

Banco de Dados Oracle 10g: Fundamentos de SQL II

Guia do Aluno • Volume 1

D17111BP10

Produção 1.0

Junho 2004

D39569

ORACLE®

Autor

Priya Vennapusa

Revisores e**Colaboradores Técnicos**

Nancy Greenberg

Priya Nathan

Andrew Brannigan

Angelika Krupp

Brian Boxx

Christopher Lawless

Joel Goodman

Malika Marghadi

Marjolein Dekkers

Stefan Grenstad

Zarko Cesljas

Rosita Hanoman

Ruediger Steffan

Editor

Joseph Fernandez

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

I Introdução

- Objetivos 1-2
- Objetivos do Curso 1-3
- Visão Geral do Curso 1-4
- Sumário 1-6

1 Controlando o Acesso dos Usuários

- Objetivos 1-2
- Controlando o Acesso dos Usuários 1-3
- Privilégios 1-4
- Privilégios de Sistema 1-5
- Criando Usuários 1-6
- Privilégios de Sistema do Usuário 1-7
- Concedendo Privilégios de Sistema 1-8
- O Que É uma Atribuição? 1-9
- Criando e Concedendo Privilégios a uma Atribuição 1-10
- Alterando a Senha 1-11
- Privilégios de Objeto 1-12
- Concedendo Privilégios de Objeto 1-14
- Passando Privilégios 1-15
- Confirmando Privilégios Concedidos com Grant 1-16
- Revogando Privilégios de Objeto 1-17
- Sumário 1-19
- Exercício 1: Visão Geral 1-20

2 Gerenciar Objetos de Esquema

- Objetivos 2-2
- A Instrução ALTER TABLE 2-3
- Adicionando uma Coluna 2-5
- Modificando uma Coluna 2-6
- Eliminando uma Coluna 2-7
- A Opção SET UNUSED 2-8
- Adicionando uma Sintaxe de Constraint 2-10
- Adicionando uma Constraint 2-11
- ON DELETE CASCADE 2-12
- Adiando Constraints 2-13
- Eliminando uma Constraint 2-14
- Desativando Constraints 2-15
- Ativando Constraints 2-16
- Constraints em Cascata 2-18
- Visão Geral de Índices 2-20

CREATE INDEX com a Instrução CREATE TABLE 2-21
 Índices Baseados em Função 2-23
 Removendo um Índice 2-25
 DROP TABLE ...PURGE 2-26
 A Instrução FLASHBACK TABLE 2-27
 Tabelas Externas 2-29
 Criando um Diretório para a Tabela Externa 2-31
 Criando uma Tabela Externa 2-33
 Criando uma Tabela Externa Usando ORACLE_LOADER 2-35
 Consultando Tabelas Externas 2-37
 Sumário 2-38
 Exercício 2: Visão Geral 2-39

3 Manipulando Grandes Conjuntos de Dados

Objetivos 3-2
 Usando Subconsultas para Manipular Dados 3-3
 Copiando Linhas de Outra Tabela 3-4
 Inserção Usando uma Subconsulta como Destino 3-5
 Recuperando Dados com uma Subconsulta como Origem 3-7
 Atualizando Duas Colunas com uma Subconsulta 3-8
 Atualizando Linhas com Base em Outra Tabela 3-9
 Deletando Linhas com Base em Outra Tabela 3-10
 Usando a Palavra-Chave WITH CHECK OPTION em Instruções DML 3-11
 Visão Geral do Recurso de Default Explícito 3-12
 Usando Valores de Default Explícitos 3-13
 Visão Geral de Instruções INSERT em Várias Tabelas 3-14
 Tipos de Instruções INSERT em Várias Tabelas 3-16
 Instruções INSERT em Várias Tabelas 3-17
 INSERT ALL Incondicional 3-19
 INSERT ALL Condicional 3-20
 FIRST INSERT Condicional 3-22
 INSERT de Criação de Pivô 3-24
 A Instrução MERGE 3-27
 A Sintaxe da Instrução MERGE 3-28
 Intercalando Linhas 3-29
 Controlando Alterações nos Dados 3-31
 Exemplo de Flashback de Consulta de Versão 3-32
 A Cláusula VERSIONS BETWEEN 3-34
 Sumário 3-35
 Exercício 3: Visão Geral 3-36

4 Gerando Relatórios por Agrupamento de Dados Relacionados

Objetivos 4-2
 Análise de Functions de Grupo 4-3

Análise da Cláusula GROUP BY 4-4
 Análise da Cláusula HAVING 4-5
 GROUP BY com Operadores ROLLUP e CUBE 4-6
 Operador ROLLUP 4-7
 Operador ROLLUP: Exemplo 4-8
 Operador CUBE 4-9
 Operador CUBE: Exemplo 4-10
 Function GROUPING 4-11
 Function GROUPING: Exemplo 4-12
 GROUPING SETS 4-13
 GROUPING SETS: Exemplo 4-15
 Colunas Compostas 4-17
 Colunas Compostas: Exemplo 4-19
 Agrupamentos Concatenados 4-21
 Agrupamentos Concatenados: Exemplo 4-22
 Sumário 4-23
 Exercício 4: Visão Geral 4-24

5 Gerenciando Dados em Diferentes Fusos Horários

Objetivos 5-2
 Fusos Horários 5-3
 Parâmetro de Sessão TIME_ZONE 5-4
 CURRENT_DATE, CURRENT_TIMESTAMP e LOCALTIMESTAMP 5-5
 CURRENT_DATE 5-6
 CURRENT_TIMESTAMP 5-7
 LOCALTIMESTAMP 5-8
 DBTIMEZONE e SESSIONTIMEZONE 5-9
 Tipo de Dados TIMESTAMP 5-10
 Tipos de Dados TIMESTAMP 5-11
 Campos TIMESTAMP 5-12
 Diferença entre DATE e TIMESTAMP 5-13
 Tipo de Dados TIMESTAMP WITH TIMEZONE 5-14
 TIMESTAMP WITH TIMEZONE: Exemplo 5-15
 TIMESTAMP WITH LOCAL TIMEZONE 5-16
 TIMESTAMP WITH LOCAL TIMEZONE: Exemplo 5-17
 Tipos de Dados INTERVAL 5-18
 Campos INTERVAL 5-20
 Tipo de Dados INTERVAL YEAR TO MONTH 5-21
 INTERVAL YEAR TO MONTH Exemplo 5-22
 Tipo de Dados INTERVAL DAY TO SECOND 5-23
 Tipo de Dados INTERVAL DAY TO SECOND: Exemplo 5-24
 EXTRACT 5-25

TZ_OFFSET 5-26
Conversão de `TIMESTAMP` Usando `FROM_TZ` 5-28
Convertendo em `TIMESTAMP` Usando `TO_TIMESTAMP` e `TO_TIMESTAMP_TZ` 5-29
Conversão de Intervalo de Tempo com `TO_YMINTERVAL` 5-30
Usando `TO_DSINTERVAL`: Exemplo 5-31
Horário de Verão 5-32
Sumário 5-34
Exercício 5: Visão Geral 5-35

6 Recuperando Dados Usando Subconsultas

Objetivos 6-2
Subconsultas de Várias Colunas 6-3
Comparações de Colunas 6-4
Subconsulta de Comparação Emparelhada 6-5
Subconsulta de Comparação Não Emparelhada 6-6
Expressões de Subconsultas Escalares 6-7
Subconsultas Escalares: Exemplos 6-8
Subconsultas Correlacionadas 6-10
Usando Subconsultas Correlacionadas 6-12
Usando o Operador `EXISTS` 6-14
Localizar Funcionários com Pelo Menos um Subordinado 6-15
Localizar Todos os Departamentos sem Funcionários 6-16
Instrução `UPDATE` Correlacionada 6-17
Usando a Subconsulta `UPDATE` Correlacionada 6-18
Instrução `DELETE` Correlacionada 6-20
Usando a Subconsulta `DELETE` Correlacionada 6-21
A Cláusula `WITH` 6-22
Cláusula `WITH`: Exemplo 6-23
Sumário 6-25
Exercício 6: Visão Geral 6-27

7 Recuperação Hierárquica

Objetivos 7-2
Dados de Amostra da Tabela `EMPLOYEES` 7-3
Estrutura em Árvore Natural 7-4
Consultas Hierárquicas 7-5
Percorrendo a Árvore 7-6
Percorrendo a Árvore: De Baixo para Cima 7-8
Percorrendo a Árvore: De Cima para Baixo 7-9
Classificando Linhas com a Pseudocoluna `LEVEL` 7-10
Formatando Relatórios Hierárquicos Usando `LEVEL` e `LPAD` 7-11
Reduzindo Ramificações 7-13
Sumário 7-14
Exercício 7: Visão Geral 7-15

8 Suporte a Expressões Comuns

- Objetivos 8-2
- Visão Geral de Expressões Comuns 8-3
- Metacaracteres 8-4
- Usando Metacaracteres 8-5
- Observações 8-6
- Functions de Expressões Comuns 8-8
- A Sintaxe da Function REGEXP 8-9
- Executando Pesquisas Básicas 8-10
- Verificando a Presença de um Padrão 8-11
- Exemplo de Extração de Substrings 8-12
- Substituindo Padrões 8-13
- Expressões Comuns e Constraints de Verificação 8-14
- Sumário 8-15
- Exercício 8: Visão Geral 8-16

Apêndice A: Soluções dos Exercícios**Apêndice B: Descrições e Dados de Tabelas****Apêndice C: Criando Scripts Avançados**

- Objetivos C-2
- Usando SQL para Gerar SQL C-3
- Criando um Script Básico C-4
- Controlando o Ambiente C-5
- O Panorama Completo C-6
- Fazendo Dump do Conteúdo de uma Tabela para um Arquivo C-7
- Gerando um Predicado Dinâmico C-9
- Sumário C-11

Apêndice D: Componentes da Arquitetura Oracle

- Objetivos D-2
- Arquitetura do Banco de Dados Oracle: Visão Geral D-3
- Arquitetura Física do Banco de Dados D-4
- Arquivos de Controle D-5
- Arquivos de Redo Log D-6
- Tablespaces e Arquivos de Dados D-7
- Segmentos, Extensões e Blocos D-8
- Gerenciamento de Instâncias Oracle D-9
- Estruturas de Memória Oracle D-10
- Processos Oracle D-12
- Outras Estruturas Físicas Importantes D-13
- Processando uma Instrução SQL D-14

| | |
|---|------|
| Estabelecendo Conexão com uma Instância | D-15 |
| Processando uma Consulta | D-17 |
| O Shared Pool | D-18 |
| Cache de Buffer do Banco de Dados | D-20 |
| PGA (Program Global Area) | |
| Processando uma Instrução DML | D-22 |
| Buffer de Redo Log | D-24 |
| Segmento de Rollback | D-25 |
| Processamento de COMMIT | D-26 |
| Sumário | D-28 |

Índice

Exercícios Adicionais

Soluções dos Exercícios Adicionais

Prefácio

Prefácio - 2

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 an eKit watermarked with their name and email. Contact OracleWDP_ww@oracle.com if you have not received your personalized eKit.

Perfil

Antes de Iniciar o Curso

- Antes de iniciar este curso, você deve ter experiência prática com SQL.

Pré-requisitos

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

Organização Deste Curso

Banco de Dados Oracle 10g: Fundamentos de SQL II é um curso conduzido por instrutor que contém informações teóricas e exercícios práticos. As sessões de demonstração on-line e os exercícios reforçam as técnicas e os conceitos apresentados.

Publicações Relacionadas

Publicações Adicionais

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

Convenções Tipográficas

Convenções Tipográficas do Texto

| Convenção | Elemento | Exemplo |
|-------------------------|--|---|
| Negrito | Palavras e expressões enfatizadas somente no conteúdo da Web | Para navegar nessa aplicação, não clique nos botões Back e Forward. |
| Itálico negrito | Termos de glossário (se houver glossário) | O <i>Algoritmo</i> insere a nova chave. |
| Colchetes | Nomes das teclas | Pressione [Enter]. |
| Maiúsculas e minúsculas | Botões, caixas de seleção, triggers, janelas | Clique no botão Executable. Marque a caixa de seleção Registration Required. Designe um trigger When-Validate-Item. Abra a janela Master Schedule. |
| Sinais de maior e menor | Menus ou caminhos | Selecione File > Save. |
| Vírgulas | Seqüências de teclas | Pressione e solte estas teclas uma de cada vez: [Alt], [F], [D] |

Convenções Tipográficas (continuação)

Convenções Tipográficas do Texto (continuação)

| Convenção | Objeto ou Termo | Exemplo |
|--|--|--|
| Courier New, distinção entre minúsculas e maiúsculas | Saída de código, elementos de código SQL e PL/SQL, elementos de código Java, nomes de diretórios, nomes de arquivos, senhas, nomes de caminho, URLs, entrada do usuário, nomes de usuários | Saída de código: <code>debug.seti('I',(300);</code> elementos de código SQL: Use o comando <code>SELECT</code> para exibir as informações armazenadas na coluna <code>last_name</code> da tabela <code>emp</code> . Elementos de código Java: A programação em Java envolve as classes <code>String</code> e <code>StringBuffer</code> . Nomes de diretórios: <code>bin</code> (DOS), <code>\$FMHOME</code> (UNIX) Nomes de arquivos: Localize o arquivo <code>init.ora</code> Senhas: Use <code>tiger</code> como sua senha. Caminhos: Abra <code>c:\my_docs\projects</code> . URLs: Vá para <code>http://www.oracle.com</code> . Entrada do usuário: Informe <code>300</code> . Nomes de usuários: Efetue logon como <code>scott</code> . |
| Inicial maiúscula | Labels de gráficos (a menos que o termo seja um nome próprio) | Endereço do cliente (<i>exceto</i> Oracle Payables) |
| Itálico | Palavras e expressões enfatizadas em publicações impressas, títulos de manuais e cursos, e variáveis | <i>Não</i> salve alterações no banco de dados. Para obter mais informações, consulte <i>Oracle7 Server SQL Language Reference Manual</i> . Informe <u><i>user_id@us.oracle.com</i></u> , onde <i>user_id</i> é o nome do usuário. |
| Sinais de adição | Combinações de teclas | Pressione e mantenha estas teclas pressionadas simultaneamente: [Control] + [Alt] + [Delete] |
| Aspas | Títulos de lições e capítulos em referências cruzadas, elementos de interface com nomes longos com apenas a inicial em maiúscula | Este assunto será abordado na Unidade II, Lição 3, "Trabalhando com Objetos". Selecione "Include a reusable module component" e clique em Finish. Use a propriedade "WHERE clause of query". |

Convenções Tipográficas (continuação)

Convenções Tipográficas em Caminhos de Navegação

Este curso usa caminhos de navegação simplificados, como o exemplo a seguir, para orientá-lo nos Aplicativos Oracle.

Exemplo:

Sumário de Lotes de NFFs

(N) Invoice > Entry > Invoice Batches Summary (M) Query > Find (B) Approve

Esse caminho simplificado pode ser traduzido da seguinte forma:

1. (N) Na janela Navigator, selecione Invoice > Entry > Invoice Batches Summary.
2. (M) No menu, selecione Query > Find.
3. (B) Clique no botão Approve.

Notação:

| | |
|------------------|-----------------|
| (N) = Navigator | (I) = Ícone |
| (M) = Menu | (H) = Hiperlink |
| (T) = Guia (Tab) | (B) = Botão |

I

Introdução

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

Objetivos

Ao concluir esta lição, você será capaz de:

- **Listar os objetivos do curso**
- **Descrever as amostras de tabelas usadas no curso**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Objetivos do Curso

Após concluir este curso, você será capaz de:

- **Usar técnicas avançadas de recuperação de dados via SQL para recuperar dados das tabelas do banco de dados**
- **Aplicar técnicas avançadas em um exercício que simula uma situação real**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Visão Geral do Curso

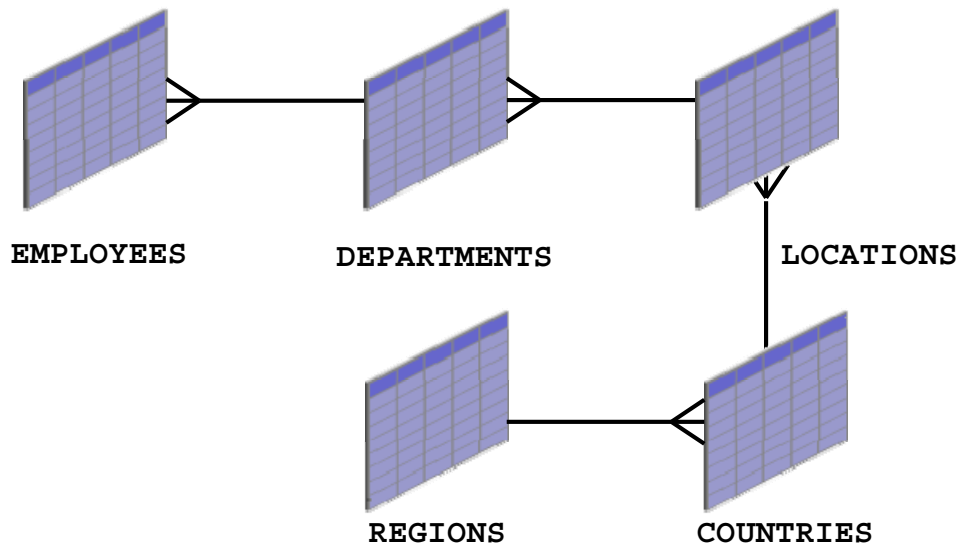
Neste curso, você usará técnicas avançadas de recuperação de dados via SQL, tais como:

- **Functions de data/horário**
- **Operadores ROLLUP, CUBE e GROUPING SETS**
- **Consultas hierárquicas**
- **Subconsultas correlacionadas**
- **Instruções INSERT em várias tabelas**
- **Operação de intercalação**
- **Tabelas externas**
- **Uso de expressões comuns**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Aplicação do Curso



ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Tabelas Usadas no Curso

As seguintes tabelas são usadas neste curso:

EMPLOYEES: A tabela EMPLOYEES contém informações sobre todos os funcionários, como nomes e sobrenomes, IDs de cargo, salários, datas de admissão, IDs de departamento e IDs de gerente. Esta tabela é filha da tabela DEPARTMENTS.

DEPARTMENTS: A tabela DEPARTMENTS contém informações como ID de departamento, nome do departamento, ID de gerente e ID de localização. Esta é a tabela de chave primária da tabela EMPLOYEES.

LOCATIONS: Esta tabela contém informações sobre a localização do departamento. Ela contém informações sobre o ID do local, endereço, estado, província, código postal e ID de país. É a tabela de chave primária da tabela DEPARTMENTS e é uma filha da tabela COUNTRIES.

COUNTRIES: Esta tabela contém os nomes e as IDs dos países e das regiões. Ela é uma filha da tabela REGIONS. Esta é a tabela de chave primária da tabela LOCATIONS.

REGIONS: Esta tabela contém IDs e nomes de regiões de vários países. É uma tabela de chave primária da tabela COUNTRIES.

Sumário

Neste lição, você deverá ter aprendido que:

- **Os objetivos do curso**
- **As amostras de tabelas usadas no curso**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

1

Controlando o Acesso dos Usuários

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Objetivos

Ao concluir esta lição, você será capaz de:

- **Diferenciar privilégios de sistema de privilégios de objeto**
- **Conceder privilégios em tabelas**
- **Exibir privilégios do dicionário de dados**
- **Conceder atribuições**
- **Distinguir privilégios de atribuições**

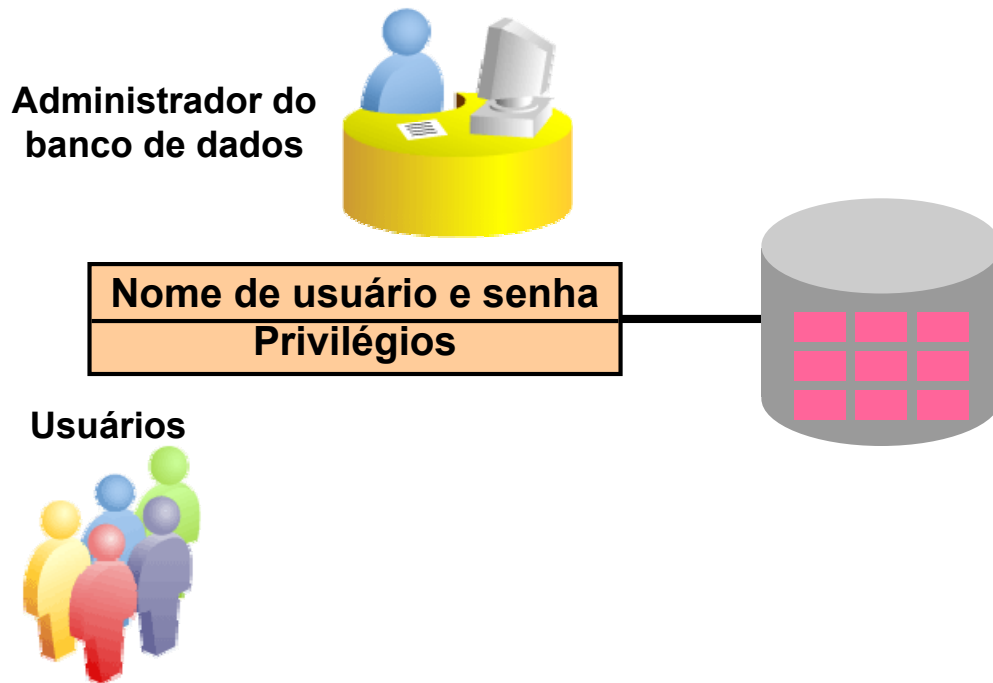
ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Objetivos

Nesta lição, você aprenderá a controlar o acesso a objetos específicos do banco de dados e a adicionar novos usuários com níveis distintos de privilégios de acesso.

Controlando o Acesso dos Usuários



ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Controlando o Acesso dos Usuários

Em um ambiente com vários usuários, você deve manter a segurança de acesso e uso do banco de dados. Com a segurança de banco de dados do Oracle Server, você pode fazer o seguinte:

- Controlar o acesso ao banco de dados
- Permitir acesso a objetos específicos do banco de dados
- Confirmar privilégios concedidos e recebidos com o dicionário de dados Oracle
- Criar sinônimos para objetos de banco de dados

É possível classificar a segurança de um banco de dados em duas categorias: segurança do sistema e segurança de dados. A segurança do sistema abrange o acesso e o uso do banco de dados no nível do sistema, como o nome de usuário e a senha, o espaço em disco alocado para os usuários e as operações do sistema que os usuários podem executar. A segurança do banco de dados abrange o acesso e o uso dos objetos do banco de dados e as ações que os usuários podem executar nesses objetos.

Privilégios

- **Segurança do banco de dados:**
 - Segurança do sistema
 - Segurança de dados
- **Privilégios de sistema: acesso ao banco de dados.**
- **Privilégios de objeto: manipulação do conteúdo dos objetos do banco de dados**
- **Esquemas: conjuntos de objetos, como tabelas, views e seqüências**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Privilégios

Privilégios são direitos de executar instruções SQL específicas. O DBA (administrador de banco de dados) é um usuário de alto nível com a capacidade de criar usuários e conceder com grant acesso de usuários ao banco de dados e seus objetos. Os usuários precisam de *privilégios do sistema* para obter acesso ao banco de dados, e de *privilégios de objeto* para manipular o conteúdo dos objetos do banco de dados. Também é possível oferecer aos usuários o privilégio de conceder com grant privilégios adicionais a outros usuários ou atribuições, que são grupos nomeados de privilégios relacionados.

Esquemas

Um *esquema* é um conjunto de objetos, como tabelas, views e seqüências. O esquema pertence a um usuário do banco de dados e tem o mesmo nome do usuário.

Para obter mais informações, consulte o manual de referência *Application Developer's Guide – Fundamentals do Banco de Dados Oracle 10g*.

Privilégios de Sistema

- **Existem mais de 100 privilégios disponíveis.**
- **O administrador de banco de dados tem privilégios de sistema de alto nível para tarefas como:**
 - **Criar novos usuários**
 - **Remover usuários**
 - **Remover tabelas**
 - **Fazer backup de tabelas**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Privilégios de Sistema

Existem mais de 100 privilégios de sistema distintos disponíveis para usuários e atribuições. Em geral, eles são fornecidos pelo administrador do banco de dados.

Privilégios Típicos de DBA

| Privilégio de Sistema | Operações Autorizadas |
|-----------------------|--|
| CREATE USER | O grantee pode criar outros usuários. |
| DROP USER | O grantee pode eliminar outro usuário. |
| DROP ANY TABLE | O grantee pode eliminar uma tabela de qualquer esquema. |
| BACKUP ANY TABLE | O grantee pode efetuar backup de qualquer tabela em qualquer esquema com o utilitário de exportação. |
| SELECT ANY TABLE | O grantee pode consultar tabelas, views ou snapshots de qualquer esquema. |
| CREATE ANY TABLE | O grantee pode criar tabelas em qualquer esquema. |

Criando Usuários

O DBA cria usuários com a instrução **CREATE USER**.

```
CREATE USER user  
IDENTIFIED BY password;
```

```
CREATE USER HR  
IDENTIFIED BY HR;  
User created.
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Criando um Usuário

O DBA (administrador de banco de dados) cria o usuário executando a instrução **CREATE USER**. Nesse momento, o usuário não tem privilégios. Depois, o DBA pode conceder com **grant** privilégios a esse usuário. Esses privilégios determinam o que o usuário pode fazer no nível do banco de dados.

O slide informa a sintaxe resumida de criação de um usuário.

Na sintaxe:

user é o nome do usuário a ser criado

Password especifica que o usuário deve efetuar login com esta senha

Para obter mais informações, consulte "GRANT" e "CREATE USER" no manual *Oracle Database 10g SQL Reference*.

Privilégios de Sistema de Usuário

- Depois de criar um usuário, o DBA pode conceder com grant privilégios de sistema específicos a ele.

```
GRANT privilege [, privilege...]  
TO user [, user | role, PUBLIC...];
```

- Um desenvolvedor de aplicações, por exemplo, pode ter os seguintes privilégios de sistema:
 - CREATE SESSION
 - CREATE TABLE
 - CREATE SEQUENCE
 - CREATE VIEW
 - CREATE PROCEDURE

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

Privilégios Típicos de Usuário

Depois de criar um usuário, o DBA pode atribuir privilégios a ele.

| Privilégio de Sistema | Operações Autorizadas |
|-----------------------|---|
| CREATE SESSION | Estabelecer conexão com o banco de dados |
| CREATE TABLE | Criar tabelas no esquema do usuário |
| CREATE SEQUENCE | Criar uma seqüência no esquema do usuário |
| CREATE VIEW | Criar uma view no esquema do usuário |
| CREATE PROCEDURE | Criar um procedure, uma function ou um package armazenado no esquema do usuário |

Na sintaxe:

privilege é o privilégio de sistema a ser concedido com grant
user | *role* | *PUBLIC* é o nome do usuário, o nome da atribuição ou
 PUBLIC designa que todo usuário receberá o
 privilégio

Observação: Os privilégios de sistema atuais podem ser encontrados na view de dicionário de dados `SESSION_PRIVS`.

Concedendo Privilégios de Sistema

O DBA pode conceder com grant privilégios de sistema específicos a um usuário.

```
GRANT  create session, create table,  
        create sequence, create view  
TO      scott;  
Grant succeeded.
```

ORACLE

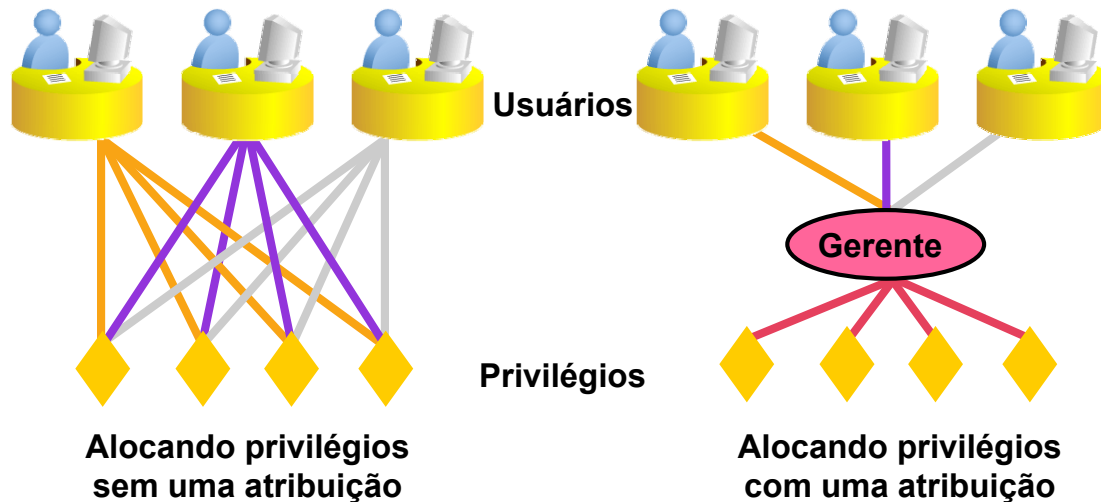
Copyright © 2004, Oracle. Todos os direitos reservados.

Concedendo Privilégios de Sistema

O DBA usa a instrução GRANT para alocar privilégios de sistema para o usuário. Após receber os privilégios, o usuário pode usá-los imediatamente.

No exemplo do slide, o usuário Scott recebeu os privilégios para criar sessões, tabelas, seqüências e views.

O Que É uma Atribuição?



ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

O Que É uma Atribuição?

Uma atribuição é um grupo nomeado de privilégios relacionados que podem ser concedidos com grant ao usuário. Esse método facilita a revogação e a manutenção de privilégios.

Um usuário pode ter acesso a várias atribuições e é possível designar a mesma atribuição a vários usuários. Em geral, as atribuições são criadas para aplicações de banco de dados.

Criando e Designando uma Atribuição

Primeiro, o DBA deve criar a atribuição. Depois, ele pode designar privilégios e usuários para a atribuição.

Sintaxe

```
CREATE ROLE role;
```

Na sintaxe:

role é o nome da atribuição criada

Após a criação da atribuição, o DBA pode usar a instrução GRANT para designar usuários e privilégios à atribuição.

Criando e Concedendo Privilégios a uma Atribuição

- Crie uma atribuição

```
CREATE ROLE manager;  
Role created.
```

- Conceda privilégios a uma atribuição

```
GRANT create table, create view  
TO manager;  
Grant succeeded.
```

- Conceda uma atribuição a usuários

```
GRANT manager TO DE HAAN, KOCHHAR;  
Grant succeeded.
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Criando uma Atribuição

O exemplo do slide cria uma atribuição de gerente e permite que os gerentes criem tabelas e views. Em seguida, a atribuição de gerente é concedida com grant a DeHaan e Kochhar. Agora, DeHaan e Kochhar podem criar tabelas e views.

Se os usuários receberam a concessão de várias atribuições, eles receberão todos os privilégios associados a todas as atribuições.

Alterando a Senha

- O DBA cria sua conta de usuário e inicializa sua senha.
- Você pode alterar sua senha com a instrução ALTER USER.

```
ALTER USER HR  
IDENTIFIED BY employ;  
User altered.
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Alterando a Senha

O DBA cria uma conta e inicializa uma senha para cada usuário. Você pode alterar sua senha com a instrução ALTER USER.

Sintaxe

```
ALTER USER user IDENTIFIED BY password;
```

Na sintaxe:

| | |
|-----------------|-------------------------|
| <i>user</i> | é o nome do usuário |
| <i>password</i> | especifica a nova senha |

Embora seja possível usar essa instrução para alterar a senha, existem várias outras opções. Você precisa ter o privilégio ALTER USER para alterar outras opções.

Para obter mais informações, consulte o manual *Oracle Database10g SQL Reference*.

Observação: O SQL*Plus conta com um comando PASSWORD (PASSW) que pode ser usado para alterar a senha de um usuário quando ele está conectado. Esse comando não se encontra disponível no iSQL*Plus.

Privilégios de Objeto

| Privilégio de Objeto | Tabela | View | Seqüência | Procedure |
|----------------------|--------|------|-----------|-----------|
| ALTER | √ | | √ | |
| DELETE | √ | √ | | |
| EXECUTE | | | | √ |
| INDEX | √ | | | |
| INSERT | √ | √ | | |
| REFERENCES | √ | | | |
| SELECT | √ | √ | √ | |
| UPDATE | √ | √ | | |

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Privilégios de Objeto

Um *privilégio de objeto* é um privilégio ou direito de realizar determinada ação em uma tabela, uma view, uma seqüência ou um procedure específico. Cada objeto tem um conjunto determinado de privilégios que podem ser concedidos. A tabela do slide lista os privilégios de vários objetos. Lembre-se de que os únicos privilégios que se aplicam a uma seqüência são SELECT e ALTER. Você pode restringir UPDATE, REFERENCES e INSERT com a especificação de um subconjunto de colunas atualizáveis. É possível restringir um privilégio SELECT criando uma view com um subconjunto de colunas e concedendo com grant esse privilégio à view. Um privilégio concedido com grant em um sinônimo é convertido em um privilégio na tabela-base referenciada pelo sinônimo.

Privilégios de Objeto

- Os privilégios de objeto variam de acordo com o objeto.
- Um proprietário tem todos os privilégios no objeto.
- Um proprietário pode conceder com grant privilégios específicos em seus próprios objetos.

```
GRANT      object_priv [(columns)]  
ON         object  
TO         {user|role|PUBLIC}  
[WITH GRANT OPTION];
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Concedendo Privilégios de Objeto

Existem privilégios de objeto distintos disponíveis para tipos diferentes de objetos de esquema. Um usuário tem automaticamente todos os privilégios para os objetos de esquema contidos no seu próprio esquema. Um usuário pode conceder com grant privilégios de objeto em qualquer objeto de seu próprio esquema a outros usuários ou atribuições. Se a concessão incluir WITH GRANT OPTION, o grantee poderá conceder com grant o privilégio de objeto a outros usuários. Caso contrário, ele poderá usar o privilégio, mas sem concedê-lo a outros usuários.

Na sintaxe:

| | |
|--------------------|--|
| <i>object_priv</i> | é um privilégio de objeto a ser concedido com grant |
| ALL | especifica todos os privilégios de objeto |
| <i>columns</i> | especifica a coluna de uma tabela ou view na qual os privilégios são concedidos com grant |
| ON <i>object</i> | é o objeto no qual os privilégios são concedidos com grant |
| TO | identifica a quem o privilégio é concedido com grant |
| PUBLIC | concede privilégios de objeto a todos os usuários |
| WITH GRANT OPTION | permite ao grantee conceder com grant os privilégios de objeto a outros usuários e atribuições |

Concedendo Privilégios de Objeto

- **Conceda privilégios de consulta na tabela EMPLOYEES.**

```
GRANT  select
ON      employees
TO      sue, rich;
Grant succeeded.
```

- **Conceda privilégios para atualizar colunas específicas a usuários e atribuições.**

```
GRANT  update (department_name, location_id)
ON      departments
TO      scott, manager;
Grant succeeded.
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Diretrizes

- Para conceder com grant privilégios em um objeto, esse objeto deve estar no seu próprio esquema ou você precisa ter recebido os privilégios WITH GRANT OPTION.
- Um proprietário de objeto pode conceder qualquer privilégio no objeto a outros usuários ou atribuições do banco de dados.
- O proprietário de um objeto adquire automaticamente todos os privilégios nesse objeto.

O primeiro exemplo do slide concede aos usuários Sue e Rich o privilégio para consultar a tabela EMPLOYEES. O segundo exemplo concede privilégios UPDATE em colunas específicas da tabela DEPARTMENTS a Scott e à atribuição de gerente.

Para usar uma instrução SELECT a fim de obter dados da tabela EMPLOYEES, Sue ou Rich deverá usar esta sintaxe:

```
SELECT * FROM HR.employees;
```

Como alternativa, esses usuários podem criar um sinônimo para a tabela e executar a instrução SELECT com o sinônimo:

```
CREATE SYNONYM emp FOR HR.employees;
SELECT * FROM emp;
```

Observação: Em geral, os DBAs alocam privilégios de sistema. Qualquer usuário que tenha um objeto pode conceder privilégios de objeto.

Passando Privilégios

- Ofereça a um usuário autoridade para passar privilégios.

```
GRANT  select, insert
ON     departments
TO     scott
WITH   GRANT OPTION;
Grant succeeded.
```

- Permita a todos os usuários do sistema consultar dados da tabela DEPARTMENTS de Alice.

```
GRANT  select
ON     alice.departments
TO     PUBLIC;
Grant succeeded.
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Palavra-chave WITH GRANT OPTION

Um grantee que tenha recebido um privilégio com a cláusula WITH GRANT OPTION poderá passá-lo a outros usuários e atribuições. Os privilégios de objeto concedidos com a cláusula WITH GRANT OPTION são revogados com revoke quando o privilégio do concessor é revogado.

O exemplo do slide concede ao usuário Scott acesso à tabela DEPARTMENTS com os privilégios para consultá-la e adicionar linhas a ela. O exemplo também mostra que Scott pode conceder com grant esses privilégios a outros.

Palavra-chave PUBLIC

Um proprietário de uma tabela pode conceder com grant acesso a todos os usuários usando a palavra-chave PUBLIC.

O segundo exemplo permite a todos os usuários do sistema consultar dados da tabela DEPARTMENTS de Alice.

Confirmando Privilégios Concedidos com Grant

| View de Dicionário de Dados | Descrição |
|-----------------------------|---|
| ROLE_SYS_PRIVS | Privilégios de sistema concedidos a atribuições |
| ROLE_TAB_PRIVS | Privilégios de tabela concedidos a atribuições |
| USER_ROLE_PRIVS | Atribuições acessíveis ao usuário |
| USER_TAB_PRIVS_MADE | Privilégios de objeto concedidos para os objetos do usuário |
| USER_TAB_PRIVS_RECD | Privilégios de objeto concedidos ao usuário |
| USER_COL_PRIVS_MADE | Privilégios de objeto concedidos nas colunas dos objetos do usuário |
| USER_COL_PRIVS_RECD | Privilégios de objeto concedidos ao usuário em colunas específicas |
| USER_SYS_PRIVS | Privilégios de sistema concedidos ao usuário |

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Confirmando Privilégios Concedidos

Se você tentar executar uma operação não autorizada – por exemplo, deletar uma linha de uma tabela na qual não tem o privilégio DELETE – o Oracle Server não permitirá que a operação ocorra.

Se o Oracle Server enviar a mensagem de erro "table or view does not exist", você executou uma destas operações:

- Nomeou uma tabela ou uma view que não existe
- Tentou executar uma operação em uma tabela ou view cujo privilégio apropriado você não tem

Você pode acessar o dicionário de dados para exibir os privilégios que tem. A tabela do slide descreve diversas views de dicionário de dados.

Revogando Privilégios de Objeto

- Use a instrução **REVOKE** para revogar privilégios concedidos com **grant** a outros usuários.
- Os privilégios concedidos a outros através da cláusula **WITH GRANT OPTION** também são revogados com **revoke**.

```
REVOKE {privilege [, privilege...] | ALL}
ON      object
FROM    {user[, user...] | role | PUBLIC}
[CASCADE CONSTRAINTS];
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

Revogando Privilégios de Objeto

Você pode remover privilégios concedidos a outros usuários usando a instrução **REVOKE**. Quando você usa a instrução **REVOKE**, os privilégios especificados são revogados dos usuários nomeados e dos outros usuários aos quais esses privilégios são concedidos pelo usuário revogado.

Na sintaxe:

A opção **CASCADE** é necessária para remover constraints de integridade referenciais feitas para o objeto **CONSTRAINTS** por meio do privilégio **REFERENCES**.

Para obter mais informações, consulte o manual *Oracle Database10g SQL Reference*.

Observação: Se um usuário sair da empresa e for necessário revogar os privilégios dele, conceda novamente os privilégios que ele possa ter concedido a outros usuários. Se você eliminar a conta do usuário sem revogar os respectivos privilégios, os privilégios de sistema concedidos por esse usuário a outros usuários não serão afetados por essa ação.

Revogando Privilégios de Objeto

Como usuária Alice, revogue os privilégios SELECT e INSERT concedidos com grant ao usuário Scott na tabela DEPARTMENTS.

```
REVOKE select, insert
ON departments
FROM scott;
Revoke succeeded.
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Revogando Privilégios de Objeto (continuação)

O exemplo do slide revoga os privilégios SELECT e INSERT concedidos com grant ao usuário Scott na tabela DEPARTMENTS.

Observação: Se for concedido a um usuário um privilégio com a cláusula WITH GRANT OPTION, esse usuário também poderá conceder o privilégio com essa cláusula, possibilitando a existência de uma longa cadeia de grantees, mas não são permitidos grants circulares. Se o proprietário revogar com revoke um privilégio de um usuário que concedeu com grant esse privilégio a outros usuários, todos os privilégios concedidos serão revogados em cascata.

Por exemplo, se o usuário A conceder com grant o privilégio SELECT em uma tabela ao usuário B, incluindo a cláusula WITH GRANT OPTION, o usuário B também poderá conceder ao usuário C o privilégio SELECT com a mesma cláusula, e o usuário C poderá, então, conceder ao usuário D o privilégio SELECT. Se o usuário A revogar com revoke os privilégios do usuário B, os privilégios concedidos com grant aos usuários C e D também serão revogados.

Sumário

Nesta lição, você conheceu as instruções que controlam o acesso ao banco de dados e aos respectivos objetos.

| Instrução | Ação |
|-------------|---|
| CREATE USER | Cria um usuário (geralmente executado por um DBA) |
| GRANT | Concede a outros usuários privilégios para acessar os objetos |
| CREATE ROLE | Cria um conjunto de privilégios (geralmente executado por um DBA) |
| ALTER USER | Altera a senha de um usuário |
| REVOKE | Remove os privilégios de usuário sobre um objeto |

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Sumário

Os DBAs estabelecem a segurança do banco de dados inicial para os usuários, designando privilégios aos usuários.

- O DBA cria usuários que precisam ter uma senha. O DBA também é responsável pela definição dos privilégios de sistema iniciais de um usuário.
- Depois que o usuário cria um objeto, ele pode passar qualquer um dos privilégios de objeto disponíveis a outros usuários ou a todos os usuários, usando a instrução GRANT.
- Um DBA pode criar atribuições usando a instrução CREATE ROLE para passar um conjunto de privilégios de sistema ou de objeto a vários usuários. As atribuições facilitam a manutenção da concessão e da revogação de privilégios.
- Os usuários podem alterar suas senhas com a instrução ALTER USER.
- Você pode remover os privilégios de usuários com a instrução REVOKE.
- Com as views de dicionário de dados, os usuários podem exibir os privilégios que receberam e os que foram concedidos aos seus objetos.
- Com os vínculos de bancos de dados, você pode acessar dados de bancos de dados remotos. Os privilégios não podem ser concedidos a objetos remotos.

Exercício 1: Visão Geral

Este exercício aborda os seguintes tópicos:

- **Concedendo privilégios a outros usuários sobre a sua tabela**
- **Modificando a tabela de outro usuário usando os privilégios concedidos a você**
- **Criando um sinônimo**
- **Consultando as views de dicionário de dados relacionadas a privilégios**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Exercício 1: Visão Geral

Forme um grupo com outros alunos para fazer este exercício sobre controle de acesso aos objetos do banco de dados.

Exercício 1

Para responder da pergunta 6 em diante, você precisará conectar-se ao banco de dados usando o *iSQL*Plus*. Para isso, acione o browser Internet Explorer no computador cliente. Informe o URL no formato *http://machinename:5561/isqlplus/* e use a conta *oraxx*, a respectiva senha e o identificador de serviço (no formato *Tx*) fornecidos pelo instrutor para efetuar o logon no banco de dados.

1. Que privilégio deve ser concedido a um usuário para que ele efetue logon no servidor Oracle? Este privilégio é de sistema ou de objeto?

2. Que privilégio deve ser concedido a um usuário para que ele crie tabelas?

3. Se você criar uma tabela, quem poderá passar privilégios a outros usuários da sua tabela?

4. Você é o DBA. Você está criando vários usuários que precisam dos mesmos privilégios de sistema. O que você deve usar para facilitar o seu trabalho?

5. Que comando você pode usar para alterar a senha?

6. Conceda a outro usuário acesso à sua tabela DEPARTMENTS. Faça com que o usuário conceda a você acesso de consulta à tabela DEPARTMENTS dele.
7. Consulte todas as linhas da tabela DEPARTMENTS.

| DEPARTMENT_ID | DEPARTMENT_NAME | MANAGER_ID | LOCATION_ID |
|---------------|------------------|------------|-------------|
| 10 | Administration | 200 | 1700 |
| 20 | Marketing | 201 | 1800 |
| 30 | Purchasing | 114 | 1700 |
| ... | | | |
| 40 | Human Resources | 203 | 2400 |
| 50 | Shipping | 121 | 1500 |
| 60 | IT | 103 | 1400 |
| 70 | Public Relations | 204 | 2700 |
| DEPARTMENT_ID | DEPARTMENT_NAME | MANAGER_ID | LOCATION_ID |
| 250 | Retail Sales | | 1700 |
| 260 | Recruiting | | 1700 |
| 270 | Payroll | | 1700 |

27 rows selected.

Exercício 1 (continuação)

8. Adicione uma nova linha à tabela DEPARTMENTS. A Equipe 1 deve adicionar Educação como departamento número 500. A Equipe 2 deve adicionar Recursos Humanos como departamento número 510. Consulte a tabela da outra equipe.
9. Crie um sinônimo para a tabela DEPARTMENTS da outra equipe.
10. Consulte todas as linhas da tabela DEPARTMENTS da outra equipe, usando o sinônimo.

Resultados da instrução SELECT da Equipe 1:

| DEPARTMENT_ID | DEPARTMENT_NAME | MANAGER_ID | LOCATION_ID |
|---------------|-----------------|------------|-------------|
| 500 | Education | | |
| 10 | Administration | 200 | 1700 |
| 20 | Marketing | 201 | 1800 |
| 30 | Purchasing | 114 | 1700 |
| 40 | Human Resources | 203 | 2400 |
| 50 | Shipping | 121 | 1500 |
| 60 | IT | 103 | 1400 |

...

| DEPARTMENT_ID | DEPARTMENT_NAME | MANAGER_ID | LOCATION_ID |
|---------------|------------------|------------|-------------|
| 240 | Government Sales | | 1700 |
| 250 | Retail Sales | | 1700 |
| 260 | Recruiting | | 1700 |
| 270 | Payroll | | 1700 |

28 rows selected.

Resultados da instrução SELECT da Equipe 2:

| DEPARTMENT_ID | DEPARTMENT_NAME | MANAGER_ID | LOCATION_ID |
|---------------|------------------|------------|-------------|
| 10 | Administration | 200 | 1700 |
| 20 | Marketing | 201 | 1800 |
| 30 | Purchasing | 114 | 1700 |
| 40 | Human Resources | 203 | 2400 |
| 50 | Shipping | 121 | 1500 |
| 60 | IT | 103 | 1400 |
| 70 | Public Relations | 204 | 2700 |

...

| DEPARTMENT_ID | DEPARTMENT_NAME | MANAGER_ID | LOCATION_ID |
|---------------|-----------------|------------|-------------|
| 250 | Retail Sales | | 1700 |
| 260 | Recruiting | | 1700 |
| 270 | Payroll | | 1700 |
| 510 | Human Resources | | |

28 rows selected.

Exercício 1 (continuação)

11. Consulte o dicionário de dados USER_TABLES para ver as informações das suas tabelas.

| TABLE_NAME |
|-------------|
| JOB_HISTORY |
| EMPLOYEES |
| JOBS |
| DEPARTMENTS |
| LOCATIONS |
| REGIONS |
| COUNTRIES |

7 rows selected.

12. Consulte a view de dicionário de dados ALL_TABLES para ver as informações de todas as tabelas que você pode acessar. Exclua as suas tabelas.

Observação: Talvez a sua lista não corresponda exatamente à lista mostrada abaixo.

| TABLE_NAME | OWNER |
|----------------------|-------|
| DUAL | SYS |
| SYSTEM_PRIVILEGE_MAP | SYS |
| ... | |
| WK\$ACL_SNAPSHOT | WKSYS |
| DEPARTMENTS | ORA2 |

13. Revogue o privilégio SELECT da outra equipe.
14. Remova a linha que você inseriu na tabela DEPARTMENTS na etapa 8 e salve as alterações.

2

Gerenciar Objetos de Esquema

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Objetivos

Ao concluir esta lição, você será capaz de:

- **Adicionar constraints**
- **Criar índices**
- **Criar índices usando a instrução `CREATE TABLE`**
- **Criar índices baseados em função**
- **Eliminar colunas e definir uma coluna como `UNUSED`**
- **Executar operações `FLASHBACK`**
- **Criar e usar tabelas externas**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Objetivos

Esta lição contém informações sobre a criação de índices e constraints e a alteração de objetos existentes. Você também conhecerá as tabelas externas e a nomeação do índice durante a criação da constraint de chave primária.

A Instrução ALTER TABLE

Use a instrução ALTER TABLE para:

- Adicionar uma nova coluna
- Modificar uma coluna existente
- Definir um valor default para a nova coluna
- Eliminar uma coluna

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

A Instrução ALTER TABLE

Depois de criar uma tabela, talvez você precise alterar sua estrutura para acrescentar uma coluna omitida, alterar a definição de uma coluna ou remover colunas. Para fazer isso, use a instrução ALTER TABLE.

A Instrução ALTER TABLE

Use a instrução ALTER TABLE para adicionar, modificar ou eliminar colunas.

```
ALTER TABLE table
ADD          (column datatype [DEFAULT expr]
             [, column datatype] ...);
```

```
ALTER TABLE table
MODIFY       (column datatype [DEFAULT expr]
             [, column datatype] ...);
```

```
ALTER TABLE table
DROP        (column);
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

A Instrução ALTER TABLE (continuação)

Você pode adicionar, modificar e eliminar colunas de uma tabela usando a instrução ALTER TABLE.

Na sintaxe:

| | |
|---------------------|---|
| <i>table</i> | é o nome da tabela |
| ADD MODIFY DROP | é o tipo de modificação |
| <i>column</i> | é o nome da nova coluna |
| <i>datatype</i> | é o tipo de dados e o tamanho da nova coluna |
| DEFAULT <i>expr</i> | especifica o valor default para uma nova coluna |

Adicionando uma Coluna

- Use a cláusula ADD para adicionar colunas.

