3500
	6400
	10033.3333
	19333.3333
	10150
	7000

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Usando a Cláusula GROUP BY (continuação)

A cláusula GROUP BY seguida pelo nome da coluna não precisa estar na cláusula SELECT. Por exemplo, a instrução SELECT do slide mostra os salários médios de cada departamento sem exibir os respectivos números de departamento. No entanto, sem os números de departamento, os resultados parecem não fazer sentido.

É possível usar a function de grupo na cláusula ORDER BY:

```
SELECT department_id, AVG(salary)
FROM employees
GROUP BY department_id
ORDER BY AVG(salary) ;
```

DEPARTMENT_ID	AVG(SALARY)
50	3500
10	4400
60	6400
...	
90	19333.3333

8 rows selected.



# Agrupando por Mais de Uma Coluna

## EMPLOYEES

DEPARTMENT_ID	JOB_ID	SALARY
90	AD_PRES	24000
90	AD_VP	17000
90	AD_VP	17000
60	IT_PROG	9000
60	IT_PROG	6000
60	IT_PROG	4200
50	ST_MAN	5800
50	ST_CLERK	3500
50	ST_CLERK	3100
50	ST_CLERK	2600
50	ST_CLERK	2500
80	SA_MAN	10500
80	SA_REP	11000
80	SA_REP	8600
...		
20	MK_REP	6000
110	AC_MGR	12000
110	AC_ACCOUNT	8300

20 rows selected.

Adicione os  
salários à tabela  
EMPLOYEES  
para cada cargo,  
agrupados por  
departamento

DEPARTMENT_ID	JOB_ID	SUM(SALARY)
10	AD_ASST	4400
20	MK_MAN	13000
20	MK_REP	6000
50	ST_CLERK	11700
50	ST_MAN	5800
60	IT_PROG	19200
80	SA_MAN	10500
80	SA_REP	19600
90	AD_PRES	24000
90	AD_VP	34000
110	AC_ACCOUNT	8300
110	AC_MGR	12000
	SA_REP	7000

13 rows selected.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Grupos Contidos em Outros Grupos

Às vezes, você precisa ver os resultados de grupos contidos em outros grupos. O slide mostra um relatório que exibe o salário total pago a cada cargo, em cada departamento.

A tabela EMPLOYEES é agrupada primeiro por número de departamento e, nesse agrupamento, por cargo. Por exemplo, os quatro estoquistas do departamento 50 são agrupados e um único resultado (salário total) é fornecido para todos os estoquistas do grupo.

## Usando a Cláusula GROUP BY em Várias Colunas

```
SELECT department_id dept_id, job_id, SUM(salary)
FROM employees
GROUP BY department id, job id ;
```

DEPT_ID	JOB_ID	SUM(SALARY)
10	AD_ASST	4400
20	MK_MAN	13000
20	MK_REP	6000
50	ST_CLERK	11700
50	ST_MAN	5800
60	IT_PROG	19200
80	SA_MAN	10500
80	SA_REP	19600
90	AD PRES	24000
90	AD_VP	34000
110	AC_ACCOUNT	8300
110	AC_MGR	12000
	SA_REP	7000

13 rows selected.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

### Grupos Contidos em Outros Grupos (continuação)

É possível obter resultados sumariados de grupos e subgrupos listando mais de uma coluna na cláusula GROUP BY. Você pode determinar a ordem de classificação default dos resultados pela ordem das colunas na cláusula GROUP BY. No exemplo do slide, a instrução SELECT com uma cláusula GROUP BY é avaliada da seguinte maneira:

- A cláusula SELECT especifica a coluna a ser recuperada:
  - O número do departamento na tabela EMPLOYEES
  - O ID do cargo na tabela EMPLOYEES
  - A soma de todos os salários no grupo especificado na cláusula GROUP BY
- A cláusula FROM especifica as tabelas que o banco de dados deve acessar: a tabela EMPLOYEES.
- A cláusula GROUP BY especifica como você deve agrupar as linhas:
  - Primeiro, as linhas são agrupadas por número de departamento.
  - Depois, as linhas são agrupadas por ID de cargo nos grupos de números de departamento.

Portanto, a function SUM é aplicada à coluna de salário para todos os IDs de cargo em cada grupo de números de departamento.

## Consultas Inválidas Usando Functions de Grupo

Qualquer coluna ou expressão da lista **SELECT** que não seja uma function agregada deverá estar na cláusula **GROUP BY**:

```
SELECT department_id, COUNT(last_name)
FROM employees;
```

```
SELECT department_id, COUNT(last_name)
      *
ERROR at line 1:
ORA-00937: not a single-group group function
```

**Coluna ausente na cláusula GROUP BY**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

## Consultas Inválidas Usando Functions de Grupo

Sempre que usar uma combinação de itens individuais (**DEPARTMENT\_ID**) e functions de grupo (**COUNT**) na mesma instrução **SELECT**, inclua uma cláusula **GROUP BY** que especifique os itens individuais (neste caso, **DEPARTMENT\_ID**). Se a cláusula **GROUP BY** não for incluída, a mensagem de erro "not a single-group group function" será exibida e um asterisco (\*) indicará a coluna afetada. Para corrigir o erro no slide, adicione a cláusula **GROUP BY**:

```
SELECT department_id, count(last_name)
FROM employees
GROUP BY department_id;
```

DEPARTMENT_ID	COUNT(LAST_NAME)
10	1
20	2
...	1

8 rows selected.

Qualquer coluna ou expressão da lista **SELECT** que não seja uma function agregada deverá estar na cláusula **GROUP BY**.

## Consultas Inválidas Usando Functions de Grupo

- Não é possível usar a cláusula **WHERE** para restringir grupos.
- Use a cláusula **HAVING** para restringir grupos.
- Não é possível usar functions de grupo na cláusula **WHERE**.

```
SELECT    department_id, AVG(salary)
FROM      employees
WHERE     AVG(salary) > 8000
GROUP BY department_id;
```

```
WHERE  AVG(salary) > 8000
      *
ERROR at line 3:
ORA-00934: group function is not allowed here
```

**Não é possível usar a cláusula **WHERE** para restringir grupos**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

### Consultas Inválidas Usando Functions de Grupo (continuação)

Não é possível usar a cláusula **WHERE** para restringir grupos. A instrução **SELECT** do exemplo do slide resulta em erro, pois utiliza a cláusula **WHERE** para restringir a exibição dos salários médios dos departamentos cujo salário médio é superior a US\$ 8.000.

Para corrigir o erro do exemplo, use a cláusula **HAVING** para restringir grupos:

```
SELECT department_id, AVG(salary)
FROM employees
HAVING  AVG(salary) > 8000
GROUP BY department_id;
```

DEPARTMENT_ID	AVG(SALARY)
20	9500
80	10033.3333
90	19333.3333
110	10150

# Restringindo Resultados de Grupos

## EMPLOYEES

DEPARTMENT_ID	SALARY
90	24000
90	17000
90	17000
60	9000
60	6000
60	4200
50	5800
50	3500
50	3100
50	2600
50	2500
80	10500
80	11000
80	8600
...	
20	6000
110	12000
110	8300

20 rows selected.

O salário máximo por departamento quando for maior que US\$ 10.000

DEPARTMENT_ID	MAX(SALARY)
20	13000
80	11000
90	24000
110	12000

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Restringindo Resultados de Grupos

Da mesma maneira que você utiliza a cláusula WHERE para restringir as linhas selecionadas, use a cláusula HAVING para restringir grupos. Para obter o salário máximo de cada departamento cujo salário máximo é superior a US\$ 10.000, você precisa fazer o seguinte:

1. Obtenha o salário médio de cada departamento agrupando por número de departamento.
2. Restrinja os grupos aos departamentos com um salário máximo maior que US\$10.000.

## Restringindo Resultados de Grupos com a Cláusula HAVING

Quando a cláusula HAVING é utilizada, o servidor Oracle restringe os grupos da seguinte forma:

1. As linhas são agrupadas.
2. A function de grupo é aplicada.
3. Os grupos que correspondem à cláusula HAVING são exibidos.

```
SELECT    column, group_function
FROM      table
[WHERE    condition]
[GROUP BY group by expression]
[HAVING   group_condition]
[ORDER BY column];
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

### Restringindo Resultados de Grupos com a Cláusula HAVING

Use a cláusula HAVING para especificar quais grupos devem ser exibidos e, dessa forma, restringir ainda mais os grupos com base nas informações agregadas.

Na sintaxe, *group\_condition* restringe os grupos de linhas retornados aos grupos cuja condição especificada é verdadeira.

Quando você usa a cláusula HAVING, o servidor Oracle executa as seguintes etapas:

1. As linhas são agrupadas.
2. A function de grupo é aplicada ao grupo.
3. Os grupos que correspondem aos critérios na cláusula HAVING são exibidos.

A cláusula HAVING pode anteceder a cláusula GROUP BY, mas é recomendável usar a cláusula GROUP BY primeiro por razões lógicas. Os grupos são formados e as functions de grupo são calculadas antes de a cláusula HAVING ser aplicada aos grupos na lista SELECT.

## Usando a Cláusula HAVING

```
SELECT department_id, MAX(salary)
FROM employees
GROUP BY department_id
HAVING MAX(salary) > 10000 ;
```

DEPARTMENT_ID	MAX(SALARY)
20	13000
80	11000
90	24000
110	12000

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Usando a Cláusula HAVING

O exemplo do slide mostra os números e os salários máximos dos departamentos cujo salário máximo é maior que US\$ 10.000.

É possível usar a cláusula GROUP BY sem uma function de grupo na lista SELECT.

Se você restringir as linhas com base no resultado de uma function de grupo, especifique as cláusulas GROUP BY e HAVING.

Este exemplo mostra os números e os salários médios dos departamentos cujo salário máximo ultrapassa US\$ 10.000:

```
SELECT department_id, AVG(salary)
FROM employees
GROUP BY department_id
HAVING max(salary) > 10000;
```

DEPARTMENT_ID	AVG(SALARY)
20	9500
80	10033.3333
90	19333.3333
110	10150

# Usando a Cláusula HAVING

```
SELECT job_id, SUM(salary) PAYROLL
FROM employees
WHERE job_id NOT LIKE '%REP%'
GROUP BY job_id
HAVING SUM(salary) > 13000
ORDER BY SUM(salary);
```

JOB_ID	PAYROLL
IT_PROG	19200
AD_PRES	24000
AD_VP	34000

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Usando a Cláusula HAVING (continuação)

O exemplo do slide mostra o ID do cargo e o salário mensal total de cada cargo com uma folha de pagamento total que ultrapassa US\$ 13.000. O exemplo exclui representantes de vendas e classifica a lista pelo salário mensal total.



# Aninhando Functions de Grupo

Exiba a média de salário máximo:

```
SELECT MAX (AVG (salary) )  
FROM employees  
GROUP BY department_id;
```

MAX(AVG(SALARY))
19333.3333

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Aninhando Functions de Grupo

É possível aninhar até duas functions de grupo. O exemplo do slide mostra a média de salário máximo.

# Sumário

Nesta lição, você aprendeu a:

- Usar as functions de grupo COUNT, MAX, MIN e AVG
- Criar consultas que utilizam a cláusula GROUP BY
- Criar consultas que utilizam a cláusula HAVING

```
SELECT      column, group_function
FROM        table
[WHERE      condition]
[GROUP BY   group_by_expression]
[HAVING     group_condition]
[ORDER BY   column];
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Sumário

Várias functions de grupo estão disponíveis em SQL, como estas:

AVG, COUNT, MAX, MIN, SUM, STDDEV e VARIANCE

Para criar subgrupos, use a cláusula GROUP BY. É possível restringir grupos com a cláusula HAVING.

Especifique as cláusulas HAVING e GROUP BY após a cláusula WHERE em uma instrução. A ordem na qual você especifica essas cláusulas após a cláusula WHERE não é importante. Informe a cláusula ORDER BY por último.

O servidor Oracle avalia as cláusulas na seguinte ordem:

1. Se a instrução contiver uma cláusula WHERE, o servidor estabelecerá as linhas candidatas.
2. O servidor identifica os grupos especificados na cláusula GROUP BY.
3. A cláusula HAVING restringe os grupos de resultados que não atendem aos critérios de grupo especificados na cláusula HAVING.

**Observação:** Para obter uma lista completa das functions de grupo, consulte o manual *Oracle SQL Reference*.

## Exercício 4: Visão Geral

**Este exercício aborda os seguintes tópicos:**

- **Criação de consultas que utilizam as functions de grupo**
- **Agrupamento por linhas para obter mais de um resultado**
- **Restrição de grupos usando a cláusula HAVING**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

### Exercício 4: Visão Geral

No final deste exercício, você deverá estar familiarizado com a utilização de functions de grupo e a seleção de grupos de dados.

## Exercício 4

Determine a validade das três instruções a seguir. Circule Verdadeiro ou Falso.

1. As functions de grupo trabalham com várias linhas para produzir um resultado por grupo.  
Verdadeiro/Falso
2. As functions de grupo incluem valores nulos em cálculos.  
Verdadeiro/Falso
3. A cláusula WHERE restringe as linhas antes da inclusão em um cálculo de grupo.  
Verdadeiro/Falso

O departamento de RH necessita dos seguintes relatórios:

4. Obtenha o salário máximo, o salário mínimo, a soma dos salários e o salário médio de todos os funcionários. Atribua os labels Maximum, Minimum, Sum e Average, respectivamente, às colunas. Arredonde os resultados para o número inteiro mais próximo. Inclua a instrução SQL no arquivo de texto lab\_04\_04.sql.

Maximum	Minimum	Sum	Average
24000	2500	175500	8775

5. Modifique a consulta em lab\_04\_04.sql para exibir o salário mínimo, o salário máximo, a soma dos salários e o salário médio de cada tipo de cargo. Salve novamente lab\_04\_04.sql como lab\_04\_05.sql. Execute a instrução em lab\_04\_05.sql.

JOB_ID	Maximum	Minimum	Sum	Average
AC_ACCOUNT	8300	8300	8300	8300
AC_MGR	12000	12000	12000	12000
AD_ASST	4400	4400	4400	4400
AD_PRES	24000	24000	24000	24000
AD_VP	17000	17000	34000	17000
IT_PROG	9000	4200	19200	6400
MK_MAN	13000	13000	13000	13000
MK_REP	6000	6000	6000	6000
SA_MAN	10500	10500	10500	10500
SA_REP	11000	7000	26600	8867
ST_CLERK	3500	2500	11700	2925
ST_MAN	5800	5800	5800	5800

12 rows selected.

## Exercício 4 (continuação)

6. Crie uma consulta para exibir o número de pessoas com o mesmo cargo.

JOB_ID	COUNT(*)
AC_ACCOUNT	1
AC_MGR	1
AD_ASST	1
AD PRES	1
AD_VP	2
IT_PROG	3
MK_MAN	1
MK_REP	1
SA_MAN	1
SA_REP	3
ST_CLERK	4
ST_MAN	1

12 rows selected.

Generalize a consulta para que o usuário do departamento de RH seja solicitado a informar um cargo. Salve o script no arquivo lab\_04\_06.sql.

7. Determine o número de gerentes sem listá-los. Atribua o label Number of Managers à coluna. *Dica: Use a coluna MANAGER\_ID para determinar o número de gerentes.*

Number of Managers
8

8. Descubra a diferença entre o salário mais alto e o mais baixo. Atribua o label DIFFERENCE à coluna.

DIFFERENCE
21500

Se tiver tempo, faça os seguintes exercícios:

9. Crie um relatório para exibir o número do gerente e o salário do funcionário com menor remuneração desse gerente. Exclua todas as pessoas cujo gerente seja desconhecido. Exclua todos os grupos em que o salário mínimo seja US\$ 6.000 ou inferior. Classifique a saída em ordem decrescente de salário.

MANAGER_ID	MIN(SALARY)
102	9000
205	8300
149	7000

## Exercício 4 (continuação)

Se quiser tomar parte em mais um desafio, faça estes exercícios:

10. Crie uma consulta que exiba o número total de funcionários e, desse total, mostre o número de funcionários admitidos em 1995, 1996, 1997 e 1998. Crie cabeçalhos de colunas apropriados.

TOTAL	1995	1996	1997	1998
20	1	2	2	3

11. Crie uma consulta matriz que exiba o cargo, o salário relativo a esse cargo com base no número do departamento e o salário total desse cargo para os departamentos 20, 50, 80 e 90, atribuindo um cabeçalho apropriado a cada coluna.

Job	Dept 20	Dept 50	Dept 80	Dept 90	Total
AC_ACCOUNT					8300
AC_MGR					12000
AD_ASST					4400
AD PRES				24000	24000
AD_VP				34000	34000
IT_PROG					19200
MK_MAN	13000				13000
MK_REP	6000				6000
SA_MAN			10500		10500
SA_REP			19600		26600
ST_CLERK		11700			11700
ST_MAN		5800			5800

12 rows selected.

# 5

## Exibindo Dados de Várias Tabelas

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

# Objetivos

**Ao concluir esta lição, você será capaz de:**

- Criar instruções **SELECT** para acessar dados de mais de uma tabela com equijoins e não-equijoins
- Juntar uma tabela a si própria com uma auto-join
- Exibir dados que normalmente não atendem a uma condição de join usando joins externas
- Gerar um produto cartesiano de todas as linhas de duas ou mais tabelas

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Objetivos

Esta lição explica como obter dados de mais de uma tabela. Uma *join* é usada para exibir informações de várias tabelas. Portanto, você pode juntar tabelas para exibir informações de mais de uma tabela.

**Observação:** Para obter informações sobre joins, consulte "SQL Queries and Subqueries: Joins" no manual *Oracle SQL Reference*.



# Obtendo Dados de Várias Tabelas

**EMPLOYEES**

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
100	King	90
101	Kochhar	90
...		
202	Fay	20
205	Higgins	110
206	Gietz	110

**DEPARTMENTS**

DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID
10	Administration	1700
20	Marketing	1800
50	Shipping	1500
60	IT	1400
80	Sales	2500
90	Executive	1700
110	Accounting	1700
190	Contracting	1700



EMPLOYEE_ID	DEPARTMENT_ID	DEPARTMENT_NAME
200	10	Administration
201	20	Marketing
202	20	Marketing
...		
102	90	Executive
205	110	Accounting
206	110	Accounting

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Obtendo Dados de Várias Tabelas

Às vezes, é necessário usar dados de mais de uma tabela. No exemplo do slide, o relatório exibe dados de duas tabelas distintas:

- Os IDs de funcionário estão na tabela EMPLOYEES.
- Os IDs de departamento estão nas tabelas EMPLOYEES e DEPARTMENTS.
- Os nomes de departamento estão na tabela DEPARTMENTS.

Para gerar o relatório, você precisa vincular as tabelas EMPLOYEES e DEPARTMENTS e acessar os dados dessas duas tabelas.

# Tipos de Join

**Estas são as joins compatíveis com o padrão SQL:1999:**

- **Joins cruzadas**
- **Joins naturais**
- **Cláusula USING**
- **Joins externas integrais (ou de dois lados)**
- **Condições arbitrárias de join para joins externas**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

## Tipos de Join

Para juntar tabelas, você pode usar a sintaxe de join compatível com o padrão SQL:1999.

**Observação:** Antes da release Oracle9i, a sintaxe de join era diferente dos padrões ANSI. A sintaxe de join compatível com o SQL:1999 não oferece benefícios de desempenho em relação à sintaxe de join de propriedade Oracle existente nas releases anteriores. Para obter informações detalhadas sobre a sintaxe de join proprietária, consulte o Apêndice C.