```
ALTER TABLE dept80
ADD      (job_id VARCHAR2(9));
Table altered.
```

- A nova coluna será a última coluna.

| EMPLOYEE_ID | LAST_NAME | ANNSAL | HIRE_DATE | JOB_ID |
|-------------|-----------|--------|-----------|--------|
| 145 | Russell | 14000 | 01-OCT-96 | |
| 146 | Partners | 13500 | 05-JAN-97 | |
| 147 | Errazuriz | 12000 | 10-MAR-97 | |
| 148 | Cambrault | 11000 | 15-OCT-99 | |
| 149 | Zlotkey | 10500 | 29-JAN-00 | |

...

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Diretrizes para Adicionar uma Coluna

- Você pode adicionar ou modificar colunas.
- Não é possível especificar onde a coluna será exibida. A nova coluna será a última coluna.

O exemplo do slide adiciona uma coluna denominada JOB_ID à tabela DEPT80. A coluna JOB_ID torna-se a última coluna da tabela.

Observação: Se uma tabela já contiver linhas quando uma coluna for adicionada, inicialmente, todas as linhas da nova coluna terão valores nulos. Não é possível adicionar uma coluna NOT NULL obrigatória a uma tabela que contém dados nas outras colunas. Só é possível adicionar uma coluna NOT NULL a uma tabela vazia.

Modificando uma Coluna

- **Você pode alterar o tipo de dados, o tamanho e o valor default de uma coluna.**

```
ALTER TABLE dept80  
MODIFY      (last_name VARCHAR2(30));  
Table altered.
```

- **Uma alteração no valor default afeta apenas as inserções subseqüentes na tabela.**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

Modificando uma Coluna

Você pode modificar a definição de uma coluna usando a instrução `ALTER TABLE` com a cláusula `MODIFY`. A modificação de uma coluna pode incluir alterações de tipo de dados, tamanho e valor default da coluna.

Diretrizes

- Você pode aumentar a largura ou a precisão de uma coluna numérica.
- É possível aumentar a largura de colunas numéricas ou de caracteres.
- Você pode diminuir a largura de uma coluna se:
 - A coluna contiver apenas valores nulos
 - A tabela não tiver nenhuma linha
 - A redução na largura da coluna não for inferior aos valores existentes nessa coluna
- Você poderá alterar o tipo de dados se a coluna contiver apenas valores nulos. A exceção a essa regra é a conversão de `CHAR` em `VARCHAR2`, que pode ser feita com dados contidos nas colunas.
- Você só poderá converter uma coluna com o tipo de dados `CHAR` em `VARCHAR2` ou vice-versa se a coluna contiver valores nulos ou se seu tamanho não for alterado.
- Uma alteração no valor default de uma coluna afeta apenas as inserções subseqüentes na tabela.

Eliminando uma Coluna

Use a cláusula **DROP COLUMN** para eliminar da tabela as colunas que não são mais necessárias.

```
ALTER TABLE dept80
DROP COLUMN job_id;
Table altered.
```

| EMPLOYEE_ID | LAST_NAME | ANNSAL | HIRE_DATE |
|-------------|-----------|--------|-----------|
| 145 | Russell | 14000 | 01-OCT-96 |
| 146 | Partners | 13500 | 05-JAN-97 |
| 147 | Errazuriz | 12000 | 10-MAR-97 |
| 148 | Cambrault | 11000 | 15-OCT-99 |
| 149 | Zlotkey | 10500 | 29-JAN-00 |

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Eliminando uma Coluna

Você pode eliminar uma coluna de uma tabela usando a instrução **ALTER TABLE** com a cláusula **DROP COLUMN**.

Diretrizes

- A coluna pode ou não conter dados.
- Com a instrução **ALTER TABLE**, só é possível eliminar uma coluna por vez.
- A tabela deve reter, pelo menos, uma coluna depois de alterada.
- Não é possível recuperar uma coluna depois de eliminada.
- Não é possível eliminar uma coluna se ela fizer parte de uma constraint ou de uma chave de índice, a menos que a opção de cascata seja adicionada.
- A eliminação de uma coluna pode demorar um pouco quando ela tem muitos valores. Nesse caso, talvez seja melhor defini-la como não utilizada e eliminá-la quando o número de usuários no sistema diminuir, a fim de evitar bloqueios de longa duração.

Observação: Algumas colunas não podem ser eliminadas. Este é o caso das colunas que fazem parte da chave de particionamento de uma tabela particionada ou das colunas que fazem parte da chave primária de uma tabela organizada por índice.

A Opção SET UNUSED

- Use a opção SET UNUSED para marcar uma ou mais colunas como não utilizadas.
- Use a opção DROP UNUSED COLUMNS para remover colunas marcadas como não utilizadas.

```
ALTER TABLE <table_name>  
SET UNUSED(<column_name>);  
OU  
ALTER TABLE <table_name>  
SET UNUSED COLUMN <column_name>;  
  
ALTER TABLE <table_name>  
DROP UNUSED COLUMNS;
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

A Opção SET UNUSED

A opção SET UNUSED marca uma ou mais colunas como não utilizadas de forma que elas possam ser eliminadas quando a demanda de recursos do sistema for menor. A especificação dessa cláusula, na verdade, não remove as colunas de destino de cada linha da tabela (isto é, não restaura o espaço em disco usado por essas colunas). Portanto, o tempo de resposta é mais rápido do que se você executasse a cláusula DROP. As colunas não utilizadas são tratadas como se tivessem sido eliminadas, embora os respectivos dados permaneçam nas linhas da tabela. Você não terá mais acesso a uma coluna depois que ela for marcada como não utilizada. Uma consulta SELECT * não recuperará dados de colunas não utilizadas. Além disso, os nomes e os tipos de colunas marcadas como não utilizadas não serão exibidos durante uma instrução DESCRIBE, e você poderá adicionar uma nova coluna à tabela com o mesmo nome da coluna não utilizada. As informações da opção SET UNUSED são armazenadas na view de dicionário de dados USER_UNUSED_COL_TABS.

Observação: As diretrizes para definir colunas como UNUSED são semelhantes às usadas para eliminá-las.

A Opção DROP UNUSED COLUMNS

A opção DROP UNUSED COLUMNS remove da tabela todas as colunas marcadas como não utilizadas no momento. Você poderá usar essa instrução quando quiser solicitar o espaço extra em disco de colunas não utilizadas da tabela. Se a tabela não contiver colunas não utilizadas, a instrução não retornará erros.

```
ALTER TABLE dept80
SET UNUSED (last_name);
Table altered.
```

```
ALTER TABLE dept80
DROP UNUSED COLUMNS;
Table altered.
```

Adicionando uma Sintaxe de Constraint

Use a instrução **ALTER TABLE** para:

- Adicionar ou eliminar uma constraint, mas não para modificar sua estrutura
- Ativar ou desativar constraints
- Adicionar uma constraint **NOT NULL** usando a cláusula **MODIFY**

```
ALTER TABLE <table_name>  
ADD [CONSTRAINT <constraint_name>]  
type (<column_name>);
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

Adicionando uma Constraint

Você pode adicionar uma constraint para tabelas existentes usando a instrução **ALTER TABLE** com a cláusula **ADD**.

Na sintaxe:

table é o nome da tabela
constraint é o nome da constraint
type é o tipo de constraint
column é o nome da coluna afetada pela constraint

A sintaxe de nome de constraint é opcional, embora recomendada. Se você não nomear suas constraints, o sistema gerará nomes para elas.

Diretrizes

- Você pode adicionar, eliminar, ativar ou desativar uma constraint, mas não pode modificar sua estrutura.
- Você pode adicionar uma constraint **NOT NULL** a uma coluna existente usando a cláusula **MODIFY** da instrução **ALTER TABLE**.

Observação: Você só poderá definir uma coluna **NOT NULL** se a tabela estiver vazia ou se a coluna tiver um valor para cada linha.

Adicionando uma Constraint

Adicione uma constraint FOREIGN KEY à tabela EMP2 para indicar que já deve existir um gerente como funcionário válido nessa tabela.

```
ALTER TABLE emp2
modify employee_id Primary Key;
Table altered.
```

```
ALTER TABLE emp2
ADD CONSTRAINT emp_mgr_fk
FOREIGN KEY(manager_id)
REFERENCES emp2(employee_id);
Table altered.
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Adicionando uma Constraint (continuação)

O primeiro exemplo do slide modifica a tabela EMP2 para adicionar uma constraint PRIMARY KEY à coluna EMPLOYEE_ID. Observe que, como não foi especificado nenhum nome de constraint, a constraint é automaticamente nomeada pelo servidor Oracle. O segundo exemplo do slide cria uma constraint FOREIGN KEY na tabela EMP2. A constraint garante que exista um gerente na condição de funcionário válido na tabela EMP2.

ON DELETE CASCADE

Delete linhas filhas quando uma chave mãe for deletada.

```
ALTER TABLE Emp2 ADD CONSTRAINT emp_dt_fk  
FOREIGN KEY (Department_id)  
REFERENCES departments ON DELETE CASCADE);  
Table altered.
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

ON DELETE CASCADE

A ação ON DELETE CASCADE permite a deleção, mas não a atualização, de dados da chave mãe referenciados pela tabela filha. Quando os dados da chave mãe são deletados, também são deletadas todas as linhas da tabela filha que dependem dos valores deletados da chave mãe. Para especificar essa ação referencial, inclua a opção ON DELETE CASCADE na definição da constraint FOREIGN KEY.

Adiando Constraints

As constraints podem ter os seguintes atributos:

- DEFERRABLE OU NOT DEFERRABLE
- INITIALLY DEFERRED OU INITIALLY IMMEDIATE

```
ALTER TABLE dept2  
ADD CONSTRAINT dept2_id_pk  
PRIMARY KEY (department_id)  
DEFERRABLE INITIALLY DEFERRED
```

Adiando uma constraint durante a criação

```
SET CONSTRAINTS dept2_id_pk IMMEDIATE
```

Alterando um atributo específico de constraint

```
ALTER SESSION  
SET CONSTRAINTS= IMMEDIATE
```

Alterando todas as constraints para uma sessão

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Adiando Constraints

Você pode adiar a verificação da validade das constraints até o fim da transação. Uma constraint será adiada se o sistema verificar que ela só é atendida no commit. Se uma constraint adiada for violada, o commit causará o rollback da transação. Se uma constraint for imediata (não adiada), ela será verificada ao final de cada instrução. Se ela for violada, o rollback da instrução ocorrerá imediatamente. Se uma constraint gerar uma ação (por exemplo, DELETE CASCADE), essa ação sempre será considerada como parte da instrução que a causou, sendo que a constraint pode ser imediata ou adiada. Use a instrução SET CONSTRAINTS para especificar, para determinada transação, se uma constraint adiável é verificada após cada instrução DML ou quando a transação é submetida a commit. A fim de criar constraints adiáveis, você deve criar um índice não exclusivo para essa constraint.

Você pode definir uma constraint como adiável ou não adiável, e como inicialmente adiável ou inicialmente imediata. Esses atributos podem ser diferentes para cada constraint.

Cenário de uso: A política da empresa determina que o número de departamento 40 deve ser alterado para 45. A alteração da coluna DEPARTMENT_ID afeta os funcionários atribuídos a esse departamento. Portanto, você torna a chave primária e as chaves estrangeiras adiáveis e inicialmente adiadas. Você atualiza as informações de departamentos e funcionários e, no momento do commit, todas as linhas são validadas.

Eliminando uma Constraint

- **Remova a constraint de gerente da tabela EMP2.**

```
ALTER TABLE emp2
DROP CONSTRAINT emp_mgr_fk;
Table altered.
```

- **Remova a constraint PRIMARY KEY da tabela DEPT2 e elimine a constraint FOREIGN KEY associada da coluna EMP2.DEPARTMENT_ID.**

```
ALTER TABLE dept2
DROP PRIMARY KEY CASCADE;
Table altered.
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

Eliminando uma Constraint

Para eliminar uma constraint, você pode identificar o respectivo nome nas views de dicionário de dados USER_CONSTRAINTS e USER_CONS_COLUMNS. Depois, use a instrução ALTER TABLE com a cláusula DROP. A opção CASCADE da cláusula DROP elimina também as constraints dependentes.

Sintaxe

```
ALTER TABLE table
DROP PRIMARY KEY | UNIQUE (column) |
CONSTRAINT constraint [CASCADE];
```

Na sintaxe:

| | |
|-------------------|--|
| <i>table</i> | é o nome da tabela |
| <i>column</i> | é o nome da coluna afetada pela constraint |
| <i>constraint</i> | é o nome da constraint |

Quando você elimina uma constraint de integridade, ela não é mais imposta pelo servidor Oracle e não continua disponível no dicionário de dados.

Desativando Constraints

- Execute a cláusula **DISABLE** da instrução **ALTER TABLE** para desativar uma constraint de integridade.
- Aplique a opção **CASCADE** para desativar as constraints de integridade dependentes.

```
ALTER TABLE emp2  
DISABLE CONSTRAINT emp_dt_fk;  
Table altered.
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

Desativando uma Constraint

Você pode desativar uma constraint sem eliminá-la, ou recriá-la usando a instrução **ALTER TABLE** com a cláusula **DISABLE**.

Sintaxe

```
ALTER TABLE table  
DISABLE CONSTRAINT constraint [CASCADE];
```

Na sintaxe:

table é o nome da tabela
constraint é o nome da constraint

Diretrizes

- Você pode usar a cláusula **DISABLE** nas instruções **CREATE TABLE** e **ALTER TABLE**.
- A cláusula **CASCADE** desativa constraints de integridade dependentes.
- A desativação de uma constraint de chave exclusiva ou primária remove o índice exclusivo.

Ativando Constraints

- **Ative uma constraint de integridade desativada no momento da definição da tabela usando a cláusula ENABLE.**

```
ALTER TABLE      emp2
ENABLE CONSTRAINT emp_dt_fk;
Table altered.
```

- **Um índice UNIQUE será automaticamente criado se você ativar uma constraint de chave UNIQUE ou PRIMARY.**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

Ativando uma Constraint

Você pode ativar uma constraint sem eliminá-la, ou recriá-la usando a instrução ALTER TABLE com a cláusula ENABLE.

Sintaxe

```
ALTER      TABLE      table
ENABLE     CONSTRAINT constraint;
```

Na sintaxe:

table é o nome da tabela
constraint é o nome da constraint

Diretrizes

- Se você ativar uma constraint, ela será aplicada a todos os dados da tabela. Todos os dados da tabela devem estar de acordo com a constraint.
- Se você ativar uma chave UNIQUE ou PRIMARY, um índice de chave UNIQUE ou PRIMARY será criado automaticamente. Caso já exista um índice, ele poderá ser usado por essas chaves.
- Você pode usar a cláusula ENABLE nas instruções CREATE TABLE e ALTER TABLE.

Adicionando uma Constraint (continuação)

Diretrizes (continuação)

- A ativação de uma constraint de chave primária que tenha sido desativada com a opção CASCADE não ativará as chaves estrangeiras dependentes da chave primária.
- Para ativar uma constraint de chave UNIQUE ou PRIMARY, você deve ter os privilégios necessários para criar um índice na tabela.

Constraints em Cascata

- A cláusula **CASCADE CONSTRAINTS** é usada com a cláusula **DROP COLUMN**.
- A cláusula **CASCADE CONSTRAINTS** elimina todas as constraints de integridade referencial que fazem referência a chaves primárias e exclusivas definidas nas colunas eliminadas.
- A cláusula **CASCADE CONSTRAINTS** também elimina as constraints de várias colunas definidas nas colunas eliminadas.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Constraints em Cascata

Esta instrução ilustra o uso da cláusula **CASCADE CONSTRAINTS**. Suponha que a tabela **TEST1** seja criada da seguinte maneira:

```
CREATE TABLE test1 (  
    pk NUMBER PRIMARY KEY,  
    fk NUMBER,  
    col1 NUMBER,  
    col2 NUMBER,  
    CONSTRAINT fk_constraint FOREIGN KEY (fk) REFERENCES test1,  
    CONSTRAINT ck1 CHECK (pk > 0 and col1 > 0),  
    CONSTRAINT ck2 CHECK (col2 > 0));
```

Será retornado um erro para as seguintes instruções:

ALTER TABLE test1 DROP (pk); —pk é uma chave mãe.

ALTER TABLE test1 DROP (col1); —col1 é referenciado pela constraint de várias colunas ck1.

Constraints em Cascata

Exemplo:

```
ALTER TABLE emp2  
DROP COLUMN employee_id CASCADE CONSTRAINTS;  
Table altered.
```

```
ALTER TABLE test1  
DROP (pk, fk, col1) CASCADE CONSTRAINTS;  
Table altered.
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Constraints em Cascata (continuação)

A submissão da seguinte instrução elimina a coluna EMPLOYEE_ID, a constraint de chave primária e todas as constraints de chave estrangeira que façam referência à constraint de chave primária para a tabela EMP2:

```
ALTER TABLE emp2 DROP COLUMN employee_id CASCADE CONSTRAINTS;
```

Se todas as colunas às quais as constraints definidas nas colunas eliminadas fizerem referência também forem eliminadas, a cláusula CASCADE CONSTRAINTS não será necessária. Por exemplo, supondo que nenhuma constraint referencial de outras tabelas faça referência à coluna PK, será válido submeter a seguinte instrução, sem a cláusula CASCADE CONSTRAINTS, para a tabela TEST1 criada na página anterior:

```
ALTER TABLE test1 DROP (pk, fk, col1);
```

Visão Geral de Índices

Os índices são criados:

- **Automaticamente:**
 - Criação de PRIMARY KEY
 - Criação de UNIQUE KEY
- **Manualmente**
 - Instrução CREATE INDEX
 - Instrução CREATE TABLE

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

Visão Geral de Índices

É possível criar dois tipos de índice. Um tipo é o índice exclusivo. O servidor Oracle cria automaticamente um índice exclusivo quando você define uma constraint de chave PRIMARY ou UNIQUE para uma coluna ou um grupo de colunas de uma tabela. O nome do índice é aquele especificado para a constraint.

O outro tipo é o índice não exclusivo, que pode ser criado pelo usuário. Por exemplo, você pode criar um índice para uma coluna FOREIGN KEY para ser usado em joins a fim de aumentar a velocidade de recuperação.

Você pode criar um índice em uma ou mais colunas executando a instrução CREATE INDEX.

Para obter mais informações, consulte o manual *Oracle Database 10g SQL Reference*.

Observação: Você pode criar manualmente um índice exclusivo, mas é recomendável criar uma constraint exclusiva, que gera implicitamente um índice exclusivo.

CREATE INDEX com a Instrução CREATE TABLE

```
CREATE TABLE NEW_EMP  
(employee_id NUMBER(6)  
    PRIMARY KEY USING INDEX  
    (CREATE INDEX emp_id_idx ON  
    NEW_EMP(employee_id)),  
first_name VARCHAR2(20),  
last_name VARCHAR2(25));  
Table created.
```

```
SELECT INDEX_NAME, TABLE_NAME  
FROM USER_INDEXES  
WHERE TABLE_NAME = 'NEW_EMP';
```

| INDEX_NAME | TABLE_NAME |
|------------|------------|
| EMP_ID_IDX | NEW_EMP |

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

CREATE INDEX com a Instrução CREATE TABLE

No exemplo do slide, a cláusula CREATE INDEX é usada com a instrução CREATE TABLE para criar um índice de chave primária explicitamente. Você pode nomear os índices no momento da criação da chave primária para diferenciá-lo do nome da constraint PRIMARY KEY. O exemplo a seguir ilustra o comportamento do banco de dados se o índice não for explicitamente nomeado:

```
CREATE TABLE EMP_UNNAMED_INDEX  
(employee_id NUMBER(6) PRIMARY KEY ,  
first_name VARCHAR2(20),  
last_name VARCHAR2(25));
```

Table created.

```
SELECT INDEX_NAME, TABLE_NAME  
FROM USER_INDEXES  
WHERE TABLE_NAME = 'EMP_UNNAMED_INDEX';
```

| INDEX_NAME | TABLE_NAME |
|-------------|-------------------|
| SYS_C002835 | EMP_UNNAMED_INDEX |

CREATE INDEX com a Instrução CREATE TABLE (continuação)

Observe que o servidor Oracle fornece um nome genérico ao índice criado para a coluna de chave primária.

Também é possível usar um índice existente para a coluna de chave primária; por exemplo, quando você estiver esperando uma carga de dados volumosa e quiser acelerar a operação. Você pode desativar as constraints enquanto executa a carga e, em seguida, ativá-las. Nesse caso, a existência de um índice exclusivo na chave primária fará com que os dados sejam verificados durante a carga. Sendo assim, você pode primeiro criar um índice não exclusivo na coluna designada como PRIMARY KEY e depois criar a coluna PRIMARY KEY e especificar que ela deve usar o índice existente. Os exemplos abaixo ilustram esse processo:

Etapa 1: Crie a tabela

```
CREATE TABLE NEW_EMP2
(  employee_id NUMBER(6)
   first_name  VARCHAR2(20),
   last_name   VARCHAR2(25)
);
```

Etapa 2: Crie o índice

```
CREATE INDEX emp_id_idx2 ON
            new_emp2(employee_id);
```

Etapa 3: Crie a Chave Primária

```
ALTER TABLE new_emp2 ADD PRIMARY KEY   (employee_id) USING INDEX
            emp_id_idx2;
```

Índices Baseados em Função

- Um índice baseado em função utiliza expressões.
- A expressão do índice é criada a partir de colunas de tabela, constantes, funções SQL e funções definidas pelo usuário.

```
CREATE INDEX upper_dept_name_idx  
ON dept2 (UPPER(department_name));
```

Index created.

```
SELECT *  
FROM   dept2  
WHERE  UPPER(department_name) = 'SALES';
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Índices Baseados em Função

Os índices baseados em função com as palavras-chave `UPPER(column_name)` ou `LOWER(column_name)` permitem pesquisas sem distinção entre maiúsculas e minúsculas. Por exemplo, o índice:

```
CREATE INDEX upper_last_name_idx ON emp2 (UPPER(last_name));
```

facilita o processamento de consultas como:

```
SELECT * FROM emp2 WHERE UPPER(last_name) = 'KING';
```

O servidor Oracle usa o índice apenas quando essa função específica é usada em uma consulta. Por exemplo, talvez a instrução abaixo use o índice, mas, sem a cláusula `WHERE`, o servidor Oracle poderá executar uma varredura integral de tabela:

```
SELECT *  
FROM   employees  
WHERE  UPPER (last_name) IS NOT NULL  
ORDER BY UPPER (last_name);
```

Observação: O parâmetro de inicialização `QUERY_REWRITE_ENABLED` deve ser definido como `TRUE` para que seja usado um índice baseado em função.

Índices Baseados em Função (continuação)

O servidor Oracle trata os índices com colunas marcadas com DESC como índices baseados em função. As colunas marcadas com DESC são classificadas em ordem decrescente.

Removendo um Índice

- Para remover um índice do dicionário de dados, use o comando `DROP INDEX`.

```
DROP INDEX index;
```

- Remova o índice `UPPER_DEPT_NAME_IDX` do dicionário de dados.

```
DROP INDEX upper_dept_name_idx;  
Index dropped.
```

- Para eliminar um índice, você precisa ser o proprietário ou ter o privilégio `DROP ANY INDEX`.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Removendo um Índice

Não é possível modificar índices. Para alterar um índice, elimine-o e, depois, recrie-o. Remova uma definição de índice do dicionário de dados executando a instrução `DROP INDEX`. Para eliminar um índice, você precisa ser o proprietário ou ter o privilégio `DROP ANY INDEX`.

Na sintaxe:

index é o nome do índice

Observação: Se você eliminar uma tabela, os índices e as constraints serão eliminados automaticamente, mas as views e as seqüências permanecerão.

DROP TABLE ...PURGE

```
DROP TABLE dept80 PURGE;
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

DROP TABLE ...PURGE

O Banco de Dados Oracle 10g apresenta um novo recurso para a eliminação de tabelas. Quando você elimina uma tabela, o banco de dados não libera imediatamente o espaço associado a ela. Em vez disso, o banco de dados renomeia a tabela e a coloca em uma lixeira, de onde pode ser posteriormente recuperada com a instrução `FLASHBACK TABLE`, caso você tenha eliminado a tabela por engano. Se quiser liberar imediatamente o espaço associado à tabela no momento da execução da instrução `DROP TABLE`, inclua a cláusula `PURGE`, conforme mostrado na instrução do slide.

Especifique `PURGE` apenas se quiser eliminar a tabela e liberar o espaço associado a ela em uma única etapa. Caso especifique `PURGE`, o banco de dados não colocará a tabela e seus objetos dependentes na lixeira.

O uso dessa cláusula equivale a eliminar a tabela e depois expurgá-la da lixeira. Essa cláusula economiza uma etapa do processo. Ela também oferece segurança avançada caso você queira impedir que material confidencial seja exibido na lixeira.

Observação: Não é possível executar rollback de uma instrução `DROP TABLE` com a cláusula `PURGE`, nem recuperar uma tabela eliminada com a cláusula `PURGE`. Esse recurso não estava disponível em releases anteriores.

A Instrução FLASHBACK TABLE

- **Ferramenta de correção de modificações acidentais em tabelas**
 - Restaura uma tabela até um momento anterior
 - Vantagens: Facilidade de uso, disponibilidade, execução rápida
 - Pode ser executada localmente
- **Sintaxe:**

```
FLASHBACK TABLE[schema.]table[,  
[ schema.]table ]...  
TO { TIMESTAMP | SCN } expr  
[ { ENABLE | DISABLE } TRIGGERS ];
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

A Instrução FLASHBACK TABLE

Recurso de Autocorreção

O Banco de Dados Oracle 10g contém um novo comando SQL DDL, a instrução `FLASHBACK TABLE`, para restaurar o estado de uma tabela até um momento anterior, caso ela tenha sido inadvertidamente deletada ou modificada. O comando `FLASHBACK TABLE` é uma ferramenta de autocorreção para restaurar dados em uma tabela, juntamente com os atributos associados, como índices ou views. Isso ocorre quando o banco de dados está on-line, fazendo o rollback apenas das alterações subsequentes da tabela. Quando comparado com mecanismos tradicionais de recuperação, o recurso oferece vantagens significativas, como facilidade de uso e restauração mais rápida. Ele também libera o DBA do trabalho de localizar e restaurar propriedades específicas da aplicação. O recurso de tabela de flashback não corrige o dano físico causado por um disco ruim.

Sintaxe

Você pode chamar uma operação de tabela de flashback em uma ou mais tabelas, mesmo em tabelas de esquemas diferentes. Você especifica o momento para o qual deseja fazer a reversão, especificando um timestamp válido. Por default, os triggers de banco de dados são desativados para todas as tabelas envolvidas. É possível sobrepor esse comportamento default, especificando a cláusula `ENABLE TRIGGERS`.

Observação: Para obter mais informações sobre a lixeira e a semântica de flashback, consulte o *Oracle Database Administrator's Reference 10g Release 1 (10.1)*.

A Instrução FLASHBACK TABLE

```
DROP TABLE emp2;  
Table dropped
```

```
SELECT original_name, operation, droptime,  
FROM recyclebin;
```

| ORIGINAL_NAME | OPERATION | DROPTIME |
|---------------|-----------|---------------------|
| EMP2 | DROP | 2004-03-03:07:57:11 |
| ... | | |

```
FLASHBACK TABLE emp2 TO BEFORE DROP;  
Flashback complete
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

A Instrução FLASHBACK TABLE (continuação)

Sintaxe e Exemplos (continuação)

O exemplo restaura a tabela EMP2 ao estado anterior a uma instrução DROP.

Na prática, a lixeira é uma tabela de dicionário de dados que contém informações sobre os objetos eliminados. As tabelas eliminadas e os objetos associados, como índices, constraints, tabelas aninhadas e outros, não são removidas e continuam ocupando espaço. Elas continuam sendo contabilizadas nas cotas de espaço do usuário, até serem especificamente expurgadas da lixeira ou até que ocorra a situação improvável na qual as tabelas precisam ser expurgadas pelo banco de dados devido a restrições de espaço no tablespaces.

Cada usuário pode ser considerado como proprietário de uma lixeira, pois, a menos que ele tenha privilégios de SYSDBA, os únicos objetos que ele poderá acessar na lixeira serão os de sua propriedade. Um usuário pode exibir seus objetos na lixeira usando a seguinte instrução:

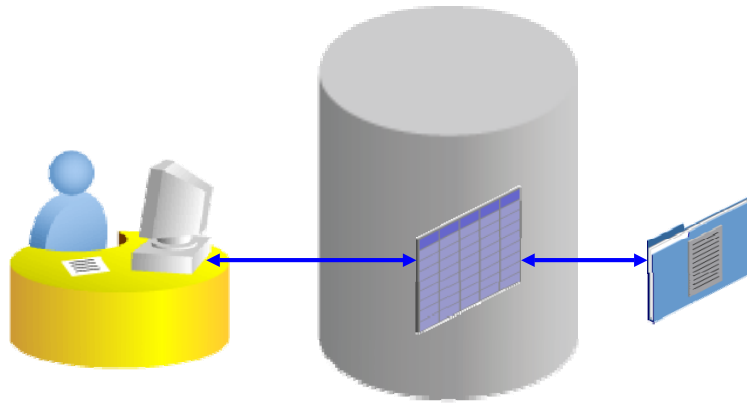
```
SELECT * FROM RECYCLEBIN;
```

Quando você elimina um usuário, os objetos que ele possui não são colocados na lixeira, e os objetos da lixeira são expurgados.

É possível expurgar a lixeira com a seguinte instrução:

```
PURGE RECYCLEBIN;
```

Tabelas Externas



ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Tabelas Externas

Uma tabela externa é uma tabela somente para leitura cujos metadados são armazenados no banco de dados, embora os dados sejam armazenados fora do banco de dados. Pense nessa definição de tabela externa como uma view que é usada para executar consultas SQL em dados externos sem carregá-los primeiramente no banco de dados. Os dados de tabela externa podem ser consultados e unidos diretamente e em paralelo sem que os dados externos sejam carregados primeiramente no banco de dados. Você pode usar SQL, PL/SQL e Java para consultar os dados de uma tabela externa.

A principal diferença entre as tabelas externas e as tabelas comuns é que as tabelas organizadas externamente são somente para leitura. Não é possível executar operações DML, e nenhum índice pode ser criado nessas tabelas. No entanto, é possível criar uma tabela externa e, assim, descarregar dados usando o comando `CREATE TABLE AS SELECT`.

O Oracle Server fornece dois drivers principais de acesso a tabelas externas. Um deles, o driver de acesso de carregador ou `ORACLE_LOADER`, permite ler os dados de arquivos externos cujo formato possa ser interpretado pelo utilitário `SQL*Loader`. Observe que nem todas as funções do `SQL*Loader` são suportadas em tabelas externas.

Tabelas Externas (continuação)

O driver de acesso ORACLE_DATAPUMP pode ser usado para importar e exportar dados usando um formato que independe de plataforma. O driver de acesso ORACLE_DATAPUMP grava as linhas de uma instrução SELECT para serem carregadas em uma tabela externa como parte de uma instrução CREATE TABLE ... ORGANIZATION EXTERNAL ... AS SELECT. Em seguida, você poderá usar SELECT para ler os dados do arquivo de dados. Também é possível criar uma definição de tabela externa em outro sistema e usar esse arquivo de dados. Com isso, é possível mover os dados entre bancos de dados Oracle.

Criando um Diretório para a Tabela Externa

Crie um objeto DIRECTORY que corresponda ao diretório no sistema de arquivos em que a origem dos dados externos reside.

```
CREATE OR REPLACE DIRECTORY emp_dir  
AS '/.../emp_dir';  
  
GRANT READ ON DIRECTORY emp_dir TO hr;
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Exemplo de Criação de uma Tabela Externa

Use a instrução `CREATE DIRECTORY` para criar um objeto de diretório. Um objeto de diretório especifica um apelido para um diretório no sistema de arquivos do servidor em que reside uma origem de dados externos. Você pode usar nomes de diretórios quando fizer referência a uma origem de dados externos, em vez de codificar o nome do caminho do sistema operacional, possibilitando uma maior flexibilidade de gerenciamento de arquivos.

É preciso ter os privilégios de sistema `CREATE ANY DIRECTORY` para criar diretórios. Ao criar um diretório, você recebe automaticamente o privilégio de objeto `READ` e `WRITE` e pode conceder privilégios `READ` e `WRITE` a outros usuários e atribuições. O DBA (administrador de banco de dados) também pode conceder com `grant` esses privilégios a outros usuários e atribuições.

Um usuário precisa de privilégios `READ` para todos os diretórios usados nas tabelas externas a serem acessadas e de privilégios `WRITE` para os diretórios em que estão sendo usados arquivos de log, de registros incorretos e de descarte.

Além disso, é necessário um privilégio `WRITE` quando o framework de tabela externa está sendo usado para descarregar dados.

O Oracle também fornece o tipo `ORACLE_DATAPUMP`, com o qual é possível descarregar dados (ou seja, ler os dados de uma tabela do banco de dados e inseri-los em uma tabela externa) e, em seguida, recarregá-los em um banco de dados Oracle. Trata-se de uma operação única que pode ser feita quando a tabela é criada. Após a criação e o preenchimento inicial da tabela, não será possível atualizar, inserir nem deletar linhas.

Exemplo de Criação de uma Tabela Externa (continuação)

Sintaxe

```
CREATE [OR REPLACE] DIRECTORY AS 'path_name';
```

Na sintaxe:

| | |
|-------------|--|
| OR REPLACE | Especifique OR REPLACE para recriar o objeto do banco de dados do diretório, caso já exista. Você pode usar esta cláusula para alterar a definição de um diretório existente sem eliminar, recriar e conceder com grant privilégios de objeto de banco de dados concedidos anteriormente no diretório. Os usuários aos quais foram anteriormente concedidos privilégios em um diretório redefinido podem continuar acessando o diretório sem precisar que os privilégios sejam novamente concedidos. |
| directory | Especifique o nome do objeto de diretório a ser criado. O tamanho máximo do nome do diretório é de 30 bytes. Não é possível qualificar um objeto de diretório com um nome de esquema. |
| 'path_name' | Especifique o nome de caminho completo do diretório do sistema operacional considerando que o nome de caminho faz distinção entre maiúsculas e minúsculas. |

A sintaxe para usar o driver de acesso ORACLE_DATAPUMP é a seguinte:

```
CREATE TABLE extract_emps
ORGANIZATION EXTERNAL (TYPE ORACLE_DATAPUMP
                        DEFAULT DIRECTORY ...
                        ACCESS PARAMETERS (... )
                        LOCATION (... )
                        PARALLEL 4
                        REJECT LIMIT UNLIMITED
AS
SELECT * FROM ...;
```

Criando uma Tabela Externa

```
CREATE TABLE <table_name>
  ( <col_name> <datatype>, ... )
  ORGANIZATION EXTERNAL
    (TYPE <access_driver_type>
      DEFAULT DIRECTORY <directory_name>
      ACCESS PARAMETERS
        (... ) )
      LOCATION ('<location_specifier>') )
  REJECT LIMIT [0 | <number> | UNLIMITED];
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Criando uma Tabela Externa

Crie tabelas externas usando a cláusula `ORGANIZATION EXTERNAL` da instrução `CREATE TABLE`. Na verdade, em vez de criar uma tabela, você estará criando metadados no dicionário de dados que permitem acessar dados externos. A cláusula `ORGANIZATION` permite especificar a ordem na qual as linhas de dados da tabela são armazenadas. Para indicar que se trata de uma tabela somente para leitura localizada fora do banco de dados, especifique `EXTERNAL` na cláusula `ORGANIZATION`. Observe que os arquivos externos já deverão existir fora do banco de dados.

`TYPE access_driver_type` indica o driver de acesso da tabela externa. O driver de acesso é a API que interpreta os dados externos para o banco de dados. Se você não especificar `TYPE`, o Oracle usará o driver de acesso default, `ORACLE_LOADER`. A outra opção é `ORACLE_DATAPUMP`.

Você usa a cláusula `DEFAULT DIRECTORY` para especificar um ou mais objetos de banco de dados de diretório Oracle que correspondem a diretórios no sistema de arquivos nos quais as origens de dados externos podem residir.

A cláusula opcional `ACCESS PARAMETERS` permite que você designe valores aos parâmetros do driver de acesso específico para essa tabela externa.

Criando uma Tabela Externa (continuação)

Use a cláusula `LOCATION` para especificar um localizador externo para cada origem de dados externos. Em geral, mas não necessariamente, o `<location_specifier>` é um arquivo.

A cláusula `REJECT LIMIT` permite especificar quantos erros de conversão podem ocorrer durante uma consulta de dados externos antes que um erro Oracle seja retornado e a consulta seja abortada. O valor default é 0.

Criando uma Tabela Externa Usando ORACLE_LOADER

```
CREATE TABLE oldemp (  
  fname char(25), lname CHAR(25))  
  ORGANIZATION EXTERNAL  
  (TYPE ORACLE_LOADER  
   DEFAULT DIRECTORY emp_dir  
   ACCESS PARAMETERS  
   (RECORDS DELIMITED BY NEWLINE  
    NOBADFILE  
    NOLOGFILE  
    FIELDS TERMINATED BY ','  
    (fname POSITION ( 1:20) CHAR,  
     lname POSITION (22:41) CHAR))  
   LOCATION ('emp.dat'))  
  PARALLEL 5  
  REJECT LIMIT 200;  
Table created.
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Exemplo de Criação de uma Tabela Externa Usando o Driver de Acesso ORACLE_LOADER

Suponha que exista um arquivo sem formatação com os seguintes registros:

```
10,jones,11-Dec-1934  
20,smith,12-Jun-1972
```

Os registros são delimitados por novas linhas e os campos são encerrados por uma vírgula (,). O nome do arquivo é: /emp_dir/emp.dat

Para converter esse arquivo na origem de dados para uma tabela externa cujos metadados residirão no banco de dados, siga estas etapas:

1. Crie um objeto de diretório emp_dir da seguinte maneira:
CREATE DIRECTORY emp_dir AS '/emp_dir' ;
2. Execute o comando CREATE TABLE mostrado no slide.

O exemplo do slide ilustra a especificação para criar uma tabela externa para o arquivo:

```
/emp_dir/emp.dat
```

Exemplo de Criação de uma Tabela Externa Usando o Driver de Acesso ORACLE_LOADER (continuação)

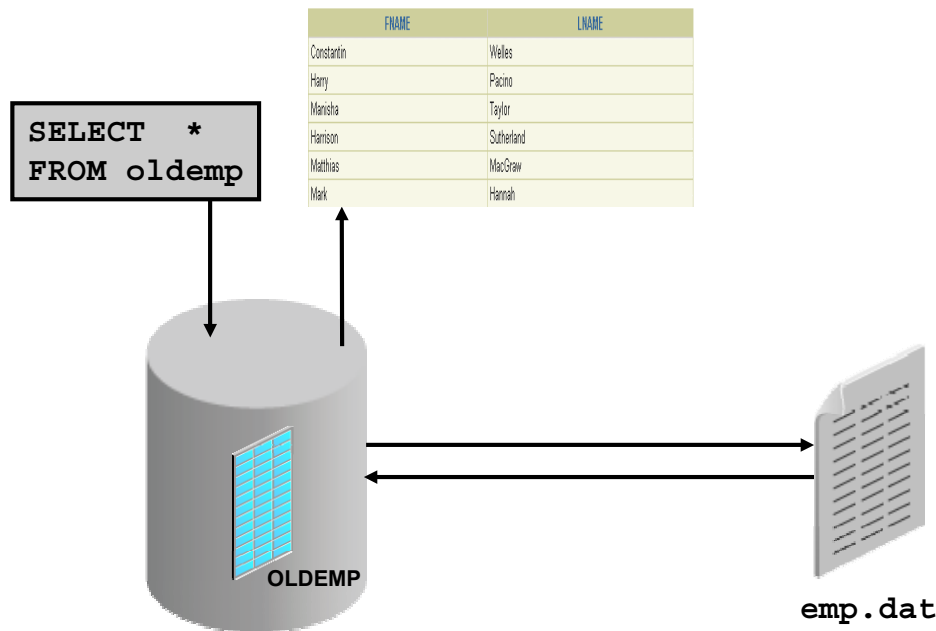
No exemplo, a especificação TYPE só é fornecida para ilustrar seu uso. ORACLE_LOADER é o driver de acesso default, caso nenhum outro seja especificado. A opção ACCESS PARAMETERS fornece os valores para os parâmetros do driver de acesso específico, os quais são interpretados por esse driver, e não pelo servidor Oracle.

A cláusula PARALLEL permite que cinco servidores de execução paralela varram simultaneamente as origens de dados externos, isto é, os arquivos dessas origens, durante a execução da instrução INSERT INTO TABLE. Por exemplo, se PARALLEL=5 tiver sido especificado, mais de um servidor de execução paralela poderá estar trabalhando em uma origem de dados. Como as tabelas externas podem ser bem grandes, por motivos de desempenho, é aconselhável especificar a cláusula PARALLEL ou uma dica paralela para a consulta.

A cláusula REJECT LIMIT especifica que, se ocorrerem mais de 200 erros de conversão durante uma consulta aos dados externos, a consulta será abortada e será retornado um erro. Esses erros de conversão poderão surgir quando o driver de acesso tentar transformar os dados do arquivo de dados para coincidirem com a definição de tabela externa.

Depois que o comando CREATE TABLE for executado com sucesso, a tabela externa OLDEMP poderá ser descrita e consultada com uma tabela relacional.

Consultando Tabelas Externas



ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Consultando Tabelas Externas

Uma tabela externa não descreve os dados armazenados no banco de dados. Ela também não descreve como os dados são armazenados na origem externa. Na verdade, ela descreve como a camada da tabela externa precisa apresentar os dados ao servidor. O driver de acesso e a camada da tabela externa são responsáveis pela execução das transformações necessárias nos dados contidos no arquivo de dados para que eles correspondam à definição da tabela externa.

Ao acessar os dados de uma origem externa, o servidor de banco de dados chama o driver de acesso apropriado para obter os dados dessa origem na forma esperada pelo servidor.

É importante lembrar que a descrição dos dados da origem de dados é separada da definição da tabela externa. O arquivo de origem pode conter um número maior ou menor de campos que o número de colunas existentes na tabela. Além disso, os tipos de dados dos campos da origem de dados podem ser diferentes dos tipos de dados das colunas da tabela. O driver de acesso é responsável por assegurar que os dados da origem de dados sejam processados para corresponderem à definição da tabela externa.

Sumário

Nesta lição, você aprendeu a:

- **Adicionar constraints**
- **Criar índices**
- **Criar uma constraint de chave primária usando um índice**
- **Criar índices usando a instrução `CREATE TABLE`**
- **Criar índices baseados em função**
- **Eliminar colunas e definir uma coluna como `UNUSED`**
- **Executar operações `FLASHBACK`**
- **Criar e usar tabelas externas**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Sumário

Altere tabelas para adicionar ou modificar colunas ou constraints. Crie índices e índices baseados em função usando a instrução `CREATE INDEX`. Elimine as colunas não utilizadas. Use mecanismos de `FLASHBACK` para restaurar tabelas. Use a cláusula `external_table` para criar uma tabela externa, que é uma tabela somente para leitura, sendo que seus metadados são armazenados no banco de dados, mas seus dados são armazenados fora do banco de dados. Use tabelas externas para consultar dados sem antes carregá-las no banco de dados. Nomeie os índices da coluna `PRIMARY KEY` ao criar a tabela com a instrução `CREATE TABLE`.

Exercício 2: Visão Geral

Este exercício aborda os seguintes tópicos:

- **Alterando tabelas**
- **Adicionando colunas**
- **Eliminando colunas**
- **Criando índices**
- **Criando tabelas externas**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Exercício 2: Visão Geral

Neste exercício, você usará o comando `ALTER TABLE` para modificar colunas e adicionar constraints. Você usará o comando `CREATE INDEX` para criar índices ao criar uma tabela, juntamente com o comando `CREATE TABLE`. Você criará tabelas externas. Você eliminará colunas e usará a operação `FLASHBACK`.

Exercício 2

1. Crie a tabela DEPT2 com base no gráfico de instâncias de tabela a seguir. Inclua a sintaxe em um script denominado lab02_01.sql e execute a instrução do script para criar a tabela. Confirme a criação da tabela.

| | | |
|---------------------|--------|----------|
| Column Name | ID | NAME |
| Key Type | | |
| Nulls/Unique | | |
| FK Table | | |
| FK Column | | |
| Data type | NUMBER | VARCHAR2 |
| Length | 7 | 25 |

| Name | Null? | Type |
|------|-------|--------------|
| ID | | NUMBER(7) |
| NAME | | VARCHAR2(25) |

2. Preencha a tabela DEPT2 com dados da tabela DEPARTMENTS. Inclua apenas as colunas de que necessita.
3. Crie a tabela EMP2 com base no gráfico de instâncias de tabela a seguir. Inclua a sintaxe em um script denominado lab02_03.sql e execute a instrução do script para criar a tabela. Confirme a criação da tabela.

| | | | | |
|---------------------|--------|-----------|------------|---------|
| Column Name | ID | LAST_NAME | FIRST_NAME | DEPT_ID |
| Key Type | | | | |
| Nulls/Unique | | | | |
| FK Table | | | | |
| FK Column | | | | |
| Data type | NUMBER | VARCHAR2 | VARCHAR2 | NUMBER |
| Length | 7 | 25 | 25 | 7 |

| Name | Null? | Type |
|------------|-------|--------------|
| ID | | NUMBER(7) |
| LAST_NAME | | VARCHAR2(25) |
| FIRST_NAME | | VARCHAR2(25) |
| DEPT_ID | | NUMBER(7) |

Exercício 2 (continuação)

4. Modifique a tabela EMP2 para que aceite sobrenomes mais longos de funcionários. Confirme a modificação.

| Name | Null? | Type |
|------------|-------|--------------|
| ID | | NUMBER(7) |
| LAST_NAME | | VARCHAR2(50) |
| FIRST_NAME | | VARCHAR2(25) |
| DEPT_ID | | NUMBER(7) |

5. Confirme se as tabelas DEPT2 e EMP2 foram armazenadas no dicionário de dados. (**Dica:** USER_TABLES)

| TABLE_NAME |
|------------|
| DEPT2 |
| EMP2 |

6. Crie a tabela EMPLOYEES2 com base na estrutura da tabela EMPLOYEES. Inclua apenas as colunas EMPLOYEE_ID, FIRST_NAME, LAST_NAME, SALARY e DEPARTMENT_ID. Nomeie as colunas da nova tabela como ID, FIRST_NAME, LAST_NAME, SALARY e DEPT_ID, respectivamente.
7. Elimine a tabela EMP2.
8. Veja se a tabela está na lixeira.

| ORIGINAL_NAME | OPERATION | DROPTIME |
|---------------|-----------|---------------------|
| EMP2 | DROP | 2004-02-13:10:40:22 |

9. Cancele a eliminação da tabela EMP2.

| Name | Null? | Type |
|------------|-------|--------------|
| ID | | NUMBER(7) |
| LAST_NAME | | VARCHAR2(50) |
| FIRST_NAME | | VARCHAR2(25) |
| DEPT_ID | | NUMBER(7) |

10. Elimine a coluna FIRST_NAME da tabela EMPLOYEES2. Confirme a modificação verificando a descrição da tabela.
11. Na tabela EMPLOYEES2, marque a coluna DEPT_ID como UNUSED. Confirme a modificação verificando a descrição da tabela.
12. Elimine todas as colunas UNUSED da tabela EMPLOYEES2. Confirme a modificação verificando a descrição da tabela.
13. Adicione uma constraint PRIMARY KEY em nível de tabela para a tabela EMP2 na coluna ID. A constraint deve ser nomeada durante a criação. Nomeie-a como my_emp_id_pk.

Exercício 2 (continuação)

14. Crie uma constraint PRIMARY KEY para a tabela DEPT2 usando a coluna ID. A constraint deve ser nomeada durante a criação. Nomeie-a como my_dept_id_pk.
15. Adicione uma referência de chave estrangeira à tabela EMP2 que garante que o funcionário não foi designado para um departamento inexistente. Nomeie a constraint como my_emp_dept_id_fk.
16. Confirme a adição das constraints consultando a view USER_CONSTRAINTS. Anote os tipos e os nomes das constraints.

| CONSTRAINT_NAME | CON |
|-------------------|-----|
| MY_DEPT_ID_PK | P |
| MY_EMP_ID_PK | P |
| MY_EMP_DEPT_ID_FK | R |

17. Exiba os nomes e os tipos de objetos da view de dicionário de dados USER_OBJECTS para as tabelas EMP2 e DEPT2. Observe que foram criadas novas tabelas e um novo índice.

Se tiver tempo, faça o seguinte exercício:

18. Modifique a tabela EMP2. Adicione uma coluna COMMISSION do tipo de dados NUMBER, precisão 2, escala 2. Adicione uma constraint à coluna COMMISSION que garanta um valor de comissão maior do que zero.
19. Elimine as tabelas EMP2 e DEPT2 de modo que não possam ser restauradas. Verifique a lixeira.
20. Crie a tabela DEPT_NAMED_INDEX com base no gráfico de instâncias de tabela a seguir. Nomeie o índice para a coluna PRIMARY KEY como DEPT_PK_IDX.

| Nome da Coluna | Deptno | Dname |
|----------------|--------|----------|
| Chave primária | Sim | |
| Tipo de Dados | Número | VARCHAR2 |
| Length | 4 | 30 |

3

Manipulando Grandes Conjuntos de Dados

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Objetivos

Ao concluir esta lição, você será capaz de:

- **Manipular dados usando subconsultas**
- **Descrever os recursos de inserções em várias tabelas**
- **Usar os seguintes tipos de inserções em várias tabelas:**
 - **INSERT Incondicional**
 - **INSERT de Criação de Pivô**
 - **ALL INSERT Condicional**
 - **FIRST INSERT Condicional**
- **Intercalar linhas em uma tabela**
- **Controlar as alterações de dados durante um período**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Objetivos

Nesta lição, você aprenderá a manipular os dados do banco de dados Oracle usando subconsultas. Você também conhecerá as instruções de inserção em várias tabelas e a instrução MERGE, além de aprender a controlar as alterações feitas no banco de dados.

Usando Subconsultas para Manipular Dados

É possível usar subconsultas em instruções DML para:

- **Copiar dados de uma tabela para outra**
- **Recuperar dados de uma view em linha**
- **Atualizar dados em uma tabela com base nos valores de outra tabela**
- **Deletar linhas de uma tabela com base nas linhas de outra tabela**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Usando Subconsultas para Manipular Dados

As subconsultas podem ser usadas para recuperar dados a partir de uma tabela usada como entrada para fazer um INSERT em uma tabela diferente. Desse modo, você pode copiar facilmente grandes volumes de dados de uma tabela para outra com uma única instrução SELECT. Da mesma forma, você pode usar subconsultas para fazer atualizações e deleções em massa, incluindo-as na cláusula WHERE das instruções UPDATE e DELETE. Também é possível usar subconsultas na cláusula FROM de uma instrução SELECT. Esse processo se chama view em linha.

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 EMPL3
SELECT *
FROM employees;
```

Para obter mais informações, consulte o manual *Oracle Database 10g SQL Reference*.

Inserção Usando uma Subconsulta como Destino

```
INSERT INTO
    (SELECT employee_id, last_name,
             email, hire_date, job_id, salary,
             department_id
     FROM   empl3
     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.

Inserção Usando uma Subconsulta como Destino

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 nem omitir um valor de uma coluna NOT NULL obrigatória.

Essa aplicação de subconsultas ajuda a evitar que seja necessário criar uma view apenas para executar uma inserção.

Inserção Usando uma Subconsulta como Destino

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 |
|-------------|-----------|----------|-----------|----------|--------|---------------|
| 120 | Weiss | MWEISS | 18-JUL-96 | ST_MAN | 8000 | 50 |
| 121 | Fripp | AFRIPP | 10-APR-97 | ST_MAN | 8200 | 50 |
| 122 | Kaufling | PKAUFLIN | 01-MAY-95 | ST_MAN | 7900 | 50 |
| ... | | | | | | |
| 193 | Everett | BEVERETT | 03-MAR-97 | SH_CLERK | 3900 | 50 |
| 194 | McCain | SMCCAIN | 01-JUL-98 | SH_CLERK | 3200 | 50 |
| 195 | Jones | VJONES | 17-MAR-99 | SH_CLERK | 2800 | 50 |
| 196 | Walsh | AWALSH | 24-APR-98 | SH_CLERK | 3100 | 50 |
| 197 | Feeney | KFEENEY | 23-MAY-98 | SH_CLERK | 3000 | 50 |
| 198 | OConnell | DOCONNEL | 21-JUN-99 | SH_CLERK | 2600 | 50 |
| 199 | Grant | DGRANT | 13-JAN-00 | SH_CLERK | 2600 | 50 |
| 99999 | Taylor | DTAYLOR | 07-JUN-99 | ST_CLERK | 5000 | 50 |

46 rows selected.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Inserção Usando uma Subconsulta como Destino (continuação)

O exemplo mostra os resultados da subconsulta usada para identificar a tabela para a instrução INSERT.

Recuperando Dados com uma Subconsulta como Origem

```
SELECT  a.last_name, a.salary,
        a.department_id, b.salavg
FROM    employees a, (SELECT  department_id,
                             AVG(salary) salavg
                     FROM    employees
                     GROUP BY department id) b
WHERE   a.department_id = b.department_id
AND     a.salary > b.salavg;
```

| LAST_NAME | SALARY | DEPARTMENT_ID | SALAVG |
|-----------|--------|---------------|------------|
| King | 24000 | 90 | 19333.3333 |
| Hunold | 9000 | 60 | 5760 |
| Ernst | 6000 | 60 | 5760 |
| Greenberg | 12000 | 100 | 8600 |
| Faviet | 9000 | 100 | 8600 |
| Raphaely | 11000 | 30 | 4150 |
| Weiss | 8000 | 50 | 3475.55556 |
| Fripp | 8200 | 50 | 3475.55556 |

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Recuperando Dados Usando uma Subconsulta como Origem

Você pode usar uma subconsulta na cláusula FROM da instrução SELECT, que é muito semelhante à forma como as views são usadas. Uma subconsulta na cláusula FROM de uma instrução SELECT também é chamada de view *em linha*. Uma subconsulta na cláusula FROM de uma instrução SELECT define uma origem de dados apenas para essa instrução SELECT específica. O exemplo do slide exibe os sobrenomes dos funcionários, os salários, os números dos departamentos e os salários médios de todos os funcionários que recebem mais que o salário médio dos respectivos departamentos. A subconsulta na cláusula FROM é denominada b, e a consulta exterior faz referência à coluna SALAVG usando esse apelido.

Atualizando Duas Colunas com uma Subconsulta

Atualize o cargo e o salário do funcionário 114 para corresponder ao cargo e ao salário do funcionário 205.

```
UPDATE emp13
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 nas instruções UPDATE para atualizar linhas de uma tabela com base em valores de outra tabela.

```
UPDATE empl3
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 EMPL3 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.

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 emp13
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 a string "Public". 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.

Usando a Palavra-Chave WITH CHECK OPTION em Instruções DML

- Uma subconsulta é usada para identificar a tabela e as colunas da instrução DML.
- A palavra-chave WITH CHECK OPTION impede a alteração de linhas que não estão na subconsulta.

```
INSERT INTO (SELECT employee_id, last_name, email,
                hire_date, job_id, salary
            FROM   emp13
            WHERE  department_id = 50
            WITH CHECK OPTION)
VALUES (99998, 'Smith', 'JSMITH',
        TO_DATE('07-JUN-99', 'DD-MON-RR'),
        'ST_CLERK', 5000);
INSERT INTO
*
```

ERROR at line 1:
ORA-01402: view WITH CHECK OPTION where-clause violation

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

A Palavra-Chave WITH CHECK OPTION

Especifique WITH CHECK OPTION para indicar que, se a subconsulta for usada no lugar de uma tabela em uma instrução INSERT, UPDATE ou DELETE, não serão permitidas alterações nessa tabela que produzam linhas não incluídas na subconsulta.

No exemplo mostrado, a palavra-chave WITH CHECK OPTION é usada. A subconsulta identifica linhas que estão no departamento 50, mas o ID do departamento não está na lista SELECT e não tem um valor especificado na lista VALUES. A inserção dessa linha resulta em um ID de departamento nulo, que não está na subconsulta.

Visão Geral do Recurso de Default Explícito

- Com o recurso de default explícito, é possível usar a palavra-chave **DEFAULT** como um valor de coluna onde se deseja especificar o valor default de coluna.
- Esse recurso é incluído para manter a compatibilidade com o padrão **SQL:1999**.
- O recurso permite ao usuário controlar onde e quando o valor default deve ser aplicado aos dados.
- É possível usar defaults explícitos em instruções **INSERT** e **UPDATE**.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Defaults Explícitos

É possível usar a palavra-chave **DEFAULT** em instruções **INSERT** e **UPDATE** para identificar um valor de coluna default. Se não houver um valor default, será usado um valor nulo.

A opção **DEFAULT** dispensa a codificação do valor default nos programas e a consulta ao dicionário para encontrá-lo, como se fazia antes da introdução deste recurso. A codificação do valor default é um problema quando ele se altera porque o código conseqüentemente precisa ser alterado. O acesso ao dicionário geralmente não é feito em um programa de aplicação. Portanto, trata-se de um recurso muito importante.

Usando Valores Default Explícitos

- **DEFAULT com INSERT:**

```
INSERT INTO deptm3  
  (department_id, department_name, manager_id)  
VALUES (300, 'Engineering', DEFAULT);
```

- **DEFAULT com UPDATE:**

```
UPDATE deptm3  
SET manager_id = DEFAULT  
WHERE department_id = 10;
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Usando Valores Default Explícitos

Especifique DEFAULT para definir a coluna para o valor especificado anteriormente como o seu valor default. Se não tiver sido especificado um valor default para a coluna correspondente, o servidor Oracle definirá a coluna como nula.

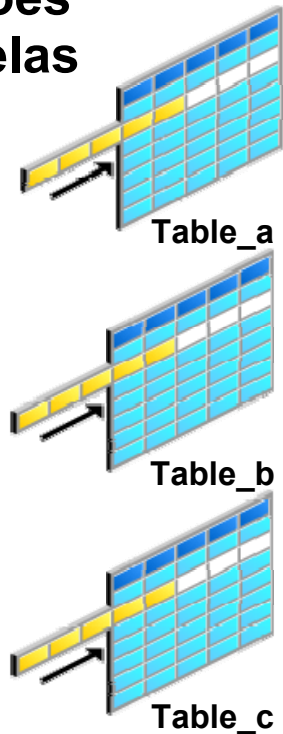
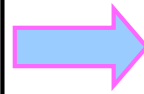
No primeiro exemplo do slide, a instrução INSERT usa um valor default para a coluna MANAGER_ID. Se não houver um valor default definido para a coluna, um valor nulo será inserido.

O segundo exemplo usa a instrução UPDATE para definir a coluna MANAGER_ID com um valor default para o departamento 10. Se nenhum valor default for definido para a coluna, o valor será alterado para nulo.

Observação: Ao criar uma tabela, você poderá especificar um valor default para uma coluna. Esse assunto será abordado na lição intitulada "Criando e Gerenciando Tabelas".

Visão Geral de Instruções INSERT em Várias Tabelas

```
INSERT ALL  
  INTO table_a VALUES (...,...,...)  
  INTO table_b VALUES (...,...,...)  
  INTO table_c VALUES (...,...,...)  
SELECT ...  
FROM sourcetab  
WHERE ...;
```



ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Visão Geral de Instruções INSERT em Várias Tabelas

Em uma instrução INSERT em várias tabelas, são inseridas as linhas calculadas derivadas das linhas retornadas da avaliação de uma subconsulta em uma ou mais tabelas.

As instruções INSERT em várias tabelas podem desempenhar um papel muito útil em um cenário de data warehouse. Carregue o data warehouse regularmente para que ele atenda ao propósito de facilitar a análise de negócios. Para isso, é necessário extrair e copiar os dados de um ou mais sistemas operacionais para o warehouse. O processo de extração de dados do sistema de origem e de adição desses dados ao data warehouse é comumente chamado de ETL (Extraction, Transformation and Loading), que significa extração, transformação e carga.

Durante a extração, os dados desejados precisam ser identificados e extraídos de várias origens distintas, tais como aplicações e sistemas de banco de dados. Depois da extração, os dados precisam ser transportados fisicamente para o sistema de destino ou para um sistema intermediário para processamento adicional. Dependendo do meio de transporte escolhido, é possível realizar algumas transformações durante esse processo. Por exemplo, uma instrução SQL que acesse diretamente um destino remoto através de um gateway pode concatenar duas colunas como parte da instrução SELECT.

Após a carga dos dados no banco de dados Oracle, é possível executar transformações nos dados usando operações SQL. Uma instrução INSERT em várias tabelas é uma das técnicas para implementar transformações de dados SQL.

Visão Geral de Instruções INSERT em Várias Tabelas

- A instrução **INSERT...SELECT** pode ser usada para inserir linhas em várias tabelas como parte de uma única instrução DML.
- Várias instruções **INSERT** podem ser usadas em sistemas de data warehouse para transferir dados de uma ou mais origens operacionais para um conjunto de tabelas de destino.
- Elas permitem uma melhoria significativa no desempenho em relação a:
 - Uma única instrução DML x várias instruções **INSERT...SELECT**
 - Uma única instrução DML x um procedimento para executar várias inserções usando a sintaxe **IF . . . THEN**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Visão Geral de Instruções **INSERT** em Várias Tabelas (Continuação)

As instruções **INSERT** em várias tabelas oferecem os benefícios da instrução **INSERT . . . SELECT** quando há várias tabelas como destino. Antes do Banco de Dados Oracle9i, para usar essa funcionalidade, era necessário lidar com n instruções **INSERT . . . SELECT** independentes, processando, assim, os mesmos dados-fonte n vezes e aumentando a carga de trabalho de transformação n vezes.

Como ocorre com a instrução **INSERT . . . SELECT** existente, a nova instrução pode ser paralelizada e usada com o mecanismo de carga direta para garantir um melhor desempenho.

Agora os registros de qualquer fluxo de entrada, como, por exemplo, uma tabela de banco de dados não relacional, podem ser convertidos em vários registros para um ambiente de tabela de banco de dados mais relacional. Para implementar essa funcionalidade opcionalmente, foi solicitado que você criasse várias instruções **INSERT**.

Tipos de Instruções INSERT em Várias Tabelas

Os diversos tipos de instruções INSERT em várias tabelas são:

- **INSERT Incondicional**
- **ALL INSERT Condicional**
- **FIRST INSERT Condicional**
- **INSERT de Criação de Pivô**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Tipos de Instruções INSERT em Várias Tabelas

Os tipos de instruções INSERT em várias tabelas são:

- INSERT Incondicional
- ALL INSERT Condicional
- FIRST INSERT Condicional
- INSERT de Criação de Pivô

Use cláusulas distintas para indicar o tipo de instrução INSERT a ser executada.

Instruções INSERT em Várias Tabelas

- **Sintaxe**

```
INSERT [ALL] [conditional_insert_clause]
[insert_into_clause values_clause] (subquery)
```

- **conditional_insert_clause**

```
[ALL] [FIRST]
[WHEN condition THEN] [insert_into_clause values_clause]
[ELSE] [insert_into_clause values_clause]
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Instruções INSERT em Várias Tabelas

O slide exibe o formato genérico para as instruções INSERT em várias tabelas.

INSERT Incondicional: ALL into_clause

Especifique ALL seguido por diversas insert_into_clauses para executar uma inserção incondicional em várias tabelas. O servidor Oracle executa cada insert_into_clause uma vez por linha retornada pela subconsulta.

INSERT Condicional: conditional_insert_clause

Especifique conditional_insert_clause para executar uma instrução INSERT condicional em várias tabelas. O servidor Oracle filtra cada insert_into_clause pela condição WHEN correspondente, que determina se essa insert_into_clause será executada. Uma única instrução INSERT em várias tabelas pode conter até 127 cláusulas WHEN.

INSERT Condicional: ALL

Se você especificar ALL, o servidor Oracle avaliará cada cláusula WHEN independentemente dos resultados da avaliação de qualquer outra cláusula WHEN. Para cada cláusula WHEN cuja condição é avaliada como verdadeira, o servidor Oracle executa a lista de cláusulas INTO correspondente.

Instruções INSERT em Várias Tabelas (continuação)

INSERT Condicional: FIRST

Se você especificar FIRST, o servidor Oracle avaliará cada cláusula WHEN na ordem em que aparece na instrução. Se a primeira cláusula WHEN for avaliada como verdadeira, o servidor Oracle executará a cláusula INTO correspondente e ignorará as cláusulas WHEN subsequentes relativas a essa linha.

INSERT Condicional: Cláusula ELSE

Para uma linha específica, se nenhuma cláusula WHEN for avaliada como verdadeira:

- Caso você tenha especificado uma cláusula ELSE, o servidor Oracle executará a lista de cláusulas INTO associada à cláusula ELSE.
- Caso você não especifique uma cláusula ELSE, o servidor Oracle não executará nenhuma ação para essa linha.

Restrições às Instruções INSERT em Várias Tabelas

- Você pode executar instruções INSERT em várias tabelas apenas em tabelas, mas não em views nem em views materializadas.
- Não é possível executar uma instrução INSERT em várias tabelas em uma tabela remota.
- Não é possível especificar uma expressão de coleta de tabelas ao executar uma instrução INSERT em várias tabelas.
- Em uma instrução INSERT em várias tabelas, não é possível combinar todas as `insert_into_clauses` para especificar mais de 999 colunas de destino.

INSERT ALL Incondicional

- **Selecione os valores de EMPLOYEE_ID, HIRE_DATE, SALARY e MANAGER_ID na tabela EMPLOYEES para os funcionários cujo EMPLOYEE_ID é maior que 200.**
- **Insira esses valores nas tabelas SAL_HISTORY e MGR_HISTORY usando uma instrução INSERT em várias tabelas.**

```
INSERT ALL
  INTO sal_history VALUES (EMPID, HIREDATE, SAL)
  INTO mgr_history VALUES (EMPID, MGR, SAL)
  SELECT employee_id EMPID, hire_date HIREDATE,
         salary SAL, manager_id MGR
  FROM employees
  WHERE employee_id > 200;
8 rows created.
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

INSERT ALL Incondicional

O exemplo do slide insere linhas nas tabelas SAL_HISTORY e MGR_HISTORY.

A instrução SELECT recupera, na tabela EMPLOYEES, os detalhes sobre o ID do funcionário, a data de admissão, o salário e o ID do gerente dos funcionários cujo ID é maior que 200. Os detalhes sobre o ID do funcionário, a data de admissão e o salário são inseridos na tabela SAL_HISTORY. Os detalhes sobre o ID do funcionário, o ID do gerente e o salário são inseridos na tabela MGR_HISTORY.

Essa instrução INSERT é denominada INSERT incondicional, pois não são aplicadas outras restrições às linhas recuperadas pela instrução SELECT. Todas as linhas recuperadas pela instrução SELECT são inseridas nas duas tabelas, SAL_HISTORY e MGR_HISTORY. A cláusula VALUES nas instruções INSERT especifica as colunas da instrução SELECT que precisam ser inseridas em cada uma das tabelas. Cada linha retornada pela instrução SELECT resulta em duas inserções, uma na tabela SAL_HISTORY e outra na tabela MGR_HISTORY.

É possível interpretar as 8 linhas criadas e retornadas como um total de oito inserções executadas nas tabelas-base, SAL_HISTORY e MGR_HISTORY.

INSERT ALL Condicional

- **Selecione os valores de EMPLOYEE_ID, HIRE_DATE, SALARY e MANAGER_ID na tabela EMPLOYEES para os funcionários cujo EMPLOYEE_ID é maior que 200.**
- **Se o valor de SALARY for maior que \$10.000, insira esse valor na tabela SAL_HISTORY usando uma instrução INSERT condicional em várias tabelas.**
- **Se o valor de MANAGER_ID for maior que 200, insira esse valor na tabela MGR_HISTORY usando uma instrução INSERT condicional em várias tabelas.**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

INSERT ALL Condicional

As orientações para criar uma instrução INSERT ALL condicional estão especificadas no slide. A solução para esse problema está indicada na próxima página.

INSERT ALL Condicional

```
INSERT ALL
  WHEN SAL > 10000 THEN
    INTO sal_history VALUES (EMPID, HIREDATE, SAL)
  WHEN MGR > 200 THEN
    INTO mgr_history VALUES (EMPID, MGR, SAL)
  SELECT employee_id EMPID, hire_date HIREDATE,
         salary SAL, manager_id MGR
  FROM   employees
  WHERE  employee_id > 200;
4 rows created.
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

INSERT ALL Condicional (continuação)

O exemplo do slide é semelhante ao exemplo do slide anterior, pois ele insere linhas nas tabelas SAL_HISTORY e MGR_HISTORY. A instrução SELECT recupera, na tabela EMPLOYEES, os detalhes sobre o ID do funcionário, a data de admissão, o salário e o ID do gerente dos funcionários cujo ID é maior que 200. Os detalhes sobre o ID do funcionário, a data de admissão e o salário são inseridos na tabela SAL_HISTORY. Os detalhes sobre o ID do funcionário, o ID do gerente e o salário são inseridos na tabela MGR_HISTORY.

Essa instrução INSERT é denominada ALL INSERT condicional, pois são aplicadas outras restrições às linhas recuperadas pela instrução SELECT. Das linhas recuperadas pela instrução SELECT, apenas aquelas cujo valor na coluna SAL é maior que 10.000 são inseridas na tabela SAL_HISTORY. Da mesma forma, apenas as linhas cujo valor na coluna MGR é maior que 200 são inseridas na tabela MGR_HISTORY.

Observe que, diferentemente do exemplo anterior, no qual oito linhas foram inseridas nas tabelas, neste exemplo, apenas quatro linhas são inseridas.

É possível interpretar as 4 linhas criadas e retornadas como um total de quatro operações INSERT executadas nas tabelas-base, SAL_HISTORY e MGR_HISTORY.

FIRST INSERT Condicional

- **Selecione DEPARTMENT_ID, SUM (SALARY) e MAX (HIRE_DATE) na tabela EMPLOYEES.**
- **Se o valor de SUM (SALARY) for maior que \$25.000, insira esse valor em SPECIAL_SAL usando uma instrução FIRST INSERT condicional em várias tabelas.**
- **Se a primeira cláusula WHEN for avaliada como verdadeira, as cláusulas WHEN subsequentes relativas a essa linha deverão ser ignoradas.**
- **Insira as linhas que não atenderem à primeira condição WHEN na tabela HIREDATE_HISTORY_00, HIREDATE_HISTORY_99 ou HIREDATE_HISTORY, com base no valor da coluna HIRE_DATE usando uma instrução INSERT condicional em várias tabelas.**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

FIRST INSERT Condicional

As orientações para criar uma instrução FIRST INSERT condicional estão especificadas no slide. A solução para esse problema está indicada na próxima página.

INSERT FIRST Condicional

```
INSERT FIRST
  WHEN SAL > 25000 THEN
    INTO special_sal VALUES (DEPTID, SAL)
  WHEN HIREDATE like ('%00%') THEN
    INTO hiredate_history_00 VALUES (DEPTID, HIREDATE)
  WHEN HIREDATE like ('%99%') THEN
    INTO hiredate_history_99 VALUES (DEPTID, HIREDATE)
  ELSE
    INTO hiredate_history VALUES (DEPTID, HIREDATE)
  SELECT department_id DEPTID, SUM(salary) SAL,
    MAX(hire_date) HIREDATE
  FROM employees
  GROUP BY department_id;
8 rows created.
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

INSERT FIRST Condicional (continuação)

O exemplo do slide insere linhas em mais de uma tabela, usando uma única instrução INSERT. A instrução SELECT recupera os detalhes sobre o ID, o salário total e a data de admissão máxima relativos a todos os departamentos da tabela EMPLOYEES.

Essa instrução INSERT é denominada FIRST INSERT condicional, pois é feita uma exceção para os departamentos cujo salário total é maior que \$25.000. A condição WHEN ALL > \$25.000 é avaliada primeiro. Se o salário total de um departamento for maior que \$25.000, o registro será inserido na tabela SPECIAL_SAL independentemente da data de admissão. Se a primeira cláusula WHEN for avaliada como verdadeira, o servidor Oracle executará a cláusula INTO correspondente e ignorará as cláusulas WHEN subsequentes relativas a essa linha.

Quando as linhas não atendem à primeira condição WHEN (WHEN SAL > 25.000), as outras condições são avaliadas exatamente como a instrução INSERT condicional, e os registros recuperados pela instrução SELECT são inseridos na tabela HIREDATE_HISTORY_00, HIREDATE_HISTORY_99 ou HIREDATE_HISTORY, com base no valor da coluna HIREDATE.

É possível interpretar as 8 linhas criadas e retornadas como um total de oito operações INSERT executadas nas tabelas-base, SPECIAL_SAL, HIREDATE_HISTORY_00, HIREDATE_HISTORY_99 e HIREDATE_HISTORY.

INSERT de Criação de Pivô

- Suponha que você receba um conjunto de registros de vendas de uma tabela de banco de dados não relacional, `SALES_SOURCE_DATA`, no seguinte formato:
`EMPLOYEE_ID, WEEK_ID, SALES_MON, SALES_TUE, SALES_WED, SALES_THUR, SALES_FRI`
- Você quer armazenar esses registros na tabela `SALES_INFO` em um formato relacional mais usado:
`EMPLOYEE_ID, WEEK, SALES`
- Com uma instrução `INSERT` de criação de pivô, converta o conjunto de registros de vendas da tabela de banco de dados não relacional em um formato relacional.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

INSERT de Criação de Pivô

A criação de pivô é uma operação na qual você precisa criar uma transformação de forma que cada registro de qualquer fluxo de entrada, como uma tabela de banco de dados não relacional, seja convertido em vários registros para um ambiente de tabela de banco de dados mais relacional.

Para solucionar o problema mencionado no slide, é preciso criar uma transformação para que cada registro da tabela de banco de dados não relacional original, `SALES_SOURCE_DATA`, seja convertido em cinco registros para a tabela `SALES_INFO` de data warehouse. Essa operação é geralmente chamada de *criação de pivô*.

As orientações para desenvolver uma instrução `INSERT` de criação de pivô estão especificadas no slide. A solução para esse problema está indicada na próxima página.

INSERT de Criação de Pivô

```
INSERT ALL
  INTO sales_info VALUES (employee_id, week_id, sales_MON)
  INTO sales_info VALUES (employee_id, week_id, sales_TUE)
  INTO sales_info VALUES (employee_id, week_id, sales_WED)
  INTO sales_info VALUES (employee_id, week_id, sales_THUR)
  INTO sales_info VALUES (employee_id, week_id, sales_FRI)
SELECT EMPLOYEE_ID, week_id, sales_MON, sales_TUE,
       sales_WED, sales_THUR, sales_FRI
FROM sales_source_data;
5 rows created.
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

INSERT de Criação de Pivô (continuação)

No exemplo do slide, os dados de vendas, relativos aos detalhes das vendas realizadas por um representante de vendas em cada dia de uma semana com um ID de semana específico, são recebidos da tabela de banco de dados não relacional SALES_SOURCE_DATA.

DESC SALES_SOURCE_DATA

| Name | Null? | Type |
|-------------|-------|-------------|
| EMPLOYEE_ID | | NUMBER(6) |
| WEEK_ID | | NUMBER(2) |
| SALES_MON | | NUMBER(8,2) |
| SALES_TUE | | NUMBER(8,2) |
| SALES_WED | | NUMBER(8,2) |
| SALES_THUR | | NUMBER(8,2) |
| SALES_FRI | | NUMBER(8,2) |

INSERT de Criação de Pivô (continuação)

```
SELECT * FROM SALES_SOURCE_DATA;
```

| EMPLOYEE_ID | WEEK_ID | SALES_MON | SALES_TUE | SALES_WED | SALES_THUR | SALES_FRI |
|-------------|---------|-----------|-----------|-----------|------------|-----------|
| 176 | 6 | 2000 | 3000 | 4000 | 5000 | 6000 |

```
DESC SALES_INFO
```

| Name | Null? | Type |
|-------------|-------|-------------|
| EMPLOYEE_ID | | NUMBER(6) |
| WEEK | | NUMBER(2) |
| SALES | | NUMBER(8,2) |

```
SELECT * FROM sales_info;
```

| EMPLOYEE_ID | WEEK | SALES |
|-------------|------|-------|
| 176 | 6 | 2000 |
| 176 | 6 | 3000 |
| 176 | 6 | 4000 |
| 176 | 6 | 5000 |
| 176 | 6 | 6000 |

No exemplo anterior, observe que, ao usar uma instrução INSERT de criação de pivô, uma linha da tabela SALES_SOURCE_DATA é convertida em cinco registros para a tabela relacional, SALES_INFO.

A Instrução MERGE

- **Permite atualizar ou inserir dados de forma condicional em uma tabela de banco de dados**
- **Executa uma operação UPDATE se a linha existir e uma operação INSERT se a linha for nova**
 - **Evita atualizações separadas**
 - **Melhora o desempenho e facilita o uso**
 - **É útil nas aplicações de data warehouse**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Instruções MERGE

O servidor Oracle suporta a instrução MERGE para as operações INSERT, UPDATE e DELETE. Ao usar essa instrução, você pode atualizar, inserir ou deletar uma linha de forma condicional em uma tabela, evitando, assim, várias instruções DML. A decisão de efetuar uma atualização, inserção ou deleção na tabela de destino baseia-se na condição na cláusula ON.

Você precisa ter privilégios de objeto INSERT e UPDATE na tabela de destino e o privilégio de objeto SELECT na tabela de origem. Para especificar a cláusula DELETE de merge_update_clause, é preciso ter o privilégio de objeto DELETE na tabela de destino.

A instrução MERGE é determinante. Não é possível atualizar a mesma linha da tabela de destino várias vezes na mesma instrução MERGE.

Uma abordagem alternativa é usar loops PL/SQL e várias instruções DML. No entanto, a instrução MERGE é fácil de usar e é expressa de forma mais simples como uma única instrução SQL.

A instrução MERGE é apropriada para várias aplicações de data warehouse. Por exemplo, em uma aplicação de data warehouse, talvez seja necessário trabalhar com dados provenientes de várias origens, alguns dos quais podem ser duplicados. Com a instrução MERGE, é possível adicionar ou modificar linhas de forma condicional.

A Sintaxe da Instrução MERGE

É possível inserir ou atualizar as linhas de uma tabela de forma condicional usando a instrução MERGE.

```
MERGE INTO table_name table_alias
  USING (table/view/sub_query) alias
  ON (join condition)
  WHEN MATCHED THEN
    UPDATE SET
      col1 = col_val1,
      col2 = col2_val
  WHEN NOT MATCHED THEN
    INSERT (column_list)
    VALUES (column_values);
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Intercalando Linhas

É possível atualizar linhas existentes e inserir novas linhas de forma condicional usando a instrução MERGE.

Na sintaxe:

| | |
|------------------|--|
| Cláusula INTO | especifica a tabela de destino para atualização ou inserção |
| Cláusula USING | identifica a origem dos dados a serem atualizados ou inseridos; pode ser uma tabela, uma view ou uma subconsulta |
| Cláusula ON | a condição com base na qual a operação MERGE efetua a atualização ou inserção |
| WHEN MATCHED | instrui o servidor sobre como responder aos resultados da condição de join |
| WHEN NOT MATCHED | |

Para obter mais informações, consulte o item "MERGE" do manual *Oracle Database 10g SQL Reference*.

Intercalando Linhas

Insira ou atualize linhas da tabela EMPL3 para corresponder à tabela EMPLOYEES.

```
MERGE INTO empl3 c
  USING employees e
  ON (c.employee_id = e.employee_id)
  WHEN MATCHED THEN
    UPDATE SET
      c.first_name      = e.first_name,
      c.last_name       = e.last_name,
      ...
      c.department_id  = e.department_id
  WHEN NOT MATCHED THEN
    INSERT VALUES(e.employee_id, e.first_name, e.last_name,
      e.email, e.phone_number, e.hire_date, e.job_id,
      e.salary, e.commission_pct, e.manager_id,
      e.department_id);
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Exemplo de Intercalação de Linhas

```
MERGE INTO empl3 c
  USING employees e
  ON (c.employee_id = e.employee_id)
  WHEN MATCHED THEN
    UPDATE SET
      c.first_name      = e.first_name,
      c.last_name       = e.last_name,
      c.email           = e.email,
      c.phone_number    = e.phone_number,
      c.hire_date       = e.hire_date,
      c.job_id          = e.job_id,
      c.salary          = e.salary,
      c.commission_pct  = e.commission_pct,
      c.manager_id      = e.manager_id,
      c.department_id   = e.department_id
  WHEN NOT MATCHED THEN
    INSERT VALUES(e.employee_id, e.first_name, e.last_name,
      e.email, e.phone_number, e.hire_date, e.job_id,
      e.salary, e.commission_pct, e.manager_id,
      e.department_id);
```

Intercalando Linhas

```
TRUNCATE TABLE empl3;
```

```
SELECT *  
FROM empl3;  
no rows selected
```

```
MERGE INTO empl3 c  
  USING employees e  
  ON (c.employee_id = e.employee_id)  
WHEN MATCHED THEN  
  UPDATE SET  
    ...  
WHEN NOT MATCHED THEN  
  INSERT VALUES...;
```

```
SELECT *  
FROM empl3;  
20 rows selected.
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

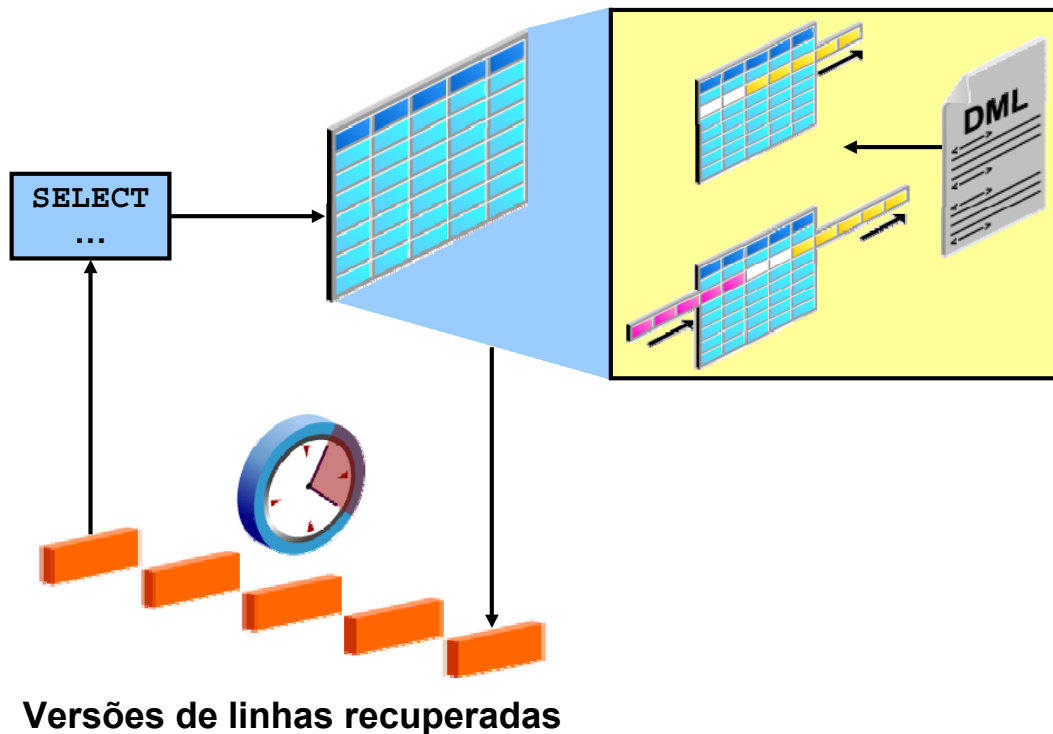
Exemplo de Intercalação de Linhas (continuação)

O exemplo do slide estabelece a correspondência entre `EMPLOYEE_ID` da tabela `EMPL3` e `EMPLOYEE_ID` da tabela `EMPLOYEES`. Caso seja encontrada uma correspondência, a linha da tabela `EMPL3` será atualizada para corresponder à linha da tabela `EMPLOYEES`. Se não for encontrada, a linha será inserida na tabela `EMPL3`.

A condição `c.employee_id = e.employee_id` será avaliada. Como a tabela `EMPL3` está vazia, a condição retorna `FALSE`, indicando que não há correspondências. A lógica corresponde à cláusula `WHEN NOT MATCHED`, e o comando `MERGE` insere as linhas da tabela `EMPLOYEES` na tabela `EMPL3`.

Se houver linhas na tabela `EMPL3`, e os IDs dos funcionários corresponderem nas duas tabelas (`EMPL3` e `EMPLOYEES`), as linhas existentes na tabela `EMPL3` serão atualizadas para corresponderem à tabela `EMPLOYEES`.

Controlando Alterações nos Dados



ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Controlando Alterações nos Dados

Você poderá perceber que, de alguma maneira, os dados de uma tabela foram alterados de forma inadequada. Para pesquisar isso, é possível usar várias consultas de flashback para exibir dados das linhas em momentos específicos. Com mais eficiência, é possível usar o recurso Flashback de Consulta de Versão para exibir todas as alterações feitas em uma linha durante um período. Esse recurso permite que você anexe a cláusula `VERSIONS` a uma instrução `SELECT` que especifique um SCN ou uma faixa de timestamp dentro da qual deseja exibir as alterações feitas nos valores das linhas. A consulta também pode retornar metadados associados, tais como a transação responsável pela alteração.

Além disso, após identificar uma transação errada, você poderá usar o recurso Flashback de Consulta de Transação para identificar outras alterações feitas por essa transação. Em seguida, você poderá usar o recurso Flashback de Tabela para restaurar a tabela até um estado anterior às alterações.

É possível consultar uma tabela com a cláusula `VERSIONS` para produzir todas as versões de todas as linhas que existem ou que já existiram entre o momento da consulta e o momento da execução do parâmetro `undo_retention`, segundos antes do momento atual. `undo_retention` é um parâmetro de inicialização auto-ajustável. A consulta que inclui uma cláusula `VERSIONS` denomina-se consulta de versão. Os resultados de uma consulta de versão se comportam como se a cláusula `WHERE` fosse aplicada às versões das linhas. A consulta de versão retorna versões das linhas apenas durante as transações.

SCN (número de alteração do sistema): O servidor Oracle atribui um SCN (System Change Number) para identificar os registros de redo para cada transação submetida a `commit`.

Exemplo de Flashback de Consulta de Versão

```
SELECT salary FROM employees3
WHERE employee_id = 107;
```

1

| SALARY |
|--------|
| 4200 |

```
UPDATE employees3 SET salary = salary * 1.30
WHERE employee_id = 107;
```

2

```
COMMIT;
```

```
SELECT salary FROM employees3
  VERSIONS BETWEEN SCN MINVALUE AND MAXVALUE
WHERE employee_id = 107;
```

3

| SALARY |
|--------|
| 5460 |
| 4200 |

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Exemplo de Flashback de Consulta de Versão

No exemplo do slide, o salário do funcionário 107 é recuperado (1). O salário do funcionário 107 é aumentado em 30%, e essa alteração é submetida a commit (2). São exibidas as diferentes versões de salário (3).

A cláusula `VERSIONS` não altera o plano da consulta. Por exemplo, se você executar uma consulta para uma tabela que usa o método de acesso por índice, a mesma consulta na mesma tabela com uma cláusula `VERSIONS` continuará usando o método de acesso por índice. As versões de linhas retornadas pela consulta são as versões das linhas durante as transações. A cláusula `VERSIONS` não tem efeito sobre o comportamento transacional de uma consulta. Isso significa que uma consulta a uma tabela com a cláusula `VERSIONS` também herda o ambiente de consulta da transação em andamento.

A cláusula `VERSIONS` default pode ser especificada como `VERSIONS BETWEEN {SCN|TIMESTAMP} MINVALUE AND MAXVALUE`.

A cláusula `VERSIONS` é uma extensão SQL apenas para consultas. É possível ter operações DML e DDL que usam uma cláusula `VERSIONS` dentro de subconsultas. A consulta de versão da linha recupera todas as versões submetidas a commit das linhas selecionadas. As alterações feitas pela transação ativa atual não são retornadas. A consulta de versão recupera todas as versões de linhas. Isso significa, essencialmente, que as versões retornadas incluem versões de linhas deletadas e subsequentemente reinseridas.

Exemplo de Obtenção de Versões de Linhas

O acesso a linha para uma consulta de versão pode ser definido em uma destas duas categorias:

- Acesso a linha baseado no ID de linha: Em caso de acesso baseado no ID de linha, todas as versões do ID de linha especificado são retornadas, não importando o conteúdo da linha. Isso significa, essencialmente, que são retornadas todas as versões do slot do bloco indicado pelo ID de linha.
- Todos os demais acessos a linha: Para os demais acessos a linha, são retornadas todas as versões de linha.

A Cláusula VERSIONS BETWEEN

```
SELECT versions_starttime "START_DATE",  
       versions_endtime   "END_DATE",  
       salary  
FROM   employees  
       VERSIONS BETWEEN SCN MINVALUE  
       AND MAXVALUE  
WHERE  last_name = 'Lorentz';
```

| START_DATE | END_DATE | SALARY |
|-----------------------|-----------------------|--------|
| 13-FEB-04 11.16.41 AM | | 5460 |
| | 13-FEB-04 11.16.41 AM | 4200 |

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

A Cláusula VERSIONS BETWEEN

Você pode usar a cláusula VERSIONS BETWEEN para recuperar todas as versões das linhas que existem ou que já existiram entre o momento da consulta e um momento passado.

Se o tempo de retenção de undo for menor que o limite inferior de tempo/SCN da cláusula BETWEEN, a consulta recuperará apenas as versões até o período de retenção de undo. O intervalo de tempo da cláusula BETWEEN pode ser especificado como um intervalo SCN ou como uma faixa de horários. Esse intervalo de tempo é definido pelos limites inferior e superior.

No exemplo, as alterações do salário de Lorentz são recuperadas. O valor nulo para END_DATE na primeira versão indica que esta era a versão existente no momento da consulta. O valor nulo para START_DATE na última versão indica que essa versão foi criada em um momento anterior ao tempo de retenção de undo.

Sumário

Nesta lição, você aprendeu a:

- **Usar instruções DML e controlar transações**
- **Descrever os recursos de inserções em várias tabelas**
- **Usar os seguintes tipos de inserções em várias tabelas:**
 - **INSERT Incondicional**
 - **INSERT de Criação de Pivô**
 - **ALL INSERT Condicional**
 - **FIRST INSERT Condicional**
- **Intercalar linhas em uma tabela**
- **Manipular dados usando subconsultas**
- **Controlar as alterações de dados durante um período**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Sumário

Nesta lição, você aprendeu a manipular os dados do banco de dados Oracle usando subconsultas. Você também conheceu as instruções INSERT em várias tabelas e a instrução MERGE, além de aprender a controlar as alterações feitas no banco de dados.

Exercício 3: Visão Geral

Este exercício aborda os seguintes tópicos:

- Executando **INSERTs** em várias tabelas
- Executando operações **MERGE**
- Controlando versões de linhas

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Exercício 3: Visão Geral

Neste exercício, você adiciona linhas à tabela `emp_data`, atualiza e deleta dados da tabela e controla suas transações.

Exercício 3

1. Execute o script `lab_03_01.sql` da pasta lab para criar a tabela `SAL_HISTORY`.
2. Exiba a estrutura da tabela `SAL_HISTORY`.

| Name | Null? | Type |
|-------------|-------|-------------|
| EMPLOYEE_ID | | NUMBER(6) |
| HIRE_DATE | | DATE |
| SALARY | | NUMBER(8,2) |

3. Execute o script `lab_03_03.sql` da pasta lab para criar a tabela `MGR_HISTORY`.
4. Exiba a estrutura da tabela `MGR_HISTORY`.

| Name | Null? | Type |
|-------------|-------|-------------|
| EMPLOYEE_ID | | NUMBER(6) |
| MANAGER_ID | | NUMBER(6) |
| SALARY | | NUMBER(8,2) |

5. Execute o script `lab_03_05.sql` da pasta lab para criar a tabela `SPECIAL_SAL`.
6. Exiba a estrutura da tabela `SPECIAL_SAL`.

| Name | Null? | Type |
|-------------|-------|-------------|
| EMPLOYEE_ID | | NUMBER(6) |
| SALARY | | NUMBER(8,2) |

7. a. Crie uma consulta que faça o seguinte:
 - Recupere na tabela `EMPLOYEES` os detalhes de ID do funcionário, data de admissão, salário e o ID do gerente desses funcionários cujo ID é inferior a 125.
 - Se o salário for superior a \$20.000, insira os detalhes sobre o ID do funcionário e o salário na tabela `SPECIAL_SAL`.
 - Insira o ID do funcionário, a data de admissão e o salário na tabela `SAL_HISTORY`.
 - Insira os detalhes sobre o ID do funcionário, o ID do gerente e o salário na tabela `MGR_HISTORY`.

Exercício 3 (continuação)

b. Exiba os registros da tabela SPECIAL_SAL.

| EMPLOYEE_ID | SALARY |
|-------------|--------|
| 100 | 24000 |

c. Exiba os registros da tabela SAL_HISTORY.

| EMPLOYEE_ID | HIRE_DATE | SALARY |
|-------------|-----------|--------|
| 101 | 21-SEP-89 | 17000 |
| 102 | 13-JAN-93 | 17000 |
| 103 | 03-JAN-90 | 9000 |
| 104 | 21-MAY-91 | 6000 |
| 105 | 25-JUN-97 | 4800 |
| 106 | 05-FEB-98 | 4800 |
| 107 | 07-FEB-99 | 4200 |
| 108 | 17-AUG-94 | 12000 |
| 109 | 16-AUG-94 | 9000 |
| 110 | 28-SEP-97 | 8200 |
| 111 | 30-SEP-97 | 7700 |
| 112 | 07-MAR-98 | 7800 |
| 113 | 07-DEC-99 | 6900 |

| | | |
|-----|-----------|-------|
| 114 | 07-DEC-94 | 11000 |
| 115 | 18-MAY-95 | 3100 |
| 116 | 24-DEC-97 | 2900 |
| 117 | 24-JUL-97 | 2800 |
| 118 | 15-NOV-98 | 2600 |
| 119 | 10-AUG-99 | 2500 |
| 120 | 18-JUL-96 | 8000 |
| 121 | 10-APR-97 | 8200 |
| 122 | 01-MAY-95 | 7900 |
| 123 | 10-OCT-97 | 6500 |
| 124 | 16-NOV-99 | 5800 |

24 rows selected.

Exercício 3 (continuação)

d. Exiba os registros da tabela MGR_HISTORY.

| EMPLOYEE_ID | MANAGER_ID | SALARY |
|-------------|------------|--------|
| 101 | 100 | 17000 |
| 102 | 100 | 17000 |
| 103 | 102 | 9000 |
| 104 | 103 | 6000 |
| 105 | 103 | 4800 |
| 106 | 103 | 4800 |
| 107 | 103 | 4200 |
| 108 | 101 | 12000 |
| 109 | 108 | 9000 |
| 110 | 108 | 8200 |
| 111 | 108 | 7700 |
| 112 | 108 | 7800 |
| 113 | 108 | 6900 |
| 114 | 100 | 11000 |
| 115 | 114 | 3100 |
| 116 | 114 | 2900 |
| 117 | 114 | 2800 |
| 118 | 114 | 2600 |
| 119 | 114 | 2500 |
| 120 | 100 | 8000 |
| 121 | 100 | 8200 |
| 122 | 100 | 7900 |
| 123 | 100 | 6500 |
| 124 | 100 | 5800 |

24 rows selected.

Exercício 3 (continuação)

8. a. Execute o script `lab_03_08a.sql` da pasta `lab` para criar a tabela `SALES_SOURCE_DATA`.
- b. Execute o script `lab_03_08b.sql` da pasta `lab` para inserir registros na tabela `SALES_SOURCE_DATA`.
- c. Exiba a estrutura da tabela `SALES_SOURCE_DATA`.

| Name | Null? | Type |
|-------------|-------|-------------|
| EMPLOYEE_ID | | NUMBER(6) |
| WEEK_ID | | NUMBER(2) |
| SALES_MON | | NUMBER(8,2) |
| SALES_TUE | | NUMBER(8,2) |
| SALES_WED | | NUMBER(8,2) |
| SALES_THUR | | NUMBER(8,2) |
| SALES_FRI | | NUMBER(8,2) |

- d. Exiba os registros da tabela `SALES_SOURCE_DATA`.

| EMPLOYEE_ID | WEEK_ID | SALES_MON | SALES_TUE | SALES_WED | SALES_THUR | SALES_FRI |
|-------------|---------|-----------|-----------|-----------|------------|-----------|
| 178 | 6 | 1750 | 2200 | 1500 | 1500 | 3000 |

- e. Execute o script `lab_03_08c.sql` da pasta `lab` para criar a tabela `SALES_INFO`.
- f. Exiba a estrutura da tabela `SALES_INFO`.

| Name | Null? | Type |
|-------------|-------|-------------|
| EMPLOYEE_ID | | NUMBER(6) |
| WEEK | | NUMBER(2) |
| SALES | | NUMBER(8,2) |

Exercício 3 (continuação)

- g. Crie uma consulta que faça o seguinte:

Recupere da tabela `SALES_SOURCE_DATA` os detalhes sobre o ID do funcionário, o ID da semana, vendas na segunda-feira, vendas na terça-feira, vendas na quarta-feira, vendas na quinta-feira e vendas na sexta-feira.

Crie uma transformação de modo que cada registro recuperado da tabela `SALES_SOURCE_DATA` seja convertido em vários registros para a tabela `SALES_INFO`.

Dica: Use uma instrução `INSERT` de criação de pivô.

- h. Exiba os registros da tabela `SALES_INFO`.

| EMPLOYEE_ID | WEEK | SALES |
|-------------|------|-------|
| 178 | 6 | 1750 |
| 178 | 6 | 2200 |
| 178 | 6 | 1500 |
| 178 | 6 | 1500 |
| 178 | 6 | 3000 |

9. Você tem os dados dos antigos funcionários armazenados em um arquivo sem formatação denominado `emp.data` e deseja armazenar em uma tabela os nomes e os IDs de e-mail de todos os funcionários, antigos e atuais. Para isso, primeiro crie uma tabela externa denominada `EMP_DATA` usando o arquivo de origem `emp.dat` no diretório `emp_dir`. Você pode usar o script `lab_03_09.sql` para essa tarefa.
10. Em seguida, execute o script `lab_03_10.sql` para criar a tabela `EMP_HIST`.
 - a. Aumente o tamanho da coluna de e-mail para 45.
 - b. Intercale os dados da tabela `EMP_DATA` criada no último laboratório com os dados da tabela `EMP_HIST`. Suponha que os dados da tabela externa `EMP_DATA` sejam os mais atualizados. Se uma linha da tabela `EMP_DATA` corresponde à tabela `EMP_HIST`, atualize a coluna de e-mail da tabela `EMP_HIST` para corresponder à linha da tabela `EMP_DATA`. Se uma linha da tabela `EMP_DATA` não corresponder à tabela `EMP_HIST`, insira-a na tabela `EMP_HIST`. As linhas são coincidentes quando o nome e o sobrenome do funcionário são idênticos.
 - c. Recupere as linhas da tabela `EMP_HIST` após a intercalação.

Exercício 3 (continuação)

| FIRST_NAME | LAST_NAME | EMAIL |
|------------|-----------|------------------|
| Steven | King | SKING |
| Neena | Kochhar | nkochh@pipit.com |
| Lex | De Haan | LDEHAAN |
| Alexander | Hunold | AHun@MOORHEN.COM |
| Bruce | Ernst | BERNST |
| David | Austin | DAUSTIN |
| Valli | Pataballa | VPATABAL |
| Diana | Lorentz | DLORENTZ |
| Nancy | Greenberg | NGREENBE |
| Daniel | Faviet | DFAVIET |
| John | Chen | JCHEN |
| Ismael | Sciarra | ISCIARRA |

...

| FIRST_NAME | LAST_NAME | EMAIL |
|------------|-----------|---------------------------|
| Diana | lorentz | dlor@limpkin.com |
| Stephen | King | sking@merganser.com |
| Hema | Voight | Hema.Voight@PHALAROPE.COM |
| Nancy | greenberg | ngreenb@plover.com |

148 rows selected.