# Unindo Tabelas com a Sintaxe SQL:1999

Use uma join para consultar dados de mais de uma tabela:

```
SELECT    table1.column, table2.column
FROM      table1
[NATURAL JOIN table2] |
[JOIN table2 USING (column_name)] |
[JOIN table2
  ON (table1.column_name = table2.column_name)] |
[LEFT|RIGHT|FULL OUTER JOIN table2
  ON (table1.column_name = table2.column_name)] |
[CROSS JOIN table2];
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Definindo Joins

Na sintaxe:

*table1.column* indica a tabela e a coluna das quais os dados são recuperados

`NATURAL JOIN` junta duas tabelas com base no mesmo nome de coluna

`JOIN table USING column_name` executa uma operação de equijoin com base no nome da coluna

`JOIN table ON table1.column_name` executa uma operação de equijoin com base na condição da cláusula `ON`, `= table2.column_name`

`LEFT/RIGHT/FULL OUTER` executa joins externas

`CROSS JOIN` retorna um produto cartesiano das duas tabelas

Para obter mais informações, consulte "SELECT" no manual *Oracle SQL Reference*.

# Criando Joins Naturais

- A cláusula **NATURAL JOIN** baseia-se em todas as colunas das duas tabelas que têm o mesmo nome.
- Ela seleciona linhas das duas tabelas que têm valores iguais em todas as colunas correspondentes.
- Se as colunas com nomes idênticos tiverem tipos de dados distintos, será retornado um erro.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Criando Joins Naturais

É possível unir tabelas automaticamente com base nas colunas das duas tabelas com tipos de dados e nomes correspondentes. Para uni-las, use as palavras-chave **NATURAL JOIN**.

**Observação:** A operação de join só pode ocorrer em colunas com os mesmos nomes e tipos de dados nas duas tabelas. Se as colunas tiverem nomes idênticos, mas tipos de dados distintos, a sintaxe **NATURAL JOIN** causará um erro.

# Recuperando Registros com Joins Naturais

```
SELECT department_id, department_name,  
       location_id, city  
FROM   departments  
NATURAL JOIN locations ;
```

DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID	CITY
60	IT	1400	Southlake
50	Shipping	1500	South San Francisco
10	Administration	1700	Seattle
90	Executive	1700	Seattle
110	Accounting	1700	Seattle
190	Contracting	1700	Seattle
20	Marketing	1800	Toronto
80	Sales	2500	Oxford

8 rows selected.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Recuperando Registros com Joins Naturais

No exemplo do slide, a tabela LOCATIONS é unida à tabela DEPARTMENT pela coluna LOCATION\_ID, que é a única coluna com o mesmo nome nas duas tabelas. Se houvesse outras colunas comuns, a join usaria todas elas.

### Joins Naturais com uma Cláusula WHERE

Para implementar outras restrições a uma join natural, use uma cláusula WHERE. O exemplo a seguir limita as linhas da saída àquelas com um ID de departamento igual a 20 ou 50:

```
SELECT  department_id, department_name,  
        location_id, city  
FROM    departments  
NATURAL JOIN locations  
WHERE   department_id IN (20, 50);
```

## Criando Joins com a Cláusula USING

- Se diversas colunas tiverem os mesmos nomes, mas os tipos de dados não forem coincidentes, a cláusula **NATURAL JOIN** poderá ser modificada com a cláusula **USING** para especificar as colunas a serem usadas em uma equijoin.
- Use a cláusula **USING** para estabelecer uma correspondência com apenas uma coluna quando houver correspondência com mais de uma coluna.
- Não use um nome ou apelido de tabela nas colunas referenciadas.
- As cláusulas **NATURAL JOIN** e **USING** são mutuamente exclusivas.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

### Cláusula USING

As joins naturais usam todas as colunas com nomes e tipos de dados correspondentes para unir as tabelas. É possível usar a cláusula **USING** para especificar apenas as colunas que deverão ser usadas em uma equijoin. As colunas referenciadas na cláusula **USING** não devem ter um qualificador (nome ou apelido de tabela) em qualquer ponto da instrução SQL.

Por exemplo, esta instrução é válida:

```
SELECT l.city, d.department_name
FROM   locations l JOIN departments d USING (location_id)
WHERE  location_id = 1400;
```

A instrução a seguir é inválida, pois **LOCATION\_ID** é qualificado na cláusula **WHERE**:

```
SELECT l.city, d.department_name
FROM locations l JOIN departments d USING (location_id)
WHERE d.location_id = 1400;
ORA-25154: column part of USING clause cannot have qualifier
```

A mesma restrição também é aplicada às joins naturais. Portanto, as colunas com o mesmo nome nas duas tabelas devem ser usadas sem qualificadores.

# Unindo Nomes de Colunas

**EMPLOYEES**

EMPLOYEE_ID	DEPARTMENT_ID
200	10
201	20
202	20
124	50
141	50
142	50
143	50
144	50
103	60
104	60
107	60
149	80
174	80
176	80

...

**DEPARTMENTS**

DEPARTMENT_ID	DEPARTMENT_NAME
10	Administration
20	Marketing
20	Marketing
50	Shipping
50	Shipping
50	Shipping
50	Shipping
50	Shipping
50	Shipping
60	IT
60	IT
60	IT
80	Sales
80	Sales
80	Sales



**Chave estrangeira      Chave primária**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

## A Cláusula USING para Equijoins

Para determinar o nome do departamento de um funcionário, compare o valor da coluna DEPARTMENT\_ID na tabela EMPLOYEES com os valores de DEPARTMENT\_ID na tabela DEPARTMENTS. O relacionamento entre as tabelas EMPLOYEES e DEPARTMENTS é uma *equijoin*, isto é, os valores da coluna DEPARTMENT\_ID nas duas tabelas devem ser iguais. Com frequência, esse tipo de join abrange complementos de chave primária e estrangeira.

**Observação:** As equijoins também são chamadas de *joins simples* ou *joins internas*.

# Recuperando Registros com a Cláusula USING

```
SELECT employees.employee_id, employees.last_name,  
       departments.location_id, department_id  
FROM   employees JOIN departments  
       USING (department_id) ;
```

EMPLOYEE_ID	LAST_NAME	LOCATION_ID	DEPARTMENT_ID
200	Whalen	1700	10
201	Hartstein	1800	20
202	Fay	1800	20
124	Mourgos	1500	50
141	Rajs	1500	50
142	Davies	1500	50
144	Vargas	1500	50
143	Matos	1500	50

...  
19 rows selected.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Recuperando Registros com a Cláusula USING

O exemplo do slide une a coluna DEPARTMENT\_ID das tabelas EMPLOYEES e DEPARTMENTS e indica o local de trabalho de um funcionário.



## Qualificando Nomes de Colunas Ambíguos

- Use prefixos de tabela para qualificar nomes de colunas presentes em várias tabelas.
- Use prefixos de tabela para melhorar o desempenho.
- Use apelidos de coluna para distinguir as colunas com nomes idênticos, mas que residem em tabelas diferentes.
- Não use apelidos em colunas identificadas na cláusula `USING` e listadas em alguma parte da instrução `SQL`.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

### Qualificando Nomes de Colunas Ambíguos

É necessário qualificar os nomes das colunas com o nome da tabela para evitar ambigüidades. Sem os prefixos das tabelas, a coluna `DEPARTMENT_ID` na lista `SELECT` poderá ser da tabela `DEPARTMENTS` ou `EMPLOYEES`. É necessário adicionar o prefixo da tabela para executar a consulta:

```
SELECT employees.employee_id, employees.last_name,  
       departments.department_id, departments.location_id  
FROM   employees JOIN departments  
ON     employees.department_id = departments.department_id;
```

Se não houver nomes de colunas comuns entre as duas tabelas, não será preciso qualificar as colunas. No entanto, o uso do prefixo da tabela melhora o desempenho, pois você informa ao servidor Oracle exatamente onde localizar as colunas.

**Observação:** Ao efetuar uma operação de join com a cláusula `USING`, você não poderá qualificar uma coluna usada nessa própria cláusula. Além disso, se essa coluna for usada em alguma parte da instrução `SQL`, ela não poderá ser utilizada como apelido.

# Usando Apelidos de Tabelas

- Use apelidos de tabelas para simplificar consultas.
- Use apelidos de tabelas para melhorar o desempenho.

```
SELECT e.employee_id, e.last_name,  
       d.location_id, department_id  
FROM   employees e JOIN departments d  
USING (department_id) ;
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

## Usando Apelidos de Tabelas

A qualificação dos nomes de colunas com nomes de tabelas pode consumir muito tempo, especialmente se os nomes das tabelas forem longos. Você pode usar os *apelidos das tabelas* em vez dos nomes. Assim como um apelido de coluna fornece outro nome a uma coluna, um apelido de tabela fornece outro nome a uma tabela. Os apelidos de tabelas ajudam a reduzir o tamanho do código SQL, utilizando menos memória.

Observe como os apelidos de tabelas são identificados na cláusula FROM do exemplo. O nome da tabela é especificado por inteiro, seguido por um espaço e, depois, o apelido da tabela. A tabela EMPLOYEES recebeu o apelido e, e a tabela DEPARTMENTS, o apelido d.

### Diretrizes

- Um apelido de tabela pode conter até 30 caracteres, mas é recomendável especificar o menor nome possível.
- Se um apelido de tabela for usado para um nome de tabela específico na cláusula FROM, ele deverá ser substituído pelo nome da tabela em toda a instrução SELECT.
- Os apelidos de tabelas devem ser significativos.
- O apelido de tabela é válido somente para a instrução SELECT atual.

## Criando Joins com a Cláusula ON

- A condição de join para a join natural é basicamente uma equijoin de todas as colunas com o mesmo nome.
- Use a cláusula ON para especificar condições arbitrárias ou colunas a serem utilizadas em operações de join.
- A condição de join é separada de outras condições de pesquisa.
- A cláusula ON facilita a compreensão do código.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

### Cláusula ON

Use a cláusula ON para especificar uma condição de join. Assim, você pode especificar condições de join separadas de condições de filtro e pesquisa na cláusula WHERE.

## Recuperando Registros com a Cláusula ON

```
SELECT e.employee_id, e.last_name, e.department_id,  
       d.department_id, d.location_id  
FROM   employees e JOIN departments d  
ON     (e.department_id = d.department_id);
```

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_ID	LOCATION_ID
200	Whalen	10	10	1700
201	Hartstein	20	20	1800
202	Fay	20	20	1800
124	Mourgos	50	50	1500
141	Rajs	50	50	1500
142	Davies	50	50	1500
143	Matos	50	50	1500

\*\*\*

19 rows selected.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

### Criando Joins com a Cláusula ON

Neste exemplo, as colunas DEPARTMENT\_ID das tabelas EMPLOYEES e DEPARTMENTS são unidas com a cláusula ON. Sempre que um ID de departamento na tabela EMPLOYEES for igual ao ID de departamento na tabela DEPARTMENTS, a linha será retornada.

Também é possível usar a cláusula ON para unir colunas com nomes distintos.

# Auto-Joins Usando a Cláusula ON

**EMPLOYEES (WORKER)**

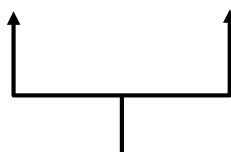
EMPLOYEE_ID	LAST_NAME	MANAGER_ID
100	King	
101	Kochhar	100
102	De Haan	100
103	Hunold	102
104	Ernst	103
107	Lorentz	103
124	Mourgos	100

...

**EMPLOYEES (MANAGER)**

EMPLOYEE_ID	LAST_NAME
100	King
101	Kochhar
102	De Haan
103	Hunold
104	Ernst
107	Lorentz
124	Mourgos

...



**MANAGER\_ID na tabela WORKER é igual a  
EMPLOYEE\_ID na tabela MANAGER.**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Unindo uma Tabela a Ela Própria

Às vezes, é necessário unir uma tabela a ela própria. Para descobrir o nome do gerente de cada funcionário, você precisa unir a tabela EMPLOYEES a ela própria ou executar uma auto-join. Por exemplo, para descobrir o nome do gerente de Lorentz, você precisa:

- Localizar Lorentz na tabela EMPLOYEES examinando a coluna LAST\_NAME.
- Localizar o número do gerente de Lorentz examinando a coluna MANAGER\_ID. O número do gerente de Lorentz é 103.
- Localizar o nome do gerente com o valor de EMPLOYEE\_ID 103 examinando a coluna LAST\_NAME. O número de funcionário de Hunold é 103, portanto, Hunold é o gerente de Lorentz.

Nesse processo, você examinará a tabela duas vezes. Na primeira vez, você examinará a tabela para localizar Lorentz na coluna LAST\_NAME e o valor 103 relativo a MANAGER\_ID. Na segunda vez, você examinará a coluna EMPLOYEE\_ID para localizar 103 e a coluna LAST\_NAME para localizar Hunold.

## Auto-Joins Usando a Cláusula ON

```
SELECT e.last_name emp, m.last_name mgr
FROM   employees e JOIN employees m
ON     (e.manager_id = m.employee_id);
```

EMP	MGR
Hartstein	King
Zlotkey	King
Mourgos	King
De Haan	King
Kochhar	King

\*\*\*  
19 rows selected.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

### Unindo uma Tabela a Ela Própria (continuação)

Também é possível usar a cláusula ON para unir colunas com nomes distintos na mesma tabela ou em uma tabela diferente.

O exemplo mostrado é uma auto-join da tabela EMPLOYEES, com base nas colunas EMPLOYEE\_ID e MANAGER\_ID.

## Aplicando Outras Condições a uma Join

```
SELECT e.employee_id, e.last_name, e.department_id,  
       d.department_id, d.location_id  
FROM   employees e JOIN departments d  
ON      (e.department_id = d.department_id)  
AND     e.manager_id = 149 ;
```

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_ID	LOCATION_ID
174	Abel	80	80	2500
176	Taylor	80	80	2500

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Aplicando Outras Condições a uma Join

Você pode aplicar outras condições à join.

O exemplo mostrado executa uma operação de join nas tabelas EMPLOYEES e DEPARTMENTS, além de exibir apenas os funcionários com o ID de gerente 149. Para adicionar outras condições à cláusula ON, especifique cláusulas AND. Como opção, você pode usar uma cláusula WHERE para aplicar outras condições:

```
SELECT e.employee_id, e.last_name, e.department_id,  
       d.department_id, d.location_id  
FROM   employees e JOIN departments d  
ON      (e.department_id = d.department_id)  
WHERE   e.manager_id = 149;
```

## Criando Joins Tridimensionais com a Cláusula ON

```
SELECT employee_id, city, department_name
FROM   employees e
JOIN    departments d
ON      d.department_id = e.department_id
JOIN    locations l
ON      d.location_id = l.location_id;
```

EMPLOYEE_ID	CITY	DEPARTMENT_NAME
103	Southlake	IT
104	Southlake	IT
107	Southlake	IT
124	South San Francisco	Shipping
141	South San Francisco	Shipping
142	South San Francisco	Shipping
143	South San Francisco	Shipping
144	South San Francisco	Shipping

...

19 rows selected.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

### Joins Tridimensionais

Uma join tridimensional é uma join de três tabelas. Na sintaxe compatível com o padrão SQL:1999, as joins são executadas da esquerda para a direita. Portanto, a primeira join a ser executada é EMPLOYEES JOIN DEPARTMENTS. A primeira condição de join pode fazer referência a colunas de EMPLOYEES e DEPARTMENTS, mas não a colunas de LOCATIONS. A segunda condição de join pode fazer referência a colunas de todas as três tabelas.



# Não-Equijoins

**EMPLOYEES**

LAST_NAME	SALARY
King	24000
Kochhar	17000
De Haan	17000
Hunold	9000
Ernst	6000
Lorentz	4200
Mourgos	5800
Rajs	3500
Davies	3100
Matos	2600
Vargas	2500
Zlotkey	10500
Abel	11000
Taylor	8600

...

20 rows selected.

**JOB\_GRADES**

GRA	LOWEST_SAL	HIGHEST_SAL
A	1000	2999
B	3000	5999
C	6000	9999
D	10000	14999
E	15000	24999
F	25000	40000

← O salário na tabela **EMPLOYEES** deve estar compreendido entre o menor e o maior salário na tabela **JOB\_GRADES**.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Não-Equijoins

Uma não-equijoin é uma condição de join que contém algo diferente de um operador de igualdade.

O relacionamento entre as tabelas **EMPLOYEES** e **JOB\_GRADES** é um exemplo de uma não-equijoin. Em um relacionamento entre as duas tabelas, os valores da coluna **SALARY** da tabela **EMPLOYEES** devem estar compreendidos entre os valores das colunas **LOWEST\_SALARY** e **HIGHEST\_SALARY** da tabela **JOB\_GRADES**. O relacionamento é obtido com um operador diferente de igualdade ( $\neq$ ).

# Recuperando Registros com Não-Equijoins

```
SELECT e.last_name, e.salary, j.grade_level
FROM   employees e JOIN job_grades j
ON     e.salary
      BETWEEN j.lowest_sal AND j.highest_sal;
```

LAST_NAME	SALARY	GRA
Matos	2600	A
Vargas	2500	A
Lorentz	4200	B
Mourgos	5800	B
Rajs	3500	B
Davies	3100	B
Whalen	4400	B
Hunold	9000	C
Ernst	6000	C

\*\*\*  
20 rows selected.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Não-Equijoins (continuação)

O exemplo do slide cria uma não-equijoin para avaliar o nível salarial de um funcionário. O salário deve estar compreendido *entre* qualquer par de faixas de salário mais baixo e mais alto.

É importante observar que todos os funcionários aparecem uma única vez quando essa consulta é executada. Os funcionários não são repetidos na lista. Existem dois motivos para isso:

- Nenhuma das linhas da tabela de níveis de cargos contém níveis sobrepostos. Isto é, o valor do salário de um funcionário somente pode estar entre os valores de salário mais alto e mais baixo de uma das linhas da tabela de níveis salariais.
- Os salários de todos os funcionários estão compreendidos entre os limites fornecidos pela tabela de níveis de cargos. Isto é, nenhum funcionário recebe menos que o valor mais baixo contido na coluna LOWEST\_SAL ou mais que o valor mais alto contido na coluna HIGHEST\_SAL.

**Observação:** É possível usar outras condições (como  $\leq$  e  $\geq$ ), mas BETWEEN é a mais simples. Quando usar BETWEEN, lembre-se de informar primeiro o valor mais baixo e depois o valor mais alto.

Foram especificados apelidos de tabelas no exemplo do slide por questões de desempenho, e não para evitar uma possível ambigüidade.

# Joins Externas

## DEPARTMENTS

DEPARTMENT_NAME	DEPARTMENT_ID
Administration	10
Marketing	20
Shipping	50
IT	60
Sales	80
Executive	90
Accounting	110
Contracting	190

8 rows selected.

## EMPLOYEES

DEPARTMENT_ID	LAST_NAME
90	King
90	Kochhar
90	De Haan
60	Hunold
60	Ernst
60	Lorentz
50	Mourgos
50	Rajs
50	Davies
50	Matos
50	Vargas
80	Zlotkey

...

20 rows selected.

**Não há funcionários no departamento 190.**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Retornando Registros sem Correspondência Direta com Joins Externas

Se não atender a uma condição de join, a linha não aparecerá no resultado da consulta. Por exemplo, na condição de equijoin das tabelas EMPLOYEES e DEPARTMENTS, o ID de departamento 190 não é exibido, pois não existem funcionários com esse ID registrado na tabela EMPLOYEES. Em vez de conter 20 funcionários, o conjunto de resultados conterá 19 registros.