11. Crie a tabela EMP3 usando o script lab_03_11.sql. Na tabela EMP3, altere o departamento de Kochhar para 60 e faça commit da alteração. Em seguida, altere o departamento de Kochhar para 50 e faça commit da alteração. Controle as alterações de Kochhar usando o recurso Row Versions.

| START_DATE | END_DATE | DEPARTMENT_ID |
|-----------------------|-----------------------|---------------|
| 13-FEB-04 12.33.56 PM | | 50 |
| 13-FEB-04 12.33.53 PM | 13-FEB-04 12.33.56 PM | 60 |
| | 13-FEB-04 12.33.53 PM | 90 |

4

Gerando Relatórios por Agrupamento de Dados Relacionados

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Objetivos

Ao concluir esta lição, você será capaz de:

- **Usar a operação ROLLUP para produzir valores de subtotais**
- **Usar a operação CUBE para produzir valores de tabelas de referência cruzada**
- **Usar a function GROUPING para identificar os valores das linhas criadas por ROLLUP ou CUBE**
- **Usar GROUPING SETS para produzir um único conjunto de resultados**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

Objetivos

Nesta lição, você aprenderá a:

- Agrupar dados para obter:
 - Valores de subtotais usando o operador ROLLUP
 - Valores de tabelas de referência cruzada usando o operador CUBE
- Usar a function GROUPING para identificar o nível de agregação no conjunto de resultados produzido por um operador ROLLUP ou CUBE.
- Usar GROUPING SETS para produzir um único conjunto de resultados que seja equivalente a um método UNION ALL

Análise de Functions de Grupo

- As functions de grupo operam em conjuntos de linhas para fornecer um resultado por grupo.

```
SELECT      [column,] group_function(column) . . .
FROM        table
[WHERE      condition]
[GROUP BY   group_by_expression]
[ORDER BY   column];
```

- Exemplo:

```
SELECT AVG(salary), STDDEV(salary),
       COUNT(commission_pct), MAX(hire_date)
FROM   employees
WHERE  job_id LIKE 'SA%';
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

Functions de Grupo

Você pode usar a cláusula GROUP BY para dividir as linhas de uma tabela em grupos. Em seguida, você pode usar functions de grupo para retornar informações resumidas de cada grupo. As functions de grupo podem aparecer em listas de seleção e nas cláusulas ORDER BY e HAVING. O servidor Oracle aplica as functions de grupo a cada grupo de linhas e retorna uma única linha de resultados para cada grupo.

Tipos de function de grupo: Cada uma das functions AVG, SUM, MAX, MIN, COUNT, STDDEV e VARIANCE aceita apenas um argumento. As functions AVG, SUM, STDDEV e VARIANCE operam apenas em valores numéricos. MAX e MIN podem operar em valores numéricos, de caracteres e de dados de datas. COUNT retorna o número de linhas não nulas para determinada expressão. No exemplo do slide, são calculados o salário médio, o desvio padrão relativo ao salário, o número de funcionários que recebem comissão e a data de admissão máxima para os funcionários cujo JOB_ID começa com SA.

Diretrizes do Uso de Functions de Grupo

- Os tipos de dados para os argumentos podem ser CHAR, VARCHAR2, NUMBER ou DATE.
- Todas as functions de grupo, exceto COUNT (*), ignoram valores nulos. Para substituir um valor por valores nulos, use a function NVL. COUNT retorna um número ou zero.
- Quando você usa a cláusula GROUP BY, o servidor Oracle classifica, implicitamente, o conjunto de resultados das colunas de agrupamento especificadas em ordem crescente. Para sobrepor essa ordenação default, é possível usar DESC em uma cláusula ORDER BY.

Análise da Cláusula GROUP BY

- **Sintaxe:**

```
SELECT      [column,] group_function(column). . .
FROM        table
[WHERE      condition]
[GROUP BY   group_by_expression]
[ORDER BY   column];
```

- **Exemplo:**

```
SELECT  department_id, job_id, SUM(salary),
        COUNT(employee_id)
FROM    employees
GROUP BY department_id, job_id ;
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

Análise da Cláusula GROUP BY

O exemplo ilustrado no slide é avaliado pelo servidor Oracle da seguinte forma:

- A cláusula SELECT especifica que as seguintes colunas devem ser recuperadas:
 - Colunas de ID do departamento e ID do cargo da tabela EMPLOYEES
 - A soma de todos os salários e o número de funcionários de cada grupo especificado na cláusula GROUP BY
- A cláusula GROUP BY especifica como as linhas devem ser agrupadas na tabela. O salário total e o número de funcionários são calculados para cada ID de cargo em cada departamento. As linhas são agrupadas por ID de departamento e, em seguida, são agrupadas por cargo, dentro de cada departamento.

Análise da Cláusula HAVING

- Use a cláusula HAVING para especificar quais grupos devem ser exibidos.
- Você pode restringir ainda mais os grupos com base em uma condição limitante.

```
SELECT      [column,] group_function(column)...  
FROM        table  
[WHERE      condition]  
[GROUP BY   group_by_expression]  
[HAVING     having_expression]  
[ORDER BY   column];
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

A Cláusula HAVING

Os grupos são formados e as functions de grupo são calculadas antes de a cláusula HAVING ser aplicada aos grupos. A cláusula HAVING pode anteceder a cláusula GROUP BY, mas, por motivos lógicos, é recomendável usar primeiro a cláusula GROUP BY.

Quando você usa a cláusula HAVING, o servidor Oracle segue as seguintes etapas:

1. Agrupa linhas
2. Aplica as functions de grupo aos grupos e exibe os grupos que correspondem aos critérios da cláusula HAVING

GROUP BY com Operadores ROLLUP e CUBE

- Use o operador ROLLUP ou CUBE com GROUP BY para produzir linhas superagregadas por colunas de referência cruzada.
- O agrupamento de ROLLUP produz um conjunto de resultados com as linhas agrupadas normais e os valores dos subtotais.
- O agrupamento de CUBE produz um conjunto de resultados com as linhas de ROLLUP e as linhas de tabelas de referência cruzada.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

GROUP BY com os Operadores ROLLUP e CUBE

Especifique os operadores ROLLUP e CUBE na cláusula GROUP BY de uma consulta. O agrupamento de ROLLUP produz um conjunto de resultados com as linhas agrupadas normais e as linhas de subtotais. A operação CUBE na cláusula GROUP BY agrupa as linhas selecionadas com base nos valores de todas as combinações possíveis de expressões na especificação e retorna uma única linha de informações resumidas para cada grupo. Você pode usar o operador CUBE para produzir linhas de tabelas de referência cruzada.

Observação: Quando estiver trabalhando com ROLLUP e CUBE, certifique-se de que as colunas após a cláusula GROUP BY tenham relacionamentos significativos e reais umas com as outras; caso contrário, os operadores retornarão informações irrelevantes.

Operador ROLLUP

- ROLLUP é uma extensão da cláusula GROUP BY.
- Use a operação ROLLUP para produzir agregados cumulativos, como subtotais.

```
SELECT      [column,] group_function(column). . .  
FROM        table  
[WHERE      condition]  
[GROUP BY   [ROLLUP] group_by_expression]  
[HAVING     having_expression];  
[ORDER BY   column];
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

O Operador ROLLUP

O operador ROLLUP fornece agregados e superagregados para expressões contidas na instrução GROUP BY. Os autores de relatórios podem usar o operador ROLLUP para extrair estatísticas e informações resumidas de conjuntos de resultados. É possível usar agregados cumulativos em relatórios, tabelas e gráficos.

O operador ROLLUP cria agrupamentos movendo-se em uma direção, da direita para a esquerda, pela lista de colunas especificada na cláusula GROUP BY. Depois, ele aplica a function agregada a esses agrupamentos.

Observação

- Para produzir subtotais em n dimensões (isto é, n colunas na cláusula GROUP BY) sem um operador ROLLUP, é necessário vincular $n+1$ instruções SELECT com UNION ALL. Isso torna a execução da consulta ineficiente, pois cada uma das instruções SELECT produz um acesso à tabela. O operador ROLLUP reúne os resultados com apenas um acesso à tabela. Esse operador será útil quando houver várias colunas envolvidas na produção de subtotais.
- Os subtotais e totais são produzidos com ROLLUP. CUBE também produz totais e faz um roll-up eficiente dos valores em todas as direções possíveis, produzindo dados de referência cruzada.

Operador ROLLUP: Exemplo

```
SELECT department_id, job_id, SUM(salary)
FROM employees
WHERE department_id < 60
GROUP BY ROLLUP(department_id, job_id);
```

| DEPARTMENT ID | JOB ID | SUM(SALARY) | |
|---------------|----------|-------------|---|
| 10 | AD_ASST | 4400 | 1 |
| 10 | | 4400 | |
| 20 | MK_MAN | 13000 | |
| 20 | MK_REP | 6000 | |
| 20 | | 19000 | |
| 30 | PU_MAN | 11000 | |
| 30 | PU_CLERK | 13900 | |
| 30 | | 24900 | |
| 40 | HR_REP | 6500 | 2 |
| 40 | | 6500 | |
| 50 | ST_MAN | 36400 | |
| 50 | SH_CLERK | 64300 | |
| 50 | ST_CLERK | 55700 | |
| 50 | | 156400 | |
| | | 211200 | 3 |

15 rows selected.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Exemplo de um Operador ROLLUP

No exemplo do slide:

- O total dos salários de todos os IDs de cargos relativos aos departamentos cujo ID é menor que 60 são exibidos pela cláusula GROUP BY.
- O operador ROLLUP exibe:
 - O salário total de cada departamento cujo ID é menor que 60
 - O salário total de todos os departamentos cujo ID é menor que 60, independentemente dos IDs dos cargos

Neste exemplo, 1 indica um grupo totalizado por DEPARTMENT_ID e JOB_ID, 2 indica um grupo totalizado apenas por DEPARTMENT_ID e 3 indica o total geral.

O operador ROLLUP cria subtotais nos quais ocorre um rollup do nível mais detalhado para um total geral, seguindo a lista de agrupamento especificada na cláusula GROUP BY.

Primeiro, ele calcula os valores agregados padrão para os grupos especificados na cláusula GROUP BY (no exemplo, a soma dos salários agrupada em cada cargo de um departamento).

Em seguida, ele cria subtotais progressivamente mais altos, movendo-se da direita para a esquerda pela lista de colunas de agrupamento. (No exemplo, a soma dos salários de cada departamento é calculada, seguida pela soma dos salários de todos os departamentos.)

- Fornecidas n expressões no operador ROLLUP da cláusula GROUP BY, a operação resulta em agrupamentos $n + 1$ (neste caso, $2 + 1 = 3$).
- As linhas baseadas nos valores das primeiras n expressões são denominadas linhas ou linhas normais e as outras são denominadas linhas superagregadas.

Operador CUBE

- CUBE é uma extensão da cláusula GROUP BY.
- Você pode usar o operador CUBE para produzir valores de tabelas de referência cruzada com uma única instrução SELECT.

```
SELECT      [column,] group_function(column)...  
FROM        table  
[WHERE      condition]  
[GROUP BY   CUBE group_by_expression]  
[HAVING     having_expression]  
[ORDER BY   column];
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

O Operador CUBE

O operador CUBE é uma alternância adicional na cláusula GROUP BY de uma instrução SELECT. É possível aplicar esse operador a todas as functions agregadas, inclusive AVG, SUM, MAX, MIN e COUNT. Ele é usado para produzir conjuntos de resultados usados geralmente em relatórios de tabelas de referência cruzada. Enquanto ROLLUP produz somente uma fração das combinações possíveis de subtotais, CUBE produz subtotais para todas as combinações possíveis de agrupamentos especificados na cláusula GROUP BY e um total geral.

O operador CUBE é usado com uma function agregada para gerar outras linhas em um conjunto de resultados. É feita referência cruzada às colunas incluídas na cláusula GROUP BY para produzir um superconjunto de grupos. A function agregada especificada na lista de seleção é aplicada a esses grupos para produzir valores resumidos para as linhas superagregadas adicionais. O número de grupos extras no conjunto de resultados é determinado pelo número de colunas incluídas na cláusula GROUP BY.

Na verdade, todas as combinações possíveis das colunas ou expressões da cláusula GROUP BY são usadas para produzir superagregados. Se você tiver n colunas ou expressões na cláusula GROUP BY, haverá 2^n combinações superagregadas possíveis. Matematicamente, essas combinações formam um cubo n dimensional, que é a origem do nome do operador.

Com ferramentas de programação e aplicações, é possível incluir esses valores superagregados em tabelas e gráficos para apresentar os resultados e os relacionamentos de maneira visual e eficiente.

Operador CUBE: Exemplo

```
SELECT department_id, job_id, SUM(salary)
FROM employees
WHERE department_id < 60
GROUP BY CUBE (department_id, job_id) ;
```

| DEPARTMENT_ID | JOB_ID | SUM(SALARY) |
|---------------|----------|-------------|
| | | 211200 |
| | HR_REP | 6500 |
| | MK_MAN | 13000 |
| | MK_REP | 6000 |
| | PU_MAN | 11000 |
| | ST_MAN | 36400 |
| | AD_ASST | 4400 |
| | PU_CLERK | 13900 |
| | SH_CLERK | 64300 |
| | ST_CLERK | 55700 |
| 10 | | 4400 |
| 10 | AD_ASST | 4400 |
| 20 | | 19000 |
| 20 | MK_MAN | 13000 |
| 20 | MK_REP | 6000 |
| 30 | | 24900 |
| 30 | PU_MAN | 11000 |

1

2

3

4

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Exemplo de um Operador CUBE

É possível interpretar a saída da instrução SELECT no exemplo da seguinte maneira:

- O salário total de todos os cargos de um departamento (para os departamentos com ID menor que 60) é exibido na cláusula GROUP BY.
- O salário total dos departamentos cujo ID é menor que 60.
- O salário total de todos os cargos independentemente do departamento.
- O salário total dos departamentos cujo ID é menor que 60, independentemente dos cargos.

Neste exemplo, 1 indica o total geral. 2 indica as linhas totalizadas apenas por JOB_ID. 3 indica algumas linhas totalizadas por DEPARTMENT_ID e JOB_ID. 4 indica algumas linhas totalizadas apenas por DEPARTMENT_ID.

O operador CUBE também executou a operação ROLLUP para exibir os subtotais, para os departamentos cujo ID é menor que 60, e o salário total dos departamentos cujo ID é menor que 60, independentemente dos cargos. Além disso, o operador CUBE exibe o salário total de todos os cargos independentemente do departamento.

Observação: De maneira semelhante ao operador ROLLUP, a produção de subtotais em n dimensões (isto é, n colunas na cláusula GROUP BY) sem um operador CUBE exige a vinculação de 2^n instruções SELECT com UNION ALL. Portanto, um relatório com três dimensões exige a vinculação de $2^3 = 8$ instruções SELECT com UNION ALL.

Function GROUPING

A Function GROUPING

- É usada com o operador CUBE ou ROLLUP
- É usada para localizar os grupos que formam o subtotal em uma linha
- É usada para diferenciar valores NULL armazenados de valores NULL criados por ROLLUP ou CUBE
- Retorna 0 ou 1

```
SELECT      [column,] group_function(column) .. ,  
            GROUPING(expr)  
FROM        table  
[WHERE      condition]  
[GROUP BY  [ROLLUP] [CUBE] group_by_expression]  
[HAVING    having_expression]  
[ORDER BY  column];
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

A Function GROUPING

É possível usar a function GROUPING com o operador CUBE ou ROLLUP para compreender como um valor resumido foi obtido.

A function GROUPING usa uma única coluna como argumento. O termo expr na function GROUPING deve corresponder a uma das expressões na cláusula GROUP BY. A function retorna o valor 0 ou 1.

Os valores retornados pela function GROUPING são úteis para:

- Determinar o nível de agregação de um subtotal fornecido; isto é, o grupo (ou grupos) no qual o subtotal é baseado
- Identificar se um valor NULL na coluna da expressão de uma linha do conjunto de resultados indica:
 - Um valor NULL da tabela-base (valor NULL armazenado)
 - Um valor NULL criado por ROLLUP ou CUBE (como resultado da function de grupo nessa expressão)

O valor 0 retornado pela function GROUPING com base em uma expressão indica uma das seguintes situações:

- A expressão foi usada para calcular o valor agregado.
- O valor NULL na coluna da expressão é um valor NULL armazenado.

O valor 1 retornado pela function GROUPING com base em uma expressão indica uma das seguintes situações:

- A expressão não foi usada para calcular o valor agregado.
- O valor NULL na coluna da expressão é criado por ROLLUP/CUBE como resultado de agrupamento.

Function GROUPING: Exemplo

```
SELECT  department_id DEPTID, job_id JOB,
        SUM(salary),
        GROUPING(department_id) GRP_DEPT,
        GROUPING(job_id) GRP_JOB
FROM    employees
WHERE   department_id < 50
GROUP BY ROLLUP(department_id, job_id);
```

| DEPTID | JOB | SUM(SALARY) | GRP_DEPT | GRP_JOB |
|--------|----------|-------------|----------|---------|
| 10 | AD_ASST | 4400 | 0 | 0 |
| 10 | | 4400 | 0 | 1 |
| 20 | MK_MAN | 13000 | 0 | 0 |
| 20 | MK_REP | 6000 | 0 | 0 |
| 20 | | 19000 | 0 | 1 |
| 30 | PU_MAN | 11000 | 0 | 0 |
| 30 | PU_CLERK | 13900 | 0 | 0 |
| 30 | | 24900 | 0 | 1 |
| 40 | HR_REP | 6500 | 0 | 0 |
| 40 | | 6500 | 0 | 1 |
| | | 54800 | 1 | 1 |

11 rows selected.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Exemplo de uma Function GROUPING

No exemplo do slide, considere o valor resumido 4400 na primeira linha (indicado por 1). Esse valor resumido é o salário total de cada ID do cargo AD_ASST no departamento 10. Para calcular esse valor resumido, as colunas DEPARTMENT_ID e JOB_ID foram consideradas. Assim, o valor 0 é retornado para as expressões GROUPING(department_id) e GROUPING(job_id).

Considere o valor resumido 4400 na segunda linha (indicado por 2). Esse valor é o salário total do departamento 10 e foi calculado considerando a coluna DEPARTMENT_ID. Portanto, o valor 0 foi retornado por GROUPING(department_id). Como a coluna JOB_ID não foi considerada no cálculo desse valor, o valor 1 foi retornado para GROUPING(job_id). Você pode observar uma saída semelhante na quinta linha.

Na última linha, considere o valor resumido 54800 (indicado por 3). Este é o salário total de todos os cargos dos departamentos cujo ID é menor que 50. Para calcular esse valor resumido, não foram consideradas as colunas DEPARTMENT_ID e JOB_ID. Assim, é retornado o valor 1 para as expressões GROUPING(department_id) e GROUPING(job_id).

GROUPING SETS

- **A sintaxe de GROUPING SETS é usada para definir vários agrupamentos na mesma consulta.**
- **Todos os agrupamentos especificados na cláusula GROUPING SETS são computados, e os resultados de agrupamentos individuais são combinados com uma operação UNION ALL.**
- **Eficiência de conjuntos de agrupamentos:**
 - **Só é preciso uma passagem pela tabela-base**
 - **Não há necessidade de criar instruções UNION complexas.**
 - **Quanto mais elementos GROUPING SETS houver, mais vantagens de desempenho serão obtidas.**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

GROUPING SETS

GROUPING SETS é uma extensão da cláusula GROUP BY que você pode usar para especificar vários agrupamentos de dados. Essa especificação facilita a agregação eficiente, portanto facilita também a análise de dados em várias dimensões.

Agora, é possível criar uma única instrução SELECT usando GROUPING SETS para especificar vários agrupamentos (que também podem incluir os operadores ROLLUP ou CUBE), em vez de diversas instruções SELECT combinadas por operadores UNION ALL. Por exemplo:

```
SELECT department_id, job_id, manager_id, AVG(salary)
FROM employees
GROUP BY
GROUPING SETS
((department_id, job_id, manager_id),
 (department_id, manager_id), (job_id, manager_id));
```

Essa instrução calcula agregados em três agrupamentos:

```
((department_id, job_id, manager_id), (department_id,
manager_id) e (job_id, manager_id)
```

Sem esse recurso, são necessárias várias consultas combinadas com UNION ALL para obter a saída da instrução SELECT precedente. Uma abordagem de várias consultas não é eficaz, pois ela requer várias varreduras nos mesmos dados.

GROUPING SETS (continuação)

Compare o exemplo anterior com a seguinte alternativa:

```
SELECT department_id, job_id, manager_id, AVG(salary)
FROM employees
GROUP BY CUBE(department_id, job_id, manager_id);
```

Essa instrução engloba os 8 agrupamentos (2 * 2 * 2), apesar de apenas os grupos (department_id, job_id, manager_id), (department_id, manager_id) e (job_id, manager_id) serem relevantes para você.

Outra alternativa seria a seguinte instrução:

```
SELECT department_id, job_id, manager_id, AVG(salary)
FROM employees
GROUP BY department_id, job_id, manager_id
UNION ALL
SELECT department_id, NULL, manager_id, AVG(salary)
FROM employees
GROUP BY department_id, manager_id
UNION ALL
SELECT NULL, job_id, manager_id, AVG(salary)
FROM employees
GROUP BY job_id, manager_id;
```

Essa instrução requer três varreduras da tabela-base, o que a torna ineficiente.

Os operadores CUBE e ROLLUP podem ser considerados conjuntos de agrupamento com semântica muito específica. As seguintes equivalências confirmam esse fato:

| | |
|-------------------------------------|--|
| CUBE (a, b, c) é equivalente a | GROUPING SETS ((a, b, c), (a, b), (a, c), (b, c), (a), (b), (c), ()) |
| ROLLUP (a, b, c) é equivalente a | GROUPING SETS ((a, b, c), (a, b), (a), ()) |

GROUPING SETS: Exemplo

```
SELECT  department_id, job_id,
        manager_id, avg(salary)
FROM    employees
GROUP BY GROUPING SETS
        ((department_id, job_id), (job_id, manager_id));
```

| DEPARTMENT_ID | JOB_ID | MANAGER_ID | AVG(SALARY) |
|---------------|--------|------------|-------------|
| | AD_VP | 100 | 17000 |
| | AC_MGR | 101 | 12000 |
| | FI_MGR | 101 | 12000 |
| | HR_REP | 101 | 6500 |
| | MK_MAN | 100 | 13000 |
| | MK_REP | 201 | 6000 |
| ... | PR_REP | 101 | 10000 |

1

| DEPARTMENT_ID | JOB_ID | MANAGER_ID | AVG(SALARY) |
|---------------|------------|------------|-------------|
| 100 | FI_MGR | | 12000 |
| 100 | FI_ACCOUNT | | 7920 |
| 110 | AC_MGR | | 12000 |
| ... | AC_ACCOUNT | | 8300 |

2

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

GROUPING SETS: Exemplo

A consulta do slide calcula agregados em dois agrupamentos. A tabela é dividida nos seguintes grupos:

- ID do Cargo, ID do Gerente
- ID do Departamento, ID do Cargo

Os salários médios de cada um desses grupos são calculados. O conjunto de resultados exibe o salário médio de cada um dos dois grupos.

Na saída, é possível interpretar o grupo marcado por 1 como:

- O salário médio de todos os funcionários com o ID de cargo AD_VP abaixo do gerente 100 é \$17.000.
- O salário médio de todos os funcionários com o ID de cargo AC_MGR abaixo do gerente 101 é \$12.000 e assim por diante.

O grupo marcado com 2 na saída é interpretado como:

- O salário médio de todos os funcionários com o ID de cargo FI_MGR no departamento 100 é \$12.000.
- O salário médio de todos os funcionários com o ID de cargo FI_ACCOUNT no departamento 100 é \$7.920 e assim por diante.

GROUPING SETS: Exemplo (continuação)

O exemplo do slide também pode ser criado assim:

```
SELECT department_id, job_id, NULL as manager_id,  
       AVG(salary) as AVGSAL  
  
FROM employees  
  
GROUP BY department_id, job_id  
  
UNION ALL  
  
SELECT NULL, job_id, manager_id, avg(salary) as AVGSAL  
  
FROM employees  
  
GROUP BY job_id, manager_id;
```

Na ausência de um otimizador que verifique os blocos de consulta para gerar o plano de execução, a consulta precedente precisaria de duas varreduras da tabela-base (EMPLOYEES). Isso pode ser muito ineficiente. Portanto, recomenda-se o uso da instrução GROUPING SETS.

Colunas Compostas

- Uma coluna composta é um conjunto de colunas tratadas como uma unidade.
`ROLLUP (a, (b, c), d)`
- Use parênteses na cláusula GROUP BY para agrupar colunas, de modo que sejam tratadas como uma unidade no cálculo das operações ROLLUP ou CUBE.
- Quando usadas com ROLLUP ou CUBE, as colunas compostas exigiriam que a agregação fosse ignorada em certos níveis.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Colunas Compostas

Uma coluna composta é um conjunto de colunas tratadas como uma unidade durante o cálculo de agrupamentos. Especifique as colunas entre parênteses como na seguinte instrução:

```
ROLLUP (a, (b, c), d)
```

Em que (b, c) forma uma coluna composta e é tratado como uma unidade. Em geral, as colunas compostas são úteis em ROLLUP, CUBE e GROUPING SETS. Por exemplo, em CUBE ou ROLLUP, as colunas compostas exigiriam que a agregação fosse ignorada em certos níveis.

Isto é, GROUP BY ROLLUP(a, (b, c)) equivale a

```
GROUP BY a, b, c UNION ALL  
GROUP BY a UNION ALL  
GROUP BY ()
```

Em que (b, c) é tratado como uma unidade e não é aplicado ROLLUP em (b, c). É como se você tivesse um apelido, por exemplo, z, para (b, c) e a expressão GROUP BY fosse reduzida a GROUP BY ROLLUP(a, z).

Observação: Normalmente, GROUP BY () é uma instrução SELECT com valores NULL para as colunas a e b, e apenas a function agregada. Quase sempre ela é usada para gerar totais gerais.

```
SELECT NULL, NULL, aggregate_col  
FROM <table_name>  
GROUP BY ( );
```

Colunas Compostas (continuação)

Compare com o ROLLUP normal, como em:

```
GROUP BY ROLLUP (a, b, c)
```

que seria

```
GROUP BY a, b, c UNION ALL
```

```
GROUP BY a, b UNION ALL
```

```
GROUP BY a UNION ALL
```

```
GROUP BY ()
```

Da mesma forma,

```
GROUP BY CUBE ((a, b), c)
```

seria equivalente a

```
GROUP BY a, b, c UNION ALL
```

```
GROUP BY a, b UNION ALL
```

```
GROUP BY c UNION ALL
```

```
GROUP BY ()
```

A tabela a seguir mostra a especificação de conjuntos de agrupamento e a especificação da cláusula GROUP BY equivalente.

| Instruções GROUPING SETS | Instruções GROUP BY Equivalentes |
|--|---|
| GROUP BY GROUPING SETS (a, b, c) | GROUP BY a UNION ALL GROUP BY b UNION ALL GROUP BY c |
| GROUP BY GROUPING SETS (a, b, (b, c)) (A expressão GROUPING SETS tem uma coluna composta.) | GROUP BY a UNION ALL GROUP BY b UNION ALL GROUP BY b, c |
| GROUP BY GROUPING SETS ((a, b, c)) | GROUP BY a, b, c |
| GROUP BY GROUPING SETS (a, (b), ()) | GROUP BY a UNION ALL GROUP BY b UNION ALL GROUP BY () |
| GROUP BY GROUPING SETS (a, ROLLUP (b, c)) (A expressão GROUPING SETS tem uma coluna composta.) | GROUP BY a UNION ALL GROUP BY ROLLUP (b, c) |

Colunas Compostas: Exemplo

```
SELECT  department_id, job_id, manager_id,
        SUM(salary)
FROM    employees
GROUP BY ROLLUP( department_id, (job_id, manager_id));
```

| DEPARTMENT ID | JOB ID | MANAGER ID | SUM(SALARY) |
|---------------|----------------|------------|-------------|
| 1 | SA_REP | 149 | 7000 |
| | | | 7000 |
| | 10 AD_ASST | 101 | 4400 |
| | 10 | | 4400 |
| | 20 MK_MAN | 100 | 13000 |
| | 20 MK_REP | 201 | 6000 |
| | 20 | | 19000 |
| ... | | | |
| | 100 FI_MGR | 101 | 12000 |
| | 100 FI_ACCOUNT | 108 | 39600 |
| | 100 | | 51600 |
| | 110 AC_MGR | 181 | 12000 |
| | 110 AC_ACCOUNT | 205 | 8300 |
| | 110 | | 20300 |
| | | | 691400 |

46 rows selected.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Colunas Compostas: Exemplo

Considere o exemplo:

```
SELECT department_id, job_id, manager_id, SUM(salary)
FROM employees
GROUP BY ROLLUP( department_id, job_id, manager_id);
```

Esta consulta resulta no cálculo do servidor Oracle dos seguintes agrupamentos:

1. (job_id, manager_id)
2. (department_id, job_id, manager_id)
3. (department_id)
4. Total geral

Se você estiver interessado apenas em grupos específicos, não poderá limitar o cálculo aos agrupamentos sem usar colunas compostas. Com colunas compostas, isso é possível tratando as colunas JOB_ID e MANAGER_ID como uma unidade durante o rollup. As colunas entre parênteses são tratadas como uma unidade no cálculo de ROLLUP e CUBE. Essa situação é ilustrada no exemplo do slide. Colocando as colunas JOB_ID e MANAGER_ID entre parênteses, você informa ao servidor Oracle para tratar JOB_ID e MANAGER_ID como uma unidade, ou seja, como uma coluna composta.

Colunas Compostas: Exemplo (continuação)

O exemplo do slide calcula os seguintes agrupamentos:

- (department_id, job_id, manager_id)
- (department_id)
- ()

O exemplo do slide exibe o seguinte:

- O salário total de cada cargo e gerente (indicado por 1)
- O salário total de cada departamento, cargo e gerente (indicado por 2)
- O salário total de cada departamento (indicado por 3)
- Total geral (indicado por 4)

O exemplo do slide também pode ser criado assim:

```
SELECT    department_id, job_id, manager_id, SUM(salary)
FROM      employees
GROUP BY  department_id, job_id, manager_id
UNION ALL
SELECT    department_id, TO_CHAR(NULL), TO_NUMBER(NULL), SUM(salary)
FROM      employees
GROUP BY  department_id
UNION ALL
SELECT    TO_NUMBER(NULL), TO_CHAR(NULL), TO_NUMBER(NULL), SUM(salary)
FROM      employees
GROUP BY  ();
```

Na ausência de um otimizador que verifique os blocos de consulta para gerar o plano de execução, a consulta precedente precisaria de três varreduras da tabela-base (EMPLOYEES). Isso pode ser muito ineficiente. Portanto, recomenda-se o uso de colunas compostas.

Agrupamentos Concatenados

- Os agrupamentos concatenados oferecem uma maneira concisa de gerar combinações úteis de agrupamentos.
- Para especificar conjuntos de agrupamentos concatenados, separe vários conjuntos de agrupamentos, bem como operações ROLLUP e CUBE, por vírgulas para que o servidor Oracle os combine em uma única cláusula GROUP BY.
- O resultado é um produto híbrido de agrupamentos de cada conjunto de agrupamentos.

```
GROUP BY GROUPING SETS(a, b), GROUPING SETS(c, d)
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

Colunas Concatenadas

Os agrupamentos concatenados oferecem uma maneira concisa de gerar combinações úteis de agrupamentos. Os agrupamentos concatenados são especificados pela listagem de vários conjuntos de agrupamentos, cubos e rollups, separados por vírgulas. Este é um exemplo de conjuntos de agrupamentos concatenados:

```
GROUP BY GROUPING SETS(a, b), GROUPING SETS(c, d)
```

Esta instrução SQL define os seguintes agrupamentos:

```
(a, c), (a, d), (b, c), (b, d)
```

A concatenação de conjuntos de agrupamentos é muito útil por estes motivos:

- **Facilidade de desenvolvimento de consultas:** não é necessário enumerar todos os agrupamentos manualmente.
- **Uso por aplicações:** a instrução SQL gerada por aplicações OLAP envolve, com frequência, a concatenação de conjuntos de agrupamentos; cada conjunto define os agrupamentos necessários para uma dimensão

Agrupamentos Concatenados: Exemplo

```
SELECT  department_id, job_id, manager_id,
        SUM(salary)
FROM    employees
GROUP BY department_id,
        ROLLUP(job_id),
        CUBE(manager_id);
```

| 1 | DEPARTMENT_ID | JOB_ID | MANAGER_ID | SUM(SALARY) |
|---|---------------|---------|------------|-------------|
| | | SA_REP | 149 | 7000 |
| | 10 | AD_ASST | 101 | 4400 |
| | 20 | MK_MAN | 100 | 13000 |
| | 20 | MK_REP | 201 | 6000 |
| 2 | ... | | | |
| | 90 | AD_VP | 100 | 34000 |
| | 90 | AD PRES | 149 | 24000 |
| | | | | 7000 |
| 3 | ... | | | |
| | | SA_REP | | 7000 |
| | 10 | AD_ASST | | 4400 |
| | ... | | | |
| 4 | | 110 | 101 | 12000 |
| | | 110 | 205 | 8300 |
| | | 110 | | 20300 |
| 5 | | | | |

93 rows selected.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Agrupamentos Concatenados: Exemplo

O exemplo do slide resulta nos seguintes agrupamentos:

- (job_id, manager_id) (1)
- (department_id, job_id, manager_id) (2)
- (job_id) (3)
- (department_id, manager_id) (4)
- (department_id) (5)

O salário total de cada um desses grupos é calculado.

Sumário

Nesta lição, você aprendeu a usar a:

- **operação ROLLUP para produzir valores de subtotais**
- **operação CUBE para produzir valores de tabelas de referência cruzada**
- **function GROUPING para identificar os valores das linhas criadas por ROLLUP ou CUBE**
- **sintaxe de GROUPING SETS para definir vários agrupamentos na mesma consulta**
- **cláusula GROUP BY para combinar expressões de várias formas:**
 - **Colunas compostas**
 - **Conjuntos de agrupamentos concatenados**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Sumário

- ROLLUP e CUBE são extensões da cláusula GROUP BY.
- ROLLUP é usado para exibir valores de subtotais e de totais gerais.
- CUBE é usado para exibir valores de tabelas de referência cruzada.
- A function GROUPING permite determinar se uma linha é um agregado produzido por um operador CUBE ou por um operador ROLLUP.
- Com a sintaxe de GROUPING SETS, é possível definir vários agrupamentos na mesma consulta. A cláusula GROUP BY calcula todos os agrupamentos especificados e combina-os com UNION ALL.
- Na cláusula GROUP BY, é possível combinar expressões de várias formas:
 - Para especificar colunas compostas, agrupe as colunas entre parênteses de modo que o servidor Oracle as trate como uma unidade durante o cálculo de operações ROLLUP ou CUBE.
 - Para especificar conjuntos de agrupamentos concatenados, separe vários conjuntos de agrupamentos, bem como operações ROLLUP e CUBE, por vírgulas para que o servidor Oracle os combine em uma única cláusula GROUP BY. O resultado é um produto híbrido de agrupamentos de cada conjunto de agrupamentos.

Exercício 4: Visão Geral

Este exercício abrange o uso de:

- **operadores ROLLUP**
- **operadores CUBE**
- **functions GROUPING**
- **GROUPING SETS**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Exercício 4: Visão Geral

Neste exercício, você usará os operadores ROLLUP e CUBE como extensões da cláusula GROUP BY. Você também usará GROUPING SETS.

Exercício 4

1. Crie uma consulta para exibir as seguintes informações sobre os funcionários cujo ID de gerente é menor que 120:
 - ID do gerente
 - ID do cargo e salário total para cada ID de cargo para funcionários que estão subordinados ao mesmo gerente
 - Salário total desses gerentes
 - Salário total desses gerentes, independentemente dos IDs dos cargos

| MANAGER_ID | JOB_ID | SUM(SALARY) |
|------------|------------|-------------|
| 100 | AD_VP | 34000 |
| 100 | MK_MAN | 13000 |
| 100 | PU_MAN | 11000 |
| 100 | SA_MAN | 61000 |
| 100 | ST_MAN | 36400 |
| 100 | | 155400 |
| 101 | AC_MGR | 12000 |
| 101 | FI_MGR | 12000 |
| 101 | HR_REP | 6500 |
| 101 | PR_REP | 10000 |
| 101 | AD_ASST | 4400 |
| ... | | |
| 101 | AD_ASST | 4400 |
| 101 | | 44900 |
| 102 | IT_PROG | 9000 |
| 102 | | 9000 |
| 103 | IT_PROG | 19800 |
| 103 | | 19800 |
| 108 | FI_ACCOUNT | 39600 |
| 108 | | 39600 |
| 114 | PU_CLERK | 13900 |
| 114 | | 13900 |
| | | 282600 |

21 rows selected.

Exercício 4 (continuação)

- Observe a resposta da questão 1. Crie uma consulta usando a function GROUPING para determinar se os valores NULL nas colunas correspondentes às expressões GROUP BY são causados pela operação ROLLUP.

| MGR | JOB | SUM(SALARY) | GROUPING(MANAGER_ID) | GROUPING(JOB_ID) |
|-----|------------|-------------|----------------------|------------------|
| 100 | AD_VP | 34000 | 0 | 0 |
| 100 | MK_MAN | 13000 | 0 | 0 |
| 100 | PU_MAN | 11000 | 0 | 0 |
| 100 | SA_MAN | 61000 | 0 | 0 |
| 100 | ST_MAN | 36400 | 0 | 0 |
| 100 | | 155400 | 0 | 1 |
| ... | | | | |
| 102 | IT_PROG | 9000 | 0 | 0 |
| 102 | | 9000 | 0 | 1 |
| 103 | IT_PROG | 19800 | 0 | 0 |
| 103 | | 19800 | 0 | 1 |
| 108 | FI_ACCOUNT | 39600 | 0 | 0 |
| 108 | | 39600 | 0 | 1 |
| 114 | PU_CLERK | 13900 | 0 | 0 |
| 114 | | 13900 | 0 | 1 |
| | | 282600 | 1 | 1 |

21 rows selected.

Exercício 4 (continuação)

3. Crie uma consulta para exibir as seguintes informações sobre os funcionários cujo ID de gerente é menor que 120:
- ID do gerente
 - Cargo e salários totais de cada cargo para funcionários que estão subordinados ao mesmo gerente
 - Salário total desses gerentes
 - Valores de tabelas de referência para exibir o salário total para cada cargo, independentemente do gerente
 - Salário total, independentemente dos cargos

| MANAGER_ID | JOB_ID | SUM(SALARY) |
|------------|--------|-------------|
| | | 282600 |
| | AD_VP | 34000 |
| | AC_MGR | 12000 |
| | FI_MGR | 12000 |
| | HR_REP | 6500 |
| ... | MK_MAN | 13000 |

| MANAGER_ID | JOB_ID | SUM(SALARY) |
|------------|------------|-------------|
| 101 | PR_REP | 10000 |
| 101 | AD_ASST | 4400 |
| 102 | | 9000 |
| 102 | IT_PROG | 9000 |
| 103 | | 19800 |
| 103 | IT_PROG | 19800 |
| 108 | | 39600 |
| 108 | FI_ACCOUNT | 39600 |
| 114 | | 13900 |
| 114 | PU_CLERK | 13900 |

34 rows selected.

Exercício 4 (continuação)

4. Observe a resposta da questão 3. Crie uma consulta usando a function GROUPING para determinar se os valores NULL nas colunas correspondentes às expressões GROUP BY são causados pela operação CUBE.

| MGR | JOB | SUM(SALARY) | GROUPING(MANAGER_ID) | GROUPING(JOB_ID) |
|-----|------------|-------------|----------------------|------------------|
| | | 282600 | 1 | 1 |
| | AD_VP | 34000 | 1 | 0 |
| | AC_MGR | 12000 | 1 | 0 |
| | FI_MGR | 12000 | 1 | 0 |
| | HR_REP | 6500 | 1 | 0 |
| | MK_MAN | 13000 | 1 | 0 |
| | PR_REP | 10000 | 1 | 0 |
| | PU_MAN | 11000 | 1 | 0 |
| | SA_MAN | 61000 | 1 | 0 |
| | ST_MAN | 36400 | 1 | 0 |
| | AD_ASST | 4400 | 1 | 0 |
| | IT_PROG | 28800 | 1 | 0 |
| | PU_CLERK | 13900 | 1 | 0 |
| | FI_ACCOUNT | 39600 | 1 | 0 |
| 100 | | 155400 | 0 | 1 |

...

| MGR | JOB | SUM(SALARY) | GROUPING(MANAGER_ID) | GROUPING(JOB_ID) |
|-----|------------|-------------|----------------------|------------------|
| 101 | PR_REP | 10000 | 0 | 0 |
| 101 | AD_ASST | 4400 | 0 | 0 |
| 102 | | 9000 | 0 | 1 |
| 102 | IT_PROG | 9000 | 0 | 0 |
| 103 | | 19800 | 0 | 1 |
| 103 | IT_PROG | 19800 | 0 | 0 |
| 108 | | 39600 | 0 | 1 |
| 108 | FI_ACCOUNT | 39600 | 0 | 0 |
| 114 | | 13900 | 0 | 1 |
| 114 | PU_CLERK | 13900 | 0 | 0 |

34 rows selected.

Exercício 4 (continuação)

5. Usando GROUPING SETS, crie uma consulta para exibir os seguintes agrupamentos:

- department_id, manager_id, job_id
- department_id, job_id
- manager_id, job_id

A consulta deve calcular a soma dos salários para cada um desses grupos.

| DEPARTMENT_ID | MANAGER_ID | JOB_ID | SUM(SALARY) |
|---------------|------------|------------|-------------|
| 90 | | AD_PRES | 24000 |
| 90 | 100 | AD_VP | 34000 |
| 20 | 100 | MK_MAN | 13000 |
| 30 | 100 | PU_MAN | 11000 |
| 80 | 100 | SA_MAN | 61000 |
| 50 | 100 | ST_MAN | 36400 |
| 110 | 101 | AC_MGR | 12000 |
| 100 | 101 | FI_MGR | 12000 |
| ... | | AD_PRES | 24000 |
| | 100 | AD_VP | 34000 |
| | 100 | MK_MAN | 13000 |
| | 100 | PU_MAN | 11000 |
| ... | | SA_REP | 7000 |
| 10 | | AD_ASST | 4400 |
| 20 | | MK_MAN | 13000 |
| 20 | | MK_REP | 6000 |
| ... | | ST_MAN | 36400 |
| 50 | | SH_CLERK | 64300 |
| 50 | | ST_CLERK | 55700 |
| 60 | | IT_PROG | 28800 |
| 70 | | PR_REP | 10000 |
| 80 | | SA_MAN | 61000 |
| 80 | | SA_REP | 243500 |
| 90 | | AD_VP | 34000 |
| 90 | | AD_PRES | 24000 |
| 100 | | FI_MGR | 12000 |
| 100 | | FI_ACCOUNT | 39600 |
| 110 | | AC_MGR | 12000 |
| 110 | | AC_ACCOUNT | 8300 |

85 rows selected.

5

Gerenciando Dados em Diferentes Fusos Horários

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Objetivos

Ao concluir esta lição, você será capaz de usar as seguintes functions de data/horário:

- TZ_OFFSET
- FROM_TZ
- TO_TIMESTAMP
- TO_TIMESTAMP_TZ
- TO_YMINTERVAL
- CURRENT_DATE
- CURRENT_TIMESTAMP
- LOCALTIMESTAMP
- DBTIMEZONE
- SESSIONTIMEZONE
- EXTRACT

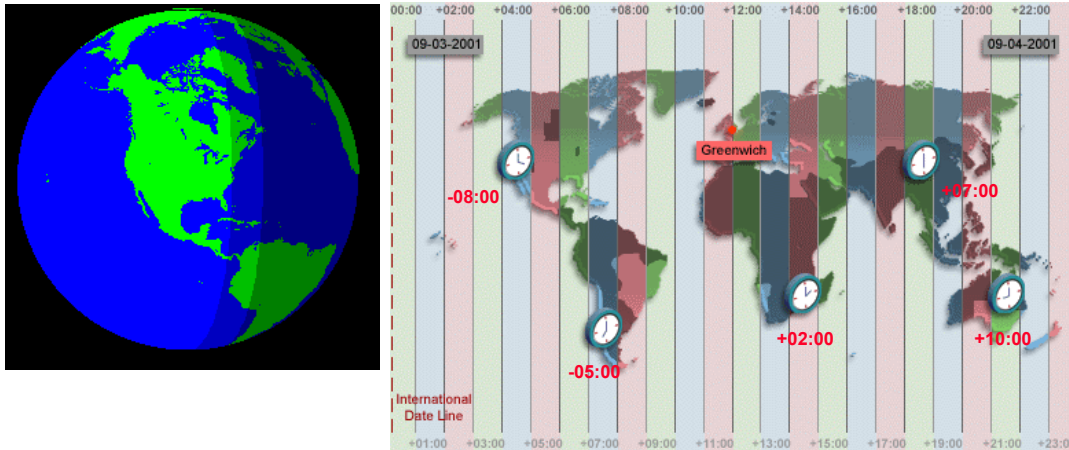
ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Objetivos

Esta lição aborda algumas das functions de data/horário disponíveis no banco de dados Oracle.

Fusos Horários



A imagem representa os horários de cada área quando for 12:00 em Greenwich.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Fusos Horários

As horas do dia são medidas de acordo com o movimento da Terra. O horário relativo a um momento em particular depende de onde você está. Quando é meio-dia em Greenwich, Inglaterra, é meia-noite na linha de data internacional. A Terra divide-se em 24 fusos horários, um para cada hora do dia. A hora no meridiano principal em Greenwich, Inglaterra, é conhecida como GMT (Greenwich Mean Time). GMT é o horário padrão pelo qual todos os outros fusos horários do mundo se orientam. O GMT é o mesmo durante o ano inteiro e não é afetado pelo horário de verão. A linha do meridiano é uma linha imaginária que vai do Pólo Norte ao Pólo Sul. Ela é conhecida como longitude zero e é a partir dela que se medem as demais linhas de longitude. Todos os horários são medidos em relação ao GMT, e todos os locais estão associados a uma latitude (a distância ao norte ou ao sul do Equador) e uma longitude (a distância a leste ou a oeste do meridiano de Greenwich).

Parâmetro de Sessão TIME_ZONE

O parâmetro de sessão TIME_ZONE pode ser definido como:

- Um deslocamento absoluto
- Um fuso horário do banco de dados
- Um fuso horário local do sistema operacional
- Determinada região

```
ALTER SESSION SET TIME_ZONE = '-05:00';  
ALTER SESSION SET TIME_ZONE = dbtimezone;  
ALTER SESSION SET TIME_ZONE = local;  
ALTER SESSION SET TIME_ZONE = 'America/New_York';
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Parâmetro de Sessão TIME_ZONE

O banco de dados Oracle suporta o armazenamento do fuso horário nos seus dados de data e horário, bem como frações de segundos. O comando ALTER SESSION pode ser usado para alterar os valores de fuso horário em uma sessão do usuário. Os valores de fuso horário podem ser definidos como um deslocamento absoluto, um fuso horário determinado, um fuso horário do banco de dados ou o fuso horário local.

CURRENT_DATE, CURRENT_TIMESTAMP e LOCALTIMESTAMP

- **CURRENT_DATE**
 - Retorna a data atual do sistema
 - Tem o tipo de dados de DATE
- **CURRENT_TIMESTAMP**
 - Retorna o timestamp atual do sistema
 - Tem um tipo de dados de TIMESTAMP WITH TIME ZONE
- **LOCALTIMESTAMP**
 - Retorna o timestamp atual da sessão do usuário
 - Tem um tipo de dados de TIMESTAMP

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

CURRENT_DATE, CURRENT_TIMESTAMP e LOCALTIMESTAMP

As functions `CURRENT_DATE` e `CURRENT_TIMESTAMP` retornam a data e o timestamp atuais, respectivamente. O tipo de dados de `CURRENT_DATE` é `DATE`. O tipo de dados de `CURRENT_TIMESTAMP` é `TIMESTAMP WITH TIME ZONE`. Os valores retornados exibem o deslocamento de fuso horário da sessão SQL que está executando as functions. O deslocamento de fuso horário representa a diferença (em horas e minutos) entre o horário local e o UTC. O tipo de dados `TIMESTAMP WITH TIME ZONE` tem o formato:

`TIMESTAMP [(fractional_seconds_precision)] WITH TIME ZONE`

onde `fractional_seconds_precision` especifica, opcionalmente, o número de dígitos da parte fracionária do campo de data/horário `SECOND` e pode ser um número entre 0 e 9. O default é 6.

A function `LOCALTIMESTAMP` retorna a data e o horário atuais do fuso horário da sessão. A diferença entre `LOCALTIMESTAMP` e `CURRENT_TIMESTAMP` é que `LOCALTIMESTAMP` retorna um valor `TIMESTAMP`, enquanto `CURRENT_TIMESTAMP` retorna um valor `TIMESTAMP WITH TIME ZONE`.

Essas functions utilizam NLS, ou seja, os resultados terão os formatos NLS atuais de calendário e data/horário.

CURRENT_DATE

Exibe a data e a hora atuais no fuso horário da sessão.

```
ALTER SESSION  
SET NLS_DATE_FORMAT = 'DD-MON-YYYY HH24:MI:SS';
```

```
ALTER SESSION SET TIME_ZONE = '-5:0';  
SELECT SESSIONTIMEZONE, CURRENT_DATE FROM DUAL;
```

| SESSIONTIMEZONE | CURRENT_DATE |
|-----------------|----------------------|
| -05:00 | 03-OCT-2001 09:37:06 |

```
ALTER SESSION SET TIME_ZONE = '-8:0';  
SELECT SESSIONTIMEZONE, CURRENT_DATE FROM DUAL;
```

| SESSIONTIMEZONE | CURRENT_DATE |
|-----------------|----------------------|
| -08:00 | 03-OCT-2001 06:38:07 |

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

CURRENT_DATE

A function `CURRENT_DATE` retorna a data atual no fuso horário da sessão. O valor retornado é uma data do calendário gregoriano.

Os exemplos do slide mostram que `CURRENT_DATE` é sensível ao fuso horário da sessão. No primeiro exemplo, a sessão é alterada para definir o parâmetro `TIME_ZONE` como `-5:0`. O parâmetro `TIME_ZONE` especifica o deslocamento do fuso horário local default para a sessão SQL atual. `TIME_ZONE` é somente um parâmetro de sessão, e não um parâmetro de inicialização. O parâmetro `TIME_ZONE` é definido da seguinte maneira:

`TIME_ZONE = '[+ | -] hh:mm'`

A máscara de formato (`[+ | -] hh:mm`) indica as horas e os minutos antes ou depois do UTC (Coordinated Universal Time, anteriormente conhecido como Greenwich Mean Time).

Observe na saída que o valor de `CURRENT_DATE` é alterado quando o valor do parâmetro `TIME_ZONE` é modificado para `-8:0` no segundo exemplo.

Observação: O comando `ALTER SESSION` define o formato de data da sessão como `'DD-MON-YYYY HH24:MI:SS'`, isto é, dia do mês (1 a 31)-nome abreviado do mês-ano com 4 dígitos hora do dia (0 a 23):minuto (0 a 59):segundo (0 a 59).

CURRENT_TIMESTAMP

Exibe a data e a fração do horário atuais no fuso horário da sessão.

```
ALTER SESSION SET TIME_ZONE = '-5:0';  
SELECT SESSIONTIMEZONE, CURRENT_TIMESTAMP  
FROM DUAL;
```

| SESSIONTIMEZONE | CURRENT_TIMESTAMP |
|-----------------|-------------------------------------|
| -05:00 | 03-OCT-01 09.40.59.000000 AM -05:00 |

```
ALTER SESSION SET TIME_ZONE = '-8:0';  
SELECT SESSIONTIMEZONE, CURRENT_TIMESTAMP  
FROM DUAL;
```

| SESSIONTIMEZONE | CURRENT_TIMESTAMP |
|-----------------|-------------------------------------|
| -08:00 | 03-OCT-01 06.41.38.000000 AM -08:00 |

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

CURRENT_TIMESTAMP

A function `CURRENT_TIMESTAMP` retorna a data e o horário atuais no fuso horário da sessão como um valor do tipo de dados `TIMESTAMP WITH TIME ZONE`. O deslocamento do fuso horário reflete o horário local atual da sessão SQL. A sintaxe da function `CURRENT_TIMESTAMP` é:

`CURRENT_TIMESTAMP (precision)`

em que *precision* é um argumento opcional que especifica a precisão de frações de segundos do valor do horário retornado. Se você omitir a precisão, o default será 6.

Os exemplos do slide mostram que `CURRENT_TIMESTAMP` é sensível ao fuso horário da sessão. No primeiro exemplo, a sessão é alterada para definir o parâmetro `TIME_ZONE` como `-5:0`. Observe na saída que o valor de `CURRENT_TIMESTAMP` é alterado quando o valor do parâmetro `TIME_ZONE` é modificado para `-8:0` no segundo exemplo.

LOCALTIMESTAMP

- **Exibe a data e o horário atuais no fuso horário da sessão em um valor de tipo de dados `TIMESTAMP`.**

```
ALTER SESSION SET TIME_ZONE = '-5:0';  
SELECT CURRENT_TIMESTAMP, LOCALTIMESTAMP  
FROM DUAL;
```

| CURRENT_TIMESTAMP | LOCALTIMESTAMP |
|-------------------------------------|------------------------------|
| 03-OCT-01 09.44.21.000000 AM -05:00 | 03-OCT-01 09.44.21.000000 AM |

```
ALTER SESSION SET TIME_ZONE = '-8:0';  
SELECT CURRENT_TIMESTAMP, LOCALTIMESTAMP  
FROM DUAL;
```

| CURRENT_TIMESTAMP | LOCALTIMESTAMP |
|-------------------------------------|------------------------------|
| 03-OCT-01 06.45.21.000001 AM -08:00 | 03-OCT-01 06.45.21.000001 AM |

- **`LOCALTIMESTAMP` retorna um valor `TIMESTAMP`, enquanto `CURRENT_TIMESTAMP` retorna um valor `TIMESTAMP WITH TIME ZONE`.**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

LOCALTIMESTAMP

A function `LOCALTIMESTAMP` retorna a data e o horário atuais no fuso horário da sessão. `LOCALTIMESTAMP` retorna um valor `TIMESTAMP`. A sintaxe da function `LOCAL_TIMESTAMP` é:

```
LOCAL_TIMESTAMP (TIMESTAMP_precision)
```

Em que `TIMESTAMP_precision` é um argumento opcional que especifica a precisão de frações de segundos do valor de `TIMESTAMP` retornado.

Os exemplos do slide ilustram a diferença entre `LOCALTIMESTAMP` e `CURRENT_TIMESTAMP`. Observe que `LOCALTIMESTAMP` não exibe o valor de fuso horário, enquanto `CURRENT_TIMESTAMP` exibe.

DBTIMEZONE e SESSIONTIMEZONE

- **Exibe o valor do fuso horário do banco de dados.**

```
SELECT DBTIMEZONE FROM DUAL;
```

| DBTIME |
|--------|
| -05:00 |

- **Exibe o valor do fuso horário da sessão.**

```
SELECT SESSIONTIMEZONE FROM DUAL;
```

| SESSIONTIMEZONE |
|-----------------|
| -08:00 |

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

DBTIMEZONE e SESSIONTIMEZONE

O DBA define o fuso horário default do banco de dados, especificando a cláusula `SET TIME_ZONE` da instrução `CREATE DATABASE`. Se omitido, o fuso horário default do banco de dados passa a ser o fuso horário do sistema operacional. O fuso horário do banco de dados não pode ser alterado para uma sessão com uma instrução `ALTER SESSION`.

A function `DBTIMEZONE` retorna o valor do fuso horário do banco de dados. O tipo retornado é um deslocamento do fuso horário (um tipo de caractere no formato '`[+ | -] TZH:TZM`') ou o nome de uma região de fuso horário, dependendo de como você especificou o valor do fuso horário do banco de dados na instrução `CREATE DATABASE` ou `ALTER DATABASE` mais recente. O exemplo do slide mostra que o horário do banco de dados está definido como "-05:00", com o parâmetro `TIME_ZONE` no formato:

`TIME_ZONE = '[+ | -] hh:mm'`

A function `SESSIONTIMEZONE` retorna o valor do fuso horário da sessão atual. O tipo retornado é um deslocamento do fuso horário (um tipo de caractere no formato '`[+ | -] TZH:TZM`') ou o nome de uma região de fuso horário, dependendo de como você especificou o valor do fuso horário da sessão na instrução `ALTER SESSION` mais recente. O exemplo do slide mostra que o fuso horário da sessão está deslocado em relação ao UTC em -8 horas. Observe que o fuso horário do banco de dados é diferente do fuso horário da sessão atual.

Tipo de Dados TIMESTAMP

- O tipo de dados **TIMESTAMP** é uma extensão do tipo de dados **DATE**.
- Ele armazena o ano, o mês e o dia do tipo de dados **DATE**, além dos valores de hora, minuto, segundo e fração de segundo.
- As variações de **TIMESTAMP** são:
 - **TIMESTAMP**
[(fractional_seconds_precision)]
 - **TIMESTAMP**
[(fractional_seconds_precision)]
WITH TIME ZONE
 - **TIMESTAMP**
[(fractional_seconds_precision)]
WITH LOCAL TIME ZONE

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Tipos de Dados de Data/Horário

O tipo de dados **TIMESTAMP** contém os campos de data/horário **YEAR**, **MONTH**, **DAY**, **HOURL**, **MINUTE** e **SECOND** e frações de segundo.

O tipo de dados **TIMESTAMP WITH TIME ZONE** contém os campos de data/horário **HOURL**, **MINUTE**, **SECOND**, **TIMEZONE_HOUR** e **TIMEZONE_MINUTE** e frações de segundo.

O tipo de dados **TIMESTAMP WITH TIME ZONE** contém os campos de data/horário **YEAR**, **MONTH**, **DAY**, **HOURL**, **MINUTE**, **SECOND**, **TIMEZONE_HOUR** e **TIMEZONE_MINUTE** e frações de segundo.

Observação: A precisão de frações de segundos especifica o número de dígitos da parte fracionária do campo de data/horário **SECOND** e pode ser um número na faixa entre 0 e 9. O default é 6.

Tipos de Dados **TIMESTAMP**

| Tipo de Dados | Campos |
|---------------------------------------|---|
| TIMESTAMP | Year, Month, Day, Hour, Minute, Second com frações de segundos |
| TIMESTAMP WITH TIME ZONE | Year, Month, Day, Hour, Minute, Second com frações de segundos, TimeZone_Hour e TimeZone_Minute ou TimeZone_Region |
| TIMESTAMP WITH LOCAL TIME ZONE | Year, Month, Day, Hour, Minute, Second com frações de segundos, |

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Tipos de Dados **TIMESTAMP**

TIMESTAMP (fractional_seconds_precision)

Este tipo de dados contém os valores de ano, mês e dia da data, bem como os valores de hora, minuto e segundo do horário, onde a precisão significativa de frações de segundos é o número de dígitos da parte fracionária do campo de data/horário **SECOND**. Os valores aceitos para a precisão significativa de frações de segundos estão entre 0 e 9. O default é 6.

TIMESTAMP (fractional_seconds_precision) WITH TIME ZONE

Este tipo de dados contém todos os valores de **TIMESTAMP**, bem como o valor de deslocamento de fuso horário.

TIMESTAMP (fractional_seconds_precision) WITH LOCAL TIME ZONE

Este tipo de dados contém todos os valores de **TIMESTAMP**, com as seguintes exceções:

- Os dados são normalizados para o fuso horário do banco de dados quando são armazenados no banco de dados.
- Quando os dados são recuperados, os usuários vêem os dados no fuso horário da sessão.

Campos TIMESTAMP

| Campo de Data/Horário | Valores Válidos |
|-----------------------|---|
| YEAR | de -4712 a 9999 (excluindo-se o ano 0) |
| MONTH | de 01 a 12 |
| DAY | de 01 a 31 |
| HOURL | de 00 a 23 |
| MINUTE | de 00 a 59 |
| SECOND | de 00 a 59,9(N), em que 9(N) é precisão |
| TIMEZONE_HOUR | de -12 a 14 |
| TIMEZONE_MINUTE | de 00 a 59 |

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Campos TIMESTAMP

Cada tipo de dados é composto por vários desses campos. Datas/horários só podem ser mutuamente comparados e designados se tiverem os mesmos campos de data/horário.

Diferença entre DATE e TIMESTAMP

A

```
-- quando hire_date  
é do tipo DATE
```

```
SELECT hire_date  
FROM emp5;
```

| HIRE_DATE |
|-----------|
| 17-JUN-87 |
| 21-SEP-89 |
| 13-JAN-93 |
| 03-JAN-90 |
| 21-MAY-91 |
| 25-JUN-97 |
| 05-FEB-98 |
| 07-FEB-99 |
| 17-AUG-94 |
| 16-AUG-94 |
| 28-SEP-97 |

...

B

```
ALTER TABLE emp5  
MODIFY hire_date TIMESTAMP;
```

```
SELECT hire_date  
FROM emp5;
```

| HIRE_DATE |
|------------------------------|
| 17-JUN-87 12.00.00.000000 AM |
| 21-SEP-89 12.00.00.000000 AM |
| 13-JAN-93 12.00.00.000000 AM |
| 03-JAN-90 12.00.00.000000 AM |
| 21-MAY-91 12.00.00.000000 AM |
| 25-JUN-97 12.00.00.000000 AM |
| 05-FEB-98 12.00.00.000000 AM |
| 07-FEB-99 12.00.00.000000 AM |
| 17-AUG-94 12.00.00.000000 AM |
| 16-AUG-94 12.00.00.000000 AM |
| 28-SEP-97 12.00.00.000000 AM |
| 30-SEP-97 12.00.00.000000 AM |

...

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

Tipo de Dados TIMESTAMP: Exemplo

No slide, o exemplo A mostra os dados da coluna hire_date da tabela EMP5 quando o tipo de dados da coluna é DATE. No exemplo B, a tabela é alterada, e o tipo de dados da coluna hire_date passa a ser TIMESTAMP. A saída mostra as diferenças na exibição. Você pode converter DATE em TIMESTAMP quando a coluna tem dados, mas não é possível converter DATE ou TIMESTAMP em TIMESTAMP WITH TIME ZONE, a menos que a coluna esteja vazia.

Você pode especificar a precisão de frações de segundos para timestamp. Se não for especificada nenhuma precisão, como no exemplo acima, o default 6 será assumido.

Por exemplo, a seguinte instrução define a precisão de frações de segundos como 7:

```
ALTER TABLE emp5  
MODIFY hire_date TIMESTAMP(7);
```

Observação: Por default, o tipo de dados date do Oracle aparece como mostra o exemplo. No entanto, o tipo de dados date também contém informações adicionais, como horas, minutos, segundos, a.m. e p.m. Para obter a data nesse formato, você pode aplicar uma máscara de formato ou uma function para o valor de data.

Tipo de Dados **TIMESTAMP WITH TIME ZONE**

- O tipo de dados **TIMESTAMP WITH TIME ZONE** é uma variante de **TIMESTAMP** cujo valor inclui um deslocamento de fuso horário.
- O deslocamento de fuso horário representa a diferença, em horas e minutos, entre o horário local e o UTC.
- Ele é especificado como:

```
TIMESTAMP [(fractional_seconds_precision)] WITH  
TIME ZONE
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Tipo de Dados **TIMESTAMP WITH TIME ZONE**

UTC significa Coordinated Universal Time (anteriormente denominado Greenwich Mean Time). Dois valores de **TIMESTAMP WITH TIME ZONE** serão considerados idênticos se representarem o mesmo instante no UTC, independentemente dos deslocamentos de **TIME ZONE** armazenados nos dados. Por exemplo:

```
TIMESTAMP '1999-04-15 8:00:00 -8:00'
```

é o mesmo que

```
TIMESTAMP '1999-04-15 11:00:00 -5:00'
```

Isto é, 8:00 a.m. Pacific Standard Time é o mesmo que 11:00 a.m. Eastern Standard Time.

Também é possível especificar esse valor como:

```
TIMESTAMP '1999-04-15 8:00:00 US/Pacific'
```

TIMESTAMP WITH TIMEZONE: Exemplo

```
CREATE TABLE web_orders  
(ord_id number primary key,  
 order_date TIMESTAMP WITH TIME ZONE);
```

```
INSERT INTO web_orders values  
(ord_seq.nextval, current_date);
```

```
SELECT * FROM web_orders;
```

| ORD_ID | ORDER_DATE |
|--------|-------------------------------------|
| 100 | 09-FEB-04 07:04:44.000000 AM -07:00 |

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

TIMESTAMP WITH TIME ZONE Exemplo

No exemplo do slide, é criada uma nova tabela `web_orders` com uma coluna de tipo de dados `TIMESTAMP WITH TIME ZONE`. Essa tabela será preenchida sempre que for feito um pedido na Web. O timestamp e o fuso horário do usuário que está fazendo o pedido são inseridos com base no valor de `CURRENT_DATE`. Desse modo, quando uma empresa baseada na Web garantir o envio, será possível estimar o horário da entrega com base no fuso horário da pessoa que está fazendo o pedido.

TIMESTAMP WITH LOCAL TIMEZONE

- O tipo de dados **TIMESTAMP WITH LOCAL TIMEZONE** é outra variante cujo valor inclui um deslocamento do fuso horário.
- Os dados armazenados no banco de dados são normalizados para o fuso horário do banco de dados.
- O deslocamento de fuso horário não é armazenado como parte dos dados da coluna.
- O banco de dados Oracle retorna os dados no fuso horário da sessão local do usuário.
- O tipo de dados **TIMESTAMP WITH LOCAL TIMEZONE** é especificado da seguinte maneira:

```
TIMESTAMP[(fractional_seconds_precision)] WITH  
LOCAL TIME ZONE
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

TIMESTAMP WITH LOCAL TIMEZONE

Diferentemente de **TIMESTAMP WITH TIMEZONE**, você pode especificar colunas do tipo **TIMESTAMP WITH LOCAL TIMEZONE** como parte de uma chave primária ou exclusiva. O deslocamento de fuso horário representa a diferença (em horas e minutos) entre o horário local e o UTC. Não existe um literal para **TIMESTAMP WITH LOCAL TIMEZONE**.

TIMESTAMP WITH LOCAL TIMEZONE: Exemplo

```
CREATE TABLE shipping (delivery_time TIMESTAMP WITH  
LOCAL TIME ZONE);  
INSERT INTO shipping VALUES(current_timestamp + 2);
```

```
SELECT * FROM shipping;
```

DELIVERY_TIME

11-FEB-04 07.09.02.000000 AM

```
ALTER SESSION SET TIME_ZONE = 'EUROPE/LONDON';  
SELECT * FROM shipping;
```

DELIVERY_TIME

11-FEB-04 02.09.02.000000 PM

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

TIMESTAMP WITH LOCAL TIME ZONE: Exemplo

No exemplo do slide, é criada uma nova tabela SHIPPING com uma coluna de tipo de dados TIMESTAMP WITH LOCAL TIME ZONE. Essa tabela será preenchida com a inserção de dois dias do valor de CURRENT_TIMESTAMP sempre que for feito um pedido. A saída da tabela DATE_TAB mostra que os dados são armazenados sem o deslocamento de fuso horário. A seguir, é executado o comando ALTER SESSION para alterar o fuso horário para o local da entrega. Uma segunda consulta à mesma tabela mostrará os dados com o fuso horário local refletido no valor de horário, de modo que o cliente possa ser notificado sobre o horário de entrega estimado.

Tipos de Dados INTERVAL

- Os tipos de dados INTERVAL são usados para armazenar a diferença entre dois valores de data/horário.
- Há dois tipos de intervalos:
 - Ano-mês
 - Dia-horário
- A precisão do intervalo é:
 - O subconjunto atual de campos que formam um intervalo
 - Especificada no qualificador de intervalo

| Tipo de Dados | Campos |
|------------------------|--|
| INTERVAL YEAR TO MONTH | Year, Month |
| INTERVAL DAY TO SECOND | Days, Hour, Minute, Second com frações de segundos |

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

Tipos de Dados INTERVAL

Os tipos de dados INTERVAL são usados para armazenar a diferença entre dois valores de data/horário. Há dois tipos de intervalos: intervalos ano-mês e intervalos dia-horário. Um intervalo ano-mês é composto por um subconjunto contíguo de campos de YEAR e MONTH, enquanto um intervalo dia-horário é composto por um subconjunto contíguo de campos como DAY, HOUR, MINUTE e SECOND. O subconjunto atual de campos que formam um intervalo é chamado de precisão do intervalo e é especificado no qualificador de intervalo. Como o número de dias no ano depende do calendário, o intervalo ano-mês depende do NLS, enquanto o intervalo dia-horário não depende do NLS.

O qualificador de intervalo também pode especificar a precisão do primeiro campo, que é o número de dígitos do primeiro (ou único) campo, e, caso o segundo campo seja SECOND, também poderá especificar uma precisão de frações de segundos, que é o número de dígitos na parte fracionária do valor de SECOND. Se não for especificado, o valor default para a precisão do primeiro campo será de 2 dígitos, e o valor default para a precisão de frações de segundos será de 6 dígitos.

Tipos de Dados INTERVAL (continuação)

INTERVAL YEAR (year_precision) TO MONTH

Este tipo de dados armazena um período em anos e meses, em que `year_precision` é o número de dígitos no campo data/horário YEAR. Os valores aceitos vão de 0 a 9. O default é 6.

INTERVAL DAY (day_precision) TO SECOND (fractional_seconds_precision)

Este tipo de dados armazena um período em dias, horas, minutos e segundos, em que `day_precision` é o número máximo de dígitos no campo de data/horário DAY (os valores aceitos vão de 0 a 9; o default é 2), e `fractional_seconds_precision` é o número de dígitos na parte fracionária do campo SECOND. Os valores aceitos vão de 0 a 9. O default é 6.

Campos INTERVAL

| Campo INTERVAL | Valores Válidos para o Intervalo |
|----------------|--|
| YEAR | Qualquer inteiro, positivo ou negativo |
| MONTH | de 00 a 11 |
| DAY | Qualquer inteiro, positivo ou negativo |
| HOURL | de 00 a 23 |
| MINUTE | de 00 a 59 |
| SECOND | de 00 a 59,9(N), onde 9(N) é precisão |

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Campos INTERVAL

INTERVAL YEAR TO MONTH pode ter os campos YEAR e MONTH.

INTERVAL DAY TO SECOND pode ter os campos DAY, HOUR, MINUTE e SECOND.

O subconjunto atual de campos que formam um item de um dos tipos de intervalo é definido por um qualificador de intervalo, e este subconjunto é conhecido como precisão do item.

Os intervalos ano-mês são mutuamente comparados e designados apenas em relação a outros intervalos ano-mês, e os intervalos dia-horário são mutuamente comparados e designados apenas em relação a outros intervalos dia-horário.

Tipo de Dados INTERVAL YEAR TO MONTH

O tipo de dados INTERVAL YEAR TO MONTH armazena um período usando os campos de data/horário YEAR e MONTH.

```
INTERVAL YEAR [(year_precision)] TO MONTH
```

- **Por exemplo:**

```
'312-2' designado a INTERVAL YEAR(3) TO MONTH  
Indica um intervalo de 312 anos e 2 meses
```

```
'312-0' designado a INTERVAL YEAR(3) TO MONTH  
Indica 312 anos e 0 meses
```

```
'0-3' designado a INTERVAL YEAR TO MONTH  
Indica um intervalo de 3 meses
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Tipo de Dados INTERVAL YEAR TO MONTH

O tipo de dados INTERVAL YEAR TO MONTH armazena um período usando os campos de data/horário YEAR e MONTH. Especifique esse tipo de dados da seguinte maneira:

```
INTERVAL YEAR [(year_precision)] TO MONTH
```

em que year_precision é o número de dígitos no campo de data/horário YEAR. O valor default de year_precision é 2.

Restrição: O primeiro campo deve conter um valor maior que o segundo. Por exemplo, INTERVAL '0-1' MONTH TO YEAR não é válido.

O próximo literal INTERVAL YEAR TO MONTH indica um intervalo de 123 anos, 3 meses:

- INTERVAL '123-3' YEAR(3) TO MONTH
- INTERVAL '123' YEAR(3) indica um intervalo de 123 anos e 0 meses.
- INTERVAL '3' MONTH indica um intervalo de 3 meses.

INTERVAL YEAR TO MONTH Exemplo

```
CREATE TABLE warranty
(prod_id number, warranty_time INTERVAL YEAR(3)
TO MONTH);
INSERT INTO warranty VALUES (123, INTERVAL '8'
MONTH);
INSERT INTO warranty VALUES (155, INTERVAL '200'
YEAR(3));
INSERT INTO warranty VALUES (678, '200-11');
SELECT * FROM warranty;
```

| PROD_ID | WARRANTY_TIME |
|---------|---------------|
| 123 | +000-08 |
| 155 | +200-00 |
| 678 | +200-11 |

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Tipo de Dados INTERVAL YEAR TO MONTH (continuação)

O tipo de dados INTERVAL YEAR TO MONTH armazena um período usando os campos de data/horário YEAR e MONTH. Especifique INTERVAL YEAR TO MONTH da seguinte maneira:

INTERVAL YEAR [(year_precision)] TO MONTH

em que year_precision é o número de dígitos no campo de data/horário YEAR. O valor default de year_precision é 2.

Restrição: O primeiro campo deve conter um valor maior que o segundo. Por exemplo, INTERVAL '0-1' MONTH TO YEAR não é válido.

O banco de dados Oracle suporta dois tipos de intervalos de dados: Intervalo de Ano para Mês e Intervalo de Dia para Segundo; o tipo de coluna, o argumento PL/SQL, a variável e o tipo de retorno devem ser um dos dois intervalos. No entanto, para os literais de intervalo, o sistema reconhece outros tipos de intervalo ANSI, como INTERVAL '2' YEAR ou INTERVAL '10' HOUR. Nesses casos, cada intervalo é convertido em um dos dois tipos suportados.

No exemplo acima, é criada uma tabela WARRANTY, contendo uma coluna warranty_time que adota o tipo de dados INTERVAL YEAR(3) TO MONTH. Nela são inseridos valores diferentes para indicar anos e meses para vários produtos. Quando essas linhas são recuperadas da tabela, é possível ver um valor de ano exibido ao lado de um valor de mês, separados por um (-).

Tipo de Dados INTERVAL DAY TO SECOND

INTERVAL DAY TO SECOND

(fractional_seconds_precision) armazena um período em dias, horas, minutos e segundos.

```
INTERVAL DAY[(day_precision)] TO Second
```

- **Por exemplo:**

```
INTERVAL '6 03:30:16' DAY TO SECOND
```

Indica um intervalo de 6 dias, 3 horas, 30 minutos e 16 segundos

```
INTERVAL '6 00:00:00' DAY TO SECOND
```

Indica um intervalo de 6 dias, 0 horas, 0 minutos e 0 segundos

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Tipo de Dados INTERVAL DAY TO SECOND

INTERVAL DAY (day_precision) TO SECOND

(fractional_seconds_precision) armazena um período em dias, horas, minutos e segundos, em que day_precision é o número máximo de dígitos no campo de data/horário DAY (os valores aceitos vão de 0 a 9; o default é 2), e fractional_seconds_precision é o número de dígitos na parte fracionária do campo SECOND. Os valores aceitos vão de 0 a 9. O default é 6.

No exemplo acima, 6 representa o número de dias, e 03:30:15 indica os valores para horas, minutos e segundos.

Tipo de Dados INTERVAL DAY TO SECOND: Exemplo

```
CREATE TABLE lab
( exp_id number, test_time INTERVAL DAY(2) TO
SECOND);

INSERT INTO lab VALUES (100012, '90 00:00:00');
INSERT INTO lab VALUES (56098,
INTERVAL '6 03:30:16' DAY TO SECOND);
```

```
SELECT * FROM lab;
```

| EXP_ID | TEST_TIME |
|--------|---------------------|
| 100012 | +90 00:00:00.000000 |
| 56098 | +06 03:30:16.000000 |

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Tipo de Dados INTERVAL DAY TO SECOND: Exemplo

No exemplo acima, você está criando uma tabela de laboratório com uma coluna `test_time` e tipo de dados `INTERVAL DAY TO SECOND`. Em seguida, insira nela o valor `"90 00:00:00"` para indicar 90 dias e 0 horas/minutos/segundos e `INTERVAL '6 03:30:16' DAY TO SECOND`. A instrução `SELECT` mostra como esses dados são exibidos no banco de dados.

EXTRACT

- Exiba o componente YEAR de SYSDATE.

| | | |
|--------------------------|-----------------------------|------------|
| SELECT | EXTRACT (YEAR FROM SYSDATE) | FROM DUAL; |
| EXTRACT(YEARFROMSYSDATE) | | |
| 2001 | | |

- Exiba o componente MONTH de HIRE_DATE para os funcionários cujo MANAGER_ID é igual a 100.

| | |
|------------------------------|--------------------------------|
| SELECT last name, hire date, | EXTRACT (MONTH FROM HIRE_DATE) |
| FROM employees | |
| WHERE manager_id = 100; | |

| LAST_NAME | HIRE_DATE | EXTRACT(MONTHFROMHIRE_DATE) |
|-----------|-----------|-----------------------------|
| Kochhar | 21-SEP-89 | 9 |
| De Haan | 13-JAN-93 | 1 |
| Mourgos | 16-NOV-99 | 11 |
| Zlotkey | 29-JAN-00 | 1 |
| Hartstein | 17-FEB-96 | 2 |

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

EXTRACT

A expressão EXTRACT extrai e retorna o valor de um campo de data/horário especificado de uma expressão de valor de data/horário ou intervalo. É possível extrair qualquer um dos componentes mencionados na sintaxe a seguir usando a function EXTRACT. A sintaxe da function EXTRACT é:

```
SELECT  EXTRACT ( [YEAR] [MONTH] [DAY] [HOUR] [MINUTE] [SECOND]
                [TIMEZONE_HOUR] [TIMEZONE_MINUTE]
                [TIMEZONE_REGION] [TIMEZONE_ABBR]
FROM [datetime_value_expression] [interval_value_expression] );
```

Quando você extrai TIMEZONE_REGION ou TIMEZONE_ABBR (abreviatura), o valor retornado é uma string com a abreviatura ou o nome do fuso horário apropriado. Depois de extraído, um dos outros valores é retornado como uma data no formato do calendário gregoriano. Quando é executada a extração de uma data/horário, com um valor de fuso horário, o valor retornado está em UTC.

No primeiro exemplo do slide, a function EXTRACT é usada para extrair o ano de SYSDATE. No segundo exemplo do slide, a function EXTRACT é usada para extrair o mês da coluna HIRE_DATE da tabela EMPLOYEES, para aqueles funcionários que estão subordinados ao gerente cujo EMPLOYEE_ID é igual a 100.

TZ_OFFSET

- **Exiba o deslocamento para o fuso horário**

'US/Eastern'.

```
SELECT TZ_OFFSET('US/Eastern') FROM DUAL;
```

| TZ_OFFSET |
|-----------|
| -04:00 |

- **Exiba o deslocamento para o fuso horário**

'Canada/Yukon'.

```
SELECT TZ_OFFSET('Canada/Yukon') FROM DUAL;
```

| TZ_OFFSET |
|-----------|
| -07:00 |

- **Exiba o deslocamento para o fuso horário**

'Europe/London'.

```
SELECT TZ_OFFSET('Europe/London') FROM DUAL;
```

| TZ_OFFSET |
|-----------|
| +01:00 |

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

TZ_OFFSET

A function TZ_OFFSET retorna o deslocamento de fuso horário correspondente ao valor informado. O valor informado depende da data em que a instrução é executada. Por exemplo, se a function TZ_OFFSET retorna o valor -08:00, o fuso horário do local onde esse comando foi executado tem oito horas a menos que o UTC. É possível informar um nome de fuso horário válido, um deslocamento de fuso horário em relação ao UTC (o próprio é retornado) ou a palavra-chave SESSIONTIMEZONE ou DBTIMEZONE. A sintaxe da function TZ_OFFSET é:

```
TZ_OFFSET ( ['time_zone_name'] '[+ | -] hh:mm' )  
          [ SESSIONTIMEZONE ] [ DBTIMEZONE ]
```

A matriz da Fold Motor Company fica em Michigan, E.U.A., que está no fuso horário US/Eastern. O presidente da empresa, Sr. Fold, quer fazer uma teleconferência com o vice-presidente de operações do Canadá e o vice-presidente de operações da Europa, que estão nos fusos horários de Canada/Yukon e Europe/London, respectivamente. O Sr. Fold quer saber o melhor horário em cada um desses locais para ter certeza de que esses executivos estarão disponíveis para participar da teleconferência. Seu secretário, Sr. Scott, está ajudando a executar as consultas mostradas no exemplo e obtém os seguintes resultados:

- O fuso horário 'US/Eastern' tem quatro horas a menos que o UTC.
- O fuso horário 'Canada/Yukon' tem sete horas a menos que o UTC.
- O fuso horário 'Europe/London' tem uma hora a mais que o UTC.

TZ_OFFSET (continuação)

Para obter uma lista com valores de nomes de fusos horários válidos, você pode consultar a view dinâmica de desempenho V\$TIMEZONE_NAMES.

```
SELECT * FROM V$TIMEZONE_NAMES;
```

| TZNAME | TZABBREV |
|-------------------|----------|
| Africa/Algiers | LMT |
| Africa/Algiers | PMT |
| Africa/Algiers | WET |
| Africa/Algiers | WEST |
| Africa/Algiers | CET |
| Africa/Algiers | CEST |
| Africa/Cairo | LMT |
| Africa/Cairo | EET |
| Africa/Cairo | EEST |
| Africa/Casablanca | LMT |
| Africa/Casablanca | WET |
| Africa/Casablanca | WEST |
| Africa/Casablanca | CET |
| Africa/Ceuta | LMT |
| Africa/Ceuta | WET |
| Africa/Ceuta | WEST |

■ ■ ■

Conversão de TIMESTAMP Usando FROM_TZ

- Exiba o valor TIMESTAMP '2000-03-28 08:00:00' como um valor TIMESTAMP WITH TIME ZONE.

```
SELECT FROM_TZ(TIMESTAMP  
                '2000-03-28 08:00:00', '3:00')  
FROM DUAL;
```

```
FROM_TZ(TIMESTAMP'2000-03-2808:00:00','3:00')  
28-MAR-00 08.00.00.000000000 AM +03:00
```

- Exiba o valor TIMESTAMP '2000-03-28 08:00:00' como um valor TIMESTAMP WITH TIME ZONE para a região de fuso horário 'Australia/North'.

```
SELECT FROM_TZ(TIMESTAMP  
                '2000-03-28 08:00:00', 'Australia/North')  
FROM DUAL;
```

```
FROM_TZ(TIMESTAMP'2000-03-2808:00:00','AUSTRALIA/NORTH')  
28-MAR-00 08.00.00.000000000 AM AUSTRALIA/NORTH
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Conversão de TIMESTAMP Usando FROM_TZ

A function FROM_TZ converte um valor TIMESTAMP em um valor TIMESTAMP WITH TIME ZONE.

A sintaxe da function FROM_TZ é:

```
FROM_TZ(TIMESTAMP timestamp_value, time_zone_value)
```

em que time_zone_value é uma string de caracteres no formato 'TZH:TZM' ou uma expressão de caracteres que retorna uma string em TZR (Time Zone Region) com formato TZD opcional. TZD é uma string abreviada de fuso horário com informações do horário de verão. TZR representa a região do fuso horário em strings de entrada de data/horário.

Alguns exemplos são 'Australia/North', 'PST' para US/Pacific Standard Time e 'PDT' para US/Pacific Daylight Time etc. Para obter uma lista com os valores válidos para os elementos de formato TZR e TZD, consulte a view de dinâmica de desempenho V\$TIMEZONE_NAMES.

O exemplo do slide converte um valor TIMESTAMP em TIMESTAMP WITH TIME ZONE.

Convertendo em TIMESTAMP Usando TO_TIMESTAMP e TO_TIMESTAMP_TZ

- Exiba a string de caracteres '2000-12-01 11:00:00' como um valor **TIMESTAMP**.

```
SELECT TO_TIMESTAMP ('2000-12-01 11:00:00',  
                     'YYYY-MM-DD HH:MI:SS')  
FROM DUAL;
```

```
TO_TIMESTAMP('2000-12-01 11:00:00','YYYY-MM-DDHH:MI:SS')  
01-DEC-00 11.00.00.000000000 AM
```

- Exiba a string de caracteres '1999-12-01 11:00:00 -8:00' como um valor **TIMESTAMP WITH TIME ZONE**.

```
SELECT  
  TO_TIMESTAMP_TZ('1999-12-01 11:00:00 -8:00',  
                  'YYYY-MM-DD HH:MI:SS TZH:TZM')  
FROM DUAL;
```

```
TO_TIMESTAMP_TZ('1999-12-01 11:00:00 -8:00','YYYY-MM-DDHH:MI:SSTZH:TZM')  
01-DEC-99 11.00.00.000000000 AM -08:00
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Convertendo em TIMESTAMP Usando TO_TIMESTAMP e TO_TIMESTAMP_TZ

A function `TO_TIMESTAMP` converte uma string de tipo de dados `CHAR`, `VARCHAR2`, `NCHAR` ou `NVARCHAR2` em um valor de tipo de dados `TIMESTAMP`. A sintaxe da function `TO_TIMESTAMP` é:

```
TO_TIMESTAMP (char, [fmt], ['nlsparam'])
```

O parâmetro `fmt` opcional especifica o formato de `char`. Caso seja omitido, a string deve estar no formato default do tipo de dados `TIMESTAMP`. O parâmetro opcional `nlsparam` especifica o idioma em que são retornados os nomes de meses e dias e as abreviações. Esse argumento pode ter a seguinte forma:

```
'NLS_DATE_LANGUAGE = language'
```

Caso você omita o parâmetro `nlsparams`, essa function usará o idioma default para datas da sua sessão. O exemplo do slide converte uma string de caracteres em um valor de `TIMESTAMP`.

A function `TO_TIMESTAMP_TZ` converte uma string de tipo de dados `CHAR`, `VARCHAR2`, `NCHAR` ou `NVARCHAR2` em um valor de tipo de dados `TIMESTAMP WITH TIME ZONE`. A sintaxe da function `TO_TIMESTAMP_TZ` é:

```
TO_TIMESTAMP_TZ (char, [fmt], ['nlsparam'])
```

O parâmetro opcional `fmt` especifica o formato de `char`. Caso seja omitido, deverá haver uma string no formato default do tipo de dados `TIMESTAMP WITH TIME ZONE`. O exemplo do slide converte uma string de caracteres em um valor de `TIMESTAMP WITH TIME ZONE`.

Conversão de Intervalo de Tempo com TO_YMINTERVAL

Exiba uma data que seja um ano e dois meses posterior à data de admissão para funcionários que trabalham em um departamento com DEPARTMENT_ID igual a 20.

```
SELECT hire_date,  
       hire_date + TO_YMINTERVAL('01-02') AS  
       HIRE_DATE_YMININTERVAL  
FROM   employees  
WHERE  department_id = 20;
```

| HIRE_DATE | HIRE_DATE_YMININTERV |
|----------------------|----------------------|
| 17-FEB-1996 00:00:00 | 17-APR-1997 00:00:00 |
| 17-AUG-1997 00:00:00 | 17-OCT-1998 00:00:00 |

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Conversão de Intervalo de Tempo com TO_YMINTERVAL

A function TO_YMINTERVAL converte uma string de caracteres de tipo de dados CHAR, VARCHAR2, NCHAR ou NVARCHAR2 em um tipo de dados INTERVAL YEAR TO MONTH. O tipo de dados INTERVAL YEAR TO MONTH armazena um período usando os campos de data/horário YEAR e MONTH. O formato de INTERVAL YEAR TO MONTH é:

INTERVAL YEAR [(year_precision)] TO MONTH

em que year_precision é o número de dígitos no campo de data/horário YEAR. O valor default de year_precision é 2.

A sintaxe da function TO_YMINTERVAL é:

TO_YMINTERVAL (char)

em que char é a string de caracteres a ser convertida.

O exemplo do slide calcula uma data que é um ano e dois meses posterior à data de admissão dos funcionários que trabalham no departamento 20 da tabela EMPLOYEES.

Também é possível fazer um cálculo inverso usando a function TO_YMINTERVAL. Por exemplo:

```
SELECT hire_date, hire_date + TO_YMINTERVAL('-02-04') AS  
       HIRE_DATE_YMININTERVAL  
FROM   EMPLOYEES WHERE department_id = 20;
```

Observe que a string de caracteres passada para a function TO_YMINTERVAL tem valor negativo. O exemplo retorna uma data que é dois anos e quatro meses anterior à data de admissão para os funcionários que trabalham no departamento 20 da tabela EMPLOYEES.

Usando TO_DSINTERVAL: Exemplo

TO_DSINTERVAL: Converte uma string de caracteres em um tipo de dados INTERVAL DAY TO SECOND

```
SELECT last_name,  
       TO_CHAR(hire_date, 'mm-dd-yy:hh:mi:ss') hire_date,  
       TO_CHAR(hire_date +  
               TO_DSINTERVAL('100 10:00:00'),  
               'mm-dd-yy:hh:mi:ss') hiredate2  
FROM employees;
```

| LAST_NAME | HIRE_DATE | HIREDATE2 |
|-----------|-------------------|-------------------|
| King | 06-17-87:12:00:00 | 09-25-87:10:00:00 |
| Kochhar | 09-21-89:12:00:00 | 12-30-89:10:00:00 |
| De Haan | 01-13-93:12:00:00 | 04-23-93:10:00:00 |
| Hunold | 01-03-90:12:00:00 | 04-13-90:10:00:00 |
| Ernst | 05-21-91:12:00:00 | 08-29-91:10:00:00 |
| Austin | 06-25-97:12:00:00 | 10-03-97:10:00:00 |
| Pataballa | 02-05-98:12:00:00 | 05-16-98:10:00:00 |
| Lorentz | 02-07-99:12:00:00 | 05-18-99:10:00:00 |
| Greenberg | 08-17-94:12:00:00 | 11-25-94:10:00:00 |
| Faviet | 08-16-94:12:00:00 | 11-24-94:10:00:00 |

...

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

TO_DSINTERVAL

A function TO_DSINTERVAL converte uma string de caracteres de tipo de dados CHAR, VARCHAR2, NCHAR ou NVARCHAR2 em um tipo de dados INTERVAL DAY TO SECOND.

No exemplo acima, a data obtida é 100 dias e 10 horas posterior à data de admissão.

TO_YMINTERVAL

A function TO_YMINTERVAL converte uma string de caracteres de tipo de dados CHAR, VARCHAR2, NCHAR ou NVARCHAR2 em um tipo de dados INTERVAL YEAR TO MONTH.

No exemplo a seguir, a data obtida é um ano e dois meses posterior à data de admissão.

```
SELECT hire_date, hire_date + TO_YMINTERVAL('01-02') ytm  
FROM employees;
```

```
HIRE_DATE  YTM  
-----  
17-JUN-87  17-AUG-88  
21-SEP-89  21-NOV-90  
13-JAN-93  13-MAR-94  
03-JAN-90  03-MAR-91  
21-MAY-91  21-JUL-92
```

...

Horário de Verão

- **Primeiro Domingo de Abril**
 - A hora passa de 01:59:59 para 03:00:00.
 - Os valores de 02:00:00 até 02:59:59 não são válidos.
- **Último Domingo de Outubro**
 - A hora passa de 02:00:00 para 01:00:01.
 - Os valores de 01:00:01 a 02:00:00 são ambíguos porque são utilizados duas vezes.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

Horário de Verão (DST)

A maioria dos países ocidentais adianta o relógio em uma hora durante os meses de verão. Esse período é chamado de horário de verão. O horário de verão vai desde o primeiro domingo de abril até o último domingo de outubro em grande parte dos Estados Unidos, México e Canadá. Os países da União Européia também seguem o horário de verão. Na Europa, o horário de verão começa uma semana mais cedo do que na América do Norte, mas termina na mesma época.

O banco de dados Oracle determina, automaticamente, para determinada região de fuso horário, se o horário de verão está em vigor e retorna os valores de hora local da forma adequada. O valor de data/horário sempre é suficiente para que o banco de dados Oracle determine se o horário de verão está em vigor em determinada região, exceto nos casos limítrofes. Um caso limítrofe ocorre por ocasião do início ou do fim do horário de verão. Por exemplo, na região US-Eastern, quando o horário de verão entra em vigor, a hora passa de 01:59:59 para 3:00:00. Esse intervalo de uma hora entre 02:00:00 e 02:59:59 não existe. Quando o horário de verão termina, a hora passa de 02:00:00 para 01:00:01, e o intervalo de hora entre 01:00:01 e 02:00:00 se repete.

Horário de Verão (DST) (continuação)

ERROR_ON_OVERLAP_TIME

ERROR_ON_OVERLAP_TIME é um parâmetro de sessão que serve para notificar ao sistema que ele deve emitir uma mensagem de erro quando encontrar uma data/horário que ocorra no período de sobreposição e não for especificada nenhuma abreviatura de fuso horário para distinguir o período.

Por exemplo, se o horário de verão terminar em 31 de outubro, às 02:00:01, os períodos de sobreposição serão:

- de 31/10/2004, à 01:00:01 até 31/10/2004, às 02:00:00 (EDT)
- de 31/10/2004, à 01:00:01 até 31/10/2004, às 02:00:00 (EST)

Caso informe uma string de data/horário que ocorre em um desses dois períodos, você precisará especificar a abreviatura de fuso horário (por exemplo, EDT ou EST) na string de entrada para que o sistema determine o período. Sem essa abreviatura de fuso horário, o sistema fará o seguinte:

Se o valor do parâmetro ERROR_ON_OVERLAP_TIME for FALSE, o sistema assumirá que o horário de entrada é o horário padrão (por exemplo, EST). Caso contrário, ocorrerá um erro.

Sumário

Nesta lição, você aprendeu a usar as seguintes functions:

- TZ_OFFSET
- FROM_TZ
- TO_TIMESTAMP
- TO_TIMESTAMP_TZ
- TO_YMINTERVAL
- CURRENT_DATE
- CURRENT_TIMESTAMP
- LOCALTIMESTAMP
- DBTIMEZONE
- SESSIONTIMEZONE
- EXTRACT

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Sumário

Esta lição abordou algumas das functions de data/horário disponíveis no banco de dados Oracle.

Exercício 5: Visão Geral

Este exercício aborda o uso das functions de data/horário.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Exercício 5: Visão Geral

Neste exercício, você exibirá os deslocamentos de fuso horário, `CURRENT_DATE`, `CURRENT_TIMESTAMP` e `LOCALTIMESTAMP`. Você também definirá fusos horários e usará a function `EXTRACT`.

Exercício 5

1. Altere a sessão para definir NLS_DATE_FORMAT como DD-MON-YYYY HH24:MI:SS.

2. a. Crie consultas para exibir os deslocamentos (TZ_OFFSET) para os seguintes fusos horários:

– *US/Pacific-New*

| TZ_OFFSET('US/PACIFIC') |
|-------------------------|
| -08:00 |

– *Singapore*

| TZ_OFFSET('SINGAPORE') |
|------------------------|
| +08:00 |

– *Egypt*

| TZ_OFFSET('EGYPT') |
|--------------------|
| +02:00 |

- b. Altere a sessão para definir o valor do parâmetro TIME_ZONE com o deslocamento de fuso horário de US/Pacific-New.
- c. Exiba CURRENT_DATE, CURRENT_TIMESTAMP e LOCALTIMESTAMP para esta sessão.

| CURRENT_DATE | CURRENT_TIMESTAMP | LOCALTIMESTAMP |
|--------------|-------------------------------------|------------------------------|
| 16-FEB-04 | 16-FEB-04 05.12.22.557032 PM -07:00 | 16-FEB-04 05.12.22.557032 PM |

- d. Altere a sessão para definir o valor do parâmetro TIME_ZONE como o deslocamento do fuso horário de Singapore.
- e. Exiba CURRENT_DATE, CURRENT_TIMESTAMP e LOCALTIMESTAMP para esta sessão.

Observação: A saída poderá ser diferente, dependendo da data de execução do comando.

| CURRENT_DATE | CURRENT_TIMESTAMP | LOCALTIMESTAMP |
|--------------|-------------------------------------|------------------------------|
| 17-FEB-04 | 17-FEB-04 08.06.18.057870 AM +08:00 | 17-FEB-04 08.06.18.057870 AM |

Observação: Observe que, no exercício anterior, CURRENT_DATE, CURRENT_TIMESTAMP e LOCALTIMESTAMP são sensíveis ao fuso horário da sessão.

3. Crie uma consulta para exibir DBTIMEZONE e SESSIONTIMEZONE.

| DBTIMEZONE | SESSIONTIMEZONE |
|------------|-----------------|
| +00:00 | -07:00 |

Exercício 5 (continuação)

4. Crie uma consulta para extrair o ano da coluna HIRE_DATE da tabela EMPLOYEES em relação aos funcionários que trabalham no departamento 80.

| LAST_NAME | EXTRACT(YEARFROMHIRE_DATE) |
|-----------|----------------------------|
| Russell | 1996 |
| Partners | 1997 |
| Errazuriz | 1997 |
| Cambrault | 1999 |
| Zlotkey | 2000 |

...

| LAST_NAME | EXTRACT(YEARFROMHIRE_DATE) |
|------------|----------------------------|
| Bloom | 1998 |
| Fox | 1998 |
| Smith | 1999 |
| Bates | 1999 |
| Kumar | 2000 |
| Abel | 1996 |
| Hutton | 1997 |
| Taylor | 1998 |
| Livingston | 1998 |
| Johnson | 2000 |

34 rows selected.

5. Altere a sessão para definir NLS_DATE_FORMAT como DD-MON-YYYY.

Exercício 5 (continuação)

6. Examine e execute o script `lab05_06.sql` para criar a tabela `SAMPLE_DATES` e preenchê-la.

a. Selecione na tabela e exiba os dados.

| DATE_COL |
|-----------|
| 16-FEB-04 |

- b. Modifique o tipo de dados da coluna `DATE_COL` e altere-o para `TIMESTAMP`. Selecione na tabela para exibir os dados.

| DATE_COL |
|------------------------------|
| 16-FEB-04 05.38.50.000000 PM |

- c. Tente modificar o tipo de dados da coluna `DATE_COL` e altere-o para `TIMESTAMP WITH TIME ZONE`. O que acontece?

7. Crie uma consulta para recuperar os sobrenomes da tabela `EMPLOYEES` e calcular o status da avaliação. Se o ano de admissão foi 2000, exiba `Needs Review` para o status da avaliação. Caso contrário, exiba `not this year!`. Nomeie a coluna de status da avaliação como `Review`. Classifique os resultados pela coluna `HIRE_DATE`.

Dica: Use uma expressão `CASE` com function `EXTRACT` para calcular o status da avaliação.

| LAST_NAME | Review |
|-----------|----------------|
| King | not this year! |
| Kochhar | not this year! |
| De Haan | not this year! |
| Hunold | not this year! |
| Ernst | not this year! |
| Austin | not this year! |
| Pataballa | Needs Review |
| Lorentz | not this year! |

...

| LAST_NAME | Review |
|-----------|----------------|
| Walsh | Needs Review |
| Feeney | Needs Review |
| OConnell | not this year! |
| Grant | not this year! |
| Whalen | not this year! |
| Hartstein | not this year! |
| Fay | not this year! |
| Mavris | not this year! |
| Baer | not this year! |
| Higgins | not this year! |
| Gietz | not this year! |

107 rows selected.

Exercício 5 (continuação)

8. Crie uma consulta para imprimir os sobrenomes e os anos de serviço de cada funcionário. Se o funcionário foi contratado há cinco anos ou mais, imprima 5 years of service. Se o funcionário foi contratado há 10 anos ou mais, imprima 10 years of service. Se o funcionário foi contratado há 15 anos ou mais, imprima 15 years of service. Se não houver correspondência com nenhuma dessas condições, imprima maybe next year! Classifique os resultados pela coluna HIRE_DATE. Use a tabela EMPLOYEES.

Dica: Use expressões CASE e TO_YMINTERVAL.

| LAST_NAME | HIRE_DATE | SYSDATE | Awards |
|-----------|-----------|-----------|---------------------|
| King | 17-JUN-87 | 16-FEB-04 | 15 years of service |
| Kochhar | 21-SEP-89 | 16-FEB-04 | 10 years of service |
| De Haan | 13-JAN-93 | 16-FEB-04 | 10 years of service |
| Hunold | 03-JAN-90 | 16-FEB-04 | 10 years of service |
| Ernst | 21-MAY-91 | 16-FEB-04 | 10 years of service |
| Austin | 25-JUN-97 | 16-FEB-04 | 5 years of service |
| Pataballa | 05-FEB-98 | 16-FEB-04 | 5 years of service |
| Lorentz | 07-FEB-99 | 16-FEB-04 | 5 years of service |

...

| LAST_NAME | HIRE_DATE | SYSDATE | Awards |
|-----------|-----------|-----------|---------------------|
| Walsh | 24-APR-98 | 16-FEB-04 | 5 years of service |
| Feeney | 23-MAY-98 | 16-FEB-04 | 5 years of service |
| OConnell | 21-JUN-99 | 16-FEB-04 | maybe next year! |
| Grant | 13-JAN-00 | 16-FEB-04 | maybe next year! |
| Whalen | 17-SEP-87 | 16-FEB-04 | 15 years of service |
| Hartstein | 17-FEB-96 | 16-FEB-04 | 5 years of service |
| Fay | 17-AUG-97 | 16-FEB-04 | 5 years of service |
| Mavris | 07-JUN-94 | 16-FEB-04 | 5 years of service |
| Baer | 07-JUN-94 | 16-FEB-04 | 5 years of service |
| Higgins | 07-JUN-94 | 16-FEB-04 | 5 years of service |
| Gietz | 07-JUN-94 | 16-FEB-04 | 5 years of service |

107 rows selected.

6

Recuperando Datos Usando Subconsultas

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Objetivos

Ao concluir esta lição, você será capaz de:

- **Criar uma subconsulta de várias colunas**
- **Usar subconsultas escalares em SQL**
- **Solucionar problemas com subconsultas correlacionadas**
- **Atualizar e deletar linhas usando subconsultas correlacionadas**
- **Usar os operadores EXISTS e NOT EXISTS**
- **Usar a cláusula WITH**

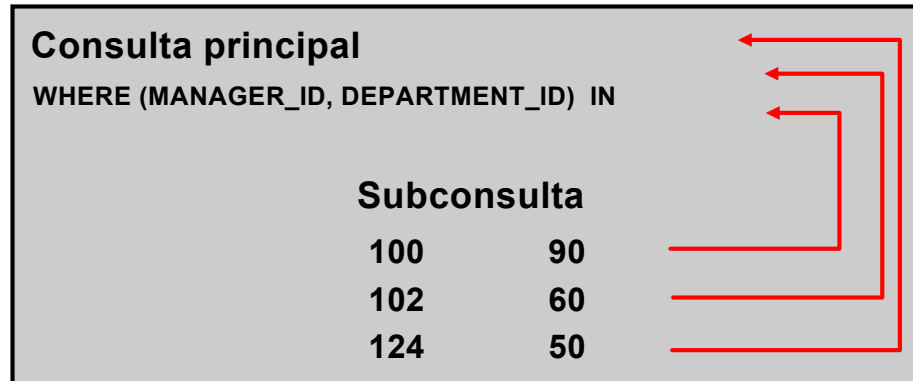
ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Objetivos

Nesta lição, você aprenderá a criar subconsultas de várias colunas e subconsultas na cláusula FROM de uma instrução SELECT. Você também aprenderá a solucionar problemas usando as subconsultas escalares, as subconsultas correlacionadas e a cláusula WITH.

Subconsultas de Várias Colunas



Cada linha da consulta principal é comparada a valores de uma subconsulta de várias linhas e várias colunas.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Subconsultas de Várias Colunas

Até então, você criou subconsultas de uma única linha e subconsultas de várias linhas em que apenas uma coluna é retornada pela instrução `SELECT` interna; essa coluna é usada para avaliar a expressão na instrução `SELECT` mãe. Para comparar duas ou mais colunas, crie uma cláusula `WHERE` composta usando operadores lógicos. Usando subconsultas de várias colunas, é possível combinar condições `WHERE` duplicadas em uma única cláusula `WHERE`.

Sintaxe

```
SELECT column, column, ...
FROM table
WHERE (column, column, ...) IN
      (SELECT column, column, ...
       FROM table
       WHERE condition);
```

O gráfico no slide mostra que os valores das colunas `MANAGER_ID` e `DEPARTMENT_ID` da consulta principal estão sendo comparados aos valores das mesmas colunas recuperados pela subconsulta. Como estão sendo comparadas mais de uma coluna, o exemplo qualifica-se como uma subconsulta de várias colunas.

Comparações de Colunas

As comparações de colunas em uma subconsulta de várias colunas podem ser:

- **Comparações emparelhadas**
- **Comparações não emparelhadas**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

Comparações Emparelhadas e Não Emparelhadas

As comparações de colunas em uma subconsulta de várias colunas podem ser comparações emparelhadas ou não emparelhadas.

No exemplo do próximo slide, uma comparação emparelhada é executada na cláusula WHERE. Cada linha candidata na instrução SELECT deve ter os mesmos valores nas colunas MANAGER_ID e DEPARTMENT_ID que os funcionários com EMPLOYEE_ID 199 ou 174.

Uma subconsulta de várias colunas também pode ser uma comparação não emparelhada. Em uma comparação não emparelhada, cada uma das colunas da cláusula WHERE da instrução SELECT mãe é comparada individualmente a diversos valores recuperados pela instrução SELECT interna. As colunas individuais podem corresponder a qualquer um dos valores recuperados pela instrução SELECT interna. Porém, coletivamente, todas as condições da instrução SELECT principal devem ser atendidas para que a linha seja exibida. O exemplo da próxima página ilustra uma comparação emparelhada.

Subconsulta de Comparação Emparelhada

Exiba os detalhes dos funcionários que estão subordinados ao mesmo gerente e que trabalham no mesmo departamento que os funcionários com EMPLOYEE_ID 199 ou 174

```
SELECT employee_id, manager_id, department_id
FROM employees
WHERE (manager_id, department_id) IN
      (SELECT manager_id, department_id
       FROM employees
       WHERE employee_id IN (199,174))
AND employee_id NOT IN (199,174);
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Subconsulta de Comparação Emparelhada

O exemplo do slide exibe uma subconsulta de várias colunas, isto é, uma subconsulta que retorna mais de uma coluna. Ela compara os valores das colunas MANAGER_ID e DEPARTMENT_ID de cada linha da tabela EMPLOYEES com os valores das mesmas colunas para os funcionários com EMPLOYEE_ID 199 ou 174.