Para retornar o registro de um departamento sem funcionários, use uma join externa.

## Joins Internas e Externas

- No padrão SQL:1999, a join de duas tabelas que retorna apenas as linhas correspondentes é uma join interna.
- Uma join entre duas tabelas que retorna os resultados da join interna, bem como as linhas não correspondentes da tabela esquerda (ou direita), é chamada de join externa esquerda (ou direita).
- Uma join entre duas tabelas que retorna os resultados de uma join interna, bem como os resultados de uma join esquerda e direita, é uma join externa integral.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

### Joins Internas e Externas

A união de tabelas com as cláusulas `NATURAL JOIN`, `USING` ou `ON` resulta em uma join interna. As linhas não correspondentes não são exibidas na saída. Para retornar as linhas não correspondentes, use uma join externa. Uma join externa retorna todas as linhas que atendem à condição de join, bem como algumas ou todas as linhas de uma tabela para as quais nenhuma linha da outra tabela atende à condição de join.

Há três tipos de joins externas:

- Externa Esquerda (`LEFT OUTER JOIN`)
- Externa Direita (`RIGHT OUTER JOIN`)
- Externa Integral (`FULL OUTER JOIN`)

## LEFT OUTER JOIN

```
SELECT e.last_name, e.department_id, d.department_name
FROM   employees e LEFT OUTER JOIN departments d
ON     (e.department_id = d.department_id) ;
```

LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
Whalen	10	Administration
Fay	20	Marketing
Hartstein	20	Marketing
...		
De Haan	90	Executive
Kochhar	90	Executive
King	90	Executive
Gietz	110	Accounting
Higgins	110	Accounting
Grant		

20 rows selected.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

### Exemplo de Join Externa Esquerda (LEFT OUTER JOIN)

Esta consulta recupera todas as linhas da tabela EMPLOYEES, que é a tabela esquerda, mesmo quando não há correspondência na tabela DEPARTMENTS.

## RIGHT OUTER JOIN

```
SELECT e.last_name, e.department_id, d.department_name
FROM   employees e RIGHT OUTER JOIN departments d
ON     (e.department_id = d.department_id) ;
```

LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
Whalen	10	Administration
Fay	20	Marketing
Hartstein	20	Marketing
Davies	50	Shipping
...		
Kochhar	90	Executive
Gietz	110	Accounting
Higgins	110	Accounting
	190	Contracting

20 rows selected.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

### Exemplo de Join Externa Direita (RIGHT OUTER JOIN)

Esta consulta recupera todas as linhas da tabela DEPARTMENTS, que é a tabela direita, mesmo quando não há correspondência na tabela EMPLOYEES.

## FULL OUTER JOIN

```
SELECT e.last_name, d.department_id, d.department_name
FROM   employees e FULL OUTER JOIN departments d
ON     (e.department_id = d.department_id) ;
```

LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
Whalen	10	Administration
Fay	20	Marketing
Hartstein	20	Marketing
...		
King	90	Executive
Gietz	110	Accounting
Higgins	110	Accounting
Grant		
	190	Contracting

21 rows selected.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

### Exemplo de Join Externa Integral (FULL OUTER JOIN)

Esta consulta recupera todas as linhas da tabela EMPLOYEES, mesmo quando não há correspondência na tabela DEPARTMENTS. Ela também recupera todas as linhas da tabela DEPARTMENTS, mesmo quando não há correspondência na tabela EMPLOYEES.

# Produtos Cartesianos

- **Um produto cartesiano será formado quando:**
  - Uma condição de join for omitida
  - Uma condição de join for inválida
  - Todas as linhas da primeira tabela se unirem a todas as linhas da segunda tabela
- **Para evitar um produto cartesiano, inclua sempre uma condição de join válida.**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Produtos Cartesianos

Quando uma condição de join é inválida ou completamente omitida, o resultado é um *produto cartesiano*, no qual todas as combinações de linhas são exibidas. Todas as linhas da primeira tabela são unidas a todas as linhas da segunda tabela.

Um produto cartesiano tende a gerar um grande número de linhas e o resultado raramente é útil. Inclua sempre uma condição de join válida, a menos que exista uma necessidade específica de combinar todas as linhas de todas as tabelas.

Os produtos cartesianos são úteis em alguns testes quando é necessário gerar muitas linhas para simular um volume razoável de dados.



# Gerando um Produto Cartesiano

## EMPLOYEES (20 linhas)

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
100	King	90
101	Kochhar	90
...		
202	Fay	20
205	Higgins	110
206	Gietz	110

20 rows selected.

## DEPARTMENTS (8 linhas)

DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID
10	Administration	1700
20	Marketing	1800
50	Shipping	1500
60	IT	1400
80	Sales	2500
90	Executive	1700
110	Accounting	1700
190	Contracting	1700

8 rows selected.

**Produto cartesiano:**  
**20 x 8 = 160 linhas**

EMPLOYEE_ID	DEPARTMENT_ID	LOCATION_ID
100	90	1700
101	90	1700
102	90	1700
103	60	1700
104	60	1700
107	60	1700
...		

160 rows selected.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Produtos Cartesianos (continuação)

Um produto cartesiano será gerado se uma condição de join for omitida. O exemplo do slide exibe o sobrenome e o nome do departamento dos funcionários com base nas tabelas EMPLOYEES e DEPARTMENTS. Como não foi especificada uma condição de join, todas as linhas (20) da tabela EMPLOYEES são unidas a todas as linhas (8) da tabela DEPARTMENTS, gerando 160 linhas na saída.

## Criando Joins Cruzadas

- A cláusula **CROSS JOIN** gera o produto cruzado de duas tabelas.
- Ele também é chamado de produto cartesiano entre as duas tabelas.

```
SELECT last_name, department_name  
FROM employees  
CROSS JOIN departments ;
```

LAST_NAME	DEPARTMENT_NAME
King	Administration
Kochhar	Administration
De Haan	Administration
Hunold	Administration

\*\*\*  
160 rows selected.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Criando Joins Cruzadas

O exemplo do slide resulta em um produto cartesiano das tabelas EMPLOYEES e DEPARTMENTS.

# Sumário

**Nesta lição, você aprendeu a usar joins para exibir dados de várias tabelas por meio de:**

- **Equijoins**
- **Não-equijoins**
- **Joins externas**
- **Auto-joins**
- **Joins cruzadas**
- **Joins naturais**
- **Joins externas integrais (ou de dois lados)**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

## Sumário

Há várias maneiras de unir tabelas.

### Tipos de Join

- Equijoins
- Não-equijoins
- Joins externas
- Auto-joins
- Joins cruzadas
- Joins naturais
- Joins externas integrais (ou de dois lados)

### Produtos Cartesianos

Um produto cartesiano resulta na exibição de todas as combinações de linhas. Para obter esse resultado, omita a cláusula WHERE ou especifique a cláusula CROSS JOIN.

### Apelidos de Tabelas

- Os apelidos de tabelas aceleram o acesso ao banco de dados.
- Eles podem ajudar a reduzir o código SQL, preservando a memória.

## Exercício 5: Visão Geral

**Este exercício aborda os seguintes tópicos:**

- **União de tabelas com uma equijoin**
- **Execução de auto-joins e joins externas**
- **Inclusão de condições**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

### Exercício 5: Visão Geral

Este exercício tem como objetivo proporcionar a você um treinamento prático de como extrair dados de mais de uma tabela com joins compatíveis com o padrão SQL:1999.

## Exercício 5

1. Crie uma consulta para o departamento de recursos humanos a fim de gerar os endereços de todos os departamentos. Use as tabelas `LOCATIONS` e `COUNTRIES`. Mostre o ID do local, o endereço, a cidade, o estado e o país na saída. Use `NATURAL JOIN` para gerar os resultados.

LOCATION_ID	STREET_ADDRESS	CITY	STATE_PROVINCE	COUNTRY_NAME
1400	2014 Jabberwocky Rd	Southlake	Texas	United States of America
1500	2011 Interiors Blvd	South San Francisco	California	United States of America
1700	2004 Charade Rd	Seattle	Washington	United States of America
1800	460 Bloor St. W.	Toronto	Ontario	Canada
2500	Magdalen Centre, The Oxford Science Park	Oxford	Oxford	United Kingdom

2. O departamento de recursos humanos precisa de um relatório de todos os funcionários. Crie uma consulta para exibir o sobrenome, o número do departamento e o nome do departamento de todos os funcionários.

LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
Whalen	10	Administration
Hartstein	20	Marketing
Fay	20	Marketing
Mourgos	50	Shipping
Rajs	50	Shipping
Davies	50	Shipping
Vargas	50	Shipping
■ ■ ■		
De Haan	90	Executive
Higgins	110	Accounting
Gietz	110	Accounting

19 rows selected.

## Exercício 5 (continuação)

- O departamento de recursos humanos precisa de um relatório dos funcionários em Toronto. Exiba o sobrenome, o cargo, o número do departamento e o nome do departamento de todos os funcionários que trabalham em Toronto.

LAST_NAME	JOB_ID	DEPARTMENT_ID	DEPARTMENT_NAME
Hartstein	MK_MAN	20	Marketing
Fay	MK_REP	20	Marketing

- Crie um relatório para exibir o sobrenome e o número dos funcionários, bem como o sobrenome e o número dos respectivos gerentes. Atribua às colunas os labels Employee, Emp#, Manager e Mgr#, respectivamente. Inclua a instrução SQL no arquivo de texto lab\_05\_04.sql.

Employee	EMP#	Manager	Mgr#
Kochhar	101	King	100
De Haan	102	King	100
Mourgos	124	King	100
Zlotkey	149	King	100
Hartstein	201	King	100
Whalen	200	Kochhar	101
Higgins	205	Kochhar	101
Hunold	103	De Haan	102
Ernst	104	Hunold	103
Lorentz	107	Hunold	103
Rajs	141	Mourgos	124
Davies	142	Mourgos	124
Matos	143	Mourgos	124
Vargas	144	Mourgos	124
Employee	EMP#	Manager	Mgr#
Abel	174	Zlotkey	149
Taylor	176	Zlotkey	149
Grant	178	Zlotkey	149
Fay	202	Hartstein	201
Gietz	206	Higgins	205

19 rows selected.

## Exercício 5 (continuação)

5. Modifique `lab_05_04.sql` para exibir todos os funcionários, inclusive King, que não possui gerente. Ordene os resultados pelo número do funcionário. Inclua a instrução SQL no arquivo de texto `lab_05_05.sql`. Execute a consulta em `lab_05_05.sql`.

Employee	EMP#	Manager	Mgr#
King	100		
Kochhar	101	King	100
De Haan	102	King	100
Hunold	103	De Haan	102
Ernst	104	Hunold	103
Lorentz	107	Hunold	103
Mourgos	124	King	100

■ ■ ■

20 rows selected.

6. Crie um relatório para o departamento de recursos humanos que exiba os sobrenomes e os números de departamento dos funcionários, bem como todos os funcionários que trabalham no mesmo departamento como um funcionário específico. Atribua um label apropriado a cada coluna. Salve o script no arquivo `lab_05_06.sql`.

DEPARTMENT	EMPLOYEE	COLLEAGUE
20	Fay	Hartstein
20	Hartstein	Fay
50	Davies	Matos
50	Davies	Mourgos
50	Davies	Rajs
50	Davies	Vargas
50	Matos	Davies
50	Matos	Mourgos
50	Matos	Rajs
50	Matos	Vargas
50	Mourgos	Davies
50	Mourgos	Matos
50	Mourgos	Rajs
50	Mourgos	Vargas

■ ■ ■

42 rows selected.

## Exercício 5 (continuação)

7. O departamento de recursos humanos precisa de um relatório sobre níveis de cargos e salários. Para se familiarizar com a tabela `JOB_GRADES`, primeiro mostre a estrutura dessa tabela. Em seguida, crie uma consulta que exiba o nome, o cargo, o nome do departamento, o salário e o nível de todos os funcionários.

Name	Null?	Type
GRADE_LEVEL		VARCHAR2(3)
LOWEST_SAL		NUMBER
HIGHEST_SAL		NUMBER

LAST_NAME	JOB_ID	DEPARTMENT_NAME	SALARY	GRA
Matos	ST_CLERK	Shipping	2600	A
Vargas	ST_CLERK	Shipping	2500	A
Lorentz	IT_PROG	IT	4200	B
Mourgos	ST_MAN	Shipping	5800	B
Rajs	ST_CLERK	Shipping	3500	B
Davies	ST_CLERK	Shipping	3100	B
Whalen	AD_ASST	Administration	4400	B

■ ■ ■

19 rows selected.

Se quiser tomar parte em mais um desafio, faça estes exercícios:

8. O departamento de recursos humanos deseja determinar os nomes de todos os funcionários admitidos após Davies. Crie uma consulta para exibir o nome e a data de admissão de todos os funcionários admitidos após Davies.

LAST_NAME	HIRE_DATE
Lorentz	07-FEB-99
Mourgos	16-NOV-99
Matos	15-MAR-98
Vargas	09-JUL-98
Zlotkey	29-JAN-00
Taylor	24-MAR-98
Grant	24-MAY-99
Fay	17-AUG-97

8 rows selected.



## Exercício 5 (continuação)

9. O departamento de recursos humanos precisa obter os nomes e as datas de admissão de todos os funcionários admitidos antes dos respectivos gerentes, além dos nomes e das datas de admissão desses gerentes. Salve o script no arquivo `lab5_09.sql`.

LAST_NAME	HIRE_DATE	LAST_NAME	HIRE_DATE
Whalen	17-SEP-87	Kochhar	21-SEP-89
Hunold	03-JAN-90	De Haan	13-JAN-93
Rajs	17-OCT-95	Mourgos	16-NOV-99
Davies	29-JAN-97	Mourgos	16-NOV-99
Matos	15-MAR-98	Mourgos	16-NOV-99
Vargas	09-JUL-98	Mourgos	16-NOV-99
Abel	11-MAY-96	Zlotkey	29-JAN-00
Taylor	24-MAR-98	Zlotkey	29-JAN-00
Grant	24-MAY-99	Zlotkey	29-JAN-00

9 rows selected.



# 6

## Usando Subconsultas para Solucionar Consultas

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

# Objetivos

**Ao concluir esta lição, você será capaz de:**

- **Definir subconsultas**
- **Descrever os tipos de problemas que as subconsultas podem solucionar**
- **Listar os tipos de subconsultas**
- **Criar subconsultas de uma única linha e de várias linhas**

ORACLE

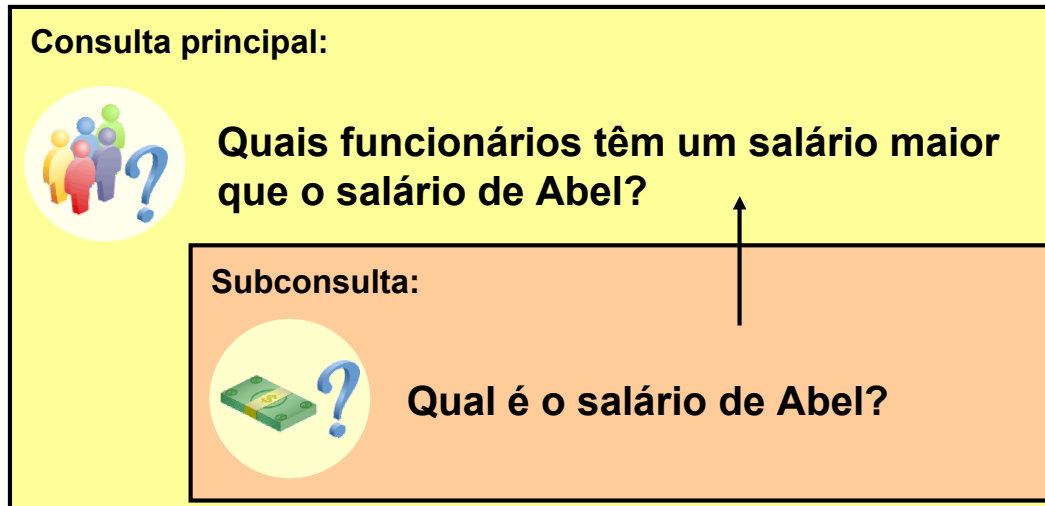
Copyright © 2004, Oracle. Todos os direitos reservados.

## Objetivos

Nesta lição, você conhecerá os recursos mais avançados da instrução `SELECT`. É possível criar subconsultas na cláusula `WHERE` de outra instrução SQL para obter valores baseados em um valor condicional desconhecido. Esta lição aborda subconsultas de uma única linha e de várias linhas.

# Usando uma Subconsulta para Solucionar um Problema

Quem tem um salário maior que o salário de Abel?



ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Usando uma Subconsulta para Solucionar um Problema

Suponha que você queira criar uma consulta para saber quem ganha um salário maior que o salário de Abel.

Para solucionar esse problema, são necessárias *duas* consultas: uma para saber quanto Abel ganha e outra para saber quem ganha uma quantia maior.

Você pode solucionar esse problema com a combinação das duas consultas, *inserindo* uma na outra.

A consulta interna (ou *subconsulta*) retorna um valor usado pela consulta externa (ou *consulta principal*). Usar uma subconsulta é o mesmo que executar duas consultas em seqüência e utilizar o resultado da primeira como o valor de pesquisa na segunda.

# Sintaxe da Subconsulta

```
SELECT    select_list
FROM      table
WHERE     expr operator
          (SELECT    select_list
          FROM      table);
```

- A subconsulta (consulta interna) é executada uma vez antes da consulta principal (consulta externa).
- O resultado da subconsulta é usado pela consulta principal.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

## Sintaxe da Subconsulta

Uma subconsulta é uma instrução `SELECT` incorporada a uma cláusula de outra instrução `SELECT`. É possível criar instruções complexas a partir de instruções simples usando subconsultas. Elas poderão ser muito úteis quando for necessário selecionar linhas de uma tabela com uma condição que dependa dos dados da própria tabela.

É possível inserir a subconsulta em várias cláusulas SQL, inclusive nestas:

- Cláusula `WHERE`
- Cláusula `HAVING`
- Cláusula `FROM`

Na sintaxe:

*operator* inclui uma condição de comparação, como `>`, `=` ou `IN`

**Observação:** As condições de comparação estão incluídas em duas classes: operadores de uma única linha (`>`, `=`, `>=`, `<`, `<>`, `<=`) e de várias linhas (`IN`, `ANY`, `ALL`).

Em geral, a subconsulta é denominada instrução `SELECT` interna, sub-`SELECT` ou `SELECT` aninhada. A subconsulta normalmente é executada primeiro e o seu resultado é usado para concluir a condição da consulta principal (ou externa).

## Usando uma Subconsulta

```
SELECT last_name  
FROM employees 11000  
WHERE salary >  
      (SELECT salary  
       FROM employees  
       WHERE last_name = 'Abel');
```

LAST_NAME
King
Kochhar
De Haan
Hartstein
Higgins

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

### Usando uma Subconsulta

No slide, a consulta interna determina o salário do funcionário Abel. A consulta externa recebe o resultado da consulta interna e o utiliza para exibir todos os funcionários que ganham mais que essa quantia.