Primeiro, a subconsulta para recuperar os valores de MANAGER_ID e DEPARTMENT_ID para os funcionários com EMPLOYEE_ID 199 ou 174 é executada. Esses valores são comparados com as colunas MANAGER_ID e DEPARTMENT_ID de cada linha da tabela EMPLOYEES. Se houver correspondência entre os valores, a linha será exibida. Na saída, os registros dos funcionários com EMPLOYEE_ID 199 ou 174 não serão exibidos. A saída da consulta no slide é a seguinte:

| EMPLOYEE_ID | MANAGER_ID | DEPARTMENT_ID |
|-------------|------------|---------------|
| 176 | 149 | 80 |

Subconsulta de Comparação Não Emparelhada

Exiba os detalhes dos funcionários que estão subordinados ao mesmo gerente que os funcionários com EMPLOYEE_ID 174 ou 199 e que trabalham no mesmo departamento desses funcionários.

```
SELECT employee_id, manager_id, department_id
FROM employees
WHERE manager_id IN
      (SELECT manager_id
       FROM employees
       WHERE employee_id IN (174,199))
AND department_id IN
      (SELECT department_id
       FROM employees
       WHERE employee_id IN (174,199))
AND employee_id NOT IN(174,199);
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Subconsulta de Comparação Não Emparelhada

O exemplo mostra uma comparação não emparelhada de colunas. Ele exibe as colunas EMPLOYEE_ID, MANAGER_ID e DEPARTMENT_ID dos funcionários cujo ID do gerente corresponde a um dos IDs de gerente dos funcionários com os IDs 174 ou 199 e com um valor de DEPARTMENT_ID correspondente aos IDs de departamento desses mesmos funcionários.

Primeiro, a subconsulta para recuperar os valores de MANAGER_ID para os funcionários com EMPLOYEE_ID 174 ou 199 é executada. Depois, a segunda subconsulta para recuperar os valores de DEPARTMENT_ID para os funcionários com EMPLOYEE_ID 174 ou 199 é executada. Os valores recuperados das colunas MANAGER_ID e DEPARTMENT_ID são comparados com os valores das mesmas colunas para cada linha da tabela EMPLOYEES. Se a coluna MANAGER_ID da linha da tabela EMPLOYEES corresponder a qualquer um dos valores da coluna MANAGER_ID recuperados pela subconsulta interna e se a coluna DEPARTMENT_ID da linha da tabela EMPLOYEES corresponder a qualquer um dos valores da coluna DEPARTMENT_ID recuperados pela segunda subconsulta, o registro será exibido. A saída da consulta no slide é a seguinte:

| EMPLOYEE_ID | MANAGER_ID | DEPARTMENT_ID |
|-------------|------------|---------------|
| 142 | 124 | 50 |
| 143 | 124 | 50 |
| 144 | 124 | 50 |
| 176 | 149 | 80 |

Expressões de Subconsultas Escalares

- Uma expressão de subconsulta escalar é uma subconsulta que retorna exatamente um valor de coluna de uma linha.
- As subconsultas escalares podem ser usadas:
 - Na parte da expressão e da condição `DECODE` e `CASE`
 - Em todas as cláusulas de `SELECT`, com exceção de `GROUP BY`

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

Subconsultas Escalares em SQL

Uma subconsulta que retorna exatamente um valor de coluna de uma linha também é denominada subconsulta escalar. As subconsultas de várias colunas criadas para comparar duas ou mais colunas, usando uma cláusula `WHERE` composta e operadores lógicos, não se qualificam como subconsultas escalares.

O valor da expressão da subconsulta escalar corresponde ao valor do item da lista de seleção da subconsulta. Se a subconsulta retornar 0 linhas, o valor da expressão da subconsulta escalar será `NULL`. Se a subconsulta retornar mais de uma linha, o servidor Oracle retornará um erro. O servidor Oracle sempre suportou o uso de uma subconsulta escalar em uma instrução `SELECT`. É possível usar subconsultas escalares:

- Na parte da expressão e da condição `DECODE` e `CASE`
- Em todas as cláusulas de `SELECT`, com exceção de `GROUP BY`
- Na cláusula `SET` e na cláusula `WHERE` de uma instrução `UPDATE`

No entanto, as subconsultas escalares não são válidas em expressões nos seguintes locais:

- Como valores default para colunas e expressões hash para clusters
- Na cláusula `RETURNING` de instruções `DML`
- Como a base de um índice baseado em function
- Em cláusulas `GROUP BY`, constraints `CHECK` e condições `WHEN`
- Em cláusulas `CONNECT BY`
- Em instruções não relacionadas a consultas, como `CREATE PROFILE`

Subconsultas Escalares: Exemplos

- Subconsultas escalares em expressões CASE

```
SELECT employee_id, last_name,  
       (CASE  
         WHEN department_id = 20  
           (SELECT department_id  
            FROM departments  
            WHERE location_id = 1800)  
         THEN 'Canada' ELSE 'USA' END) location  
FROM   employees;
```

- Subconsultas escalares em cláusulas ORDER BY

```
SELECT  employee_id, last_name  
FROM    employees e  
ORDER BY (SELECT department_name  
          FROM departments d  
          WHERE e.department_id = d.department_id);
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

Subconsultas Escalares: Exemplos

O primeiro exemplo do slide demonstra que é possível usar subconsultas escalares em expressões CASE. A consulta interna retorna o valor 20, que é o ID do departamento cujo ID de local é 1800. A expressão CASE na consulta externa usa o resultado da consulta interna para exibir o ID dos funcionários, os sobrenomes e o valor Canada ou USA, dependendo de o ID do departamento do registro recuperado pela consulta externa ser ou não 20.

Este é o resultado do primeiro exemplo do slide:

...

| EMPLOYEE_ID | LAST_NAME | LOCATION |
|-------------|-----------|----------|
| 196 | Walsh | USA |
| 197 | Feeney | USA |
| 198 | OConnell | USA |
| 199 | Grant | USA |
| 200 | Whalen | USA |
| 201 | Hartstein | Canada |
| 202 | Fay | Canada |
| 203 | Mavris | USA |
| 204 | Baer | USA |
| 205 | Higgins | USA |
| 206 | Gietz | USA |

107 rows selected.

Subconsultas Escalares: Exemplos (continuação)

O segundo exemplo do slide demonstra que as subconsultas escalares podem ser usadas na cláusula ORDER BY. O exemplo ordena a saída com base no valor de DEPARTMENT_NAME, estabelecendo uma correspondência entre DEPARTMENT_ID da tabela EMPLOYEES e DEPARTMENT_ID da tabela DEPARTMENTS. Essa comparação é feita em uma subconsulta escalar na cláusula ORDER BY. Este é o resultado do segundo exemplo:

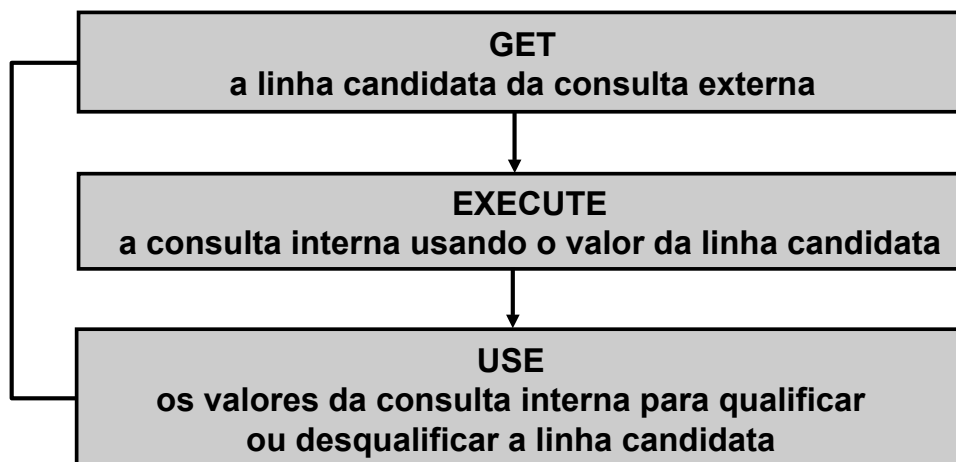
| EMPLOYEE_ID | LAST_NAME |
|-------------|------------|
| 205 | Higgins |
| 206 | Gietz |
| 200 | Whalen |
| 100 | King |
| 101 | Kochhar |
| 102 | De Haan |
| 108 | Greenberg |
| 109 | Faviet |
| ... | |
| EMPLOYEE_ID | LAST_NAME |
| 135 | Gee |
| 136 | Philtanker |
| 137 | Ladwig |
| 138 | Stiles |
| 139 | Seo |
| 140 | Patel |
| 141 | Rajs |
| 142 | Davies |
| 143 | Matos |
| 144 | Vargas |
| 178 | Grant |

107 rows selected.

O segundo exemplo usa uma subconsulta correlacionada. Em uma subconsulta correlacionada, a subconsulta faz referência a uma coluna de uma tabela referenciada na instrução mãe. As subconsultas correlacionadas serão explicadas posteriormente nesta lição.

Subconsultas Correlacionadas

As subconsultas correlacionadas são usadas para processamento por linha. Cada subconsulta é executada uma vez para cada linha da consulta externa.



ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Subconsultas Correlacionadas

O servidor Oracle executa uma subconsulta correlacionada quando essa subconsulta faz referência a uma coluna de uma tabela referenciada na instrução mãe. Uma subconsulta correlacionada é avaliada uma vez para cada linha processada pela instrução mãe. A instrução mãe pode ser uma instrução **SELECT**, **UPDATE** ou **DELETE**.

Subconsultas Aninhadas e Subconsultas Correlacionadas

Em uma subconsulta aninhada normal, a consulta **SELECT** interna é executada primeiro e apenas uma vez, retornando os valores a serem usados pela consulta principal. No entanto, uma subconsulta correlacionada é executada uma vez para cada linha candidata considerada pela consulta externa. Em outras palavras, a consulta interna é orientada pela consulta externa.

Execução de Subconsulta Aninhada

- A consulta interna é executada primeiro e localiza um valor.
- A consulta externa é executada uma vez, usando o valor da consulta interna.

Execução de Subconsulta Correlacionada

- Obtenha uma linha candidata (resultado do fetch da consulta externa).
- Execute a consulta interna usando o valor da linha candidata.
- Use os valores resultantes da consulta interna para qualificar ou desqualificar a linha candidata.
- Repita até que não existam mais linhas candidatas.

Subconsultas Correlacionadas

A subconsulta faz referência a uma coluna de uma tabela da consulta mãe.

```
SELECT column1, column2, ...
FROM   table1 outer
WHERE  column1 operator
      (SELECT column1, column2
       FROM   table2
       WHERE  expr1 =
            outer.expr2);
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Subconsultas Correlacionadas (continuação)

Uma subconsulta correlacionada é uma forma de ler todas as linhas de uma tabela e comparar os valores de cada linha com os dados relacionados. Ela é usada sempre que uma subconsulta precisa retornar um resultado ou um conjunto de resultados diferente para cada linha candidata considerada pela consulta principal. Em outras palavras, use uma subconsulta correlacionada para responder a uma pergunta com várias partes cuja resposta depende do valor de cada linha processada pela instrução mãe.

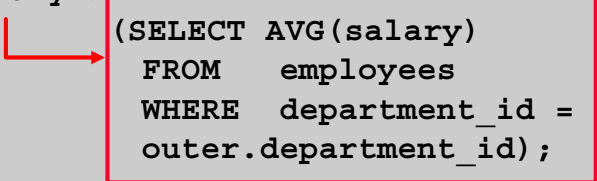
O servidor Oracle executa uma subconsulta correlacionada quando essa subconsulta faz referência a uma coluna de uma tabela na consulta mãe.

Observação: Você pode usar os operadores ANY e ALL em uma subconsulta correlacionada.

Usando Subconsultas Correlacionadas

Localize todos os funcionários que recebem mais que o salário médio nos respectivos departamentos.

```
SELECT last_name, salary, department_id
FROM   employees outer
WHERE  salary >
      (SELECT AVG(salary)
       FROM   employees
       WHERE  department_id =
             outer.department_id);
```



Sempre que uma linha da consulta externa for processada, a consulta interna será avaliada.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Usando Subconsultas Correlacionadas

O exemplo do slide determina quais funcionários recebem mais que o salário médio dos respectivos departamentos. Nesse caso, a subconsulta correlacionada calcula especificamente o salário médio de cada departamento.

Como as consultas externa e interna usam a tabela EMPLOYEES na cláusula FROM, é fornecido um apelido a essa tabela na instrução SELECT externa para maior clareza. Além de o apelido tornar a instrução SELECT inteira mais legível, sem ele a consulta não funcionará adequadamente, porque a instrução interna não conseguirá diferenciar a coluna da tabela interna da coluna da tabela externa.

Usando Subconsultas Correlacionadas

Exiba os detalhes dos funcionários que mudaram de cargo pelo menos duas vezes.

```
SELECT e.employee_id, last_name, e.job_id
FROM   employees e
WHERE  2 <= (SELECT COUNT(*)
             FROM   job_history
             WHERE  employee_id = e.employee_id);
```

| EMPLOYEE_ID | LAST_NAME | JOB_ID |
|-------------|-----------|---------|
| 101 | Kochhar | AD_VP |
| 176 | Taylor | SA_REP |
| 200 | Whalen | AD_ASST |

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Usando Subconsultas Correlacionadas (continuação)

O exemplo do slide exibe os detalhes dos funcionários que mudaram de cargo pelo menos duas vezes. O servidor Oracle avalia uma subconsulta correlacionada da seguinte maneira:

1. Seleciona uma linha da tabela especificada na consulta externa. Essa será a linha candidata atual.
2. Armazena o valor da coluna referenciada na subconsulta dessa linha candidata. (No exemplo do slide, a coluna à qual é feita referência na subconsulta é `E.EMPLOYEE_ID`.)
3. Executa a subconsulta cuja condição faz referência ao valor da linha candidata da consulta externa. (No exemplo do slide, a função de grupo `COUNT(*)` é avaliada com base no valor da coluna `E.EMPLOYEE_ID` obtido na etapa 2.)
4. Avalia a cláusula `WHERE` da consulta externa com base nos resultados da subconsulta executada na etapa 3. Essa ação determina se a linha candidata será selecionada para a saída. (No exemplo, o número de vezes que um funcionário mudou de cargo, avaliado pela subconsulta, é comparado com 2 na cláusula `WHERE` da consulta externa. Se a condição for atendida, o registro desse funcionário será exibido.)
5. Repita o procedure para a próxima linha candidata da tabela, e assim por diante, até que todas as linhas da tabela tenham sido processadas.

A correlação é estabelecida usando um elemento da consulta externa na subconsulta. Neste exemplo, você compara o valor de `EMPLOYEE_ID` da tabela na subconsulta com o valor de `EMPLOYEE_ID` da tabela na consulta externa.

Usando o Operador EXISTS

- O operador **EXISTS** verifica a existência de linhas no conjunto de resultados da subconsulta.
- Se for localizado um valor de linha na subconsulta:
 - A pesquisa na consulta interna não continuará
 - A condição será marcada como **TRUE**
- Se não for localizado um valor de linha na subconsulta:
 - A condição será marcada como **FALSE**
 - A pesquisa na consulta interna continuará

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

O Operador EXISTS

Em instruções **SELECT** aninhadas, todos os operadores lógicos são válidos. Além disso, você pode usar o operador **EXISTS**. Esse operador é frequentemente usado com subconsultas correlacionadas para verificar se um valor recuperado pela consulta externa existe no conjunto de resultados dos valores recuperados pela consulta interna. Se a subconsulta retornar, pelo menos, uma linha, o operador retornará **TRUE**. Se o valor não existir, o operador retornará **FALSE**. Da mesma forma, **NOT EXISTS** verifica se um valor recuperado pela consulta externa não faz parte do conjunto de resultados dos valores recuperados pela consulta interna.

Localizar Funcionários com Pelo Menos um Subordinado

```
SELECT employee_id, last_name, job_id, department_id
FROM   employees outer
WHERE  EXISTS ( SELECT 'X'
                FROM   employees
                WHERE  manager_id =
                      outer.employee_id);
```

| 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 |
| 108 | Greenberg | FL_MGR | 100 |
| 114 | Raphaely | PU_MAN | 30 |
| 120 | Weiss | ST_MAN | 50 |
| 121 | Fripp | ST_MAN | 50 |
| 122 | Kaufling | ST_MAN | 50 |
| 123 | Vollman | ST_MAN | 50 |
| 124 | Mourgos | ST_MAN | 50 |
| 145 | Russell | SA_MAN | 80 |
| 146 | Partners | SA_MAN | 80 |
| 147 | Errazuriz | SA_MAN | 80 |
| 148 | Cambrault | SA_MAN | 80 |
| 149 | Zlotkey | SA_MAN | 80 |
| 201 | Hartstein | MK_MAN | 20 |
| 205 | Higgins | AC_MGR | 110 |

18 rows selected.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Usando o Operador EXISTS

O operador EXISTS garantirá que a pesquisa na consulta interna não continuará quando pelo menos uma correspondência for encontrada para esse número de funcionário e de gerente com a condição:

```
WHERE manager_id = outer.employee_id.
```

Observe que a consulta SELECT interna não precisa retornar um valor específico. Portanto, será possível selecionar uma constante.

Localizar Todos os Departamentos sem Funcionários

```
SELECT department_id, department_name
FROM departments d
WHERE NOT EXISTS (SELECT 'X'
                   FROM employees
                   WHERE department_id
                     = d.department_id);
```

| DEPARTMENT_ID | DEPARTMENT_NAME |
|---------------|----------------------|
| 120 | Treasury |
| 130 | Corporate Tax |
| 140 | Control And Credit |
| 150 | Shareholder Services |
| 160 | Benefits |
| 170 | Manufacturing |
| 180 | Construction |
| 190 | Contracting |
| 200 | Operations |
| 210 | IT Support |
| 220 | NOC |
| 230 | IT Helpdesk |
| 240 | Government Sales |
| 250 | Retail Sales |
| 260 | Recruiting |
| 270 | Payroll |
| 510 | Human Resources |

17 rows selected.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Usando o Operador NOT EXISTS

Solução Alternativa

É possível usar um construct NOT IN como uma alternativa para um operador NOT EXISTS, como mostra o seguinte exemplo:

```
SELECT department_id, department_name
FROM departments
WHERE department_id NOT IN (SELECT department_id
                           FROM employees);
```

no rows selected

No entanto, NOT IN será avaliado como FALSE se um membro do conjunto for um valor NULL. Assim, a consulta não retornará linhas mesmo se houver linhas na tabela de departamentos que atendam à condição WHERE.

Instrução UPDATE Correlacionada

Use uma subconsulta correlacionada para atualizar as linhas de uma tabela com base nas linhas de outra tabela.

```
UPDATE table1 alias1
SET    column = (SELECT expression
                  FROM    table2 alias2
                  WHERE    alias1.column =
                          alias2.column);
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Instrução UPDATE Correlacionada

No caso da instrução UPDATE, você pode usar uma subconsulta correlacionada para atualizar linhas de uma tabela com base nas linhas de outra tabela.

Usando a Subconsulta UPDATE Correlacionada

- **Desnormalize a tabela EMP6 adicionando uma coluna para armazenar o nome do departamento.**
- **Preencha a tabela usando uma instrução UPDATE correlacionada.**

```
ALTER TABLE empl6  
ADD(department_name VARCHAR2(25));
```

```
UPDATE empl6 e  
SET    department_name =  
        (SELECT department_name  
         FROM   departments d  
         WHERE  e.department_id = d.department_id);
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Instrução UPDATE Correlacionada (continuação)

O exemplo do slide desnormaliza a tabela EMPL6 adicionando uma coluna para armazenar o nome do departamento e, depois, preenche a tabela usando uma instrução UPDATE correlacionada.

A seguir, outro exemplo de uma instrução UPDATE correlacionada.

Instrução em Questão

A tabela REWARDS tem uma lista de funcionários que superaram as expectativas de desempenho. Use uma subconsulta correlacionada para atualizar as linhas da tabela EMP6 com base nas linhas da tabela REWARDS:

```
UPDATE empl6  
SET    salary = (SELECT employees.salary + rewards.pay_raise  
                 FROM   rewards  
                 WHERE  employee_id =  
                        employees.employee_id  
                 AND    payraise_date =  
                        (SELECT MAX(payraise_date)  
                         FROM   rewards  
                         WHERE  employee_id = employees.employee_id))  
WHERE  employees.employee_id  
IN      (SELECT employee_id FROM rewards);
```

Instrução UPDATE Correlacionada (continuação)

Este exemplo usa a tabela REWARDS. A tabela REWARDS tem as colunas EMPLOYEE_ID, PAY_RAISE e PAYRAISE_DATE. Sempre que um funcionário recebe um aumento de salário, é inserido um registro com os detalhes de ID do funcionário, o valor do aumento de salário e a data de recebimento do aumento na tabela REWARDS. A tabela REWARDS pode conter mais de um registro para um funcionário. A coluna PAYRAISE_DATE é usada para identificar o aumento de salário mais recente de um funcionário.

No exemplo, a coluna SALARY da tabela EMPL6 é atualizada para refletir o aumento de salário mais recente recebido pelo funcionário. Para isso, é adicionado o salário atual do funcionário com o respectivo aumento de salário da tabela REWARDS.

Instrução DELETE Correlacionada

Use uma subconsulta correlacionada para deletar as linhas de uma tabela com base nas linhas de outra tabela.

```
DELETE FROM table1 alias1
WHERE column operator
      (SELECT expression
       FROM table2 alias2
       WHERE alias1.column = alias2.column);
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Instrução DELETE Correlacionada

No caso de uma instrução DELETE, você pode usar uma subconsulta correlacionada para deletar apenas as linhas que também existem em outra tabela. Se você decidir manter apenas os últimos quatro registros do histórico de cargos na tabela JOB_HISTORY, quando um funcionário for transferido para o quinto cargo, delete a linha mais antiga dessa tabela. Para isso, consulte o valor de MIN(START_DATE) relativo ao funcionário na tabela JOB_HISTORY. O código a seguir ilustra como é possível executar a operação precedente usando uma instrução DELETE correlacionada.

```
DELETE FROM emp_history JH
WHERE employee_id =
      (SELECT employee_id
       FROM employees E
       WHERE JH.employee_id = E.employee_id
       AND START_DATE =
            (SELECT MIN(start_date)
             FROM job_history JH
             WHERE JH.employee_id = E.employee_id)
       AND 5 > (SELECT COUNT(*)
                FROM job_history JH
                WHERE JH.employee_id = E.employee_id
                GROUP BY EMPLOYEE_ID
                HAVING COUNT(*) >= 4));
```

Usando a Subconsulta DELETE Correlacionada

Use uma subconsulta correlacionada para deletar apenas as linhas da tabela EMPL6 que também existem na tabela EMP_HISTORY:

```
DELETE FROM empl6 E
WHERE employee_id =
      (SELECT employee_id
       FROM   emp_history
       WHERE  employee_id = E.employee_id);
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Instrução DELETE Correlacionada (continuação)

Exemplo

Nesse exemplo, são usadas duas tabelas. São elas:

- A tabela EMPL6, que fornece os detalhes de todos os funcionários atuais
- A tabela EMP_HISTORY, que fornece os detalhes dos ex-funcionários

A tabela EMP_HISTORY contém dados relativos a ex-funcionários. Portanto, será um erro se o registro do mesmo funcionário existir nas tabelas EMPL6 e EMP_HISTORY. Você pode deletar esses registros errados usando a subconsulta correlacionada mostrada no slide.

A Cláusula WITH

- **Permite usar o mesmo bloco de consulta em uma instrução SELECT quando ele ocorre mais de uma vez em uma consulta complexa.**
- **A cláusula WITH recupera os resultados de um bloco de consulta e os armazena no tablespace temporário do usuário.**
- **A cláusula WITH melhora o desempenho.**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

A Cláusula WITH

Usando a cláusula WITH, é possível definir um bloco de consulta antes de usá-lo em uma consulta. Essa cláusula (anteriormente conhecida como `subquery_factoring_clause`) permite reutilizar o mesmo bloco de consulta em uma instrução SELECT quando ele ocorre mais de uma vez em uma consulta complexa. Ela é especialmente útil quando uma consulta faz diversas referências ao mesmo bloco de consulta e existem joins e agregações.

Com a cláusula WITH, você poderá reutilizar a mesma consulta quando for custoso avaliar o bloco de consulta e ele ocorrer mais de uma vez em uma consulta complexa. Usando a cláusula WITH, o servidor Oracle recupera os resultados de um bloco de consulta e os armazena no tablespace temporário do usuário. Essa característica pode melhorar o desempenho.

Vantagens da Cláusula WITH

- Facilita a leitura da consulta
- Avalia uma cláusula apenas uma vez, mesmo quando ela aparece diversas vezes na consulta
- Na maioria dos casos, pode melhorar o desempenho para consultas grandes

Cláusula WITH: Exemplo

Com a cláusula WITH, crie uma consulta para exibir o nome e o salário total dos departamentos cujo salário total é maior que o salário médio de todos os departamentos.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

Cláusula WITH: Exemplo

O problema do slide exigirá os seguintes cálculos intermediários:

1. Calcule o salário total de cada departamento e armazene o resultado usando uma cláusula WITH.
2. Calcule o salário médio dos departamentos e armazene o resultado usando uma cláusula WITH.
3. Compare o salário total calculado na primeira etapa com o salário médio calculado na segunda etapa. Se o salário total de um departamento específico for maior que o salário médio dos departamentos, exiba o nome do departamento e o salário total correspondente.

A solução para esse problema é mostrada na próxima página.

Cláusula WITH: Exemplo

```
WITH
dept_costs AS (
  SELECT d.department_name, SUM(e.salary) AS dept_total
  FROM   employees e, departments d
  WHERE  e.department_id = d.department_id
  GROUP BY d.department_name),
avg_cost AS (
  SELECT SUM(dept_total)/COUNT(*) AS dept_avg
  FROM   dept_costs)
SELECT *
FROM   dept_costs
WHERE  dept_total >
      (SELECT dept_avg
       FROM avg_cost)
ORDER BY department_name;
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Cláusula WITH: Exemplo (continuação)

O código SQL do slide é um exemplo de situação na qual é possível melhorar o desempenho e criar código SQL mais simples usando a cláusula WITH. A consulta cria os nomes de consultas DEPT_COSTS e AVG_COST e os utiliza no corpo da consulta principal. Internamente, a cláusula WITH é resolvida como uma view em linha ou uma tabela temporária. O otimizador escolhe a resolução apropriada de acordo com o custo/benefício do armazenamento temporário dos resultados da cláusula WITH.

A saída gerada pelo código SQL no slide será esta:

| DEPARTMENT_NAME | DEPT_TOTAL |
|-----------------|------------|
| Sales | 304500 |
| Shipping | 156400 |

Notas de Uso da Cláusula WITH

- É usada apenas com instruções SELECT.
- Um nome de consulta pode ser visto por todos os blocos de consulta de elemento WITH (incluindo os respectivos blocos de subconsulta) definidos depois dele e pelo próprio bloco de consulta (incluindo os respectivos blocos de subconsulta).
- Quando o nome da consulta é igual ao de uma tabela existente, o parser pesquisa de dentro para fora. O nome do bloco de consulta tem precedência sobre o nome da tabela.
- A cláusula WITH pode conter mais de uma consulta. Nesse caso, cada consulta é separada por uma vírgula.

Sumário

Neste lição, você deverá ter aprendido que:

- **Uma subconsulta de várias colunas retorna mais de uma coluna.**
- **Comparações de várias colunas podem ser emparelhadas ou não emparelhadas.**
- **Uma subconsulta de várias colunas também pode ser usada na cláusula FROM de uma instrução SELECT.**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Sumário

Você pode usar subconsultas de várias colunas para combinar várias condições WHERE em uma única cláusula WHERE. As comparações de colunas em uma subconsulta de várias colunas podem ser comparações emparelhadas ou não emparelhadas.

É possível usar uma subconsulta para definir uma tabela para ser usada por uma consulta.

As subconsultas escalares podem ser usadas:

- Na parte da expressão e da condição DECODE e CASE
- Em todas as cláusulas de SELECT, com exceção de GROUP BY
- Na cláusula SET e na cláusula WHERE da instrução UPDATE

Sumário

- **As subconsultas correlacionadas são úteis sempre que uma subconsulta tiver que retornar um resultado diferente para cada linha candidata.**
- **O operador EXISTS é um operador booleano que testa a presença de um valor.**
- **As subconsultas correlacionadas podem ser usadas com instruções SELECT, UPDATE e DELETE.**
- **É possível utilizar a cláusula WITH para usar o mesmo bloco de consulta em uma instrução SELECT quando ocorrer mais de uma vez.**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Sumário (continuação)

O servidor Oracle executa uma subconsulta correlacionada quando essa subconsulta faz referência a uma coluna de uma tabela referenciada na instrução mãe. Uma subconsulta correlacionada é avaliada uma vez para cada linha processada pela instrução mãe. A instrução mãe pode ser uma instrução SELECT, UPDATE ou DELETE. Com a cláusula WITH, você poderá reutilizar a mesma consulta quando for custoso reavaliar o bloco de consulta e ele ocorrer mais de uma vez em uma consulta complexa.

Exercício 6: Visão Geral

Este exercício aborda os seguintes tópicos:

- Criando subconsultas de várias colunas
- Criando subconsultas correlacionadas
- Usando o operador EXISTS
- Usando subconsultas escalares
- Usando a cláusula WITH

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Exercício 6: Visão Geral

Neste exercício, você cria subconsultas de várias colunas e subconsultas correlacionadas e escalares. Você também soluciona problemas com a criação da cláusula WITH.

Exercício 6

1. Crie uma consulta para exibir o sobrenome, o número de departamento e o salário de qualquer funcionário cujo número de departamento e salário coincidam com o número de departamento e salário de qualquer funcionário que receba comissão.

| LAST_NAME | DEPARTMENT_ID | SALARY |
|-----------|---------------|--------|
| Russell | 80 | 14000 |
| Partners | 80 | 13500 |
| Errazuriz | 80 | 12000 |
| Abel | 80 | 11000 |
| Cambrault | 80 | 11000 |

...

2. Exiba o sobrenome, o nome do departamento e o salário de qualquer funcionário cujo salário e comissão coincidam com o salário e a comissão de qualquer funcionário que esteja na localização ID 1700.

...

| LAST_NAME | DEPARTMENT_NAME | SALARY |
|------------|-----------------|--------|
| Matos | Shipping | 2600 |
| OConnell | Shipping | 2600 |
| Grant | Shipping | 2600 |
| Himuro | Purchasing | 2600 |
| Vargas | Shipping | 2500 |
| Sullivan | Shipping | 2500 |
| Perkins | Shipping | 2500 |
| Patel | Shipping | 2500 |
| Marlow | Shipping | 2500 |
| Colmenares | Purchasing | 2500 |
| Whalen | Administration | 4400 |
| Gietz | Accounting | 8300 |

36 rows selected.

3. Crie uma consulta para exibir o sobrenome, a data de admissão e o salário de todos os funcionários que tenham o mesmo salário e a mesma comissão de Kochhar.

Observação: Não exiba Kochhar no conjunto de resultados.

| LAST_NAME | HIRE_DATE | SALARY |
|-----------|-----------|--------|
| De Haan | 13-JAN-93 | 17000 |

4. Crie uma consulta para exibir os funcionários que recebem um salário mais alto do que o salário dos gerentes de vendas (JOB_ID = 'SA_MAN'). Classifique os resultados dos salários do maior para o menor.

| LAST_NAME | JOB_ID | SALARY |
|-----------|---------|--------|
| King | AD_PRES | 24000 |
| Kochhar | AD_VP | 17000 |
| De Haan | AD_VP | 17000 |

Exercício 6 (continuação)

5. Exiba os detalhes do ID do funcionário, o sobrenome e o ID do departamento dos funcionários que vivem em cidades que começam com a letra *T*.

| EMPLOYEE_ID | LAST_NAME | DEPARTMENT_ID |
|-------------|-----------|---------------|
| 202 | Fay | 20 |
| 201 | Hartstein | 20 |

6. Crie uma consulta para localizar todos os funcionários que recebem mais que o salário médio nos respectivos departamentos.

Exiba o sobrenome, o salário, o ID do departamento e o salário médio do departamento. Classifique pelo salário médio. Use apelidos para as colunas recuperadas pela consulta, conforme indicado no exemplo de saída.

| ENAME | SALARY | DEPTNO | DEPT_AVG |
|-----------|--------|--------|------------|
| Bell | 4000 | 50 | 3475.55556 |
| Bull | 4100 | 50 | 3475.55556 |
| Rajs | 3500 | 50 | 3475.55556 |
| Dilly | 3600 | 50 | 3475.55556 |
| Weiss | 8000 | 50 | 3475.55556 |
| Fripp | 8200 | 50 | 3475.55556 |
| Everett | 3900 | 50 | 3475.55556 |
| Vollman | 6500 | 50 | 3475.55556 |
| Kaufling | 7900 | 50 | 3475.55556 |
| Sarchand | 4200 | 50 | 3475.55556 |
| Mourgos | 5800 | 50 | 3475.55556 |
| Ladwig | 3600 | 50 | 3475.55556 |
| ■ ■ ■ | | | |
| Vishney | 10500 | 80 | 8955.88235 |
| Russell | 14000 | 80 | 8955.88235 |
| Tucker | 10000 | 80 | 8955.88235 |
| McEwen | 9000 | 80 | 8955.88235 |
| Greene | 9500 | 80 | 8955.88235 |
| Sully | 9500 | 80 | 8955.88235 |
| King | 10000 | 80 | 8955.88235 |
| Bloom | 10000 | 80 | 8955.88235 |
| Ozer | 11500 | 80 | 8955.88235 |
| Hall | 9000 | 80 | 8955.88235 |
| Abel | 11000 | 80 | 8955.88235 |
| Hartstein | 13000 | 20 | 9500 |
| Higgins | 12000 | 110 | 10150 |
| King | 24000 | 90 | 19333.3333 |

38 rows selected.

Exercício 6 (continuação)

7. Descubra todos os funcionários que não são supervisores.
 - a. Primeiramente, faça isto usando o operador `NOT EXISTS`.

| LAST_NAME |
|-----------|
| Ernst |
| Austin |
| Pataballa |
| Lorentz |
| Faviet |
| OConnell |
| Grant |
| Whalen |
| Fay |
| Mavris |
| Baer |
| Gietz |

89 rows selected.

- b. Isto pode ser feito usando o operador `NOT IN`? Como ou por que não?
8. Crie uma consulta para exibir os sobrenomes dos funcionários que recebem menos que o salário médio nos respectivos departamentos.

| LAST_NAME |
|------------|
| Fay |
| Khoo |
| Baida |
| Tobias |
| Himuro |
| Colmenares |
| Nayer |
| Kochhar |
| De Haan |
| Chen |
| Sciarra |
| Urman |
| Popp |
| Gietz |

65 rows selected.

Exercício 6 (continuação)

9. Crie uma consulta para exibir os sobrenomes dos funcionários que têm um ou mais colegas de trabalho nos respectivos departamentos com datas de admissão posteriores, mas com salários mais altos.

| LAST_NAME |
|-----------|
| Faviet |
| Sciarra |
| Tobias |
| Bell |
| Sarchand |
| ... |
| Marvins |
| Tuvault |
| Grant |
| Perkins |
| Gee |

66 rows selected.

10. Crie uma consulta para exibir o ID do funcionário, os sobrenomes e os nomes de departamento de todos os funcionários.

Observação: Use uma subconsulta escalar para recuperar o nome do departamento na instrução SELECT.

| EMPLOYEE_ID | LAST_NAME | DEPARTMENT |
|-------------|-----------|-----------------|
| 205 | Higgins | Accounting |
| 206 | Gietz | Accounting |
| 200 | Whalen | Administration |
| 100 | King | Executive |
| 101 | Kochhar | Executive |
| 102 | De Haan | Executive |
| 108 | Greenberg | Finance |
| 109 | Faviet | Finance |
| 110 | Chen | Finance |
| 111 | Sciarra | Finance |
| 113 | Popp | Finance |
| 112 | Urman | Finance |
| 203 | Mavris | Human Resources |
| ... | | |
| 140 | Patel | Shipping |
| 141 | Rajs | Shipping |
| 142 | Davies | Shipping |
| 143 | Matos | Shipping |
| 144 | Vargas | Shipping |
| 178 | Grant | |

107 rows selected.

Exercício 6 (continuação)

11. Crie uma consulta para exibir os nomes dos departamentos cujo custo total de salário está acima de 1/8 do custo total de salário de toda a empresa. Use a cláusula `WITH` para criar esta consulta. Nomeie a consulta `SUMMARY`.

| DEPARTMENT_NAME | DEPT_TOTAL |
|-----------------|------------|
| Sales | 304500 |
| Shipping | 156400 |



Recuperação Hierárquica

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Objetivos

Ao concluir esta lição, você será capaz de:

- Interpretar o conceito de uma consulta hierárquica
- Criar um relatório estruturado em árvore
- Formatar dados hierárquicos
- Excluir ramificações da estrutura em árvore

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Objetivos

Nesta lição, você aprenderá a usar consultas hierárquicas para criar relatórios estruturados em árvore.

Dados de Amostra da Tabela EMPLOYEES

| EMPLOYEE_ID | LAST_NAME | JOB_ID | MANAGER_ID |
|-------------|-----------|------------|------------|
| 100 | King | AD_PRES | |
| 101 | Kochhar | AD_VP | 100 |
| 102 | De Haan | AD_VP | 100 |
| 103 | Hunold | IT_PROG | 102 |
| 104 | Ernst | IT_PROG | 103 |
| 105 | Austin | IT_PROG | 103 |
| 106 | Pataballa | IT_PROG | 103 |
| 107 | Lorentz | IT_PROG | 103 |
| 108 | Greenberg | FI_MGR | 101 |
| ... | | | |
| EMPLOYEE_ID | LAST_NAME | JOB_ID | MANAGER_ID |
| 196 | Walsh | SH_CLERK | 124 |
| 197 | Feeney | SH_CLERK | 124 |
| 198 | OConnell | SH_CLERK | 124 |
| 199 | Grant | SH_CLERK | 124 |
| 200 | Whalen | AD_ASST | 101 |
| 201 | Hartstein | MK_MAN | 100 |
| 202 | Fay | MK_REP | 201 |
| 203 | Mavris | HR_REP | 101 |
| 204 | Baer | PR_REP | 101 |
| 205 | Higgins | AC_MGR | 101 |
| 206 | Gietz | AC_ACCOUNT | 205 |

107 rows selected.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Dados de Amostra da Tabela EMPLOYEES

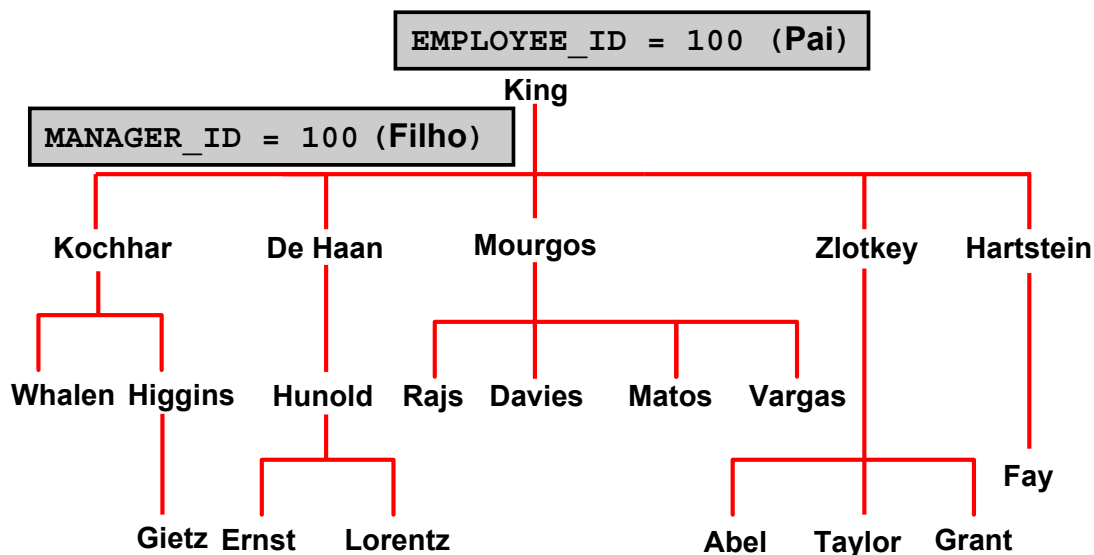
Ao usar consultas hierárquicas, você poderá recuperar dados com base em um relacionamento hierárquico natural entre as linhas de uma tabela. Um banco de dados relacional não armazena registros de forma hierárquica. No entanto, quando existe um relacionamento hierárquico entre as linhas de uma tabela, um processo denominado percurso da árvore permite que a hierarquia seja construída. Uma consulta hierárquica é um método de organizar os galhos de uma árvore em uma ordem específica.

Imagine uma árvore genealógica onde os membros mais velhos ficam na base da árvore, e os mais novos representam os galhos. Os galhos podem ter suas próprias ramificações e assim por diante.

Pode ser feita uma consulta hierárquica quando existe um relacionamento entre as linhas de uma tabela. Por exemplo, no slide, é possível verificar que os funcionários com IDs de cargos AD_VP, ST_MAN, SA_MAN e MK_MAN estão diretamente subordinados ao presidente da empresa. Sabemos disso porque a coluna MANAGER_ID desses registros contém um ID de funcionário igual a 100, que pertence ao presidente (AD_PRES).

Observação: As árvores hierárquicas são usadas em várias áreas, como genealogia humana (árvores genealógicas), rebanhos (para fins de fertilização), gerência corporativa (hierarquias de gerência), indústria (montagem de produtos), pesquisa evolutiva (desenvolvimento de espécies) e pesquisa científica.

Estrutura em Árvore Natural



ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Estrutura em Árvore Natural

A tabela EMPLOYEES tem uma estrutura em árvore que representa a linha de relatórios de gerenciamento. É possível criar a hierarquia examinando o relacionamento entre os valores equivalentes nas colunas EMPLOYEE_ID e MANAGER_ID. Para explorar esse relacionamento, é possível unir a tabela a ela mesma. A coluna MANAGER_ID contém o número de funcionário do gerente do funcionário.

O relacionamento pai/filho de uma estrutura em árvore permite controlar:

- A direção em que a hierarquia é percorrida
- A posição inicial na hierarquia

Observação: O slide exibe uma estrutura em árvore invertida da hierarquia de gerenciamento dos funcionários na tabela EMPLOYEES.

Consultas Hierárquicas

```
SELECT [LEVEL], column, expr...  
FROM table  
[WHERE condition(s)]  
[START WITH condition(s)]  
[CONNECT BY PRIOR condition(s)] ;
```

Condição WHERE:

```
expr comparison_operator expr
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

Palavras-Chave e Cláusulas

É possível identificar as consultas hierárquicas pela presença das cláusulas CONNECT BY e START WITH.

Na sintaxe:

| | |
|-------------------|---|
| SELECT | É a cláusula SELECT padrão |
| LEVEL | Para cada linha retornada por uma consulta hierárquica, a pseudocoluna LEVEL retorna 1 para uma linha-raiz, 2 para uma linha filha da raiz, e assim por diante. |
| FROM <i>table</i> | Especifica a tabela, a view ou o snapshot que contém as colunas. Você só pode fazer seleções em uma tabela. |
| WHERE | Restringe as linhas retornadas pela consulta sem afetar outras linhas da hierarquia |
| <i>condition</i> | É uma comparação com expressões |
| START WITH | Especifica as linhas-raiz da hierarquia (onde começar). Esta cláusula é necessária para uma consulta hierárquica verdadeira. |
| CONNECT BY | PRIOR especifica as colunas nas quais existe o relacionamento entre linhas mães e filhas. Esta cláusula é necessária para uma consulta hierárquica. |

A instrução SELECT não pode conter uma join ou uma consulta de uma view que contém uma join.

Percorrendo a Árvore

Posição Inicial

- Especifica a condição a ser atendida
- Aceita qualquer condição válida

```
START WITH column1 = value
```

Usando a tabela EMPLOYEES, comece com o funcionário cujo sobrenome é Kochhar.

```
...START WITH last_name = 'Kochhar'
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

Percorrendo a Árvore

As linhas a serem usadas como raiz da árvore são determinadas pela cláusula START WITH. É possível usar a cláusula START WITH com qualquer condição válida.

Exemplos

Usando a tabela EMPLOYEES, comece com King, o presidente da empresa.

```
... START WITH manager_id IS NULL
```

Usando a tabela EMPLOYEES, comece com o funcionário Kochhar. Uma condição START WITH pode conter uma subconsulta.

```
... START WITH employee_id = (SELECT employee_id
                                FROM   employees
                                WHERE  last_name = 'Kochhar')
```

Se a cláusula START WITH for omitida, o percurso da árvore começará com todas as linhas da tabela como linhas-raiz. Se for usada uma cláusula WHERE, o percurso começará com todas as linhas que atenderem à condição WHERE. Esse percurso não refletirá mais a verdadeira hierarquia.

Observação: As cláusulas CONNECT BY PRIOR e START WITH não seguem o padrão ANSI SQL.

Percorrendo a Árvore

```
CONNECT BY PRIOR column1 = column2
```

Percorra a árvore de cima para baixo usando a tabela EMPLOYEES.

```
... CONNECT BY PRIOR employee_id = manager_id
```

Direção

| | | |
|-----------------------|---|--|
| De cima para baixo | → | Column1 = Chave Mãe Column2 = Chave Filha |
| De baixo para cima | → | Column1 = Chave Filha Column2 = Chave Mãe |

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

Percorrendo a Árvore (continuação)

A direção da consulta, de mãe para filha ou de filha para mãe, é determinada pela posição da coluna `CONNECT BY PRIOR`. O operador `PRIOR` faz referência à linha mãe. Para localizar as linhas filhas de uma linha mãe, o servidor Oracle avalia a expressão `PRIOR` da linha mãe e as outras expressões de cada linha da tabela. As linhas para as quais a condição é verdadeira são as linhas filhas da linha mãe. O servidor Oracle sempre seleciona as linhas filhas avaliando a condição `CONNECT BY` relativa a uma linha mãe atual.

Exemplos

Percorra a árvore de cima para baixo usando a tabela `EMPLOYEES`. Defina um relacionamento hierárquico no qual o valor de `EMPLOYEE_ID` da linha mãe seja igual ao valor de `MANAGER_ID` da linha filha.

```
... CONNECT BY PRIOR employee_id = manager_id
```

Percorra a árvore de baixo para cima usando a tabela `EMPLOYEES`.

```
... CONNECT BY PRIOR manager_id = employee_id
```

O operador `PRIOR` não precisa ser necessariamente codificado logo após `CONNECT BY`. Portanto, a cláusula `CONNECT BY PRIOR` abaixo fornece o mesmo resultado que o exemplo anterior.

```
... CONNECT BY employee_id = PRIOR manager_id
```

Observação: A cláusula `CONNECT BY` não pode conter uma subconsulta.

Percorrendo a Árvore: De Baixo para Cima

```
SELECT employee_id, last_name, job_id, manager_id
FROM employees
START WITH employee_id = 101
CONNECT BY PRIOR manager_id = employee_id ;
```

| EMPLOYEE_ID | LAST_NAME | JOB_ID | MANAGER_ID |
|-------------|-----------|---------|------------|
| 101 | Kochhar | AD_VP | 100 |
| 100 | King | AD_PRES | |

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Percorrendo a Árvore: De Baixo para Cima

O exemplo do slide exibe uma lista de gerentes começando com o funcionário cujo ID é 101.

Exemplo

No exemplo a seguir, os valores de EMPLOYEE_ID são avaliados para a linha mãe e os valores de MANAGER_ID e SALARY são avaliados para as linhas filhas. O operador PRIOR só se aplica ao valor de EMPLOYEE_ID.

```
... CONNECT BY PRIOR employee_id = manager_id
                AND salary > 15000;
```

Para qualificar-se como uma linha filha, a linha deve ter um valor de MANAGER_ID igual ao valor de EMPLOYEE_ID da linha mãe e um valor de SALARY maior que \$15.000.

Percorrendo a Árvore: De Cima para Baixo

```
SELECT last_name || ' reports to ' ||  
PRIOR last_name "Walk Top Down"  
FROM employees  
START WITH last_name = 'King'  
CONNECT BY PRIOR employee_id = manager_id ;
```

Walk Top Down

| |
|------------------------------|
| King reports to |
| King reports to |
| Kochhar reports to King |
| Greenberg reports to Kochhar |
| Faviet reports to Greenberg |
| Chen reports to Greenberg |

...

108 rows selected.

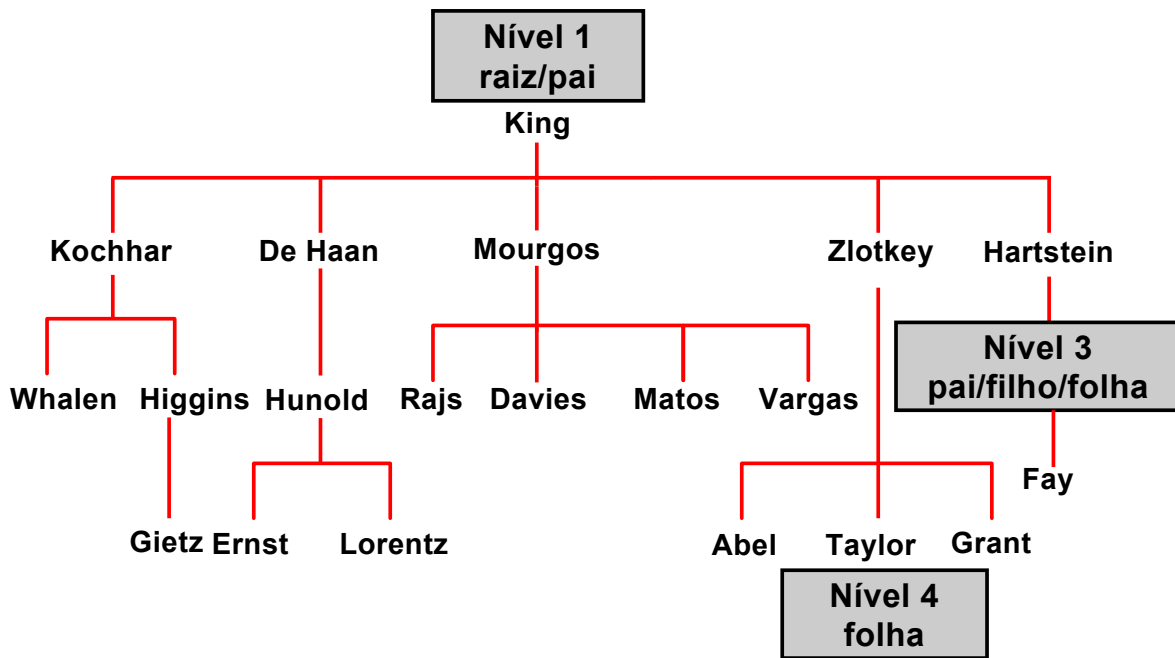
ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Percorrendo a Árvore: De Cima para Baixo

Se a árvore for percorrida de cima para baixo, os nomes dos funcionários e respectivos gerentes serão exibidos. Use o funcionário King como a posição inicial. Imprima apenas uma coluna.

Classificando Linhas com a Pseudocoluna LEVEL



ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Classificando Linhas com a Pseudocoluna LEVEL

Você pode mostrar explicitamente a classificação ou o nível de uma linha na hierarquia usando a pseudocoluna LEVEL. Assim, seu relatório será mais legível. As bifurcações onde uma ou mais ramificações separam-se de uma ramificação maior são chamadas de nós, e a extremidade de uma ramificação é chamada de folha ou nó-folha. O diagrama do slide mostra os nós da árvore invertida com os respectivos valores de LEVEL. Por exemplo, o funcionário Higgins é um nó pai e um nó filho; enquanto o funcionário Davies é um nó filho e um nó-folha.

A Pseudocoluna LEVEL

| Valor | Nível |
|-------|--|
| 1 | Um nó-raiz |
| 2 | Um filho de um nó-raiz |
| 3 | Um filho de um filho, e assim por diante |

No slide, King é o nó-raiz ou pai (LEVEL = 1). Kochhar, De Hann, Mourgos, Zlotkey, Hartstein, Higgins e Hunold são nós filhos e também nós pais (LEVEL = 2). Whalen, Rajs, Davies, Matos, Vargas, Gietz, Ernst, Lorentz, Abel, Taylor, Grant e Fay são nós filhos e nós-folha. (LEVEL = 3 e LEVEL = 4)

Observação: Um *nó-raiz* é o nó mais alto em uma árvore invertida. Um *nó filho* é qualquer nó diferente do nó-raiz. Um nó pai tem nós filhos. Um nó-folha não tem nós filhos. O número de níveis retornados por uma consulta hierárquica pode ser limitado pela memória disponível do usuário.

Formatando Relatórios Hierárquicos Usando LEVEL e LPAD

Crie um relatório exibindo os níveis de gerenciamento da empresa, começando com o nível mais elevado e recuando cada nível subsequente.

```
COLUMN org_chart FORMAT A12
SELECT LPAD(last_name, LENGTH(last_name) + (LEVEL*2) - 2, '_')
       AS org_chart
FROM   employees
START WITH last_name='King'
CONNECT BY PRIOR employee_id=manager_id
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Formatando Relatórios Hierárquicos Usando a Pseudocoluna LEVEL

São designados números de nível aos nós de uma árvore a partir da raiz. Use a function LPAD com a pseudocoluna LEVEL para exibir um relatório hierárquico como uma árvore recuada.

No exemplo do slide:

- `LPAD(char1, n [, char2])` retorna `char1`, preenchido à esquerda até o tamanho `n` com a seqüência de caracteres em `char2`. O argumento `n` é o tamanho total do valor retornado como aparece na tela do terminal.
- `LPAD(last_name, LENGTH(last_name) + (LEVEL*2) - 2, '_')` define o formato da exibição.
- `char1` é o valor de `LAST_NAME`, `n` é o tamanho total do valor retornado, o tamanho de `LAST_NAME` + $(LEVEL * 2) - 2$ e `char2` é `'_'`.

Em outras palavras, o exemplo informa ao SQL para preencher à esquerda o valor de `LAST_NAME` com o caractere `'_'` até que o tamanho da string de caracteres resultante seja igual ao valor determinado por $LENGTH(last_name) + (LEVEL * 2) - 2$.

Para King, `LEVEL` = 1. Portanto, $(2 * 1) - 2 = 2 - 2 = 0$. Assim, King não é preenchido com caracteres `'_'` e é exibido na coluna 1.

Para Kochhar, `LEVEL` = 2. Portanto, $(2 * 2) - 2 = 4 - 2 = 2$. Assim, Kochhar é preenchido com 2 caracteres `'_'` e exibido recuado.

Os outros registros da tabela `EMPLOYEES` são exibidos de forma semelhante.

Formatando Relatórios Hierárquicos Usando LEVEL (continuação)

| ORG_CHART | |
|----------------|--|
| King | |
| King | |
| __Kochhar | |
| ___Greenber g | |
| ____Faviet | |
| ____Chen | |
| ____Sciarr a | |
| ____Urman | |
| ____Popp | |
| ___Whalen | |
| ___Mavris | |
| ___Baer | |
| ___Higgins | |
| ___Gietz | |
| ... | |
| ___Kumar | |
| ___Zlotkey | |
| ___Abel | |
| ___Hutton | |
| ___Taylor | |
| ___Livingst on | |
| ___Grant | |
| ___Johnson | |
| ___Hartstein | |
| ___Fay | |

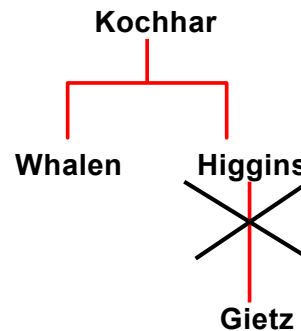
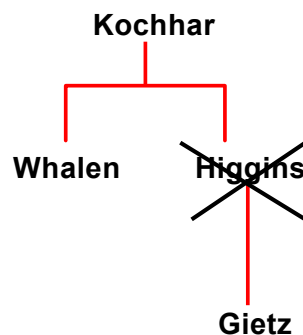
108 rows selected.

Reduzindo Ramificações

Use a cláusula WHERE
para eliminar um nó.

Use a cláusula CONNECT BY
para eliminar uma ramificação.

```
WHERE last_name != 'Higgins'
CONNECT BY PRIOR
employee_id = manager_id
AND last_name != 'Higgins'
```



ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

Reduzindo Ramificações

Você pode usar as cláusulas WHERE e CONNECT BY para reduzir a árvore, isto é, para controlar quais nós ou linhas serão exibidos. O predicado usado age como uma condição booleana.

Exemplos

Começando na raiz, percorra a árvore de cima para baixo e elimine o funcionário Higgins do resultado, mas processe as linhas filhas.

```
SELECT department_id, employee_id, last_name, job_id, salary
FROM employees
WHERE last_name != 'Higgins'
START WITH manager_id IS NULL
CONNECT BY PRIOR employee_id = manager_id;
```

Começando na raiz, percorra a árvore de cima para baixo e elimine o funcionário Higgins e todas as linhas filhas.

```
SELECT department_id, employee_id, last_name, job_id, salary
FROM employees
START WITH manager_id IS NULL
CONNECT BY PRIOR employee_id = manager_id
AND last_name != 'Higgins';
```

Sumário

Nesta lição, você aprendeu que:

- **É possível usar consultas hierárquicas para exibir um relacionamento hierárquico entre as linhas de uma tabela.**
- **Você especifica a direção e o ponto inicial da consulta.**
- **Você pode remover nós ou ramificações.**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Sumário

É possível usar consultas hierárquicas para recuperar dados com base em um relacionamento hierárquico natural entre as linhas de uma tabela. A pseudocoluna LEVEL contabiliza o quanto você percorreu uma árvore hierárquica. Você pode especificar a direção da consulta usando a cláusula CONNECT BY PRIOR. É possível especificar o ponto de partida usando a cláusula START WITH. Você pode usar as cláusulas WHERE e CONNECT BY para reduzir as ramificações da árvore.

Exercício 7: Visão Geral

Este exercício aborda os seguintes tópicos:

- **Diferenciando consultas hierárquicas de consultas não hierárquicas**
- **Percorrendo uma árvore**
- **Produzindo um relatório recuado usando a pseudocoluna LEVEL**
- **Reduzindo a estrutura da árvore**
- **Classificando a saída**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Exercício 7: Visão Geral

Neste exercício, você irá adquirir experiência na produção de relatórios hierárquicos.

Observação: A Pergunta 1 é dissertativa.

Exercício 7

1. Observe os exemplos de saída a seguir. Essas saídas são o resultado de uma consulta hierárquica? Explique por que sim e por que não.

Exemplo 1:

| EMPLOYEE_ID | LAST_NAME | MANAGER_ID | SALARY | DEPARTMENT_ID |
|-------------|-----------|------------|--------|---------------|
| 100 | King | | 24000 | 90 |
| 101 | Kochhar | 100 | 17000 | 90 |
| 102 | De Haan | 100 | 17000 | 90 |
| 201 | Hartstein | 100 | 13000 | 20 |
| 205 | Higgins | 101 | 12000 | 110 |
| 174 | Abel | 149 | 11000 | 80 |
| 149 | Zlotkey | 100 | 10500 | 80 |
| 103 | Hunold | 102 | 9000 | 60 |
| ... | | | | |
| 200 | Whalen | 101 | 4400 | 10 |
| 107 | Lorentz | 103 | 4200 | 60 |
| 141 | Rajs | 124 | 3500 | 50 |
| 142 | Davies | 124 | 3100 | 50 |
| 143 | Matos | 124 | 2600 | 50 |
| 144 | Vargas | 124 | 2500 | 50 |

20 rows selected.

Exemplo 2:

| EMPLOYEE_ID | LAST_NAME | DEPARTMENT_ID | DEPARTMENT_NAME |
|-------------|-----------|---------------|-----------------|
| 205 | Higgins | 110 | Accounting |
| 206 | Gietz | 110 | Accounting |
| 100 | King | 90 | Executive |
| 101 | Kochhar | 90 | Executive |
| 102 | De Haan | 90 | Executive |
| 149 | Zlotkey | 80 | Sales |
| 174 | Abel | 80 | Sales |
| 176 | Taylor | 80 | Sales |
| 103 | Hunold | 60 | IT |
| 104 | Ernst | 60 | IT |
| 107 | Lorentz | 60 | IT |

11 rows selected.

Exercício 7 (continuação)

Exemplo 3:

| RANK | LAST_NAME |
|------|-----------|
| 1 | King |
| 2 | Kochhar |
| 2 | De Haan |
| 3 | Hunold |
| 4 | Ernst |

2. Gere um relatório que mostre um organograma do departamento de Mourgos. Imprima os sobrenomes, os salários e os IDs dos departamentos.

| LAST_NAME | SALARY | DEPARTMENT_ID |
|-----------|--------|---------------|
| Mourgos | 5800 | 50 |
| Rajs | 3500 | 50 |
| Davies | 3100 | 50 |
| Matos | 2600 | 50 |
| Vargas | 2500 | 50 |
| Walsh | 3100 | 50 |
| Feeney | 3000 | 50 |
| OConnell | 2600 | 50 |
| Grant | 2600 | 50 |

9 rows selected.

3. Crie um relatório que mostre a hierarquia de gerentes para o funcionário Lorentz. Exiba primeiramente o seu gerente imediato.

| LAST_NAME |
|-----------|
| Hunold |
| De Haan |
| King |

Exercício 7 (continuação)

4. Crie um relatório recuado mostrando a hierarquia de gerenciamento, começando pelo funcionário cujo `LAST_NAME` é Kochhar. Imprima o sobrenome, o ID do gerente e o ID do departamento do funcionário. Defina apelidos para as colunas, conforme indicado no exemplo de saída.

| NAME | MGR | DEPTNO |
|-------------|-----|--------|
| Kochhar | 100 | 90 |
| __Greenberg | 101 | 100 |
| ___Faviet | 108 | 100 |
| ___Chen | 108 | 100 |
| ___Sciarra | 108 | 100 |
| ___Urman | 108 | 100 |
| ___Popp | 108 | 100 |
| __Whalen | 101 | 10 |
| __Mavris | 101 | 40 |
| __Baer | 101 | 70 |
| __Higgins | 101 | 110 |
| ___Gietz | 205 | 110 |

12 rows selected.

Se tiver tempo, faça o seguinte exercício:

5. Produza um organograma que mostre a hierarquia de gerenciamento da empresa. Comece pela pessoa que está no nível mais alto e exclua todas as outras que tenham um ID do cargo igual a `IT_PROG`. Exclua também De Haan e respectivos subordinados.

| LAST_NAME | EMPLOYEE_ID | MANAGER_ID |
|-----------|-------------|------------|
| King | 100 | |
| Kochhar | 101 | 100 |
| Greenberg | 108 | 101 |
| Faviet | 109 | 108 |
| Chen | 110 | 108 |
| Sciarra | 111 | 108 |

...

| LAST_NAME | EMPLOYEE_ID | MANAGER_ID |
|------------|-------------|------------|
| Livingston | 177 | 149 |
| Grant | 178 | 149 |
| Johnson | 179 | 149 |
| Hartstein | 201 | 100 |
| Fay | 202 | 201 |

101 rows selected.

Suporte a Expressões Comuns



ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Objetivos

Ao concluir esta lição, você será capaz de usar o suporte a expressões SQL comuns para pesquisar, substituir e estabelecer uma correspondência com strings específicas.



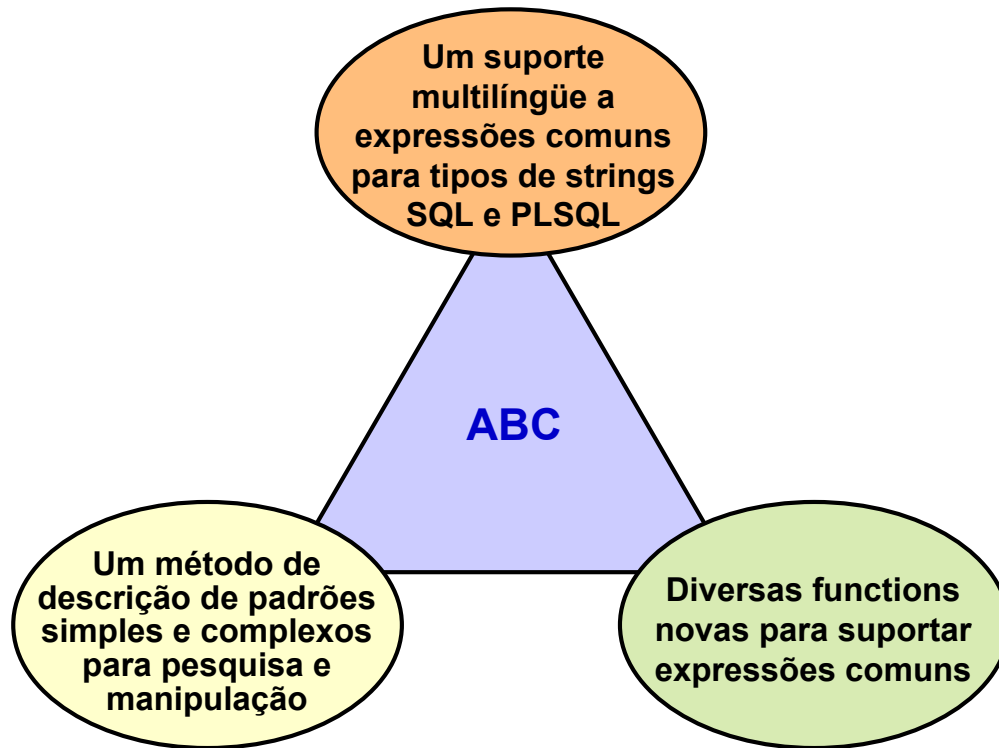
ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Objetivos

Nesta lição, você aprenderá a usar o recurso de suporte a expressões comuns que foi apresentado no Banco de Dados Oracle 10g.

Visão Geral de Expressões Comuns



ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Visão Geral de Expressões Comuns

O Banco de Dados Oracle 10g apresenta o suporte a Expressões Comuns. Esta implementação é compatível com o POSIX (Portable Operating System for UNIX) padrão, controlado pelo IEEE (Institute of Electrical and Electronics Engineers), para semântica e sintaxe de correspondência de dados ASCII. Os recursos multilíngües do Oracle ampliam os recursos de correspondência dos operadores, superando o padrão do POSIX. As expressões comuns são um método de descrição de padrões simples e complexos para pesquisa e manipulação de dados.

A manipulação e a pesquisa de strings contribuem para um grande percentual de lógica em uma aplicação baseada na Web. O uso varia desde a tarefa mais simples (encontrar as palavras "San Francisco" em um texto especificado) até a mais complexa (extrair todos os URLs do texto), ou tarefas ainda mais complexas (encontrar todas as palavras cujo segundo caractere é uma vogal).

Quando conjugado com o SQL nativo, o uso de expressões comuns possibilita operações de pesquisa e manipulação muito eficientes com os dados armazenados no banco de dados Oracle. É possível usar esse recurso para solucionar problemas que, de outro modo, teriam uma programação muito complexa.

Metacaracteres

| Símbolo | Descrição |
|---------|---|
| * | Estabelece correspondência com zero ou mais ocorrências |
| | Operação de alteração para especificar correspondências alternativas |
| ^/\$ | Estabelece correspondência com o início de linha/fim de linha |
| [] | Expressão entre colchetes para uma lista de correspondências que contenha qualquer uma das expressões representadas na lista |
| {m} | Estabelece correspondência exatamente <i>m</i> vezes |
| {m,n} | Estabelece correspondência pelo menos <i>m</i> vezes, mas não mais do que <i>n</i> vezes |
| [:] | Especifica uma classe de caracteres e corresponde a qualquer caractere dessa classe |
| \ | Pode ter 4 significados diferentes: 1. Significa uma barra propriamente dita. 2. Colocar o próximo caractere entre aspas. 3. Introduzir um operador. 4. Não fazer nada. |
| + | Corresponde a uma ou mais ocorrências |
| ? | Estabelece correspondência com zero ou uma ocorrência |
| . | Corresponde a qualquer caractere no conjunto de caracteres suportado, exceto NULL |
| () | Expressão de agrupamento, tratada como uma única subexpressão |
| [==] | Especifica as classes de equivalência |
| \n | Expressão de referência retroativa |
| [..] | Especifica um elemento de comparação, como um elemento de multicaracteres |

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

Metacaracteres

Os metacaracteres são caracteres especiais que têm um significado especial, como um curinga, um caractere repetitivo, um caractere não correspondente ou uma faixa de caracteres. Você pode usar diversos símbolos de metacaracteres predefinidos no padrão de correspondência.

Usando Metacaracteres

Problema: Localizar 'abc' em uma string:

Solução: 'abc'

Corresponde a: abc

Não corresponde a: 'def'

1

Problema: Localizar um 'a' seguido de qualquer caractere, seguido de 'c'

Metacaractere: qualquer caractere é definido por '.'

Solução: 'a.c'

Corresponde a: abc

Corresponde a: adc

Corresponde a: alc

Corresponde a: a&c

Não corresponde a: abb

2

Problema: Localizar uma ou mais ocorrências de 'a'

Metacaractere: Use o sinal de adição '+' para estabelecer uma correspondência com um ou mais dos caracteres anteriores

Solução: 'a+'

Corresponde a: a

Corresponde a: aa

Não corresponde a: bbb

3

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

Usando Metacaracteres

1. No primeiro exemplo, é executada uma correspondência simples.
2. No segundo exemplo, qualquer caractere é definido como '.'. Este exemplo pesquisa o caractere "a" seguido de qualquer caractere, seguido do caractere "c".
3. O terceiro exemplo pesquisa uma ou mais ocorrências da letra "a". O caractere "+" é usado aqui para indicar uma correspondência de um ou mais dos caracteres dos caracteres anteriores.

Você também pode pesquisar listas de caracteres não correspondentes. Uma lista de caracteres não correspondentes possibilita definir um conjunto de caracteres para os quais uma correspondência é inválida. Por exemplo, para localizar todos os caracteres, exceto "a", "b" e "c", você pode definir "^" para indicar uma não correspondência.

Expressão: [^abc]

Corresponde a: abcdef

Corresponde a: ghi

Não corresponde a: abc

Para estabelecer uma correspondência com qualquer letra que não esteja entre "a" e "i", você pode usar:

Expressão: [^a-i]

Corresponde a: hijk

Corresponde a: lmn

Não corresponde a: abcdefghi

Usando Metacaracteres (continuação)

| Sintaxe do Metacaractere | Nome do Operador | Descrição |
|--|--|---|
| . | Qualquer caractere – Ponto | Estabelece correspondência com qualquer caractere |
| + | Um ou Mais – Quantificador Sinal de Adição | Estabelece correspondência com uma ou mais ocorrências da subexpressão precedente |
| ? | Zero ou Um – Quantificador Ponto de Interrogação | Estabelece correspondência com zero ou uma ocorrência da subexpressão precedente |
| * | Zero ou Mais – Quantificador Asterisco | Estabelece correspondência com zero ou mais ocorrências da subexpressão precedente |
| { <i>m</i> } { <i>m</i> ,} { <i>m</i> , <i>n</i> } | Intervalo – Contagem Exata | Estabelece correspondência com <ul style="list-style-type: none"> • exatamente <i>m</i> ocorrências • pelo menos <i>m</i> ocorrências • pelo menos <i>m</i>, mas não mais do que <i>n</i> ocorrências da subexpressão precedente |
| [...] | Lista de Caracteres de Correspondência | Estabelece correspondência com qualquer caractere da lista ... |
| [^...] | Lista de Caracteres Não Correspondentes | Estabelece correspondência com qualquer caractere que não esteja na lista ... |
| | Ou | 'a b' corresponde ao caractere 'a' ou 'b'. |
| (...) | Subexpressão ou Agrupamento | Trata a expressão ... como uma unidade. |
| \n | Referência retroativa | Estabelece correspondência com a subexpressão precedente de número <i>n</i> th , onde <i>n</i> é um inteiro entre 1 e 9 |
| \ | Caractere de Escape | Trata o metacaractere subsequente da expressão como um literal. |

Usando Metacaracteres (continuação)

| Sintaxe do Metacaractere | Nome do Operador | Descrição |
|--------------------------|-------------------------------|---|
| ^ | Início de Âncora de Linha | Estabelece correspondência com a expressão subsequente quando ele ocorre no início de uma linha. |
| \$ | Fim de Âncora de Linha | Estabelece correspondência apenas quando ocorre no fim de uma linha. |
| [:class:] | Classe de Caracteres do POSIX | Estabelece correspondência com qualquer caractere que pertença à classe de caracteres especificada. |

Functions de Expressões Comuns

| Nome da Function | Descrição |
|-----------------------|---|
| REGEXP_LIKE | É semelhante ao operador LIKE , mas executa a correspondência de expressões comuns em vez de fazer a correspondência de padrão simples |
| REGEXP_REPLACE | Pesquisa um padrão de expressão comum e o substitui por uma string de substituição |
| REGEXP_INSTR | Procura em determinada string um padrão de expressão comum e retorna a posição em que o correspondente foi localizado |
| REGEXP_SUBSTR | Pesquisa um padrão de expressão comum em determinada string e retorna a substring correspondente |

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

Functions de Expressões Comuns

O Banco de Dados Oracle 10g oferece um conjunto de functions SQL que podem ser usadas para pesquisar e manipular strings usando expressões comuns. É possível usar essas functions com qualquer tipo de dados que contenha dados de caractere como CHAR, NCHAR, CLOB, NCLOB, NVARCHAR2 e VARCHAR2. Uma expressão comum deve ser delimitada ou encapsulada em aspas simples. Assim, garante-se que a expressão inteira será interpretada pela function SQL e que haverá maior legibilidade do código.

REGEXP_LIKE: Esta function pesquisa um padrão em uma coluna de caracteres. Use esta function na cláusula WHERE de uma consulta para retornar linhas que correspondam à expressão comum especificada.

REGEXP_REPLACE: Esta function pesquisa um padrão em uma coluna de caracteres e substitui cada ocorrência pelo padrão especificado.

REGEXP_INSTR: Esta function pesquisa determinada ocorrência de um padrão de expressão comum em uma string. Você especifica qual ocorrência deseja localizar e a posição de início da pesquisa. Esta function retorna um inteiro indicando a posição na string em que o correspondente foi localizado.

REGEXP_SUBSTR: Esta function retorna a substring real que corresponde ao padrão de expressão comum especificado.

A Sintaxe da Function REGEXP

```
REGEXP_LIKE (srcstr, pattern [,match_option])
```

```
REGEXP_INSTR (srcstr, pattern [, position [, occurrence  
[, return_option [, match_option]]]])
```

```
REGEXP_SUBSTR (srcstr, pattern [, position  
[, occurrence [, match_option]]])
```

```
REGEXP_REPLACE(srcstr, pattern [,replacestr [, position  
[, occurrence [, match_option]]]])
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

A Sintaxe da Function REGEXP

A tabela a seguir contém descrições dos termos mostrados no slide de sintaxe.

| | |
|---------------|---|
| srcstr | Valor de pesquisa |
| pattern | Expressão comum |
| occurrence | Ocorrência a ser pesquisada |
| position | Posição de início da pesquisa |
| return_option | Posição de início ou fim da ocorrência |
| replacestr | Padrão de substituição da string de caracteres |
| match_option | Opção para alterar a correspondência default. Pode incluir um ou mais dos seguintes valores: "C" —usa correspondência com distinção entre maiúsculas e minúsculas (default) "I" —usa correspondência sem distinção entre maiúsculas e minúsculas "n" — permite operador de correspondência com qualquer caractere "m" —trata a string de origem como uma linha múltipla |

Executando Pesquisas Básicas

```
SELECT first_name, last_name  
FROM employees  
WHERE REGEXP_LIKE (first_name, '^Ste(v|ph)en$');
```

| FIRST_NAME | LAST_NAME |
|------------|-----------|
| Steven | King |
| Steven | Markle |
| Stephen | Stiles |

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Exemplo de REGEXP_LIKE

Nesta consulta à tabela EMPLOYEES, serão exibidos todos os funcionários cujos nomes contenham Steven ou Stephen. Na expressão usada,

'**^Ste(v|ph)en\$**' :

- ^ indica o início da sentença
- \$ indica o fim da sentença
- | indica ou

Verificando a Presença de um Padrão

```
SELECT street_address,  
       REGEXP_INSTR(street_address, '^[:alpha:]')  
FROM   locations  
WHERE  
       REGEXP_INSTR(street_address, '^[:alpha:]') > 1;
```

| STREET_ADDRESS | REGEXP_INSTR(STREET_ADDRESS, '^[:ALPHA:]') |
|--|--|
| Magdalen Centre, The Oxford Science Park | 9 |
| Schwanthalerstr. 7031 | 16 |
| Rua Frei Caneca 1360 | 4 |
| Murtenstrasse 921 | 14 |
| Pieter Breughelstraat 837 | 7 |
| Mariano Escobedo 9991 | 8 |

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

Verificando a Presença de um Padrão

Neste exemplo, a function `REGEXP_INSTR` é usada para pesquisar o endereço a fim de encontrar o local do primeiro caractere não alfabético, independentemente se está em maiúsculas ou minúsculas. A pesquisa é executada apenas para os endereços que não começam com um número. Observe que `[:<class>:]` implica uma classe de caracteres e corresponde a qualquer caractere dessa classe; `[:alpha:]` corresponde a qualquer caractere alfabético. Os resultados são exibidos.

Na expressão usada na consulta `'^[:alpha:]'`:

- `[` inicia a expressão
- `^` indica NOT
- `[:alpha:]` indica a classe de caracteres alfa
- `]` finaliza a expressão

Observação: O operador de classe de caracteres do POSIX permite pesquisar uma expressão em uma lista de caracteres que é membro de uma classe de caracteres específica do POSIX. É possível usar este operador para pesquisar uma formatação específica, como caracteres maiúsculos, ou para pesquisar caracteres especiais, como dígitos ou caracteres de pontuação. Todo o conjunto de classes de caracteres do POSIX é suportado. Use a sintaxe `[:class:]` em que *class* é o nome da classe de caracteres do POSIX a ser pesquisada. A expressão comum abaixo pesquisa um ou mais caracteres maiúsculos consecutivos: `[[upper:]]+`.

Exemplo de Extração de Substrings

```
SELECT REGEXP_SUBSTR(street_address , ' [^ ]+ ')  
"Road" FROM locations;
```

| Road |
|-------------|
| Via |
| Calle |
| |
| |
| Jabberwocky |
| Interiors |
| Zagora |
| Charade |
| ... |

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Exemplo de uma Extração de Substring

Neste exemplo, os nomes de ruas são extraídos da tabela LOCATIONS. Para isso, o conteúdo da coluna STREET_ADDRESS que está antes do primeiro espaço é retornado usando a function REGEXP_SUBSTR. Na expressão usada na consulta ' [^]+ ':

- [inicia a expressão
- ^ indica NOT
- indica espaço
-] finaliza a expressão
- + indica 1 ou mais
- indica espaço

Substituindo Padrões

```
SELECT REGEXP_REPLACE( country_name, '(.)',  
                        '\1 ') "REGEXP_REPLACE"  
FROM countries;
```

| REGEXP_REPLACE(COUNTRY_NAME,'(',')','1') |
|--|
| Argentina |
| Australia |
| Belgium |
| Brazil |
| Canada |
| Switzerland |
| China |

...

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Substituindo Padrões

Este exemplo examina COUNTRY_NAME. O banco de dados Oracle reformata este padrão com um espaço após cada caractere não nulo da string. Os resultados são mostrados.

Expressões Comuns e Constraints de Verificação

```
ALTER TABLE emp8  
  ADD CONSTRAINT email_addr  
  CHECK (REGEXP_LIKE(email, '@')) NOVALIDATE ;
```

1

```
INSERT INTO emp8 VALUES  
  (500, 'Christian', 'Patel',  
   'ChrisP2creme.com', 1234567890,  
   '12-Jan-2004', 'HR_REP', 2000, null, 102, 40) ;
```

2

```
INSERT INTO emp8 VALUES  
*
```

ERROR at line 1:
ORA-02290: check constraint (ORA20.EMAIL_ADDR) violated

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Expressões Comuns e Constraints de Verificação

As expressões comuns também podem ser usadas em constraints de verificação. Neste exemplo, uma constraint de verificação é adicionada à coluna EMAIL da tabela EMPLOYEES. Isso garantirá que apenas as strings que contiverem um símbolo "@" serão aceitas. A constraint é testada. A constraint de verificação é violada porque o endereço de e-mail não contém o símbolo solicitado. A cláusula NOVALIDATE garante que os dados existentes não serão verificados.

Sumário

Nesta lição, você aprendeu a usar o suporte a expressões SQL comuns para pesquisar, substituir e estabelecer uma correspondência com strings específicas.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

Sumário

Nesta lição, você aprendeu a usar o recurso de suporte a expressões comuns que foi apresentado no Banco de Dados Oracle 10g.

Exercício 8: Visão Geral

Este exercício abrange o uso de expressões comuns.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

Exercício 8: Visão Geral

Este exercício abrange a pesquisa e a substituição de dados usando expressões comuns.

Exercício 8

1. Crie uma consulta para pesquisar, na tabela EMPLOYEES, todos os funcionários cujos nomes começam com "Ne" ou "Na".

| FIRST_NAME | LAST_NAME |
|------------|-----------|
| Nanette | Cambrault |
| Nancy | Greenberg |
| Neena | Kochhar |
| Nandita | Sarchand |

2. Crie uma consulta que remova, na exibição, os espaços da coluna STREET_ADDRESS da tabela LOCATIONS.

| REGEXP_REPLACE(STREET_ADDRESS, ",") |
|-------------------------------------|
| 1297ViaColadiRie |
| 93091CalledellaTesta |
| 2017Shinjuku-ku |
| 9450Kamiya-cho |
| 2014JabberwockyRd |
| 2011InteriorsBlvd |
| 2007ZagoraSt |
| 2004CharadeRd |
| 147SpadinaAve |
| 6092BoxwoodSt |
| 40-5-12Laogianggen |
| 1298Vileparle(E) |
| 12-98VictoriaStreet |
| 198ClementiNorth |
| 8204ArthurSt |
| MagdalenCentre,TheOxfordSciencePark |
| 9702ChesterRoad |
| Schwanthalerstr.7031 |
| RuaFreiCaneca1360 |
| 20RuedesCorps-Saints |
| Murtenstrasse921 |
| PieterBreughelstraat837 |
| MarianoEscobedo9991 |

23 rows selected.

Exercício 8 (continuação)

3. Crie uma consulta que, na exibição, substitua "St" por "Street" na coluna STREET_ADDRESS da tabela LOCATIONS. Tenha cuidado para não afetar as linhas que já incluam "Street". Exiba apenas as linhas que forem afetadas.

| REGEXP_REPLACE(STREET_ADDRESS,'ST\$','STREET') |
|--|
| 2007 Zagora Street |
| 6092 Boxwood Street |
| 12-98 Victoria Street |
| 8204 Arthur Street |

Apêndice A

Soluções dos Exercícios

Exercício 1: Soluções

Para responder da pergunta 6 em diante, você precisará conectar-se ao banco de dados usando o iSQL*Plus. Para isso, acione o browser Internet Explorer no computador cliente. Informe o URL no formato *http://machinename:5561/isqlplus/* e use a conta *oraxx* e a respectiva *senha* e *identificador de serviço* (no formato *Tx*) fornecidos pelo instrutor para efetuar logon no banco de dados.

1. Que privilégio deve ser concedido a um usuário para que ele efetue logon no servidor Oracle? Este privilégio é de sistema ou de objeto?
O privilégio de sistema CREATE SESSION
2. Que privilégio deve ser concedido a um usuário para que ele crie tabelas?
O privilégio CREATE TABLE
3. Se você criar uma tabela, quem poderá passar privilégios a outros usuários da sua tabela?
Você ou qualquer outra pessoa a quem tenha concedido esses privilégios que utilize a cláusula WITH GRANT OPTION.
4. Você é o DBA. Você está criando vários usuários que precisam dos mesmos privilégios de sistema. O que você deve usar para facilitar o seu trabalho?
Criar uma atribuição que contenha os privilégios de sistema e concedê-la aos usuários.
5. Que comando você pode usar para alterar a senha?
A instrução ALTER USER
6. Conceda a outro usuário acesso à sua tabela DEPARTMENTS. Faça com que o usuário conceda a você acesso de consulta à tabela DEPARTMENTS dele.

A Equipe 2 executa a instrução GRANT.

```
GRANT select
ON      departments
TO      <user1>;
```

A Equipe 1 executa a instrução GRANT.

```
GRANT select
ON      departments
TO      <user2>;
```

Onde *user1* é o nome da Equipe 1, e *user2* é o nome da Equipe 2.

7. Consulte todas as linhas da tabela DEPARTMENTS.

```
SELECT  *
FROM    departments;
```

Exercício 1: Soluções (continuação)

8. Adicione uma nova linha à tabela DEPARTMENTS. A Equipe 1 deve adicionar Educação como departamento número 500. A Equipe 2 deve adicionar Recursos Humanos como departamento número 510. Consulte a tabela da outra equipe.

```
A Equipe 1 executa esta instrução INSERT.
INSERT INTO departments(department_id, department_name)
VALUES (500, 'Education');
COMMIT;

A Equipe 2 executa esta instrução INSERT.
INSERT INTO departments(department_id, department_name)
VALUES (510, 'Human Resources');
COMMIT;
```

9. Crie um sinônimo para a tabela DEPARTMENTS da outra equipe.

```
A Equipe 1 cria um sinônimo denominado team2.
CREATE SYNONYM team2
FOR <oraxx>.DEPARTMENTS;

A Equipe 2 cria um sinônimo denominado team1.
CREATE SYNONYM team1
FOR <oraxx>. DEPARTMENTS;
```

10. Consulte todas as linhas da tabela DEPARTMENTS da outra equipe usando o sinônimo.

```
A Equipe 1 executa esta instrução SELECT.
SELECT *
FROM team2;

A Equipe 2 executa esta instrução SELECT.
SELECT *
FROM team1;
```

Exercício 1: Soluções (continuação)

11. Consulte o dicionário de dados USER_TABLES para ver as informações das suas tabelas.

```
SELECT table_name
FROM   user_tables;
```

12. Consulte a view de dicionário de dados ALL_TABLES para ver as informações de todas as tabelas que você pode acessar. Exclua as suas tabelas.

```
SELECT table_name, owner
FROM   all_tables
WHERE  owner <> 'Oraxx';
```

13. Revogue o privilégio SELECT da outra equipe.

```
A Equipe 1 revoga o privilégio.
REVOKE select
  ON    departments
  FROM  <oraxx>;

A Equipe 2 revoga o privilégio.
REVOKE select
  ON    departments
  FROM  <oraxx>;
```

14. Remova a linha que você inseriu na tabela DEPARTMENTS na etapa 8 e salve as alterações.

```
A Equipe 1 executa esta instrução INSERT.
DELETE FROM departments
WHERE department_id = 500;
COMMIT;

A Equipe 2 executa esta instrução INSERT.
DELETE FROM departments
WHERE department_id = 510;
COMMIT;
```


Exercício 2: Soluções

1. Crie a tabela DEPT2 com base no gráfico de instâncias de tabela a seguir. Inclua a sintaxe em um script denominado lab_02_01.sql e execute a instrução do script para criar a tabela. Confirme a criação da tabela.

| | | |
|----------------|--------|----------|
| Nome da Coluna | ID | NAME |
| Tipo de Chave | | |
| Nulo/Exclusivo | | |
| Tabela FK | | |
| Coluna FK | | |
| Tipo de dados | NUMBER | VARCHAR2 |
| Tamanho | 7 | 25 |

```
CREATE TABLE dept2
(id NUMBER(7),
 name VARCHAR2(25));

DESCRIBE dept2
```

2. Preencha a tabela DEPT2 com dados da tabela DEPARTMENTS. Inclua apenas as colunas de que necessita.

```
INSERT INTO dept2
SELECT department_id, department_name
FROM departments;
```

3. Crie a tabela EMP2 com base no gráfico de instâncias de tabela a seguir. Inclua a sintaxe em um script denominado lab_02_03.sql e execute a instrução do script para criar a tabela. Confirme a criação da tabela.

Exercício 2: Soluções (continuação)

```
CREATE TABLE emp2
(id          NUMBER(7),
last_name    VARCHAR2(25),
first_name   VARCHAR2(25),
dept_id      NUMBER(7));

DESCRIBE emp2
```

4. Modifique a tabela EMP2 para que aceite sobrenomes mais longos de funcionários. Confirme a modificação.

```
ALTER TABLE emp2
MODIFY (last_name    VARCHAR2(50));

DESCRIBE emp2
```

5. Confirme se as tabelas DEPT2 e EMP2 foram armazenadas no dicionário de dados. (Dica: USER_TABLES)

```
SELECT table_name
FROM user_tables
WHERE table_name IN ('DEPT2', 'EMP2');
```

6. Crie a tabela EMPLOYEES2 com base na estrutura da tabela EMPLOYEES. Inclua apenas as colunas EMPLOYEE_ID, FIRST_NAME, LAST_NAME, SALARY e DEPARTMENT_ID. Nomeie as colunas da nova tabela como ID, FIRST_NAME, LAST_NAME, SALARY e DEPT_ID, respectivamente.

```
CREATE TABLE employees2 AS
SELECT employee_id id, first_name, last_name, salary,
       department_id dept_id
FROM employees;
```

7. Elimine a tabela EMP2.

```
DROP TABLE emp2;
```

Exercício 2: Soluções (continuação)

8. Verifique se a tabela está na lixeira.

```
SELECT original_name, operation, droptime  
FROM recyclebin;
```

9. Cancele a eliminação da tabela EMP2.

```
FLASHBACK TABLE emp2 TO BEFORE DROP;  
DESC emp2;
```

10. Elimine a coluna FIRST_NAME da tabela EMPLOYEES2. Confirme a sua modificação verificando a descrição da tabela.

```
ALTER TABLE employees2  
DROP COLUMN first_name;  
  
DESCRIBE employees2
```

11. Na tabela EMPLOYEES2, marque a coluna DEPT_ID como UNUSED. Confirme a sua modificação verificando a descrição da tabela.

```
ALTER TABLE employees2  
SET UNUSED (dept_id);  
  
DESCRIBE employees2
```

12. Elimine todas as colunas UNUSED da tabela EMPLOYEES2. Confirme a sua modificação verificando a descrição da tabela.

```
ALTER TABLE employees2  
DROP UNUSED COLUMNS;  
  
DESCRIBE employees2
```

Exercício 2: Soluções (continuação)

13. Adicione uma constraint PRIMARY KEY em nível de tabela para a tabela EMP2 na coluna ID. A constraint deve ser nomeada durante a criação. Nomeie-a como my_emp_id_pk.

```
ALTER TABLE    emp2
ADD CONSTRAINT my_emp_id_pk PRIMARY KEY (id);
```

14. Crie uma constraint PRIMARY KEY para a tabela DEPT2 usando a coluna ID. A constraint deve ser nomeada durante a criação. Nomeie-a como my_dept_id_pk.

```
ALTER TABLE    dept2
ADD CONSTRAINT my_dept_id_pk PRIMARY KEY(id);
```

15. Adicione uma referência de chave estrangeira à tabela EMP2 que garante que o funcionário não foi designado para um departamento inexistente. Nomeie a constraint como my_emp_dept_id_fk.

```
ALTER TABLE emp2
ADD CONSTRAINT my_emp_dept_id_fk
FOREIGN KEY (dept_id) REFERENCES dept2(id);
```

16. Confirme se as constraints foram adicionadas, consultando a view USER_CONSTRAINTS.

Anote os tipos e os nomes das constraints.

```
SELECT    constraint_name, constraint_type
FROM      user_constraints
WHERE     table_name IN ('EMP2', 'DEPT2');
```

17. Exiba os nomes e os tipos dos objetos da view do dicionário de dados USER_OBJECTS para as tabelas EMP2 e DEPT2. Observe que foram criadas tabelas novas e um índice novo.

```
SELECT    object_name, object_type
FROM      user_objects
WHERE     object_name LIKE 'EMP%'
OR        object_name LIKE 'DEPT%';
```

Exercício 2: Soluções (continuação)

Se tiver tempo, faça o seguinte exercício:

18. Modifique a tabela EMP2. Adicione uma coluna COMMISSION do tipo de dados NUMBER, precisão 2, escala 2. Adicione uma constraint à coluna COMMISSION que garanta um valor de comissão maior do que zero.

```
ALTER TABLE emp2
ADD commission NUMBER(2,2)
CONSTRAINT my_emp_comm_ck CHECK (commission > 0);
```

19. Elimine as tabelas EMP2 e DEPT2 de modo que não possam ser restauradas. Verifique a lixeira.

```
DROP TABLE emp2 PURGE;
DROP TABLE dept2 PURGE;

SELECT original_name, operation, droptime
FROM recyclebin;
```

20. Crie a tabela DEPT_NAMED_INDEX com base no gráfico de instâncias de tabela a seguir. Nomeie o índice para a coluna PRIMARY KEY como DEPT_PK_IDX.

| Nome da Coluna | Deptno | Dname |
|----------------|--------|----------|
| Chave primária | Sim | |
| Tipo de Dados | Número | VARCHAR2 |
| Tamanho | 4 | 30 |

```
CREATE TABLE DEPT_NAMED_INDEX
(deptno NUMBER(4)
PRIMARY KEY USING INDEX
(CREATE INDEX dept_pk_idx ON
DEPT_NAMED_INDEX(deptno)),
dname VARCHAR2(30));
```

Exercício 3: Soluções

1. Execute o script `lab_03_01.sql` da pasta lab para criar a tabela `SAL_HISTORY`.
2. Exiba a estrutura da tabela `SAL_HISTORY`.

```
DESC sal_history
```

3. Execute o script `lab_03_03.sql.sql` da pasta lab para criar a tabela `MGR_HISTORY`.
4. Exiba a estrutura da tabela `MGR_HISTORY`.

```
DESC mgr_history
```

5. Execute o script `lab_03_05.sql.sql` da pasta lab para criar a tabela `SPECIAL_SAL`.
6. Exiba a estrutura da tabela `SPECIAL_SAL`.

```
DESC special_sal
```

7. a. Crie uma consulta que faça o seguinte:
 - Recupere na tabela `EMPLOYEES` os detalhes de ID do funcionário, data de contratação, salário e o ID do gerente desses funcionários cujo ID é inferior a 125.
 - Se o salário for superior a \$20.000, insira os detalhes sobre o ID do funcionário e o salário na tabela `SPECIAL_SAL`.
 - Insira os detalhes sobre o ID do funcionário, a data de contratação e o salário na tabela `SAL_HISTORY`.
 - Insira os detalhes sobre o ID do funcionário, o ID do gerente e o salário na tabela `MGR_HISTORY`.

```
INSERT ALL
WHEN SAL > 20000 THEN
  INTO special_sal VALUES (EMPID, SAL)
ELSE
  INTO sal_history VALUES(EMPID,HIREDATE,SAL)
  INTO mgr_history VALUES(EMPID,MGR,SAL)
SELECT employee_id EMPID, hire_date HIREDATE,
```

Exercício 3: Soluções (continuação)

```
salary SAL, manager_id MGR  
FROM employees  
WHERE employee_id < 125;
```

- b. Exiba os registros da tabela SPECIAL_SAL.

```
SELECT * FROM special_sal;
```

- c. Exiba os registros da tabela SAL_HISTORY.

```
SELECT * FROM sal_history;
```

- d. Exiba os registros da tabela MGR_HISTORY.

```
SELECT * FROM mgr_history;
```

8. a. Execute o script lab_03_08a.sql da pasta lab para criar a tabela SALES_SOURCE_DATA.
- b. Execute o script lab_03_08b.sql da pasta lab para inserir registros na tabela SALES_SOURCE_DATA.
- c. Exiba a estrutura da tabela SALES_SOURCE_DATA.

```
DESC sales_source_data
```

- d. Exiba os registros da tabela SALES_SOURCE_DATA.

```
SELECT * FROM SALES_SOURCE_DATA;
```

- e. Execute o script lab_03_08c.sql da pasta lab para criar a tabela SALES_INFO.
- f. Exiba a estrutura da tabela SALES_INFO.

```
DESC sales_info
```

Exercício 3: Soluções (continuação)

- g. Crie uma consulta que faça o seguinte:
- Na tabela `SALES_SOURCE_DATA`, recuperar os detalhes sobre o ID do funcionário, o ID da semana, vendas na segunda-feira, vendas na terça-feira, vendas na quarta-feira, vendas na quinta-feira e vendas na sexta-feira.
 - Criar uma transformação de modo que cada registro recuperado da tabela `SALES_SOURCE_DATA` seja convertido em vários registros para a tabela `SALES_INFO`.