## Diretrizes de Uso de Subconsultas

- **Delimite subconsultas por parênteses.**
- **Posicione subconsultas à direita da condição de comparação.**
- **A cláusula ORDER BY não será necessária na subconsulta, a menos que uma análise Top-N seja executada.**
- **Use operadores de uma única linha com subconsultas de uma única linha e operadores de várias linhas com subconsultas de várias linhas.**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

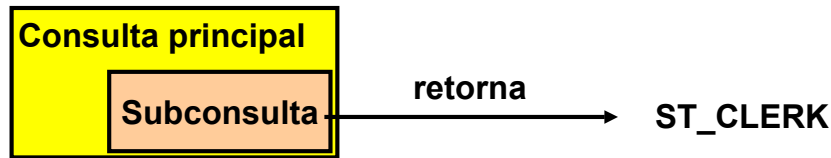
### Diretrizes de Uso de Subconsultas

- Uma subconsulta deve ser delimitada por parênteses.
- Posicione a subconsulta à direita da condição de comparação para fins de legibilidade.
- No Oracle8i e em releases mais recentes, uma cláusula ORDER BY pode ser usada, além de ser obrigatória na subconsulta para executar uma análise Top-N.
  - Entretanto, antes do Oracle8i, as subconsultas não podiam conter uma cláusula ORDER BY. Apenas uma cláusula ORDER BY era permitida em uma instrução SELECT e, se fosse especificada, deveria ser a última cláusula na instrução SELECT principal.
- Duas classes de condições de comparação são usadas em subconsultas: operadores de uma única linha e operadores de várias linhas.

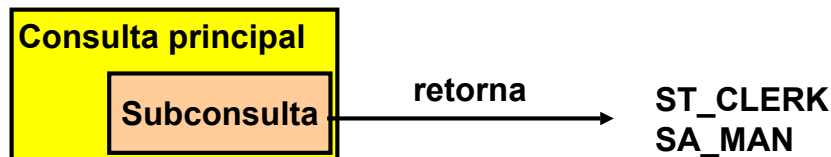


# Tipos de Subconsultas

- Subconsulta de uma única linha



- Subconsulta de várias linhas



ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## 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

**Observação:** Há também subconsultas de várias colunas, ou seja, consultas que retornam mais de uma coluna da instrução SELECT interna. Elas são abordadas no curso *Banco de Dados Oracle 10g: Fundamentos de SQL II*.

# Subconsultas de uma Única Linha

- Retornam somente uma linha
- Usam operadores de comparação de uma única linha

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

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

## Subconsultas de uma Única Linha

Uma 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

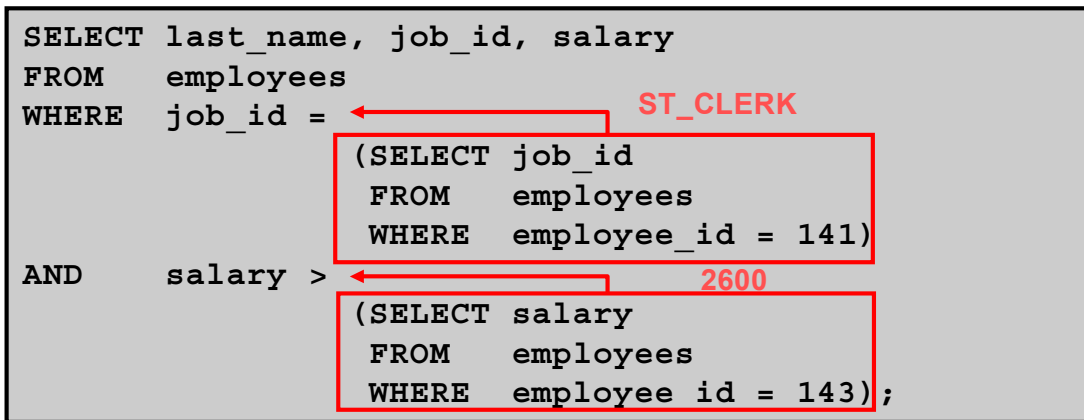
Exiba os funcionários cujo ID de cargo é igual ao do funcionário 141:

```
SELECT last_name, job_id
FROM employees
WHERE job_id =
      (SELECT job_id
       FROM employees
       WHERE employee_id = 141);
```

LAST_NAME	JOB_ID
Rajs	ST_CLERK
Davies	ST_CLERK
Matos	ST_CLERK
Vargas	ST_CLERK

# Executando Subconsultas de uma Única Linha

```
SELECT last_name, job_id, salary
FROM   employees
WHERE  job_id = (SELECT job_id
                 FROM   employees
                 WHERE  employee_id = 141)
AND    salary > (SELECT salary
                 FROM   employees
                 WHERE  employee_id = 143);
```



LAST_NAME	JOB_ID	SALARY
Rajs	ST_CLERK	3500
Davies	ST_CLERK	3100

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Executando Subconsultas de uma Única Linha

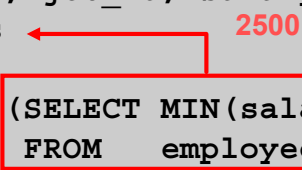
Uma instrução SELECT pode ser considerada um bloco de consulta. O exemplo do slide mostra os funcionários cujo ID de cargo é igual ao do funcionário 141 e cujo salário é maior que o do funcionário 143.

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 ST\_CLERK e 2600, respectivamente. O bloco de consulta externa é processado depois e usa os valores retornados pelas consultas internas para concluir suas condições de pesquisa. As duas consultas internas retornam valores únicos (ST\_CLERK e 2600, respectivamente); por isso, essa instrução SQL é denominada subconsulta de uma única linha.

**Observação:** As consultas internas e externas podem obter dados de tabelas distintas.

## Usando Functions de Grupo em uma Subconsulta

```
SELECT last_name, job_id, salary
FROM   employees
WHERE  salary =
      (SELECT MIN(salary)
       FROM   employees);
```



LAST_NAME	JOB_ID	SALARY
Vargas	ST_CLERK	2500

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

### Usando Functions de Grupo em uma Subconsulta

É possível exibir dados de uma consulta principal usando uma function de grupo em uma subconsulta para retornar uma única linha. A subconsulta é delimitada por parênteses e é posicionada após a condição de comparação.

O exemplo do slide mostra o sobrenome, o ID do cargo e o salário de todos os funcionários cujo salário é igual ao salário mínimo. A function de grupo MIN retorna um único valor (2500) para a consulta externa.

## A Cláusula HAVING com Subconsultas

- O servidor Oracle executa primeiro as subconsultas.
- Ele retorna os resultados para a cláusula HAVING da consulta principal.

```
SELECT  department_id, MIN(salary)
FROM    employees
GROUP BY department_id
HAVING  MIN(salary) > (SELECT MIN(salary)
                       FROM    employees
                       WHERE    department_id = 50);
```

Diagram illustrating the execution of the SQL query. A red box highlights the `HAVING MIN(salary) >` clause. A red arrow points from the `MIN(salary)` in the `HAVING` clause to the subquery result, which is `2500`. The subquery is also highlighted with a red box.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

### A Cláusula HAVING com Subconsultas

É possível usar subconsultas tanto na cláusula WHERE como na cláusula HAVING. O servidor Oracle executa a subconsulta, e os resultados são retornados para a cláusula HAVING da consulta principal.

A instrução SQL do slide mostra todos os departamentos cujo salário mínimo é maior que o do departamento 50.

DEPARTMENT_ID	MIN(SALARY)
10	4400
20	6000
...	
	7000

7 rows selected.

#### Exemplo

Localize o cargo com o menor salário médio.

```
SELECT  job_id, AVG(salary)
FROM    employees
GROUP BY job_id
HAVING  AVG(salary) = (SELECT  MIN(AVG(salary))
                       FROM    employees
                       GROUP BY job_id);
```

# O Que Está Errado Nesta Instrução?

```
SELECT employee_id, last_name
FROM   employees
WHERE  salary =
      (SELECT MIN(salary)
       FROM   employees
       GROUP BY department_id);
```

```
ERROR at line 4:
ORA-01427: single-row subquery returns more than
one row
```

**Operador de uma única linha com  
uma subconsulta de várias linhas**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Erros com Subconsultas

Um erro comum com subconsultas ocorre quando mais de uma linha é retornada para uma subconsulta de uma única linha.

Na instrução SQL do slide, a subconsulta contém uma cláusula GROUP BY, o que significa que ela retornará várias linhas, uma para cada grupo localizado. Nesse caso, os resultados da subconsulta são 4400, 6000, 2500, 4200, 7000, 17000 e 8300.

A consulta externa recebe esses resultados e os utiliza em sua cláusula WHERE. A cláusula WHERE contém um operador de igual a (=), que é um operador de comparação de uma única linha que espera apenas um valor. Esse 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 Retornará Linhas?

```
SELECT last_name, job_id
FROM   employees
WHERE  job_id =
      (SELECT job_id
       FROM   employees
       WHERE  last_name = 'Haas');
```

no rows selected

**A subconsulta não retorna nenhum valor.**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

### Problemas com Subconsultas

Um problema comum com subconsultas ocorre quando nenhuma linha é retornada pela consulta interna.

Na instrução SQL do slide, a subconsulta contém uma cláusula WHERE. Presumivelmente, o objetivo é localizar o funcionário cujo nome é Haas. A instrução está correta, mas não seleciona nenhuma linha quando é executada.

Não há nenhum funcionário com esse nome. Por isso, a subconsulta não retorna nenhuma linha. A consulta externa recebe o resultado da subconsulta (nulo) e o utiliza em sua cláusula WHERE. Ela não encontra nenhum funcionário com o ID de cargo igual a nulo e, portanto, não retorna nenhuma linha. Se existisse um cargo com o valor nulo, a linha não seria retornada porque a comparação de dois valores nulos resulta em um valor nulo; assim, a condição WHERE não é verdadeira.

# Subconsultas de Várias Linhas

- Retornam mais de uma linha
- Usam operadores de comparação de várias linhas

Operador	Significado
IN	Igual a qualquer membro da lista
ANY	Compara o valor com cada valor retornado pela subconsulta
ALL	Compara o valor com todos os valores retornados pela subconsulta

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

## Subconsultas de Várias Linhas

As subconsultas que retornam mais de uma linha são denominadas subconsultas de várias linhas. Com uma subconsulta de várias linhas, você usa um operador de várias linhas, em vez de um operador de uma única linha. O operador de várias linhas espera um ou mais valores:

```
SELECT last_name, salary, department_id
FROM employees
WHERE salary IN (SELECT MIN(salary)
                  FROM employees
                  GROUP BY department_id);
```

### Exemplo

Localize os funcionários cujo salário é igual ao salário mínimo em cada departamento.

A consulta interna é executada primeiro e produz um resultado. O bloco de consulta principal é processado depois e usa os valores retornados pela consulta interna para concluir sua condição de pesquisa. Na verdade, a consulta principal aparece para o servidor Oracle da seguinte forma:

```
SELECT last_name, salary, department_id
FROM employees
WHERE salary IN (2500, 4200, 4400, 6000, 7000, 8300,
                 8600, 17000);
```



# Usando o Operador ANY em Subconsultas de Várias Linhas

```
SELECT employee_id, last_name, job_id, salary
FROM   employees
WHERE  salary < ANY
      (SELECT salary
       FROM   employees
       WHERE  job_id = 'IT_PROG')
AND    job_id <> 'IT_PROG';
```

EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
124	Mourgos	ST_MAN	5800
141	Rajs	ST_CLERK	3500
142	Davies	ST_CLERK	3100
143	Matos	ST_CLERK	2600
144	Vargas	ST_CLERK	2500

...  
10 rows selected.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Subconsultas de Várias Linhas (continuação)

O operador ANY (e seu sinônimo, o operador SOME) compara um valor a *cada* valor retornado por uma subconsulta. O exemplo do slide mostra os funcionários que não são programadores de computação e cujo salário é menor que o de qualquer programador de computação. O salário máximo de um programador é US\$9.000.

<ANY significa menor que o máximo. >ANY significa maior que o mínimo. =ANY equivale a IN.

## Usando o Operador ALL em Subconsultas de Várias Linhas

```
SELECT employee_id, last_name, job_id, salary
FROM   employees          9000, 6000, 4200
WHERE  salary < ALL
      (SELECT salary
       FROM   employees
       WHERE  job_id = 'IT_PROG')
AND    job_id <> 'IT_PROG';
```

EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
141	Rajs	ST_CLERK	3500
142	Davies	ST_CLERK	3100
143	Matos	ST_CLERK	2600
144	Vargas	ST_CLERK	2500

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

### Subconsultas de Várias Linhas (continuação)

O operador ALL compara um valor com *todos* os valores retornados por uma subconsulta. O exemplo do slide mostra os funcionários cujo salário é menor que o salário de todos os funcionários com o ID de cargo IT\_PROG e cujo cargo não é IT\_PROG.

>ALL significa maior que o máximo e <ALL significa menor que o mínimo.

É possível usar o operador NOT com os operadores IN, ANY e ALL.

## Valores Nulos em uma Subconsulta

```
SELECT emp.last_name
FROM   employees emp
WHERE  emp.employee_id NOT IN
      (SELECT mgr.manager_id
       FROM   employees mgr);
```

no rows selected

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

### Retornando Valores Nulos no Conjunto Resultante de uma Subconsulta

A instrução SQL do slide tenta exibir todos os funcionários sem subordinados. Logicamente, essa instrução deveria ter retornado 12 linhas. No entanto, ela não retorna nenhuma linha. Um dos valores retornados pela consulta interna é um valor nulo e, por isso, a consulta inteira não retorna nenhuma linha.

O motivo é que todas as condições que comparam um valor nulo resultam em um valor nulo. Dessa forma, sempre que houver a possibilidade de valores nulos integrarem o conjunto de resultados de uma subconsulta, não use o operador NOT IN. Esse operador corresponde a <> ALL.

Observe que o valor nulo como parte do conjunto de resultados de uma subconsulta não representa um problema se você usa o operador IN. Esse operador corresponde a =ANY. Por exemplo, para exibir os funcionários com subordinados, use a seguinte instrução SQL:

```
SELECT emp.last_name
FROM   employees emp
WHERE  emp.employee_id IN
      (SELECT mgr.manager_id
       FROM   employees mgr);
```

## Retornando Valores Nulos no Conjunto Resultante de uma Subconsulta (continuação)

Como alternativa, é possível incluir uma cláusula WHERE na subconsulta para exibir todos os funcionários sem subordinados:

```
SELECT last_name FROM employees
WHERE employee_id NOT IN
      (SELECT manager_id
       FROM employees
       WHERE manager_id IS NOT NULL);
```

# Sumário

Nesta lição, você aprendeu a:

- Identificar quando uma subconsulta pode ajudar a solucionar um problema
- Criar subconsultas quando uma consulta se basear em valores desconhecidos

```
SELECT    select_list
FROM      table
WHERE     expr operator
          (SELECT select_list
           FROM    table);
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Sumário

Nesta lição, você aprendeu a usar subconsultas. Uma subconsulta é uma instrução SELECT incorporada a uma cláusula de outra instrução SQL. As subconsultas são úteis quando uma consulta se baseia em um critério de pesquisa com valores intermediários desconhecidos.

As subconsultas apresentam as seguintes características:

- Podem passar uma linha de dados para uma instrução principal que contém um operador de uma única linha, como =, <>, >, >=, < ou <=
- Podem passar várias linhas de dados para uma instrução principal que contém um operador de várias linhas, como IN
- São processadas primeiro pelo servidor Oracle. Em seguida, a cláusula WHERE ou HAVING utiliza os resultados
- Podem conter functions de grupo

## Exercício 6: Visão Geral

**Este exercício aborda os seguintes tópicos:**

- **Criação de subconsultas para consultar valores baseados em critérios desconhecidos**
- **Utilização de subconsultas para descobrir os valores existentes em um conjunto de dados, e não em outro**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

### Exercício 6: Visão Geral

Neste exercício, você criará consultas complexas com instruções SELECT aninhadas.

#### Perguntas Impressas

Convém criar a consulta interna primeiro para estas perguntas. Certifique-se de que ela seja executada e gere os dados previstos antes de você codificar a consulta externa.

## Exercício 6

1. O departamento de recursos humanos precisa de uma consulta que solicite ao usuário o sobrenome de um funcionário. A consulta exibe o sobrenome e a data de admissão de todos os funcionários no mesmo departamento do funcionário cujo nome foi fornecido (excluindo esse funcionário). Por exemplo, se o usuário informar Zlotkey, serão exibidos todos os funcionários que trabalham com Zlotkey (excluindo ele próprio).

LAST_NAME	HIRE_DATE
Abel	11-MAY-96
Taylor	24-MAR-98

2. Crie um relatório que exiba o número e o sobrenome de todos os funcionários cujo salário é maior que o salário médio. Classifique os resultados em ordem crescente de salário.

EMPLOYEE_ID	LAST_NAME	SALARY
103	Hunold	9000
149	Zlotkey	10500
174	Abel	11000
205	Higgins	12000
201	Hartstein	13000
101	Kochhar	17000
102	De Haan	17000
100	King	24000

8 rows selected.

3. Crie uma consulta que exiba o número e o sobrenome de todos os funcionários que trabalham em um departamento com funcionários cujos sobrenomes contêm a letra *u*. Inclua a instrução SQL no arquivo de texto lab\_06\_03.sql. Execute a consulta.

EMPLOYEE_ID	LAST_NAME
124	Mourgos
141	Rajs
142	Davies
143	Matos
144	Vargas
103	Hunold
104	Ernst
107	Lorentz

8 rows selected.

## Exercício 6 (continuação)

4. O departamento de recursos humanos precisa de um relatório que exiba o sobrenome, o número do departamento e o ID do cargo de todos os funcionários cujo ID de local do departamento é 1700.

LAST_NAME	DEPARTMENT_ID	JOB_ID
Whalen	10	AD_ASST
King	90	AD PRES
Kochhar	90	AD_VP
De Haan	90	AD_VP
Higgins	110	AC_MGR
Gietz	110	AC_ACCOUNT

6 rows selected.

Modifique a consulta para que um ID de local seja solicitado ao usuário. Salve-a no arquivo lab\_06\_04.sql.

5. Crie um relatório para o departamento de recursos humanos que exiba o sobrenome e o salário de todos os funcionários subordinados a King.

LAST_NAME	SALARY
Kochhar	17000
De Haan	17000
Mourgos	5800
Zlotkey	10500
Hartstein	13000

6. Crie um relatório para o departamento de recursos humanos que exiba o número do departamento, o sobrenome e o ID do cargo de todos os funcionários no departamento executivo.

DEPARTMENT_ID	LAST_NAME	JOB_ID
90	King	AD PRES
90	Kochhar	AD_VP
90	De Haan	AD_VP

Se tiver tempo, faça o seguinte exercício:

7. Modifique a consulta em lab\_06\_03.sql para exibir o número, o sobrenome, bem como o salário de todos os funcionários que ganham mais que o salário médio e trabalham em um departamento com funcionários cujos sobrenomes contêm a letra u. Salve novamente lab\_06\_03.sql como lab\_06\_07.sql. Execute a instrução em lab\_06\_07.sql.

EMPLOYEE_ID	LAST_NAME	SALARY
103	Hunold	9000



# Usando os Operadores de Conjunto

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

# Objetivos

**Ao concluir esta lição, você será capaz de:**

- **Descrever os operadores de conjunto**
- **Usar um operador de conjunto para combinar várias consultas em uma só**
- **Controlar a ordem das linhas retornadas**

ORACLE

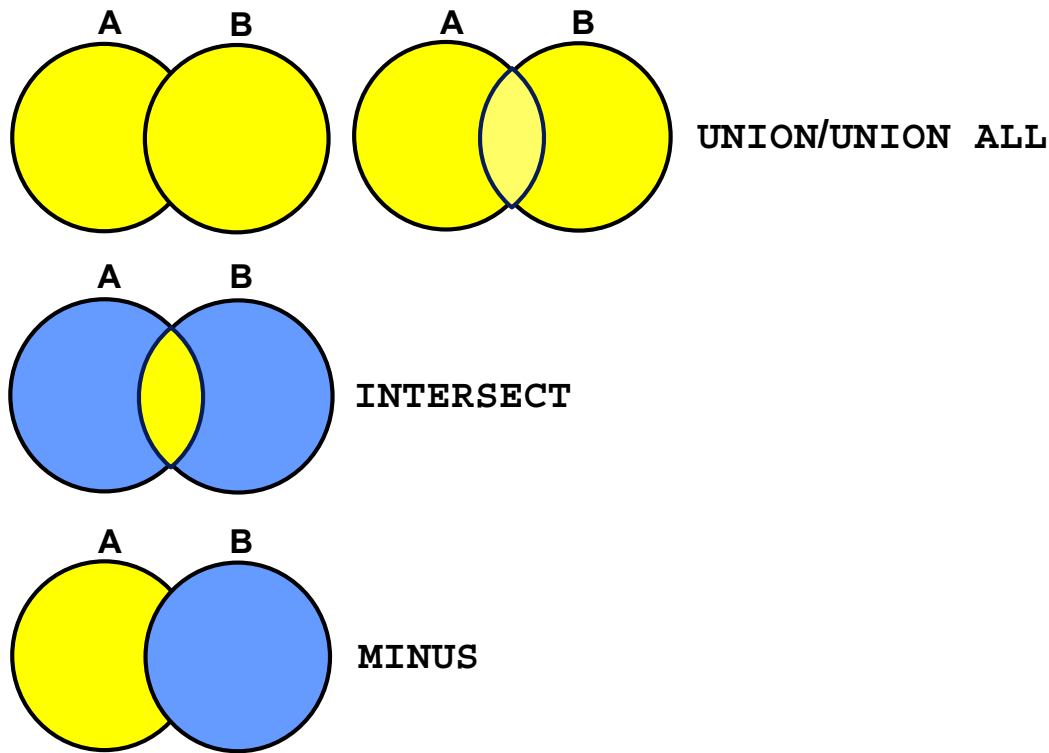
Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

## Objetivos

Nesta lição, você aprenderá a criar consultas com os operadores de conjunto.

## Operadores de Conjunto



ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Operadores de Conjunto

Os operadores de conjunto combinam os resultados de duas ou mais consultas componentes em um resultado. As consultas que contêm operadores de conjunto são denominadas *consultas compostas*.

Operador	Retorna
UNION	Todas as linhas distintas selecionadas por qualquer uma das consultas
UNION ALL	Todas as linhas selecionadas por qualquer uma das consultas, inclusive as linhas duplicadas
INTERSECT	Todas as linhas distintas selecionadas pelas duas consultas
MINUS	Todas as linhas distintas selecionadas pela primeira instrução SELECT e não selecionadas na segunda instrução SELECT

Todos os operadores de conjunto têm a mesma precedência. Se uma instrução SQL contiver vários operadores de conjunto, o servidor Oracle os avaliará da esquerda (superior) para a direita (inferior) caso não haja parênteses especificando explicitamente outra ordem. Use parênteses para especificar explicitamente a ordem de avaliação nas consultas que utilizam o operador INTERSECT com outros operadores de conjunto.

# Tabelas Usadas Nesta Lição

**As tabelas usadas nesta lição são:**

- **EMPLOYEES:** Fornece detalhes sobre todos os funcionários atuais
- **JOB\_HISTORY:** Registra os detalhes relativos às datas inicial e final do cargo antigo, ao número de identificação do cargo e ao departamento quando um funcionário muda de cargo

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

## Tabelas Usadas Nesta Lição

Duas tabelas são usadas nesta lição: EMPLOYEES e JOB\_HISTORY.

A tabela EMPLOYEES armazena detalhes dos funcionários. Nos registros de recursos humanos, essa tabela armazena um número de identificação exclusivo e um endereço de e-mail para cada funcionário. Também são armazenados os detalhes do número de identificação de cargo, salário e gerente do funcionário. Além do salário, alguns funcionários recebem comissão, e essa informação também é rastreada. A empresa organiza as atribuições dos funcionários em cargos. Alguns funcionários trabalham há muito tempo na empresa e passaram por cargos diferentes. Essas informações são monitoradas com a tabela JOB\_HISTORY. Quando um funcionário muda de cargo, os detalhes relativos às datas inicial e final do cargo antigo, ao número de identificação do cargo e ao departamento são registrados na tabela JOB\_HISTORY.

A estrutura e os dados das tabelas EMPLOYEES e JOB\_HISTORY são mostrados nas páginas seguintes.

## Tabelas Usadas Nesta Lição (continuação)

Houve casos na empresa de pessoas que ocuparam o mesmo cargo mais de uma vez durante a sua permanência nesse estabelecimento. Por exemplo, considere o funcionário Taylor, admitido na empresa em 24-MAR-1998. Taylor ocupou o cargo SA\_REP de 24-MAR-98 a 31-DEC-98 e o cargo SA\_MAN de 01-JAN-99 a 31-DEC-99. Ele voltou para o cargo SA\_REP, que é o seu cargo atual.

Da mesma forma, considere o funcionário Whalen, admitido na empresa em 17-SEP-1987. Whalen ocupou o cargo AD\_ASST de 17-SEP-87 a 17-JUN-93 e o cargo AC\_ACCOUNT de 01-JUL-94 a 31-DEC-98. Ele voltou para o cargo AD\_ASST, que é o seu cargo atual.

DESCRIBE employees

Name	Null?	Type
EMPLOYEE_ID	NOT NULL	NUMBER(6)
FIRST_NAME		VARCHAR2(20)
LAST_NAME	NOT NULL	VARCHAR2(25)
EMAIL	NOT NULL	VARCHAR2(25)
PHONE_NUMBER		VARCHAR2(20)
HIRE_DATE	NOT NULL	DATE
JOB_ID	NOT NULL	VARCHAR2(10)
SALARY		NUMBER(8,2)
COMMISSION_PCT		NUMBER(2,2)
MANAGER_ID		NUMBER(6)
DEPARTMENT_ID		NUMBER(4)
DEPARTMENT_NAME		VARCHAR2(14)

## Tabelas Usadas Nesta Lição (continuação)

```
SELECT employee_id, last_name, job_id, hire_date, department_id
FROM employees;
```

EMPLOYEE_ID	LAST_NAME	JOB_ID	HIRE_DATE	DEPARTMENT_ID
100	King	AD_PRES	17-JUN-87	90
101	Kochhar	AD_VP	21-SEP-89	90
102	De Haan	AD_VP	13-JAN-93	90
103	Hunold	IT_PROG	03-JAN-90	60
104	Ernst	IT_PROG	21-MAY-91	60
107	Lorentz	IT_PROG	07-FEB-99	60
124	Mourgos	ST_MAN	16-NOV-99	50
141	Rajs	ST_CLERK	17-OCT-95	50
142	Davies	ST_CLERK	29-JAN-97	50
143	Matos	ST_CLERK	15-MAR-98	50
144	Vargas	ST_CLERK	09-JUL-98	50
149	Zlotkey	SA_MAN	29-JAN-00	80
174	Abel	SA_REP	11-MAY-96	80
176	Taylor	SA_REP	24-MAR-98	80
EMPLOYEE_ID	LAST_NAME	JOB_ID	HIRE_DATE	DEPARTMENT_ID
178	Grant	SA_REP	24-MAY-99	
200	Whalen	AD_ASST	17-SEP-87	10
201	Hartstein	MK_MAN	17-FEB-96	20

■■■  
20 rows selected.

```
DESCRIBE job_history
```

Name	Null?	Type
EMPLOYEE_ID	NOT NULL	NUMBER(6)
START_DATE	NOT NULL	DATE
END_DATE	NOT NULL	DATE
JOB_ID	NOT NULL	VARCHAR2(10)
DEPARTMENT_ID		NUMBER(4)

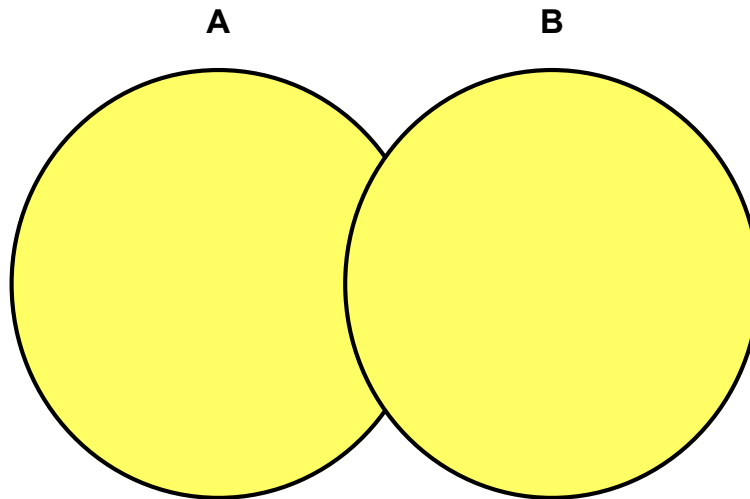
## Tabelas Usadas Nesta Lição (continuação)

```
SELECT * FROM job_history;
```

EMPLOYEE_ID	START_DAT	END_DATE	JOB_ID	DEPARTMENT_ID
102	13-JAN-93	24-JUL-98	IT_PROG	60
101	21-SEP-89	27-OCT-93	AC_ACCOUNT	110
101	28-OCT-93	15-MAR-97	AC_MGR	110
201	17-FEB-96	19-DEC-99	MK_REP	20
114	24-MAR-98	31-DEC-99	ST_CLERK	50
122	01-JAN-99	31-DEC-99	ST_CLERK	50
200	17-SEP-87	17-JUN-93	AD_ASST	90
176	24-MAR-98	31-DEC-98	SA_REP	80
176	01-JAN-99	31-DEC-99	SA_MAN	80
200	01-JUL-94	31-DEC-98	AC_ACCOUNT	90

10 rows selected.

# Operador UNION



**O operador UNION retorna resultados das duas consultas após eliminar os valores duplicados.**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Operador UNION

O operador UNION retorna todas as linhas selecionadas pelas consultas. Utilize-o para retornar todas as linhas de várias tabelas e eliminar linhas duplicadas.

### Diretrizes

- O número de colunas e os tipos de dados das colunas selecionadas devem ser idênticos em todas as instruções SELECT usadas na consulta. Os nomes das colunas não precisam ser idênticos.
- UNION opera em todas as colunas selecionadas.
- Os valores nulos não são ignorados durante a verificação de valores duplicados.
- O operador IN tem precedência em relação ao operador UNION.
- Por default, a saída é classificada em ordem crescente da primeira coluna da cláusula SELECT.



# Usando o Operador UNION

**Exiba os detalhes dos cargos atual e anterior de todos os funcionários. Exiba cada funcionário apenas uma vez.**

```
SELECT employee_id, job_id
FROM employees
UNION
SELECT employee_id, job_id
FROM job_history;
```

EMPLOYEE_ID	JOB_ID
100	AD_PRES
101	AC_ACCOUNT
...	
200	AC_ACCOUNT
200	AD_ASST
...	
205	AC_MGR
206	AC_ACCOUNT

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Usando o Operador UNION

O operador UNION elimina registros duplicados. Se houver registros idênticos nas tabelas EMPLOYEES e JOB\_HISTORY, eles serão exibidos apenas uma vez. Na saída mostrada no slide, observe que o registro para o funcionário com EMPLOYEE\_ID 200 aparece duas vezes, já que o valor em JOB\_ID é diferente em cada linha.

Considere o seguinte exemplo:

```
SELECT employee_id, job_id, department_id
FROM employees
UNION
SELECT employee_id, job_id, department_id
FROM job_history;
```

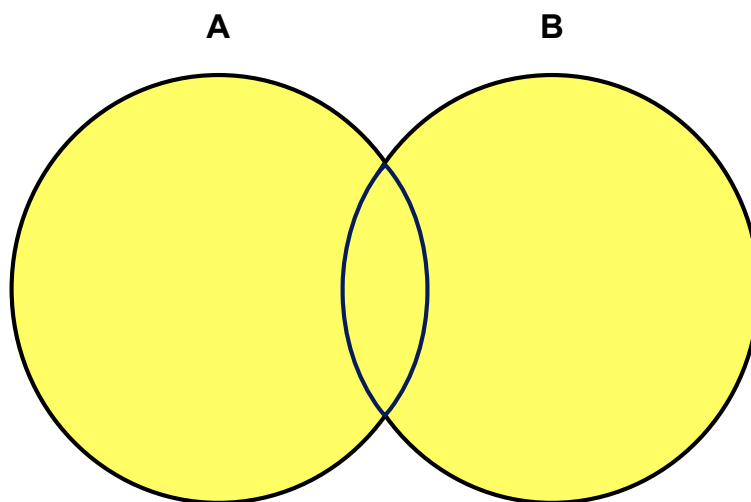
EMPLOYEE_ID	JOB_ID	DEPARTMENT_ID
...		
200	AC_ACCOUNT	90
200	AD_ASST	10
200	AD_ASST	90
...		

29 rows selected.

## Usando o Operador UNION (continuação)

Na saída anterior, o funcionário 200 aparece três vezes. Por quê? Observe os valores de DEPARTMENT\_ID relativos a esse funcionário. O valor de DEPARTMENT\_ID em uma linha é 90, em outra é 10 e na terceira é 90. Em função dessas combinações exclusivas de IDs de cargo e IDs de departamento, cada linha relativa ao funcionário 200 é exclusiva e, por isso, não é considerada duplicada. Observe que a saída é classificada em ordem crescente da primeira coluna da cláusula SELECT (nesse caso, EMPLOYEE\_ID).

## Operador UNION ALL



**O operador UNION ALL retorna resultados das duas consultas, inclusive todos os valores duplicados.**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

### Operador UNION ALL

Use o operador UNION ALL para retornar todas as linhas de várias consultas.

#### Diretrizes

As diretrizes para UNION e UNION ALL são as mesmas, apenas com duas exceções relativas a UNION ALL:

- Ao contrário de UNION, as linhas duplicadas não são eliminadas e, por default, a saída não é classificada.
- A palavra-chave DISTINCT não pode ser usada.

# Usando o Operador UNION ALL

Exiba os departamentos anterior e atual de todos os funcionários.

```
SELECT employee_id, job_id, department_id
FROM employees
UNION ALL
SELECT employee_id, job_id, department_id
FROM job_history
ORDER BY employee_id;
```

EMPLOYEE_ID	JOB_ID	DEPARTMENT_ID
100	AD_PRES	90
101	AD_VP	90
...		
200	AD_ASST	10
200	AD_ASST	90
200	AC_ACCOUNT	90
...		
205	AC_MGR	110
206	AC_ACCOUNT	110

30 rows selected.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Operador UNION ALL (continuação)

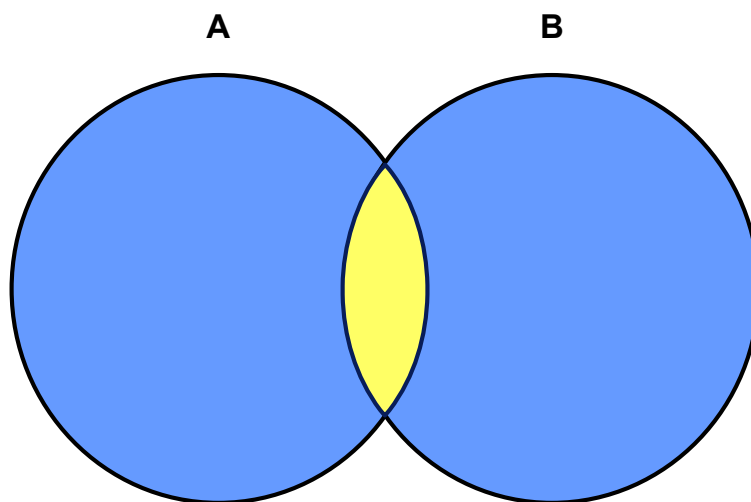
No exemplo, são selecionadas 30 linhas. A combinação das duas tabelas totaliza 30 linhas. O operador UNION ALL não elimina linhas duplicadas. UNION retorna todas as linhas distintas selecionadas pelas consultas. UNION ALL retorna todas as linhas selecionadas pelas consultas, inclusive todas as duplicadas. Considere a consulta do slide, criada agora com a cláusula UNION:

```
SELECT employee_id, job_id, department_id
FROM employees
UNION
SELECT employee_id, job_id, department_id
FROM job_history
ORDER BY employee_id;
```

A consulta anterior retorna 29 linhas. Isso porque ela elimina a seguinte linha (já que ela é duplicada):

EMPLOYEE_ID	JOB_ID	DEPARTMENT_ID
176	SA_REP	80

# Operador INTERSECT



**O operador INTERSECT retorna linhas comuns às duas consultas.**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Operador INTERSECT

Use o operador INTERSECT para retornar todas as linhas comuns a várias consultas.

### Diretrizes

- O número de colunas e os tipos de dados das colunas selecionadas pelas instruções SELECT nas consultas devem ser idênticos em todas as instruções SELECT usadas na consulta. Os nomes das colunas não precisam ser idênticos.
- O resultado não é alterado quando é invertida a ordem das tabelas de interseção.
- INTERSECT não ignora valores nulos.

# Usando o Operador INTERSECT

Exiba os IDs de funcionário e os IDs de cargo dos funcionários que, no momento, estão no mesmo cargo que ocupavam quando foram admitidos pela empresa (ou seja, eles mudaram de cargo, mas agora voltaram para o cargo original).

```
SELECT employee_id, job_id
FROM employees
INTERSECT
SELECT employee_id, job_id
FROM job_history;
```

EMPLOYEE_ID	JOB_ID
176	SA_REP
200	AD_ASST

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Operador INTERSECT (continuação)

No exemplo do slide, a consulta retorna apenas os registros com os mesmos valores nas colunas selecionadas das duas tabelas.

Quais serão os resultados se você adicionar a coluna DEPARTMENT\_ID à instrução SELECT da tabela EMPLOYEES, adicionar a coluna DEPARTMENT\_ID à instrução SELECT da tabela JOB\_HISTORY e executar essa consulta? Os resultados poderão ser diferentes em função da inclusão de outra coluna cujos valores podem ou não ser duplicados.

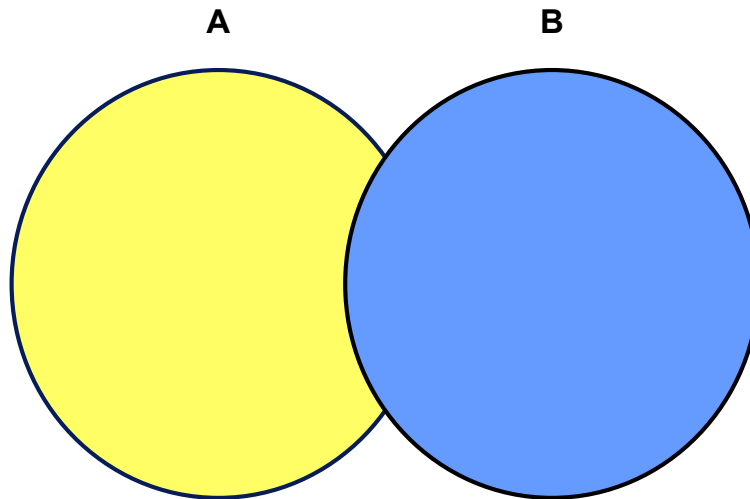
### Exemplo

```
SELECT employee_id, job_id, department_id
FROM employees
INTERSECT
SELECT employee_id, job_id, department_id
FROM job_history;
```

EMPLOYEE_ID	JOB_ID	DEPARTMENT_ID
176	SA_REP	80

O funcionário 200 não está mais incluído nos resultados, pois o valor de EMPLOYEES.DEPARTMENT\_ID é diferente do valor de JOB\_HISTORY.DEPARTMENT\_ID.

# Operador MINUS



**O operador MINUS retorna as linhas da primeira consulta que não estão presentes na segunda consulta.**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Operador MINUS

Use o operador MINUS para exibir as linhas retornadas pela primeira consulta que não estão presentes na segunda consulta (a primeira instrução SELECT subtraída da segunda instrução SELECT).

### Diretrizes

- O número de colunas e os tipos de dados das colunas selecionadas pelas instruções SELECT nas consultas devem ser idênticos em todas as instruções SELECT usadas na consulta. Os nomes das colunas não precisam ser idênticos.
- Todas as colunas da cláusula WHERE devem estar presentes na cláusula SELECT para que o operador MINUS seja executado.

# Operador MINUS

**Exiba os IDs dos funcionários que nunca mudaram de cargo.**

```
SELECT employee_id, job_id
FROM employees
MINUS
SELECT employee_id, job_id
FROM job_history;
```

EMPLOYEE_ID	JOB_ID
100	AD_PRES
101	AD_VP
102	AD_VP
103	IT_PROG
...	
201	MK_MAN
202	MK_REP
205	AC_MGR
206	AC_ACCOUNT

18 rows selected.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Operador MINUS (continuação)

No exemplo do slide, os IDs de funcionário e os IDs de cargo na tabela JOB\_HISTORY são subtraídos dos IDs correspondentes na tabela EMPLOYEES. O conjunto de resultados exibe os funcionários restantes após a subtração; eles são representados pelas linhas existentes na tabela EMPLOYEES, mas que não estão presentes na tabela JOB\_HISTORY. Esses registros são relativos aos funcionários que nunca mudaram de cargo.



## Diretrizes de Operadores de Conjunto

- As expressões das listas **SELECT** devem corresponder em número e tipo de dados.
- É possível usar parênteses para alterar a seqüência de execução.
- A cláusula **ORDER BY**:
  - Só pode aparecer no final da instrução
  - Aceitará o nome da coluna, os apelidos da primeira instrução **SELECT** ou a notação posicional

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Diretrizes de Operadores de Conjunto

- As expressões nas listas de seleção das consultas devem corresponder em número e tipo de dados. As consultas que usam operadores **UNION**, **UNION ALL**, **INTERSECT** e **MINUS** na cláusula **WHERE** devem ter o mesmo número e tipo das colunas da lista **SELECT**. Por exemplo:

```
SELECT employee_id, department_id
FROM employees
WHERE (employee_id, department_id)
      IN (SELECT employee_id, department_id
          FROM employees
          UNION
          SELECT employee_id, department_id
          FROM job_history);
```

- A cláusula **ORDER BY**:
  - Só pode aparecer no final da instrução
  - Aceitará o nome da coluna, um apelido ou a notação posicional
- O nome ou o apelido da coluna, se usado em uma cláusula **ORDER BY**, deverá originar-se da primeira lista **SELECT**.
- É possível usar operadores de conjunto em subconsultas.

# O Servidor Oracle e os Operadores de Conjunto

- **As linhas duplicadas são eliminadas automaticamente, exceto em UNION ALL.**
- **Os nomes das colunas da primeira consulta aparecem no resultado.**
- **Por default, a saída é classificada em ordem crescente, exceto em UNION ALL.**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## O Servidor Oracle e os Operadores de Conjunto

Quando uma consulta usa operadores de conjunto, o servidor Oracle elimina as linhas duplicadas automaticamente, exceto no caso do operador UNION ALL. Os nomes das colunas na saída são decididos pela lista de colunas da primeira instrução SELECT. Por default, a saída é classificada em ordem crescente da primeira coluna da cláusula SELECT.

As expressões correspondentes nas listas de seleção das consultas componentes de uma consulta composta devem coincidir em número e tipo de dados. Se as consultas componentes selecionarem dados de caractere, o tipo de dados dos valores retornados será determinado da seguinte maneira:

- Se as duas consultas selecionarem valores do tipo de dados CHAR, os valores retornados terão esse tipo de dados.
- Se uma ou as duas consultas selecionarem valores do tipo de dados VARCHAR2, os valores retornados terão esse tipo de dados.

## Correspondência entre Instruções SELECT

Com o operador UNION, exiba o ID de departamento, a localização e a data de admissão de todos os funcionários.

```
SELECT department_id, TO_NUMBER(null)
      location, hire_date
FROM   employees
UNION
SELECT department_id, location_id,  TO_DATE(null)
FROM   departments;
```

DEPARTMENT_ID	LOCATION	HIRE_DATE
10	1700	
10		17-SEP-87
20	1800	
20		17-FEB-96
...		
110	1700	
110		07-JUN-94
190	1700	
		24-MAY-99

27 rows selected.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Correspondência entre Instruções SELECT

Como as expressões nas listas de seleção das consultas devem corresponder em número, você pode usar colunas fictícias e as functions de conversão de tipos de dados para seguir essa regra. No slide, a localização do nome é fornecida como o cabeçalho da coluna fictícia. A function TO\_NUMBER é usada na primeira consulta para corresponder ao tipo de dados NUMBER da coluna LOCATION\_ID recuperada pela segunda consulta. Da mesma forma, a function TO\_DATE na segunda consulta é usada para corresponder ao tipo de dados DATE da coluna HIRE\_DATE recuperada pela primeira consulta.

## Correspondência entre Instruções SELECT: Exemplo

Com o operador UNION, exiba o ID de funcionário, o ID do cargo e o salário de todos os funcionários.

```
SELECT employee_id, job_id, salary
FROM   employees
UNION
SELECT employee_id, job_id, 0
FROM   job_history;
```

EMPLOYEE_ID	JOB_ID	SALARY
100	AD_PRES	24000
101	AC_ACCOUNT	0
101	AC_MGR	0
...		
205	AC_MGR	12000
206	AC_ACCOUNT	8300

30 rows selected.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

### Correspondência entre Instruções SELECT: Exemplo

As tabelas EMPLOYEES e JOB\_HISTORY têm várias colunas em comum (por exemplo, EMPLOYEE\_ID, JOB\_ID e DEPARTMENT\_ID). Mas e se você quiser que a consulta exiba o ID de funcionário, o ID de cargo e o salário com o operador UNION, sabendo que o salário existe apenas na tabela EMPLOYEES?

O exemplo de código do slide estabelece a correspondência entre as colunas EMPLOYEE\_ID e JOB\_ID das tabelas EMPLOYEES e JOB\_HISTORY. Um valor literal 0 é adicionado à instrução SELECT de JOB\_HISTORY para corresponder à coluna numérica SALARY na instrução SELECT de EMPLOYEES.

Nos resultados anteriores, cada linha da saída que corresponde a um registro da tabela JOB\_HISTORY contém 0 na coluna SALARY.

# Controlando a Ordem das Linhas

Produza uma frase em inglês usando dois operadores UNION.

```
COLUMN a_dummy NOPRINT
SELECT 'sing' AS "My dream", 3 a_dummy
FROM dual
UNION
SELECT 'I'd like to teach', 1 a_dummy
FROM dual
UNION
SELECT 'the world to', 2 a_dummy
FROM dual
ORDER BY a_dummy;
```

My dream
I'd like to teach
the world to
sing

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

## Controlando a Ordem das Linhas

Por default, a saída é classificada em ordem crescente na primeira coluna. Você pode usar a cláusula ORDER BY para alterar essa classificação.

Só é possível usar a cláusula ORDER BY uma vez em uma consulta composta. Se usada, essa cláusula deverá ser inserida no final da consulta. A cláusula ORDER BY aceita o nome da coluna ou um apelido. Sem a cláusula ORDER BY, o exemplo de código do slide produz a seguinte saída na ordem alfabética da primeira coluna:

My dream
I'd like to teach
sing
the world to

**Observação:** Considere uma consulta composta em que o operador de conjunto UNION é usado mais de uma vez. Nesse caso, a cláusula ORDER BY só pode usar posições em vez de expressões explícitas.

## O Comando iSQL\*Plus COLUMN

É possível usar o comando iSQL\*Plus COLUMN para personalizar cabeçalhos de colunas.

## O Comando *iSQL*\*Plus COLUMN (continuação)

Sintaxe:

```
COL[UMN] [{column|alias} [option]]
```

Em que OPTION é:

CLE [AR] : Remove todos os formatos de coluna

HEA [DING] *texto*: Define o cabeçalho da coluna

FOR [MAT] *formato*: Altera a exibição da coluna usando um modelo de formato

NOPRINT | PRINT: Suprime ou exhibe os dados e os cabeçalhos de colunas

NULL

A instrução a seguir suprime os dados e o cabeçalho da coluna A\_DUMMY. Observe que a primeira cláusula SELECT do slide anterior cria uma coluna fictícia denominada A\_DUMMY.

```
COLUMN a_dummy NOPRINT
```

# Sumário

**Nesta lição, você aprendeu a:**

- **Usar UNION para retornar todas as linhas distintas**
- **Usar UNION ALL para retornar todas as linhas, inclusive as duplicadas**
- **Usar INTERSECT para retornar todas as linhas compartilhadas pelas duas consultas**
- **Usar MINUS para retornar todas as linhas distintas selecionadas pela primeira consulta, mas não pela segunda**
- **Usar ORDER BY somente no final da instrução**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Sumário

- O operador UNION retorna todas as linhas selecionadas pelas consultas. Utilize-o para retornar todas as linhas de várias tabelas e eliminar linhas duplicadas.
- Use o operador UNION ALL para retornar todas as linhas de várias consultas. Ao contrário do que ocorre com o operador UNION, as linhas duplicadas não são eliminadas e, por default, a saída não é classificada.
- Use o operador INTERSECT para retornar todas as linhas comuns a várias consultas.
- Use o operador MINUS para exibir as linhas retornadas pela primeira consulta que não estão presentes na segunda consulta.
- Lembre-se de usar a cláusula ORDER BY somente no final da instrução composta.
- Certifique-se de que as expressões correspondentes nas listas SELECT coincidem em número e tipo de dados.

## Exercício 7: Visão Geral

**Neste exercício, você usará os operadores de conjunto para criar relatórios:**

- Usando o operador **UNION**
- Usando o operador **INTERSECTION**
- Usando o operador **MINUS**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

### Exercício 7: Visão Geral

Neste exercício, você criará consultas com os operadores de conjunto.



## Exercício 7

1. O departamento de recursos humanos precisa de uma lista de IDs dos departamentos que não contêm o ID de cargo ST\_CLERK. Use os operadores de conjunto para criar esse relatório.

DEPARTMENT_ID
10
20
60
80
90
110
190

7 rows selected.

2. O departamento de recursos humanos precisa de uma lista de países nos quais não há departamentos. Exiba o ID e o nome dos países. Use os operadores de conjunto para criar esse relatório.

CO	COUNTRY_NAME
DE	Germany

3. Produza uma lista de cargos dos departamentos 10, 50 e 20, nessa ordem. Exiba o ID de cargo e o ID de departamento usando operadores de conjunto.

JOB_ID	DEPARTMENT_ID
AD_ASST	10
ST_CLERK	50
ST_MAN	50
MK_MAN	20
MK_REP	20

4. Crie um relatório que liste os IDs de funcionário e os IDs de cargo dos funcionários que, no momento, estão no mesmo cargo que ocupavam quando foram admitidos pela empresa (ou seja, eles mudaram de cargo, mas agora voltaram para o cargo original).

EMPLOYEE_ID	JOB_ID
176	SA_REP
200	AD_ASST

## Exercício 7 (continuação)

5. O departamento de recursos humanos precisa de um relatório com as seguintes especificações:
  - Sobrenome e ID do departamento de todos os funcionários da tabela EMPLOYEES, mesmo que não pertençam a um departamento
  - ID e nome de todos os departamentos da tabela DEPARTMENTS, mesmo que não tenham funcionários

Crie uma consulta composta para isso.

LAST_NAME	DEPARTMENT_ID	TO_CHAR(NULL)
Abel	80	
Davies	50	
De Haan	90	
Ernst	60	
Fay	20	
Gietz	110	
Grant		
Hartstein	20	
Higgins	110	
Hunold	60	
King	90	
Kochhar	90	
Lorentz	60	
Matos	50	
LAST_NAME	DEPARTMENT_ID	TO_CHAR(NULL)
Mourgos	50	
Rajs	50	
Taylor	80	
Vargas	50	
Whalen	10	
Zlotkey	80	
	10	Administration
	20	Marketing
	50	Shipping
	60	IT
	80	Sales
	90	Executive
	110	Accounting
	190	Contracting

28 rows selected

# 8

## Manipulando Dados

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

# Objetivos

**Ao concluir esta lição, você será capaz de:**

- **Descrever cada instrução DML (Data Manipulation Language)**
- **Inserir linhas em uma tabela**
- **Atualizar linhas em uma tabela**
- **Deletar linhas de uma tabela**
- **Controlar transações**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Objetivo

Nesta lição, você aprenderá a usar instruções DML para inserir linhas em uma tabela, atualizar as linhas existentes em uma tabela e deletar as linhas existentes de uma tabela. Você também aprenderá a controlar transações com as instruções COMMIT, SAVEPOINT e ROLLBACK.

# Data Manipulation Language

- **Uma instrução DML é executada quando você:**
  - Adiciona novas linhas a uma tabela
  - Modifica as linhas existentes de uma tabela
  - Remove as linhas existentes de uma tabela
- **Uma transação consiste em um conjunto de instruções DML que formam uma unidade lógica de trabalho.**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Data Manipulation Language

A DML (Data Manipulation Language) é uma parte essencial de SQL. Para 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 chamado de *transação*.

Considere um banco de dados de uma instituição bancária. Quando um cliente do banco transfere dinheiro da poupança para a conta corrente, a transação pode consistir em três operações distintas: diminuição da poupança, aumento da conta corrente e registro da transação no diário de transações. O servidor Oracle deve garantir a execução de todas as três instruções SQL para manter as contas com os saldos corretos. Quando algo impedir a execução de uma das instruções da transação, será necessário desfazer as outras.

# Adicionando uma Nova Linha a uma Tabela

## DEPARTMENTS

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
50	Shipping	124	1500
60	IT	103	1400
80	Sales	149	2500
90	Executive	100	1700
110	Accounting	205	1700
190	Contracting		1700

Insira uma nova  
linha na tabela  
DEPARTMENTS

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
50	Shipping	124	1500
60	IT	103	1400
80	Sales	149	2500
90	Executive	100	1700
110	Accounting	205	1700
190	Contracting		1700
70	Public Relations	100	1700

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Adicionando uma Nova Linha a uma Tabela

O slide mostra a adição de um novo departamento à tabela DEPARTMENTS.

# Sintaxe da Instrução INSERT

- Adicione novas linhas a uma tabela usando a instrução INSERT:

```
INSERT INTO  table [(column [, column...])]  
VALUES      (value [, value...]);
```

- Com esta sintaxe, apenas uma linha é inserida por vez.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Adicionando uma Nova Linha a uma Tabela (continuação)

É possível adicionar novas linhas a uma tabela executando a instrução INSERT.

Na sintaxe:

<i>table</i>	é o nome da tabela
<i>column</i>	é o nome da coluna da tabela a ser preenchida
<i>value</i>	é o valor correspondente da coluna

**Observação:** Esta instrução com a cláusula VALUES adiciona somente uma linha por vez a uma tabela.

# Inserindo Novas Linhas

- Insira uma nova linha com valores para cada coluna.
- Liste os valores na ordem default das colunas na tabela.
- Como alternativa, liste as colunas na cláusula INSERT.

```
INSERT INTO departments(department_id,  
                        department_name, manager_id, location_id)  
VALUES (70, 'Public Relations', 100, 1700);  
1 row created.
```

- Delimite os valores de caractere e data com aspas simples.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

## Adicionando uma Nova Linha a uma Tabela (continuação)

Como é possível inserir uma nova linha com valores para cada coluna, a lista de colunas não é necessária na cláusula INSERT. No entanto, se você não usar a lista de colunas, os valores deverão ser listados de acordo com a ordem default das colunas na tabela e um valor deverá ser fornecido para cada coluna.

DESCRIBE departments

Name	Null?	Type
DEPARTMENT_ID	NOT NULL	NUMBER(4)
DEPARTMENT_NAME	NOT NULL	VARCHAR2(30)
MANAGER_ID		NUMBER(6)
LOCATION_ID		NUMBER(4)

Para fins de clareza, use a lista de colunas na cláusula INSERT.

Delimite os valores de caractere e data com aspas simples; não é recomendável delimitar valores numéricos com aspas simples.

Os valores numéricos não devem ser delimitados com aspas simples, já que poderá ocorrer conversão implícita dos valores numéricos designados às colunas com o tipo de dados NUMBER se forem incluídas aspas simples.



# Inserindo Linhas com Valores Nulos

- **Método implícito: omite a coluna da lista de colunas.**

```
INSERT INTO departments (department_id,  
                           department_name )  
VALUES (30, 'Purchasing');  
1 row created.
```

- **Método explícito: especifique a palavra-chave NULL na cláusula VALUES.**

```
INSERT INTO departments  
VALUES (100, 'Finance', NULL, NULL);  
1 row created.
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

## Métodos para Inserir Valores Nulos

Método	Descrição
Implícito	Omita a coluna da lista de colunas.
Explícito	Especifique a palavra-chave NULL na lista VALUES; especifique a string vazia ( ' ' ) na lista VALUES para strings de caracteres e datas.

Certifique-se de que a coluna de destino permita valores nulos verificando o status Null? com o comando DESCRIBE do *iSQL\*Plus*.

O servidor Oracle impõe automaticamente todos os tipos de dados, faixas de dados e constraints de integridade de dados. Todas as colunas não listadas explicitamente obtêm um valor nulo na nova linha.

Erros comuns que podem ocorrer durante a entrada do usuário:

- Valor obrigatório ausente para uma coluna NOT NULL
- Constraint de exclusividade violada por valor duplicado
- Constraint de chave estrangeira violada
- Constraint CHECK violada
- Incompatibilidade de tipo de dados
- Valor muito extenso para caber na coluna

# Inserindo Valores Especiais

A function **SYSDATE** registra a data e o horário atuais.

```
INSERT INTO employees (employee_id,
                        first_name, last_name,
                        email, phone_number,
                        hire_date, job_id, salary,
                        commission_pct, manager_id,
                        department_id)
VALUES (113,
        'Louis', 'Popp',
        'LPOPP', '515.124.4567',
        SYSDATE, 'AC_ACCOUNT', 6900,
        NULL, 205, 100);

1 row created.
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Inserindo Valores Especiais com Functions SQL

É possível usar functions para especificar valores especiais em uma tabela.

O exemplo do slide registra informações sobre o funcionário Popp na tabela **EMPLOYEES**. São fornecidos a data e o horário atuais na coluna **HIRE\_DATE**. A function **SYSDATE** é usada para a data e o horário atuais.

Você também pode usar a function **USER** ao inserir linhas em uma tabela. Essa function registra o nome do usuário atual.

### Confirmando Adições à Tabela

```
SELECT employee_id, last_name, job_id, hire_date, commission_pct
FROM   employees
WHERE  employee_id = 113;
```

EMPLOYEE_ID	LAST_NAME	JOB_ID	HIRE_DATE	COMMISSION_PCT
113	Popp	AC_ACCOUNT	27-SEP-01	

# Inserindo Valores de Data Específicos

- Adicione um novo funcionário.

```
INSERT INTO employees
VALUES      (114,
             'Den', 'Raphealy',
             'DRAPHEAL', '515.127.4561',
             TO_DATE('FEB 3, 1999', 'MON DD, YYYY'),
             'AC_ACCOUNT', 11000, NULL, 100, 30);

1 row created.
```

- Verifique a adição.

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_P
114	Den	Raphealy	DRAPHEAL	515.127.4561	03-FEB-99	AC_ACCOUNT	11000	

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Inserindo Valores Específicos de Data e Horário

O formato DD-MON-YY é normalmente usado para inserir um valor de data. Com esse formato, lembre-se de que o século usado como default é o atual. Como a data também contém informações sobre horário, o horário default é meia-noite (00:00:00).

Se for necessário informar uma data em um formato diferente do default (por exemplo, com outro século ou um horário específico), use a function TO\_DATE.

O exemplo do slide registra informações sobre o funcionário Raphealy na tabela EMPLOYEES. Ele define a coluna HIRE\_DATE como February 3, 1999. Se você usar a instrução a seguir em vez da mostrada no slide, o ano de admissão será interpretado como 2099.

```
INSERT INTO employees
VALUES      (114,
             'Den', 'Raphealy',
             'DRAPHEAL', '515.127.4561',
             '03-FEB-99',
             'AC_ACCOUNT', 11000, NULL, 100, 30);
```

Se o formato RR for usado, o sistema fornecerá o século correto automaticamente, mesmo que não seja o atual.

## Criando um Script

- Use a variável de substituição & em uma instrução SQL para solicitar valores.
- & é um marcador de espaço para o valor da variável.

```
INSERT INTO departments
      (department_id, department_name, location_id)
VALUES (&department_id, '&department_name', &location);
```

Define Substitution Variables

"department\_id"

"department\_name"

"location"

Submit for Execution

Cancel

1 row created.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Criando um Script para Manipular Dados

É possível salvar comandos com variáveis de substituição em um arquivo e executar os comandos no arquivo. O exemplo do slide registra informações de um departamento na tabela DEPARTMENTS.

Ao executar o arquivo de script, você será solicitado a especificar informações para as variáveis de substituição &. Os valores especificados serão aplicados à instrução. Isso permite executar várias vezes o mesmo arquivo de script, mas fornecer um conjunto de valores diferente a cada execução.

# Copiando Linhas de Outra Tabela

- **Crie a instrução INSERT com uma subconsulta:**

```
INSERT INTO sales_reps(id, name, salary, commission_pct)
  SELECT employee_id, last_name, salary, commission_pct
 FROM   employees
 WHERE  job_id LIKE '%REP%';
```

4 rows created.

- **Não use a cláusula VALUES.**
- **Estabeleça uma correspondência entre o número de colunas na cláusula INSERT e o número de colunas na subconsulta.**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

## Copiando Linhas de Outra Tabela

É possível usar a instrução INSERT para adicionar linhas a uma tabela cujos valores são provenientes de tabelas existentes. No lugar da cláusula VALUES, use uma subconsulta.

### Sintaxe

```
INSERT INTO table [ column (, column) ] subquery;
```

Na sintaxe:

*table*                é o nome da tabela  
*column*             é o nome da coluna da tabela a ser preenchida  
*subquery*           é a subconsulta que retorna linhas para a tabela

O número de colunas e os respectivos tipos de dados na lista de colunas da cláusula INSERT devem corresponder ao número de valores e aos respectivos tipos de dados na subconsulta. Para criar uma cópia das linhas de uma tabela, use SELECT \* na subconsulta:

```
INSERT INTO copy_emp
  SELECT *
 FROM   employees;
```


Para obter mais informações, consulte "SELECT" (seção "subqueries") no manual *Oracle Database SQL Reference*.

# Alterando os Dados de uma Tabela

## EMPLOYEES

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	HIRE_DATE	JOB_ID	SALARY	DEPARTMENT_ID	COMMISSION_F
100	Steven	King	SKING	17-JUN-87	AD_PRES	24000	90	
101	Neena	Kochhar	NKOCHHAR	21-SEP-89	AD_VP	17000	90	
102	Lex	De Haan	LDEHAAN	13-JAN-93	AD_VP	17000	90	
103	Alexander	Hunold	AHUNOLD	03-JAN-90	IT_PROG	9000	60	
104	Bruce	Ernst	BERNST	21-MAY-91	IT_PROG	6000	60	
107	Diana	Lorentz	DLORENTZ	07-FEB-99	IT_PROG	4200	60	
124	Kevin	Mourgos	KMOURGOS	16-NOV-99	ST_MAN	5800	50	

Atualize as linhas da tabela **EMPLOYEES**:



EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	HIRE_DATE	JOB_ID	SALARY	DEPARTMENT_ID	COMMISSIO
100	Steven	King	SKING	17-JUN-87	AD_PRES	24000	90	
101	Neena	Kochhar	NKOCHHAR	21-SEP-89	AD_VP	17000	90	
102	Lex	De Haan	LDEHAAN	13-JAN-93	AD_VP	17000	90	
103	Alexander	Hunold	AHUNOLD	03-JAN-90	IT_PROG	9000	30	
104	Bruce	Ernst	BERNST	21-MAY-91	IT_PROG	6000	30	
107	Diana	Lorentz	DLORENTZ	07-FEB-99	IT_PROG	4200	30	
124	Kevin	Mourgos	KMOURGOS	16-NOV-99	ST_MAN	5800	50	

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Alterando os Dados de uma Tabela

O slide mostra a alteração do número de departamento dos funcionários do departamento 60 para o departamento 30.

# Sintaxe da Instrução UPDATE

- **Modifique as linhas existentes com a instrução UPDATE:**

```
UPDATE      table
SET         column = value [, column = value, ...]
[WHERE      condition];
```

- **Atualize mais de uma linha por vez (se necessário).**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Atualizando Linhas

É possível modificar linhas existentes com a instrução UPDATE.

Na sintaxe:

<i>table</i>	é o nome da tabela
<i>column</i>	é o nome da coluna da tabela a ser preenchida
<i>value</i>	é o valor correspondente ou a subconsulta da coluna
<i>condition</i>	identifica as linhas a serem atualizadas e é composta de nomes de colunas, expressões, constantes, subconsultas e operadores de comparação

Para confirmar a operação de atualização, consulte a tabela para exibir as linhas atualizadas.

Para obter mais informações, consulte "UPDATE" no manual *Oracle Database SQL Reference*.

**Observação:** Em geral, use a chave primária para identificar uma única linha. O uso de outras colunas pode resultar na atualização inesperada de várias linhas. Por exemplo, identificar uma única linha da tabela EMPLOYEES por nome é perigoso, pois mais de um funcionário pode ter o mesmo nome.

# Atualizando Linhas em uma Tabela

- Uma ou mais linhas específicas são modificadas quando a cláusula WHERE é especificada:

```
UPDATE employees
SET    department_id = 70
WHERE  employee_id = 113;
1 row updated.
```

- Se você omitir a cláusula WHERE, todas as linhas da tabela serão modificadas:

```
UPDATE  copy_emp
SET      department_id = 110;
22 rows updated.
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

## Atualizando Linhas (continuação)

A instrução UPDATE modifica linhas específicas quando a cláusula WHERE é especificada. O exemplo do slide transfere o funcionário 113 (Popp) para o departamento 70.

Se você omitir a cláusula WHERE, todas as linhas da tabela serão modificadas.

```
SELECT last_name, department_id
FROM   copy_emp;
```

LAST_NAME	DEPARTMENT_ID
King	110
Kochhar	110
De Haan	110
Hunold	110
Ernst	110
Lorentz	110

■■■  
22 rows selected.

**Observação:** A tabela COPY\_EMP tem os mesmos dados que a tabela EMPLOYEES.



# Atualizando Duas Colunas com uma Subconsulta

**Atualize o cargo e o salário do funcionário 114 para que correspondam aos do funcionário 205.**

```
UPDATE employees
SET   job_id = (SELECT job_id
                FROM   employees
                WHERE  employee_id = 205),
      salary = (SELECT salary
                FROM   employees
                WHERE  employee_id = 205)
WHERE employee_id = 114;
1 row updated.
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Atualizando Duas Colunas com uma Subconsulta

É possível atualizar diversas colunas na cláusula SET de uma instrução UPDATE criando várias subconsultas.

### Sintaxe

```
UPDATE table
SET   column =
      (SELECT column
       FROM table
       WHERE condition)
[ ,
  column =
      (SELECT column
       FROM table
       WHERE condition)]
[WHERE condition] ;
```

**Observação:** Se nenhuma linha for atualizada, a mensagem "0 rows updated" será exibida.

## Atualizando Linhas com Base em Outra Tabela

Use subconsultas em instruções UPDATE para atualizar as linhas de uma tabela com base nos valores de outra tabela:

```
UPDATE copy_emp
SET    department_id = (SELECT department_id
                        FROM employees
                        WHERE employee_id = 100)
WHERE  job_id        = (SELECT job_id
                        FROM employees
                        WHERE employee_id = 200);

1 row updated.
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

### Atualizando Linhas com Base em Outra Tabela

É possível usar subconsultas em instruções UPDATE para atualizar as linhas de uma tabela. O exemplo do slide atualiza a tabela COPY\_EMP com base nos valores da tabela EMPLOYEES. Ele altera o número do departamento de todos os funcionários com o ID de cargo do funcionário 200 para o número do departamento atual do funcionário 100.

# Removendo uma Linha de uma Tabela

## DEPARTMENTS

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
30	Purchasing		
100	Finance		
50	Shipping	124	1500
60	IT	103	1400

## Delete uma linha da tabela DEPARTMENTS:

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
30	Purchasing		
50	Shipping	124	1500
60	IT	103	1400

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Removendo uma Linha de uma Tabela

O exemplo do slide remove o departamento Finance da tabela DEPARTMENTS (pressupondo que não haja constraints definidas nessa tabela).

# Instrução DELETE

É possível remover as linhas existentes de uma tabela com a instrução DELETE:

```
DELETE [FROM]    table
[WHERE           condition] ;
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Deletando Linhas

É possível remover linhas existentes com a instrução DELETE.

Na sintaxe:

*table*            é o nome da tabela  
*condition*      identifica as linhas a serem deletadas e é composta de nomes de colunas, expressões, constantes, subconsultas e operadores de comparação

**Observação:** Se nenhuma linha for deletada, a mensagem "0 rows deleted" será exibida.

Para obter mais informações, consulte "DELETE" no manual *Oracle Database SQL Reference*.

# Deletando Linhas de uma Tabela

- Se você usar a cláusula WHERE, as linhas específicas serão deletadas:

```
DELETE FROM departments
WHERE department_name = 'Finance';
1 row deleted.
```

- Se você omitir a cláusula WHERE, todas as linhas da tabela serão deletadas:

```
DELETE FROM copy_emp;
22 rows deleted.
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

## Deletando Linhas (continuação)

É possível deletar linhas específicas usando a cláusula WHERE na instrução DELETE. O exemplo do slide deleta o departamento Finance da tabela DEPARTMENTS. Para confirmar a operação de deleção, exiba as linhas deletadas com a instrução SELECT.

```
SELECT *
FROM departments
WHERE department_name = 'Finance';
no rows selected.
```

Se você omitir a cláusula WHERE, todas as linhas da tabela serão deletadas. O segundo exemplo do slide deleta todas as linhas da tabela COPY\_EMP porque nenhuma cláusula WHERE foi especificada.

## Exemplo

Remova as linhas identificadas na cláusula WHERE.

```
DELETE FROM employees WHERE employee_id = 114;
1 row deleted.
```

```
DELETE FROM departments WHERE department_id IN (30, 40);
2 rows deleted.
```

## Deletando Linhas com Base em Outra Tabela

Use subconsultas em instruções **DELETE** para remover linhas de uma tabela com base nos valores de outra tabela:

```
DELETE FROM employees
WHERE department_id =
    (SELECT department_id
     FROM departments
     WHERE department_name
       LIKE '%Public%');

1 row deleted.
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

### Deletando Linhas com Base em Outra Tabela

É possível usar subconsultas para deletar linhas de uma tabela com base nos valores de outra tabela. O exemplo do slide deleta todos os funcionários que trabalham em um departamento cujo nome contém a string `Public`. A subconsulta pesquisa a tabela `DEPARTMENTS` para localizar o número do departamento com base no nome do departamento que contém essa string. Em seguida, a subconsulta informa o número do departamento para a consulta principal, que deleta as linhas de dados da tabela `EMPLOYEES` com base nesse número de departamento.

# Instrução TRUNCATE

- Remove todas as linhas de uma tabela, esvaziando a tabela e mantendo a estrutura intacta
- É uma instrução DDL (Data Definition Language), e não DML; não pode ser desfeita facilmente
- Sintaxe:

```
TRUNCATE TABLE table_name;
```

- Exemplo:

```
TRUNCATE TABLE copy_emp;
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Instrução TRUNCATE

Um método mais eficiente de esvaziar uma tabela é a utilização da instrução TRUNCATE. É possível usar essa instrução para remover rapidamente todas as linhas de uma tabela ou cluster. A remoção de linhas com a instrução TRUNCATE é mais rápida do que com a instrução DELETE pelos seguintes motivos:

- A instrução TRUNCATE é uma instrução DDL (Data Definition Language) e não gera informações de rollback. As informações de rollback são abordadas posteriormente nesta lição.
- Truncar uma tabela não dispara os triggers de deleção dessa tabela.
- Se a tabela for mãe de uma constraint de integridade referencial, você não poderá truncá-la. É necessário desativar a constraint antes de executar a instrução TRUNCATE. A desativação de constraints é abordada em uma lição posterior.

## Usando uma Subconsulta em uma Instrução INSERT

```
INSERT INTO
    (SELECT employee_id, last_name,
             email, hire_date, job_id, salary,
             department_id
     FROM   employees
     WHERE  department_id = 50)
VALUES (99999, 'Taylor', 'DTAYLOR',
        TO_DATE('07-JUN-99', 'DD-MON-RR'),
        'ST_CLERK', 5000, 50);

1 row created.
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

### Usando uma Subconsulta em uma Instrução INSERT

É possível usar uma subconsulta no lugar do nome da tabela na cláusula INTO da instrução INSERT.

A lista de seleção da subconsulta deve ter o mesmo número de colunas que a lista de colunas da cláusula VALUES. Para a execução bem-sucedida da instrução INSERT, todas as regras nas colunas da tabela-base devem ser cumpridas. Por exemplo, não é possível especificar um ID de funcionário duplicado ou omitir um valor de uma coluna não nula obrigatória.



# Usando uma Subconsulta em uma Instrução INSERT

Verifique os resultados:

```
SELECT employee_id, last_name, email, hire_date,  
       job_id, salary, department_id  
FROM   employees  
WHERE  department_id = 50;
```

EMPLOYEE_ID	LAST_NAME	EMAIL	HIRE_DATE	JOB_ID	SALARY	DEPARTMENT_ID
124	Mourgos	KMOURGOS	16-NOV-99	ST_MAN	5800	50
141	Rajs	TRAJS	17-OCT-95	ST_CLERK	3500	50
142	Davies	CDAVIES	29-JAN-97	ST_CLERK	3100	50
143	Matos	RMATOS	15-MAR-98	ST_CLERK	2600	50
144	Vargas	PVARGAS	09-JUL-98	ST_CLERK	2500	50
99999	Taylor	DTAYLOR	07-JUN-99	ST_CLERK	5000	50

6 rows selected.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Usando uma Subconsulta em uma Instrução INSERT (continuação)

O exemplo mostra os resultados da subconsulta usada a fim de identificar a tabela para a instrução INSERT.

# Transações de Banco de Dados

**Uma transação de banco de dados consiste em uma das seguintes instruções:**

- **Instruções DML que constituem uma alteração consistente nos dados**
- **Uma instrução DDL**
- **Uma instrução DCL (Data Control Language)**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

## Transações de Banco de Dados

O servidor Oracle garante a consistência de dados com base em transações. As transações permitem mais flexibilidade e controle durante a alteração de dados e garantem a consistência de dados em caso de falha de processo do usuário ou falha do sistema.

As transações consistem em instruções DML que formam uma alteração consistente dos dados. Por exemplo, uma transferência de fundos entre duas contas deve incluir o débito em uma conta e o crédito em outra conta no mesmo valor. As duas ações deverão apresentar falha ou ser bem-sucedidas; o crédito não deverá ser submetido a commit sem o débito.

## Tipos de Transação

Tipo	Descrição
DML (Data Manipulation Language)	Consiste em qualquer número de instruções DML que o servidor Oracle trata como uma entidade única ou uma unidade de trabalho lógica
(Data Definition Language)	Consiste em uma única instrução DDL
DCL (Data Control Language)	Consiste em uma única instrução DCL

# Transações de Banco de Dados

- **Começam quando a primeira instrução SQL DML é executada**
- **Terminam com um destes eventos:**
  - Uma instrução COMMIT ou ROLLBACK é executada.
  - Uma instrução DDL ou DCL é executada (commit automático).
  - O usuário sai do iSQL\*Plus.
  - Ocorre uma falha do sistema.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Quando uma Transação Começa e Termina?

Uma transação começa quando a primeira instrução DML é encontrada e termina quando uma destas ações ocorre:

- Uma instrução COMMIT ou ROLLBACK é executada.
- Uma instrução DDL, como CREATE, é executada.
- Uma instrução DCL é executada.
- O usuário sai do iSQL\*Plus.
- Ocorre uma falha de máquina ou do sistema.

Após o término de uma transação, a próxima instrução SQL executável inicia automaticamente a transação seguinte.

Uma instrução DDL ou DCL é submetida a commit automaticamente e, portanto, encerra uma transação de forma implícita.

# Vantagens de Instruções COMMIT e ROLLBACK

**Com as instruções COMMIT e ROLLBACK, é possível:**

- **Garantir a consistência de dados**
- **Visualizar alterações de dados antes de torná-las permanentes**
- **Agrupar operações relacionadas logicamente**

ORACLE

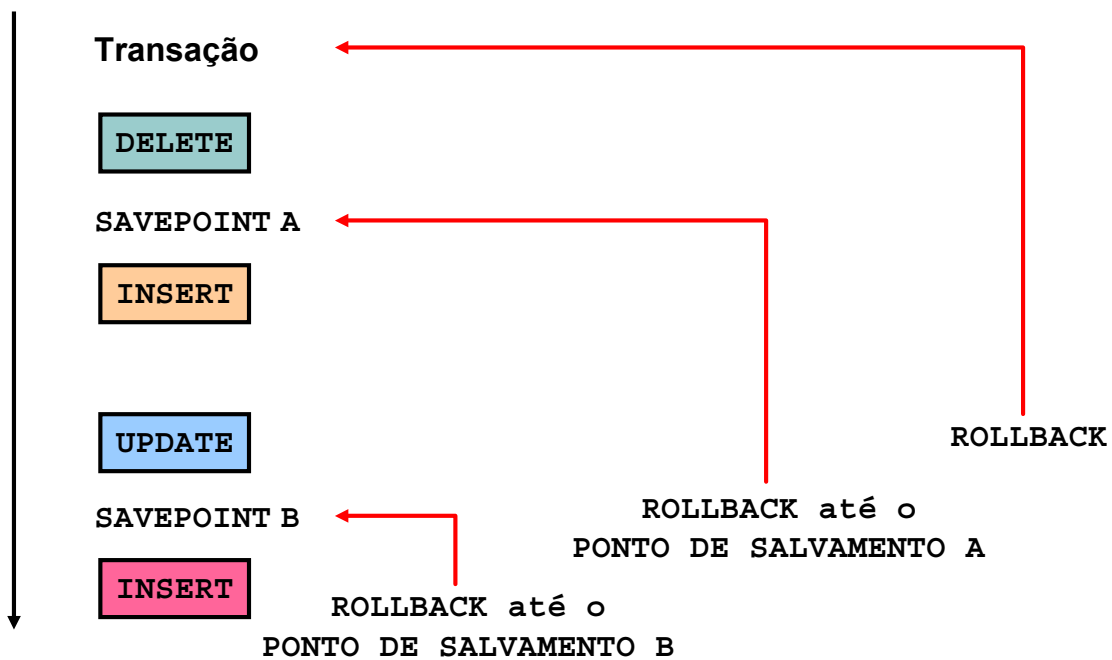
Copyright © 2004, Oracle. Todos os direitos reservados.

## Vantagens de COMMIT e ROLLBACK

As instruções COMMIT e ROLLBACK permitem controlar as alterações permanentes nos dados.

# Controlando Transações

Horário COMMIT



ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Instruções de Controle de Transação Explícita

É possível controlar a lógica de transações com as instruções COMMIT, SAVEPOINT e ROLLBACK.

Instrução	Descrição
COMMIT	Termina a transação atual, tornando permanentes todas as alterações de dados pendentes
SAVEPOINT <i>name</i>	Marca um ponto de salvamento na transação atual
ROLLBACK	ROLLBACK termina a transação atual e descarta todas as alterações de dados pendentes.
ROLLBACK TO SAVEPOINT <i>name</i>	ROLLBACK TO SAVEPOINT efetua rollback da transação atual para o ponto de salvamento especificado e, como consequência, descarta todas as alterações e/ou pontos de salvamento criados após o ponto de salvamento do rollback atual. Se você omitir a cláusula TO SAVEPOINT, a instrução ROLLBACK efetuará rollback de toda a transação. Como os pontos de salvamento são lógicos, não há como listar os pontos de salvamento criados.

**Observação:** SAVEPOINT não é uma instrução SQL que segue o padrão ANSI.

## Fazendo Rollback de Alterações até um Marcador

- Crie um marcador em uma transação atual usando a instrução **SAVEPOINT**.
- Faça rollback até esse marcador usando a instrução **ROLLBACK TO SAVEPOINT**.

```
UPDATE...  
SAVEPOINT update_done;  
Savepoint created.  
INSERT...  
ROLLBACK TO update_done;  
Rollback complete.
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

### Fazendo Rollback de Alterações até um Marcador

É possível criar um marcador na transação atual usando a instrução **SAVEPOINT**, que divide a transação em seções menores. Depois, é possível descartar as alterações pendentes até esse marcador usando a instrução **ROLLBACK TO SAVEPOINT**.

Se você criar um segundo ponto de salvamento com o mesmo nome do ponto de salvamento anterior, o ponto de salvamento anterior será deletado.

## Processamento de Transação Implícita

- Um commit automático ocorre nas seguintes circunstâncias:
  - Uma instrução DDL é executada
  - Uma instrução DCL é executada
  - Saída normal do *iSQL\*Plus*, sem a execução implícita de instruções COMMIT ou ROLLBACK
- Um rollback automático ocorre em decorrência do encerramento anormal do *iSQL\*Plus* ou de uma falha do sistema.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

### Processamento de Transação Implícita

Status	Circunstâncias
Automatic commit	A instrução DDL ou DCL é executada. O <i>iSQL*Plus</i> foi encerrado normalmente, sem a execução explícita dos comandos COMMIT ou ROLLBACK.
Automatic rollback	Encerramento anormal do <i>iSQL*Plus</i> ou falha do sistema.

**Observação:** Um terceiro comando está disponível no *iSQL\*Plus*. É possível alternar o comando AUTOCOMMIT entre os estados ativado e desativado. Se ele for *ativado*, cada instrução DML será submetida a commit logo após sua execução. Não é possível fazer rollback das alterações. Se ele for *desativado*, a instrução COMMIT ainda poderá ser executada explicitamente. Além disso, a instrução COMMIT será executada quando uma instrução DDL for executada ou quando você sair do *iSQL\*Plus*.

## Processamento de Transação Implícita (continuação)

### Falhas do Sistema

Quando uma transação for interrompida por uma falha do sistema, será feito rollback automaticamente de toda a transação. Isso impede que o erro cause alterações indesejadas nos dados e retorna as tabelas para o estado em que se encontravam no momento do último commit. Dessa forma, o servidor Oracle protege a integridade das tabelas.

No *iSQL\*Plus*, para sair normalmente da sessão, você deve clicar no botão Exit. No *SQL\*Plus*, para executar essa mesma ação, você deve digitar o comando `EXIT` no prompt. O fechamento da janela é interpretado como uma saída anormal.



## Estado dos Dados antes de COMMIT ou ROLLBACK

- É possível recuperar o estado anterior dos dados.
- O usuário atual pode verificar os resultados das operações DML usando a instrução `SELECT`.
- Outros usuários *não podem* visualizar os resultados das instruções DML executadas pelo usuário atual.
- As linhas afetadas são *bloqueadas*; outros usuários não podem alterar os dados nessas linhas.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

### Submetendo Alterações a Commit

Todas as alterações de dados feitas durante a transação serão temporárias até que ela seja submetida a commit.

O estado dos dados antes da execução da instrução `COMMIT` ou `ROLLBACK` pode ser descrito da seguinte forma:

- As operações de manipulação de dados afetam principalmente o buffer de banco de dados; portanto, é possível recuperar o estado anterior dos dados.
- O usuário atual pode visualizar os resultados das operações de manipulação de dados consultando as tabelas.
- Outros usuários não podem visualizar os resultados das operações de manipulação de dados executadas pelo usuário atual. O servidor Oracle institui a consistência de leitura para garantir que cada usuário veja os dados da forma como se encontravam no último commit.
- As linhas afetadas são bloqueadas; outros usuários não podem alterar os dados nessas linhas.

## Estado dos Dados após COMMIT

- **As alterações de dados tornam-se permanentes no banco de dados.**
- **O estado anterior dos dados é perdido permanentemente.**
- **Todos os usuários podem visualizar os resultados.**
- **Os bloqueios nas linhas afetadas são liberados; essas linhas estão disponíveis para manipulação por outros usuários.**
- **Todos os pontos de salvamento são apagados.**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

### Submetendo Alterações a Commit (continuação)

Use a instrução COMMIT para tornar permanentes todas as alterações pendentes. Veja o que acontece após uma instrução COMMIT:

- As alterações de dados são gravadas no banco de dados.
- O estado anterior dos dados não está mais disponível nas consultas SQL comuns.
- Todos os usuários podem visualizar os resultados da transação.
- Os bloqueios nas linhas afetadas são liberados; agora, as linhas estão disponíveis para que outros usuários efetuem novas alterações de dados.
- Todos os pontos de salvamento são apagados.

# Submetendo Dados a Commit

- Efetue as alterações:

```
DELETE FROM employees
WHERE employee_id = 99999;
1 row deleted.

INSERT INTO departments
VALUES (290, 'Corporate Tax', NULL, 1700);
1 row created.
```

- Submeta as alterações a commit:

```
COMMIT;
Commit complete.
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

## Submetendo Alterações a Commit (continuação)

O exemplo do slide deleta uma linha da tabela EMPLOYEES e insere uma nova linha na tabela DEPARTMENTS. Em seguida, ele torna a alteração permanente executando a instrução COMMIT.

### Exemplo

Remova os departamentos 290 e 300 da tabela DEPARTMENTS e atualize uma linha da tabela COPY\_EMP. Torne a alteração de dados permanente.

```
DELETE FROM departments
WHERE department_id IN (290, 300);
1 row deleted.
```

```
UPDATE employees
SET department_id = 80
WHERE employee_id = 206;
1 row updated.
```

```
COMMIT;
Commit Complete.
```

# Estado dos Dados após ROLLBACK

**Descarte todas as alterações pendentes com a instrução ROLLBACK:**

- **As alterações de dados são desfeitas.**
- **O estado anterior dos dados é restaurado.**
- **Os bloqueios nas linhas afetadas são liberados.**

```
DELETE FROM copy_emp;  
22 rows deleted.  
ROLLBACK ;  
Rollback complete.
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Fazendo Rollback de Alterações

Descarte todas as alterações pendentes com a instrução ROLLBACK. O resultado será:

- As alterações de dados são desfeitas.
- O estado anterior dos dados é restaurado.
- Os bloqueios nas linhas afetadas são liberados.

## Estado dos Dados após ROLLBACK

```
DELETE FROM test;  
25,000 rows deleted.  
  
ROLLBACK;  
Rollback complete.  
  
DELETE FROM test WHERE id = 100;  
1 row deleted.  
  
SELECT * FROM test WHERE id = 100;  
No rows selected.  
  
COMMIT;  
Commit complete.
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

### Exemplo

Ao tentar remover um registro da tabela TEST, você poderá esvaziar a tabela acidentalmente. É possível corrigir o erro, reexecutar a instrução correta e tornar a alteração de dados permanente.

## Rollback no Nível de Instrução

- Se houver falha de uma única instrução DML durante a execução, será feito rollback somente dessa instrução.
- O servidor Oracle implementa um ponto de salvamento implícito.
- Todas as outras alterações são retidas.
- O usuário deve encerrar as transações explicitamente executando uma instrução COMMIT ou ROLLBACK.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

### Rollback no Nível de Instrução

Parte de uma transação poderá ser descartada por um rollback implícito se for detectado um erro de execução de instrução. Se houver falha de uma única instrução DML durante a execução de uma transação, seu efeito será anulado por um rollback no nível de instrução, mas as alterações feitas pelas instruções DML anteriores na transação não serão descartadas. Elas poderão ser submetidas a commit ou rollback explicitamente pelo usuário.

O servidor Oracle executa um commit implícito antes e depois de qualquer instrução DDL. Portanto, mesmo que a instrução DDL não seja executada com êxito, você não poderá fazer rollback da instrução anterior porque o servidor executou um commit.

Encerre as transações explicitamente executando uma instrução COMMIT ou ROLLBACK.

# Consistência de Leitura

- **A consistência de leitura garante uma view consistente dos dados em todos os momentos.**
- **As alterações feitas por um usuário não entram em conflito com as alterações feitas por outro usuário.**
- **A consistência de leitura garante que nos mesmos dados:**
  - **Os usuários que efetuam operações de leitura não precisem aguardar os usuários que efetuam operações de gravação**
  - **Os usuários que efetuam operações de gravação não precisem aguardar os usuários que efetuam operações de leitura**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Consistência de Leitura

Os usuários acessam o banco de dados de duas maneiras:

- Operações de leitura (instrução `SELECT`)
- Operações de gravação (instruções `INSERT`, `UPDATE`, `DELETE`)

A consistência de leitura é necessária para permitir que:

- Os usuários que efetuam operações de leitura e gravação no banco de dados tenham uma view consistente dos dados.
- Os usuários que efetuam operações de leitura não vejam dados que estão sendo alterados.
- Os usuários que efetuam operações de gravação tenham certeza de que as alterações no banco de dados são feitas de maneira consistente.
- As alterações feitas por usuários que efetuam operações de gravação não interrompam umas às outras nem sejam conflitantes.

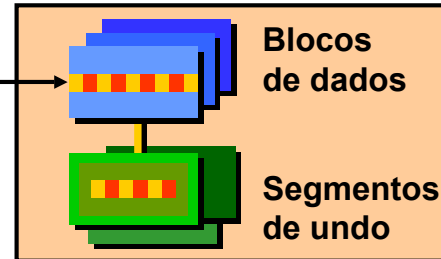
O objetivo da consistência de leitura é garantir que cada usuário veja os dados no estado em que se encontravam no momento do último commit, antes de ser iniciada uma operação DML.

# Implementação da Consistência de Leitura

## Usuário A

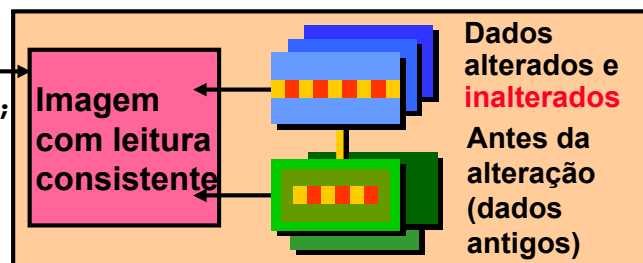


```
UPDATE employees  
SET salary = 7000  
WHERE last_name = 'Goyal';
```



## Usuário B

```
SELECT *  
FROM userA.employees;
```



ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Implementação da Consistência de Leitura

A implementação da consistência de leitura é automática. Ela mantém uma cópia parcial do banco de dados em segmentos de undo. A imagem com leitura consistente é criada a partir dos dados submetidos a commit da tabela e dos dados antigos que estão sendo alterados e ainda não foram submetidos a commit do segmento de undo.

Quando ocorre uma operação de inserção, atualização ou deleção no banco de dados, o servidor Oracle obtém uma cópia dos dados antes da sua alteração e grava essa cópia em um *segmento de undo*.

Todos os usuários que estão efetuando uma operação de leitura, exceto o usuário que executou a alteração, ainda vêem o banco de dados no estado em que se encontrava antes do início das alterações; eles vêem um snapshot dos dados no segmento de undo.

Antes do commit das alterações no banco de dados, somente o usuário que está modificando os dados vê o banco de dados com as alterações. Todos os outros usuários vêem o snapshot no segmento de undo. Isso garante que os usuários leiam dados consistentes que não estão sendo alterados no momento.

Quando uma instrução DML for submetida a commit, a alteração feita no banco de dados se tornará visível a todos os usuários que executarem uma instrução select *após* o commit. O espaço ocupado pelos dados *antigos* no arquivo de segmento de undo estará liberado para reutilização.

Se a transação for submetida a rollback, as alterações serão desfeitas:

- A versão mais antiga original dos dados no segmento de undo será gravada novamente na tabela.
- Todos os usuários verão o banco de dados no estado em que se encontrava antes do início da transação.



# Sumário

**Nesta lição, você aprendeu a usar as seguintes instruções:**

Function	Descrição
INSERT	Adiciona uma nova linha à tabela
UPDATE	Modifica as linhas existentes da tabela
DELETE	Remove as linhas existentes da tabela
COMMIT	Torna permanentes todas as alterações pendentes
SAVEPOINT	Usada para efetuar rollback até o marcador de ponto de salvamento
ROLLBACK	Descarta todas as alterações de dados pendentes

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

## Sumário

Nesta lição, você aprendeu a manipular dados no banco de dados Oracle com as instruções INSERT, UPDATE e DELETE, bem como controlar alterações nos dados com as instruções COMMIT, SAVEPOINT e ROLLBACK.

O servidor Oracle garante uma view consistente dos dados em todos os momentos.

## Exercício 8: Visão Geral

**Este exercício aborda os seguintes tópicos:**

- **Inserção de linhas nas tabelas**
- **Atualização e deleção de linhas na tabela**
- **Controle de transações**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

### Exercício 8: Visão Geral

Neste exercício, você adicionará linhas à tabela MY\_EMPLOYEE, atualizará e deletará dados na tabela e controlará as transações.

## Exercício 8

O departamento de recursos humanos deseja criar instruções SQL para inserir, atualizar e deletar dados de funcionários. Como protótipo, use a tabela MY\_EMPLOYEE antes de fornecer as instruções ao departamento de recursos humanos.

Insira dados na tabela MY\_EMPLOYEE.

1. Execute a instrução no script lab\_08\_01.sql para criar a tabela MY\_EMPLOYEE a ser usada no exercício.
2. Descreva a estrutura da tabela MY\_EMPLOYEE para identificar os nomes de colunas.

Name	Null?	Type
ID	NOT NULL	NUMBER(4)
LAST_NAME		VARCHAR2(25)
FIRST_NAME		VARCHAR2(25)
USERID		VARCHAR2(8)
SALARY		NUMBER(9,2)

3. Crie uma instrução INSERT para *adicionar a primeira linha* de dados à tabela MY\_EMPLOYEE usando estes dados de amostra. Não liste as colunas na cláusula INSERT. *Não informe todas as linhas ainda.*

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	Patel	Ralph	rpatel	895
2	Dancs	Betty	bdancs	860
3	Biri	Ben	bbiri	1100
4	Newman	Chad	cnewman	750
5	Ropeburn	Audrey	aropebur	1550

4. Preencha a tabela MY\_EMPLOYEE com a segunda linha de dados de amostra da lista anterior. Desta vez, liste as colunas explicitamente na cláusula INSERT.
5. Confirme a adição à tabela.

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	Patel	Ralph	rpatel	895
2	Dancs	Betty	bdancs	860

## Exercício 8 (continuação)

6. Crie uma instrução insert no arquivo de script dinâmico reutilizável `loademp.sql` para carregar linhas na tabela `MY_EMPLOYEE`. Concatene a primeira letra do primeiro nome e os sete primeiros caracteres do sobrenome para gerar o ID do usuário. Salve esse script no arquivo `lab_08_06.sql`.
7. Para preencher a tabela com as próximas duas linhas dos dados de amostra, execute a instrução insert no script criado.
8. Confirme as adições à tabela.

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	Patel	Ralph	rpatel	895
2	Dancs	Betty	bdancs	860
3	Biri	Ben	bbiri	1100
4	Newman	Chad	cnewman	750

9. Torne as adições de dados permanentes.

Atualize e delete dados na tabela `MY_EMPLOYEE`.

10. Altere o sobrenome do funcionário 3 para Drexler.
11. Altere o salário de todos os funcionários com salário inferior a US\$ 900 para US\$ 1.000.
12. Verifique as alterações na tabela.

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	Patel	Ralph	rpatel	1000
2	Dancs	Betty	bdancs	1000
3	Drexler	Ben	bbiri	1100
4	Newman	Chad	cnewman	1000

13. Delete Betty Dancs da tabela `MY_EMPLOYEE`.
14. Confirme as alterações na tabela.

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	Patel	Ralph	rpatel	1000
3	Drexler	Ben	bbiri	1100
4	Newman	Chad	cnewman	1000

## Exercício 8 (continuação)

15. Submeta todas as alterações pendentes a commit.

Controle a transação de dados na tabela MY\_EMPLOYEE.

16. Preencha a tabela com a última linha dos dados de amostra usando as instruções no script criado na etapa 6. Execute as instruções no script.

17. Confirme a adição à tabela.

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	Patel	Ralph	rpatel	1000
3	Drexler	Ben	bbiri	1100
4	Newman	Chad	cnewman	1000
5	Ropeburn	Audrey	aropebur	1550

18. Marque um ponto intermediário no processamento da transação.

19. Esvazie a tabela inteira.

20. Confirme se a tabela está vazia.

21. Descarte a operação DELETE mais recente sem descartar a operação INSERT anterior.

22. Confirme se a nova linha permanece intacta.

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	Patel	Ralph	rpatel	1000
3	Drexler	Ben	bbiri	1100
4	Newman	Chad	cnewman	1000
5	Ropeburn	Audrey	aropebur	1550

23. Torne a adição de dados permanente.