Dica: Use uma instrução `INSERT` de criação de pivô.

```
INSERT ALL
  INTO sales_info VALUES (employee_id, week_id, sales_MON)
  INTO sales_info VALUES (employee_id, week_id, sales_TUE)
  INTO sales_info VALUES (employee_id, week_id, sales_WED)
  INTO sales_info VALUES (employee_id, week_id,
    sales_THUR)
  INTO sales_info VALUES (employee_id, week_id, sales_FRI)
SELECT EMPLOYEE_ID, week_id, sales_MON, sales_TUE,
sales_WED, sales_THUR, sales_FRI FROM sales_source_data;
```

- h. Exiba os registros da tabela `SALES_INFO`.

```
SELECT * FROM sales_info;
```

9. Você tem os dados dos antigos funcionários armazenados em um arquivo sem formatação denominado `emp.data` e deseja armazenar em uma tabela os nomes e os IDs de e-mail de todos os funcionários, antigos e atuais. Para isso, primeiro crie uma tabela externa denominada `EMP_DATA` usando o arquivo de origem `emp.dat` no diretório `emp_dir`. Você pode usar o script `lab_03_09.sql` para essa tarefa.

Exercício 3: Soluções (continuação)

```
CREATE TABLE emp_data
  (first_name  VARCHAR2(20)
  ,last_name   VARCHAR2(20)
  , email      VARCHAR2(30)
  )
ORGANIZATION EXTERNAL
(
  TYPE oracle_loader
  DEFAULT DIRECTORY emp_dir
  ACCESS PARAMETERS
  (
    RECORDS DELIMITED BY NEWLINE CHARACTERSET US7ASCII
    NOBADFILE
    NOLOGFILE
    FIELDS
    ( first_name POSITION ( 1:20) CHAR
    , last_name POSITION (22:41) CHAR
    , email   POSITION (43:72) CHAR )
  )
  LOCATION ('emp.dat') ) ;
```

10. Em seguida, execute o script `lab_03_10.sql` para criar a tabela EMP_HIST.
- Aumente o tamanho da coluna de e-mail para 45.
 - Intercale os dados da tabela EMP_DATA criada no último laboratório com os dados da tabela EMP_HIST. Suponha que os dados da tabela externa EMP_DATA sejam os mais atualizados. Se uma linha da tabela EMP_DATA corresponde à tabela EMP_HIST, atualize a coluna de e-mail da tabela EMP_HIST para corresponder à linha da tabela EMP_DATA. Se uma linha da tabela EMP_DATA não corresponder à tabela EMP_HIST, insira-a na tabela EMP_HIST. As linhas são coincidentes quando o nome e o sobrenome do funcionário são idênticos.

```
MERGE INTO EMP_HIST f USING EMP_DATA h
  ON (f.first_name = h.first_name
  AND f.last_name = h.last_name)
WHEN MATCHED THEN
  UPDATE SET f.email = h.email
WHEN NOT MATCHED THEN
  INSERT (f.first_name
  , f.last_name
  , f.email)
  VALUES (h.first_name
  , h.last_name
  , h.email);
```

Exercício 3: Soluções (continuação)

- c. Recupere as linhas da tabela EMP_HIST após a intercalação.

```
SELECT * FROM emp_hist;
```

11. Crie a tabela EMP3 usando o script lab_03_11.sql. Na tabela EMP3, altere o departamento de Kochhar para 60 e faça commit da alteração. Em seguida, altere o departamento de Kochhar para 50 e faça commit da alteração. Controle as alterações de Kochhar usando o recurso Row Versions.

```
SELECT VERSIONS_STARTTIME "START_DATE",  
       VERSIONS_ENDTIME "END_DATE",  DEPARTMENT_ID  
FROM EMP3  
       VERSIONS BETWEEN SCN MINVALUE AND MAXVALUE  
WHERE LAST_NAME = 'Kochhar';
```

Exercício 4: Soluções

1. Crie uma consulta para exibir as seguintes informações sobre os funcionários cujo ID de gerente é menor que 120:
 - ID do gerente
 - ID do cargo e salário total para cada ID de cargo para funcionários que estão subordinados ao mesmo gerente
 - Salário total desses gerentes
 - Salário total desses gerentes, independentemente dos IDs dos cargos

```
SELECT manager_id, job_id, sum(salary)
FROM employees
WHERE manager_id < 120
GROUP BY ROLLUP(manager_id, job_id);
```

2. Observe a resposta da questão 1. Crie uma consulta usando a função GROUPING para determinar se os valores NULL nas colunas correspondentes às expressões GROUP BY são causados pela operação ROLLUP.

```
SELECT manager_id MGR , job_id JOB,
       sum(salary), GROUPING(manager_id), GROUPING(job_id)
FROM employees
WHERE manager_id < 120
GROUP BY ROLLUP(manager_id, job_id);
```

3. Crie uma consulta para exibir as seguintes informações sobre os funcionários cujo ID de gerente é menor que 120:
 - ID do gerente
 - Cargo e salários totais de cada cargo para funcionários que estão subordinados ao mesmo gerente
 - Salário total desses gerentes
 - Valores de tabelas de referência para exibir o salário total para cada cargo, independentemente do gerente
 - Salário total, independentemente dos cargos

```
SELECT manager_id, job_id, sum(salary)
FROM employees
WHERE manager_id < 120
GROUP BY CUBE(manager_id, job_id);
```

Exercício 4: Soluções (continuação)

4. Observe a resposta da questão 3. Crie uma consulta usando a função `GROUPING` para determinar se os valores `NULL` nas colunas correspondentes às expressões `GROUP BY` são causados pela operação `CUBE`.

```
SELECT manager_id MGR ,job_id JOB,
       sum(salary),GROUPING(manager_id),GROUPING(job_id)
FROM   employees
WHERE  manager_id < 120
GROUP BY CUBE(manager_id,job_id);
```

5. Usando `GROUPING SETS`, crie uma consulta para exibir os seguintes agrupamentos:
- department_id, manager_id, job_id
 - department_id, job_id
 - manager_id, job_id

A consulta deve calcular a soma dos salários para cada um desses grupos.

```
SELECT department_id, manager_id, job_id, SUM(salary)
FROM employees
GROUP BY
GROUPING SETS ((department_id, manager_id, job_id),
               (department_id, job_id),(manager_id,job_id));
```

Exercício 5: Soluções

1. Altere a sessão para definir o NLS_DATE_FORMAT como DD-MON-YYYY HH24:MI:SS.

```
ALTER SESSION SET NLS_DATE_FORMAT =  
'DD-MON-YYYY HH24:MI:SS';
```

2. a. Crie consultas para exibir os deslocamentos (TZ_OFFSET) dos seguintes fusos horários:

EUA/Pacífico Novo

```
SELECT TZ_OFFSET ('US/Pacific-New') from dual;
```

Cingapura

```
SELECT TZ_OFFSET ('Singapore') from dual;
```

Egito

```
SELECT TZ_OFFSET ('Egypt') from dual;
```

- b. Altere a sessão para definir o valor do parâmetro TIME_ZONE como o deslocamento do fuso horário de EUA/Pacífico Novo.

```
ALTER SESSION SET TIME_ZONE = '-7:00';
```

- c. Exiba SYSDATE, CURRENT_DATE, CURRENT_TIMESTAMP e LOCALTIMESTAMP para esta sessão.

Observação: A saída poderá ser diferente, dependendo da data de execução do comando.

```
SELECT CURRENT_DATE, CURRENT_TIMESTAMP,  
LOCALTIMESTAMP FROM DUAL;
```

- d. Altere a sessão para definir o valor do parâmetro TIME_ZONE como o deslocamento do fuso horário de Cingapura.

```
ALTER SESSION SET TIME_ZONE = '+8:00';
```

- e. Exiba CURRENT_DATE, CURRENT_TIMESTAMP e LOCALTIMESTAMP para esta sessão.

Observação: A saída poderá ser diferente, dependendo da data de execução do comando.

```
SELECT CURRENT_DATE, CURRENT_TIMESTAMP,  
LOCALTIMESTAMP FROM DUAL;
```

Exercício 5: Soluções (continuação)

Observação: No exercício anterior, observe que `CURRENT_DATE`, `CURRENT_TIMESTAMP` e `LOCALTIMESTAMP` são sensíveis ao fuso horário da sessão.

3. Crie uma consulta para exibir `DBTIMEZONE` e `SESSIONTIMEZONE`.

```
SELECT DBTIMEZONE,SESSIONTIMEZONE
FROM DUAL;
```

4. Crie uma consulta para extrair o ano da coluna `HIRE_DATE` da tabela `EMPLOYEES` em relação aos funcionários que trabalham no departamento 80.

```
SELECT last_name, EXTRACT (YEAR FROM HIRE_DATE)
FROM employees
WHERE department_id = 80;
```

5. Altere a sessão para definir `NLS_DATE_FORMAT` como `DD-MON-YYYY`.

```
ALTER SESSION SET NLS_DATE_FORMAT = 'DD-MON-YYYY';
```

6. Examine e execute o script `lab_05_06.sql` para criar a tabela `SAMPLE_DATES` e preenchê-la.

- a. Selecione na tabela e exiba os dados.

```
SELECT * FROM sample_dates;
```

- b. Modifique o tipo de dados da coluna `DATE_COL` e altere-o para `TIMESTAMP`. Selecione na tabela para exibir os dados.

```
ALTER TABLE sample_dates MODIFY date_col TIMESTAMP;
SELECT * FROM sample_dates;
```

- c. Tente modificar o tipo de dados da coluna `DATE_COL` e altere-o para `TIMESTAMP WITH TIME ZONE`. O que acontece?

```
ALTER TABLE sample_dates MODIFY date_col
TIMESTAMP WITH TIME ZONE;
```

Você não pode alterar o tipo de dados da coluna `DATE_COL`, pois o servidor Oracle não permite a conversão de `TIMESTAMP` em `TIMESTAMP WITH TIMEZONE` usando a instrução `ALTER`.

Exercício 5: Soluções (continuação)

7. Crie uma consulta para recuperar os sobrenomes da tabela EMPLOYEES e calcular o status da avaliação. Se o ano de admissão foi 2000, exiba Needs Review para o status da avaliação. Caso contrário, exiba not this year! Nomeie a coluna de status da avaliação como Review. Classifique os resultados pela coluna HIRE_DATE.

Dica: Use uma expressão CASE com função EXTRACT para calcular o status da avaliação.

```
SELECT e.last_name
,      (CASE extract(year from e.hire_date)
        WHEN 1998 THEN 'Needs Review'
        ELSE 'not this year!'
        END )      AS "Review "
FROM   employees e
ORDER BY e.hire_date;
```

8. Crie uma consulta para imprimir os sobrenomes e os anos de serviço de cada funcionário. Se o funcionário foi contratado há cinco anos ou mais, imprima 5 years of service. Se o funcionário foi contratado há 10 anos ou mais, imprima 10 years of service. Se o funcionário foi contratado há 15 anos ou mais, imprima 15 years of service. Se não houver correspondência com nenhuma dessas condições, imprima maybe next year! Classifique os resultados pela coluna HIRE_DATE. Use a tabela EMPLOYEES.

Dica: Use expressões CASE e TO_YMINTERVAL.

```
SELECT e.last_name, hire_date, sysdate,
       (CASE
         WHEN (sysdate -TO_YMINTERVAL('15-0'))>=
              hire_date THEN      '15 years of service'
         WHEN (sysdate -TO_YMINTERVAL('10-0'))>=
              hire_date
         THEN '10 years of service'
         WHEN (sysdate - TO_YMINTERVAL('5-0'))>=
              hire_date
         THEN '5 years of service'
         ELSE 'maybe next year!'
         END) AS "Awards"
FROM   employees e;
```

Exercício 6: Soluções

1. Crie uma consulta para exibir o sobrenome, o número de departamento e o salário de qualquer funcionário cujo número de departamento e salário coincidam com o número de departamento e salário de qualquer funcionário que receba comissão.

```
SELECT last_name, department_id, salary
FROM   employees
WHERE  (salary, department_id) IN
      (SELECT salary, department_id
       FROM   employees
       WHERE  commission_pct IS NOT NULL);
```

2. Exiba o sobrenome, o nome do departamento e o salário de qualquer funcionário cujo salário e comissão coincidam com o salário e a comissão de qualquer funcionário que esteja na localização ID1700.

```
SELECT e.last_name, d.department_name, e.salary
FROM   employees e, departments d
WHERE  e.department_id = d.department_id
AND    (salary, NVL(commission_pct,0)) IN
      (SELECT salary, NVL(commission_pct,0)
       FROM   employees e, departments d
       WHERE  e.department_id = d.department_id
       AND    d.location_id = 1700);
```

3. Crie uma consulta para exibir o sobrenome, a data de contratação e o salário de todos os funcionários que tenham o mesmo salário e a mesma comissão de Kochhar.

Observação: Não exiba Kochhar no conjunto de resultados.

```
SELECT last_name, hire_date, salary
FROM   employees
WHERE  (salary, NVL(commission_pct,0)) IN
      (SELECT salary, NVL(commission_pct,0)
       FROM   employees
       WHERE  last_name = 'Kochhar')
AND last_name != 'Kochhar';
```


Exercício 6: Soluções (continuação)

4. Crie uma consulta para exibir os funcionários que recebem um salário mais alto do que o salário dos gerentes de vendas (JOB_ID = 'SA_MAN'). Classifique os resultados dos salários do maior para o menor.

```
SELECT last_name, job_id, salary
FROM   employees
WHERE  salary > ALL
      (SELECT salary
       FROM   employees
       WHERE  job_id = 'SA_MAN')
ORDER BY salary DESC;
```

5. Exiba os detalhes do ID do funcionário, o sobrenome e o ID do departamento dos funcionários que vivem em cidades que começam com a letra T.

```
SELECT employee_id, last_name, department_id
FROM   employees
WHERE  department_id IN (SELECT department_id
                        FROM departments
                        WHERE location_id IN
                              (SELECT location_id
                               FROM locations
                               WHERE city LIKE 'T%'));
```

6. Crie uma consulta para localizar todos os funcionários que recebem mais que o salário médio nos respectivos departamentos. Exiba o sobrenome, o salário, o ID do departamento e o salário médio do departamento. Classifique pelo salário médio. Use apelidos para as colunas recuperadas pela consulta, conforme indicado no exemplo de saída.

```
SELECT e.last_name ename, e.salary salary,
       e.department_id deptno, AVG(a.salary) dept_avg
FROM   employees e, employees a
WHERE  e.department_id = a.department_id
AND    e.salary > (SELECT AVG(salary)
                  FROM   employees
                  WHERE  department_id = e.department_id )
GROUP BY e.last_name, e.salary, e.department_id
ORDER BY AVG(a.salary);
```

Exercício 6: Soluções (continuação)

7. Descubra todos os funcionários que não são supervisores.
 - a. Primeiramente, faça isso usando o operador NOT EXISTS.

```
SELECT outer.last_name
FROM   employees outer
WHERE  NOT EXISTS (SELECT 'X'
                   FROM employees inner
                   WHERE inner.manager_id =
                        outer.employee_id);
```

- b. Isso pode ser feito com o operador NOT IN? Como ou por que não?

```
SELECT outer.last_name
FROM   employees outer
WHERE  outer.employee_id
NOT IN (SELECT inner.manager_id
        FROM   employees inner);
```

Esta solução não é uma boa alternativa. A subconsulta escolhe um valor NULL, de modo que a consulta completa não retorna nenhuma linha. O motivo é que todas as condições que comparam um valor NULL resultam em um valor NULL. Sempre que os valores NULL tenderem a ser parte do valor definido, *não* use NOT IN como substituto para NOT EXISTS.

8. Crie uma consulta para exibir os sobrenomes dos funcionários que recebem menos que o salário médio nos respectivos departamentos.

```
SELECT last_name
FROM   employees outer
WHERE  outer.salary < (SELECT AVG(inner.salary)
                      FROM employees inner
                      WHERE inner.department_id
                           = outer.department_id);
```

Exercício 6: Soluções (continuação)

9. Crie uma consulta para exibir os sobrenomes dos funcionários que têm um ou mais colegas de trabalho nos respectivos departamentos com datas de contratação posteriores, mas com salários mais altos.

```
SELECT last_name
FROM employees outer
WHERE EXISTS (SELECT 'X'
              FROM employees inner
              WHERE inner.department_id =
                    outer.department_id
              AND inner.hire_date > outer.hire_date
              AND inner.salary > outer.salary);
```

10. Crie uma consulta para exibir o ID do funcionário, os sobrenomes e os nomes de departamento de todos os funcionários.

Observação: Use uma subconsulta escalar para recuperar o nome do departamento na instrução SELECT.

```
SELECT employee_id, last_name,
       (SELECT department_name
        FROM departments d
        WHERE e.department_id =
              d.department_id ) department
FROM employees e
ORDER BY department;
```

11. Crie uma consulta para exibir os nomes dos departamentos cujo custo total de salário está acima de 1/8 do custo total de salário de toda a empresa. Use a cláusula WITH para criar essa consulta. Nomeie a consulta como SUMMARY.

```
WITH
summary AS (
  SELECT d.department_name, SUM(e.salary) AS dept_total
  FROM employees e, departments d
  WHERE e.department_id = d.department_id
  GROUP BY d.department_name)
SELECT department_name, dept_total
FROM summary
WHERE dept_total > ( SELECT SUM(dept_total) * 1/8
                    FROM summary )
ORDER BY dept_total DESC;
```

Exercício 7: Soluções

1. Observe os exemplos de saída a seguir. Essas saídas são o resultado de uma consulta hierárquica? Explique por que sim e por que não.

Exemplo 1: Esta não é uma consulta hierárquica. O relatório apenas tem uma classificação decrescente baseada em SALARY.

Exemplo 2: Esta não é uma consulta hierárquica. Há duas tabelas envolvidas.

Exemplo 3: Sim, trata-se de uma consulta hierárquica, pois exibe a estrutura em árvore que representa a linha de relatórios de gerenciamento da tabela EMPLOYEES.

2. Gere um relatório que mostre um organograma do departamento de Mourgos. Imprima os sobrenomes, os salários e os IDs dos departamentos.

```
SELECT last_name, salary, department_id
FROM employees
START WITH last_name = 'Mourgos'
CONNECT BY PRIOR employee_id = manager_id;
```

3. Crie um relatório que mostre a hierarquia de gerentes para o funcionário Lorentz. Exiba primeiramente o seu gerente imediato.

```
SELECT last_name
FROM employees
WHERE last_name != 'Lorentz'
START WITH last_name = 'Lorentz'
CONNECT BY PRIOR manager_id = employee_id;
```

4. Crie um relatório recuado que mostre a hierarquia de gerenciamento, começando pelo funcionário cujo LAST_NAME é Kochhar. Imprima o sobrenome, o ID do gerente e o ID do departamento do funcionário. Defina apelidos para as colunas, conforme indicado no exemplo de saída.

```
COLUMN name FORMAT A20
SELECT LPAD(last_name, LENGTH(last_name)+(LEVEL*2)-
2, '_')
      name, manager_id mgr, department_id deptno
FROM employees
START WITH last_name = 'Kochhar'
CONNECT BY PRIOR employee_id = manager_id
/
COLUMN name CLEAR
```

Exercício 7: Soluções (continuação)

Se tiver tempo, faça os seguintes exercícios:

5. Produza um organograma que mostre a hierarquia de gerenciamento da empresa. Comece pela pessoa que está no nível mais alto e exclua todas as outras que tenham um ID do cargo igual a IT_PROG. Exclua também De Haan e respectivos subordinados.

```
SELECT last_name, employee_id, manager_id
FROM   employees
WHERE  job_id != 'IT_PROG'
START WITH manager_id IS NULL
CONNECT BY PRIOR employee_id = manager_id
AND last_name != 'De Haan';
```

Exercício 8: Soluções

1. Crie uma consulta para pesquisar, na tabela EMPLOYEES, todos os funcionários cujos nomes começam com "Ne" ou "Na".

```
SELECT first_name, last_name,  
FROM employees  
WHERE REGEXP_LIKE (first_name, '^N(e|a).');
```

2. Crie uma consulta que remova, na exibição, os espaços da coluna STREET_ADDRESS da tabela LOCATIONS.

```
SELECT regexp_replace (street_address, ' ','')  
FROM locations;
```

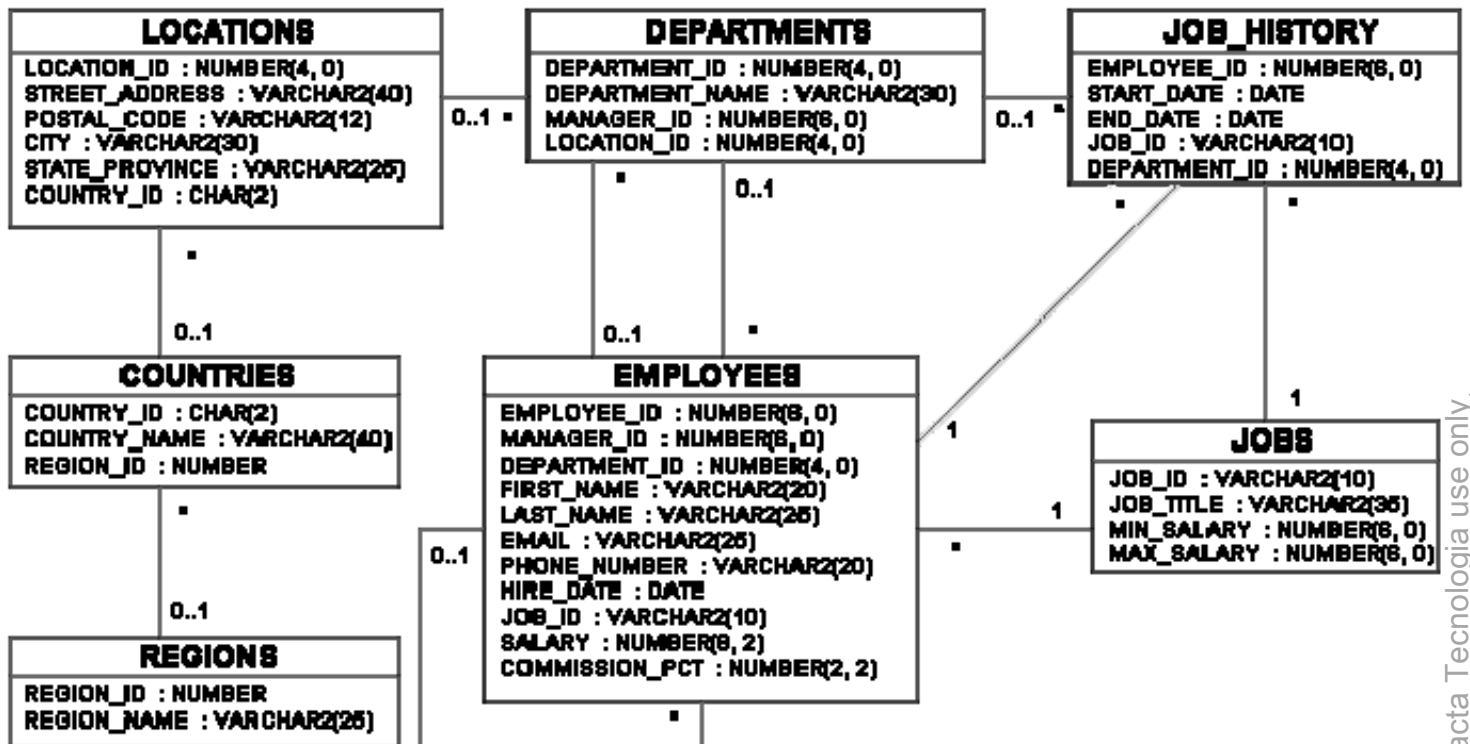
3. Crie uma consulta que, na exibição, substitua "St" por "Street" na coluna STREET_ADDRESS da tabela LOCATIONS. Tenha cuidado para não afetar as linhas que já incluam "Street". Exiba apenas as linhas que forem afetadas.

```
SELECT REGEXP_REPLACE (street_address, 'St$',  
'Street') FROM locations  
WHERE REGEXP_LIKE (street_address, 'St');
```

Apêndice B

Descrições e Dados de Tabelas

DIAGRAMA DE RELACIONAMIENTO ENTRE ENTIDADES



Tabelas do Esquema

Guia `SELECT * FROM;`

| TNAME | TABTYPE | CLUSTERID |
|------------------|---------|-----------|
| COUNTRIES | TABLE | |
| DEPARTMENTS | TABLE | |
| EMPLOYEES | TABLE | |
| EMP_DETAILS_VIEW | VIEW | |
| JOBS | TABLE | |
| JOB_HISTORY | TABLE | |
| LOCATIONS | TABLE | |
| REGIONS | TABLE | |

8 rows selected.

Tabela REGIONS

DESCRIBE regions

| Name | Null? | Type |
|-------------|----------|--------------|
| REGION_ID | NOT NULL | NUMBER |
| REGION_NAME | | VARCHAR2(25) |

SELECT * FROM regions;

| REGION_ID | REGION_NAME |
|-----------|------------------------|
| 1 | Europe |
| 2 | Americas |
| 3 | Asia |
| 4 | Middle East and Africa |

Tabela COUNTRIES

DESCRIBE countries

| Name | Null? | Type |
|--------------|----------|--------------|
| COUNTRY_ID | NOT NULL | CHAR(2) |
| COUNTRY_NAME | | VARCHAR2(40) |
| REGION_ID | | NUMBER |

SELECT * FROM countries;

| CO | COUNTRY_NAME | REGION_ID |
|----|--------------------------|-----------|
| AR | Argentina | 2 |
| AU | Australia | 3 |
| BE | Belgium | 1 |
| BR | Brazil | 2 |
| CA | Canada | 2 |
| CH | Switzerland | 1 |
| CN | China | 3 |
| DE | Germany | 1 |
| DK | Denmark | 1 |
| EG | Egypt | 4 |
| FR | France | 1 |
| HK | HongKong | 3 |
| IL | Israel | 4 |
| IN | India | 3 |
| CO | COUNTRY_NAME | REGION_ID |
| IT | Italy | 1 |
| JP | Japan | 3 |
| KW | Kuwait | 4 |
| MX | Mexico | 2 |
| NG | Nigeria | 4 |
| NL | Netherlands | 1 |
| SG | Singapore | 3 |
| UK | United Kingdom | 1 |
| US | United States of America | 2 |
| ZM | Zambia | 4 |
| ZW | Zimbabwe | 4 |

25 rows selected.

Tabela LOCATIONS

DESCRIBE locations;

| Name | Null? | Type |
|----------------|----------|--------------|
| LOCATION_ID | NOT NULL | NUMBER(4) |
| STREET_ADDRESS | | VARCHAR2(40) |
| POSTAL_CODE | | VARCHAR2(12) |
| CITY | NOT NULL | VARCHAR2(30) |
| STATE_PROVINCE | | VARCHAR2(25) |
| COUNTRY_ID | | CHAR(2) |

SELECT * FROM locations;

| LOCATION_ID | STREET_ADDRESS | POSTAL_CODE | CITY | STATE_PROVINCE | CO |
|-------------|--|-------------|---------------------|-------------------|----|
| 1000 | 1297 Via Cola di Rie | 00989 | Roma | | IT |
| 1100 | 93091 Calle della Testa | 10934 | Venice | | IT |
| 1200 | 2017 Shinjuku-ku | 1689 | Tokyo | Tokyo Prefecture | JP |
| 1300 | 9450 Kamiya-cho | 6823 | Hiroshima | | JP |
| 1400 | 2014 Jabberwocky Rd | 26192 | Southlake | Texas | US |
| 1500 | 2011 Interiors Blvd | 99236 | South San Francisco | California | US |
| 1600 | 2007 Zagora St | 50090 | South Brunswick | New Jersey | US |
| 1700 | 2004 Charade Rd | 98199 | Seattle | Washington | US |
| 1800 | 147 Spadina Ave | M5V 2L7 | Toronto | Ontario | CA |
| 1900 | 6092 Boxwood St | Y5W 9T2 | Whitehorse | Yukon | CA |
| 2000 | 40-5-12 Laogianggen | 190518 | Beijing | | CN |
| 2100 | 1298 Vileparle (E) | 490231 | Bombay | Maharashtra | IN |
| LOCATION_ID | STREET_ADDRESS | POSTAL_CODE | CITY | STATE_PROVINCE | CO |
| 2400 | 8204 Arthur St | | London | | UK |
| 2500 | Magdalen Centre, The Oxford Science Park | OX9 9ZB | Oxford | Oxford | UK |
| 2600 | 9702 Chester Road | 09629850293 | Stretford | Manchester | UK |
| 2700 | Schwanthalerstr. 7031 | 80925 | Munich | Bavaria | DE |
| 2800 | Rua Frei Caneca 1360 | 01307-002 | Sao Paulo | Sao Paulo | BR |
| 2900 | 20 Rue des Corps-Saints | 1730 | Geneva | Geneve | CH |
| 3000 | Murtenstrasse 921 | 3095 | Bern | BE | CH |
| 3100 | Pieter Breughelstraat 837 | 3029SK | Utrecht | Utrecht | NL |
| 3200 | Mariano Escobedo 9991 | 11932 | Mexico City | Distrito Federal, | MX |

23 rows selected.

Tabela DEPARTMENTS

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) |

SELECT * FROM departments;

| DEPARTMENT_ID | DEPARTMENT_NAME | MANAGER_ID | LOCATION_ID |
|---------------|----------------------|------------|-------------|
| 10 | Administration | 200 | 1700 |
| 20 | Marketing | 201 | 1800 |
| 30 | Purchasing | 114 | 1700 |
| 40 | Human Resources | 203 | 2400 |
| 50 | Shipping | 121 | 1500 |
| 60 | IT | 103 | 1400 |
| 70 | Public Relations | 204 | 2700 |
| 80 | Sales | 145 | 2500 |
| 90 | Executive | 100 | 1700 |
| 100 | Finance | 108 | 1700 |
| 110 | Accounting | 205 | 1700 |
| 120 | Treasury | | 1700 |
| 130 | Corporate Tax | | 1700 |
| 140 | Control And Credit | | 1700 |
| DEPARTMENT_ID | DEPARTMENT_NAME | MANAGER_ID | LOCATION_ID |
| 150 | Shareholder Services | | 1700 |
| 160 | Benefits | | 1700 |
| 170 | Manufacturing | | 1700 |
| 180 | Construction | | 1700 |
| 190 | Contracting | | 1700 |
| 200 | Operations | | 1700 |
| 210 | IT Support | | 1700 |
| 220 | NOC | | 1700 |
| 230 | IT Helpdesk | | 1700 |
| 240 | Government Sales | | 1700 |
| 250 | Retail Sales | | 1700 |
| 260 | Recruiting | | 1700 |
| 270 | Payroll | | 1700 |

27 rows selected.

Tabela JOBS

DESCRIBE jobs

| Name | Null? | Type |
|------------|----------|--------------|
| JOB_ID | NOT NULL | VARCHAR2(10) |
| JOB_TITLE | NOT NULL | VARCHAR2(35) |
| MIN_SALARY | | NUMBER(6) |
| MAX_SALARY | | NUMBER(6) |

SELECT * FROM jobs;

| JOB_ID | JOB_TITLE | MIN_SALARY | MAX_SALARY |
|------------|---------------------------------|------------|------------|
| AD_PRES | President | 20000 | 40000 |
| AD_VP | Administration Vice President | 15000 | 30000 |
| AD_ASST | Administration Assistant | 3000 | 6000 |
| FI_MGR | Finance Manager | 8200 | 16000 |
| FI_ACCOUNT | Accountant | 4200 | 9000 |
| AC_MGR | Accounting Manager | 8200 | 16000 |
| AC_ACCOUNT | Public Accountant | 4200 | 9000 |
| SA_MAN | Sales Manager | 10000 | 20000 |
| SA_REP | Sales Representative | 6000 | 12000 |
| PU_MAN | Purchasing Manager | 8000 | 15000 |
| PU_CLERK | Purchasing Clerk | 2500 | 5500 |
| ST_MAN | Stock Manager | 5500 | 8500 |
| ST_CLERK | Stock Clerk | 2000 | 5000 |
| SH_CLERK | Shipping Clerk | 2500 | 5500 |
| JOB_ID | JOB_TITLE | MIN_SALARY | MAX_SALARY |
| IT_PROG | Programmer | 4000 | 10000 |
| MK_MAN | Marketing Manager | 9000 | 15000 |
| MK_REP | Marketing Representative | 4000 | 9000 |
| HR_REP | Human Resources Representative | 4000 | 9000 |
| PR_REP | Public Relations Representative | 4500 | 10500 |

19 rows selected.

Tabela EMPLOYEES

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) |

Tabela EMPLOYEES

Os cabeçalhos das colunas COMMISSION_PCT, MANAGER_ID e DEPARTMENT_ID foram definidos como COMM, MGRID e DEPTID na captura de tela abaixo para que os valores da tabela se ajustassem à página.

```
SELECT * FROM employees;
```

| EMPLOYEE_ID | FIRST_NAME | LAST_NAME | EMAIL | PHONE_NUMBER | HIRE_DATE | JOB_ID | SALARY | comm | mgrid | deptid |
|-------------|-------------|-------------|----------|--------------|-----------|------------|--------|------|-------|--------|
| 100 | Steven | King | SKING | 515.123.4567 | 17-JUN-87 | AD_PRES | 24000 | | | 90 |
| 101 | Neena | Kochhar | NKOCHHAR | 515.123.4568 | 21-SEP-89 | AD_VP | 17000 | | 100 | 90 |
| 102 | Lex | De Haan | LDEHAAN | 515.123.4569 | 13-JAN-93 | AD_VP | 17000 | | 100 | 90 |
| 103 | Alexander | Hunold | AHUNOLD | 590.423.4567 | 03-JAN-90 | IT_PROG | 9000 | | 102 | 60 |
| 104 | Bruce | Ernst | BERNST | 590.423.4568 | 21-MAY-91 | IT_PROG | 6000 | | 103 | 60 |
| 105 | David | Austin | DAUSTIN | 590.423.4569 | 25-JUN-97 | IT_PROG | 4800 | | 103 | 60 |
| 106 | Valli | Pataballa | VPATABAL | 590.423.4560 | 05-FEB-98 | IT_PROG | 4800 | | 103 | 60 |
| 107 | Diana | Lorentz | DLORENTZ | 590.423.5567 | 07-FEB-99 | IT_PROG | 4200 | | 103 | 60 |
| 108 | Nancy | Greenberg | NGREENBE | 515.124.4569 | 17-AUG-94 | FI_MGR | 12000 | | 101 | 100 |
| 109 | Daniel | Faviet | DFAVET | 515.124.4169 | 16-AUG-94 | FI_ACCOUNT | 9000 | | 108 | 100 |
| 110 | John | Chen | JCHEN | 515.124.4269 | 28-SEP-97 | FI_ACCOUNT | 8200 | | 108 | 100 |
| 111 | Ismael | Sciarra | ISCIARRA | 515.124.4369 | 30-SEP-97 | FI_ACCOUNT | 7700 | | 108 | 100 |
| 112 | Jose Manuel | Urman | JMURMAN | 515.124.4469 | 07-MAR-98 | FI_ACCOUNT | 7800 | | 108 | 100 |
| 113 | Luis | Popp | LPOPP | 515.124.4567 | 07-DEC-99 | FI_ACCOUNT | 6900 | | 108 | 100 |
| EMPLOYEE_ID | FIRST_NAME | LAST_NAME | EMAIL | PHONE_NUMBER | HIRE_DATE | JOB_ID | SALARY | comm | mgrid | deptid |
| 114 | Den | Raphaely | DRAPHEAL | 515.127.4561 | 07-DEC-94 | PU_MAN | 11000 | | 100 | 30 |
| 115 | Alexander | Khoo | AKHOO | 515.127.4562 | 18-MAY-95 | PU_CLERK | 3100 | | 114 | 30 |
| 116 | Shelli | Baida | SBAIDA | 515.127.4563 | 24-DEC-97 | PU_CLERK | 2900 | | 114 | 30 |
| 117 | Sigal | Tobias | STOBIAS | 515.127.4564 | 24-JUL-97 | PU_CLERK | 2800 | | 114 | 30 |
| 118 | Guy | Himuro | GHIMURO | 515.127.4565 | 15-NOV-98 | PU_CLERK | 2600 | | 114 | 30 |
| 119 | Karen | Colmenares | KCOLMENA | 515.127.4566 | 10-AUG-99 | PU_CLERK | 2500 | | 114 | 30 |
| 120 | Matthew | Weiss | MWEISS | 650.123.1234 | 18-JUL-96 | ST_MAN | 8000 | | 100 | 50 |
| 121 | Adam | Fripp | AFRIPP | 650.123.2234 | 10-APR-97 | ST_MAN | 8200 | | 100 | 50 |
| 122 | Payam | Kaufling | PKAUFLIN | 650.123.3234 | 01-MAY-95 | ST_MAN | 7900 | | 100 | 50 |
| 123 | Shanta | Vollman | SVOLLMAN | 650.123.4234 | 10-OCT-97 | ST_MAN | 6500 | | 100 | 50 |
| 124 | Kevin | Mourgos | KMOURGOS | 650.123.5234 | 16-NOV-99 | ST_MAN | 5800 | | 100 | 50 |
| 125 | Julia | Nayer | JNAYER | 650.124.1214 | 16-JUL-97 | ST_CLERK | 3200 | | 120 | 50 |
| 126 | Irene | Mikkilineni | IMIKKILI | 650.124.1224 | 28-SEP-98 | ST_CLERK | 2700 | | 120 | 50 |
| 127 | James | Landry | JLANDRY | 650.124.1334 | 14-JAN-99 | ST_CLERK | 2400 | | 120 | 50 |

Tabela EMPLOYEES (continuação)

| EMPLOYEE_ID | FIRST_NAME | LAST_NAME | EMAIL | PHONE_NUMBER | HIRE_DATE | JOB_ID | SALARY | comm | mgrid | deptid |
|-------------|-------------|------------|----------|--------------------|-----------|----------|--------|------|-------|--------|
| 128 | Steven | Markle | SMARKLE | 650.124.1434 | 08-MAR-00 | ST_CLERK | 2200 | | 120 | 50 |
| 129 | Laura | Bissot | LBISSOT | 650.124.5234 | 20-AUG-97 | ST_CLERK | 3300 | | 121 | 50 |
| 130 | Mozhe | Atkinson | MATKINSO | 650.124.6234 | 30-OCT-97 | ST_CLERK | 2800 | | 121 | 50 |
| 131 | James | Marlow | JAMRLOW | 650.124.7234 | 16-FEB-97 | ST_CLERK | 2500 | | 121 | 50 |
| 132 | TJ | Olson | TJOLSON | 650.124.8234 | 10-APR-99 | ST_CLERK | 2100 | | 121 | 50 |
| 133 | Jason | Mallin | JMALLIN | 650.127.1934 | 14-JUN-96 | ST_CLERK | 3300 | | 122 | 50 |
| 134 | Michael | Rogers | MROGERS | 650.127.1834 | 26-AUG-98 | ST_CLERK | 2900 | | 122 | 50 |
| 135 | Ki | Gee | KGEE | 650.127.1734 | 12-DEC-99 | ST_CLERK | 2400 | | 122 | 50 |
| 136 | Hazel | Philtanker | HPHILTAN | 650.127.1634 | 06-FEB-00 | ST_CLERK | 2200 | | 122 | 50 |
| 137 | Renske | Ladwig | RLADWIG | 650.121.1234 | 14-JUL-95 | ST_CLERK | 3600 | | 123 | 50 |
| 138 | Stephen | Stiles | SSTILES | 650.121.2034 | 26-OCT-97 | ST_CLERK | 3200 | | 123 | 50 |
| 139 | John | Seo | JSEO | 650.121.2019 | 12-FEB-98 | ST_CLERK | 2700 | | 123 | 50 |
| 140 | Joshua | Patel | JPATEL | 650.121.1834 | 06-APR-98 | ST_CLERK | 2500 | | 123 | 50 |
| 141 | Trenna | Rajs | TRAJS | 650.121.8009 | 17-OCT-95 | ST_CLERK | 3500 | | 124 | 50 |
| EMPLOYEE_ID | FIRST_NAME | LAST_NAME | EMAIL | PHONE_NUMBER | HIRE_DATE | JOB_ID | SALARY | comm | mgrid | deptid |
| 142 | Curtis | Davies | CDAVES | 650.121.2994 | 29-JAN-97 | ST_CLERK | 3100 | | 124 | 50 |
| 143 | Randall | Matos | RMATOS | 650.121.2874 | 15-MAR-98 | ST_CLERK | 2600 | | 124 | 50 |
| 144 | Peter | Vargas | PVARGAS | 650.121.2004 | 09-JUL-98 | ST_CLERK | 2500 | | 124 | 50 |
| 145 | John | Russell | JRUSSEL | 011.44.1344.429268 | 01-OCT-96 | SA_MAN | 14000 | .4 | 100 | 80 |
| 146 | Karen | Partners | KPARTNER | 011.44.1344.467268 | 05-JAN-97 | SA_MAN | 13500 | .3 | 100 | 80 |
| 147 | Alberto | Errazuriz | AERRAZUR | 011.44.1344.429278 | 10-MAR-97 | SA_MAN | 12000 | .3 | 100 | 80 |
| 148 | Gerald | Cambrault | GCAMBRAU | 011.44.1344.619268 | 15-OCT-99 | SA_MAN | 11000 | .3 | 100 | 80 |
| 149 | Eleni | Zlotkey | EZLOTKEY | 011.44.1344.429018 | 29-JAN-00 | SA_MAN | 10500 | .2 | 100 | 80 |
| 150 | Peter | Tucker | PTUCKER | 011.44.1344.129268 | 30-JAN-97 | SA_REP | 10000 | .3 | 145 | 80 |
| 151 | David | Bernstein | DBERNSTE | 011.44.1344.345268 | 24-MAR-97 | SA_REP | 9500 | .25 | 145 | 80 |
| 152 | Peter | Hall | PHALL | 011.44.1344.478968 | 20-AUG-97 | SA_REP | 9000 | .25 | 145 | 80 |
| 153 | Christopher | Olsen | COLSEN | 011.44.1344.498718 | 30-MAR-98 | SA_REP | 8000 | .2 | 145 | 80 |
| 154 | Nanette | Cambrault | NCAMBRAU | 011.44.1344.987668 | 09-DEC-98 | SA_REP | 7500 | .2 | 145 | 80 |
| 155 | Oliver | Tuvault | OTUVAULT | 011.44.1344.486508 | 23-NOV-99 | SA_REP | 7000 | .15 | 145 | 80 |
| EMPLOYEE_ID | FIRST_NAME | LAST_NAME | EMAIL | PHONE_NUMBER | HIRE_DATE | JOB_ID | SALARY | comm | mgrid | deptid |
| 156 | Janette | King | JKING | 011.44.1345.429268 | 30-JAN-96 | SA_REP | 10000 | .35 | 146 | 80 |
| 157 | Patrick | Sully | PSULLY | 011.44.1345.929268 | 04-MAR-96 | SA_REP | 9500 | .35 | 146 | 80 |
| 158 | Allan | McEwen | AMCEWEN | 011.44.1345.829268 | 01-AUG-96 | SA_REP | 9000 | .35 | 146 | 80 |
| 159 | Lindsey | Smith | LSMITH | 011.44.1345.729268 | 10-MAR-97 | SA_REP | 8000 | .3 | 146 | 80 |
| 160 | Louise | Doran | LDORAN | 011.44.1345.629268 | 15-DEC-97 | SA_REP | 7500 | .3 | 146 | 80 |
| 161 | Sarath | Sewall | SSEWALL | 011.44.1345.529268 | 03-NOV-98 | SA_REP | 7000 | .25 | 146 | 80 |
| 162 | Clara | Vishney | CVISHNEY | 011.44.1346.129268 | 11-NOV-97 | SA_REP | 10500 | .25 | 147 | 80 |
| 163 | Danielle | Greene | DGREENE | 011.44.1346.229268 | 19-MAR-99 | SA_REP | 9500 | .15 | 147 | 80 |
| 164 | Mattea | Marvins | MMARVINS | 011.44.1346.329268 | 24-JAN-00 | SA_REP | 7200 | .1 | 147 | 80 |
| 165 | David | Lee | DLEE | 011.44.1346.529268 | 23-FEB-00 | SA_REP | 6800 | .1 | 147 | 80 |
| 166 | Sundar | Ande | SANDE | 011.44.1346.629268 | 24-MAR-00 | SA_REP | 6400 | .1 | 147 | 80 |
| 167 | Amit | Banda | ABANDA | 011.44.1346.729268 | 21-APR-00 | SA_REP | 6200 | .1 | 147 | 80 |
| 168 | Lisa | Ozer | LOZER | 011.44.1343.929268 | 11-MAR-97 | SA_REP | 11500 | .25 | 148 | 80 |
| 169 | Harrison | Bloom | HBLOOM | 011.44.1343.829268 | 23-MAR-98 | SA_REP | 10000 | .2 | 148 | 80 |

Tabela EMPLOYEES (continuação)

| EMPLOYEE_ID | FIRST_NAME | LAST_NAME | EMAIL | PHONE_NUMBER | HIRE_DATE | JOB_ID | SALARY | comm | mgrid | deptid |
|-------------|------------|------------|----------|--------------------|-----------|------------|--------|------|-------|--------|
| 170 | Taylor | Fox | TFOX | 011.44.1343.729268 | 24-JAN-98 | SA_REP | 9600 | .2 | 148 | 80 |
| 171 | William | Smith | WSMITH | 011.44.1343.629268 | 23-FEB-99 | SA_REP | 7400 | .15 | 148 | 80 |
| 172 | Elizabeth | Bates | EBATES | 011.44.1343.529268 | 24-MAR-99 | SA_REP | 7300 | .15 | 148 | 80 |
| 173 | Sundita | Kumar | SKUMAR | 011.44.1343.329268 | 21-APR-00 | SA_REP | 6100 | .1 | 148 | 80 |
| 174 | Elen | Abel | EABEL | 011.44.1644.429267 | 11-MAY-96 | SA_REP | 11000 | .3 | 149 | 80 |
| 175 | Alyssa | Hutton | AHUTTON | 011.44.1644.429266 | 19-MAR-97 | SA_REP | 8800 | .25 | 149 | 80 |
| 176 | Jonathon | Taylor | JTAYLOR | 011.44.1644.429265 | 24-MAR-98 | SA_REP | 8600 | .2 | 149 | 80 |
| 177 | Jack | Livingston | JLIVINGS | 011.44.1644.429264 | 23-APR-98 | SA_REP | 8400 | .2 | 149 | 80 |
| 178 | Kimberely | Grant | KGRANT | 011.44.1644.429263 | 24-MAY-99 | SA_REP | 7000 | .15 | 149 | |
| 179 | Charles | Johnson | CJOHNSON | 011.44.1644.429262 | 04-JAN-00 | SA_REP | 6200 | .1 | 149 | 80 |
| 180 | Winston | Taylor | WTAYLOR | 650.507.9876 | 24-JAN-98 | SH_CLERK | 3200 | | 120 | 50 |
| 181 | Jean | Fleaur | JFLEAUR | 650.507.9877 | 23-FEB-98 | SH_CLERK | 3100 | | 120 | 50 |
| 182 | Martha | Sullivan | MSULLIVA | 650.507.9878 | 21-JUN-99 | SH_CLERK | 2500 | | 120 | 50 |
| 183 | Girard | Geoni | GGEONI | 650.507.9879 | 03-FEB-00 | SH_CLERK | 2800 | | 120 | 50 |
| EMPLOYEE_ID | FIRST_NAME | LAST_NAME | EMAIL | PHONE_NUMBER | HIRE_DATE | JOB_ID | SALARY | comm | mgrid | deptid |
| 184 | Nandita | Sarchand | NSARCHAN | 650.509.1876 | 27-JAN-96 | SH_CLERK | 4200 | | 121 | 50 |
| 185 | Alexis | Bull | ABULL | 650.509.2876 | 20-FEB-97 | SH_CLERK | 4100 | | 121 | 50 |
| 186 | Julia | Dellinger | JDELLING | 650.509.3876 | 24-JUN-98 | SH_CLERK | 3400 | | 121 | 50 |
| 187 | Anthony | Cabrio | ACABRIO | 650.509.4876 | 07-FEB-99 | SH_CLERK | 3000 | | 121 | 50 |
| 188 | Kelly | Chung | KCHUNG | 650.505.1876 | 14-JUN-97 | SH_CLERK | 3800 | | 122 | 50 |
| 189 | Jennifer | Dilly | JDILLY | 650.505.2876 | 13-AUG-97 | SH_CLERK | 3600 | | 122 | 50 |
| 190 | Timothy | Gates | TGATES | 650.505.3876 | 11-JUL-98 | SH_CLERK | 2900 | | 122 | 50 |
| 191 | Randall | Perkins | RPERKINS | 650.505.4876 | 19-DEC-99 | SH_CLERK | 2500 | | 122 | 50 |
| 192 | Sarah | Bell | SBELL | 650.501.1876 | 04-FEB-96 | SH_CLERK | 4000 | | 123 | 50 |
| 193 | Britney | Everett | BEVERETT | 650.501.2876 | 03-MAR-97 | SH_CLERK | 3900 | | 123 | 50 |
| 194 | Samuel | McCain | SMCCAIN | 650.501.3876 | 01-JUL-98 | SH_CLERK | 3200 | | 123 | 50 |
| 195 | Vance | Jones | VJONES | 650.501.4876 | 17-MAR-99 | SH_CLERK | 2800 | | 123 | 50 |
| 196 | Alana | Walsh | AWALSH | 650.507.9811 | 24-APR-98 | SH_CLERK | 3100 | | 124 | 50 |
| 197 | Kevin | Feeney | KFEENEY | 650.507.9822 | 23-MAY-98 | SH_CLERK | 3000 | | 124 | 50 |
| EMPLOYEE_ID | FIRST_NAME | LAST_NAME | EMAIL | PHONE_NUMBER | HIRE_DATE | JOB_ID | SALARY | comm | mgrid | deptid |
| 198 | Donald | OConnell | DOCONNEL | 650.507.9833 | 21-JUN-99 | SH_CLERK | 2600 | | 124 | 50 |
| 199 | Douglas | Grant | DGRANT | 650.507.9844 | 13-JAN-00 | SH_CLERK | 2600 | | 124 | 50 |
| 200 | Jennifer | Whalen | JWHALEN | 515.123.4444 | 17-SEP-87 | AD_ASST | 4400 | | 101 | 10 |
| 201 | Michael | Hartstein | MHARTSTE | 515.123.5555 | 17-FEB-96 | MK_MAN | 13000 | | 100 | 20 |
| 202 | Pat | Fay | PFAY | 603.123.6666 | 17-AUG-97 | MK_REP | 6000 | | 201 | 20 |
| 203 | Susan | Mavris | SMAVRIS | 515.123.7777 | 07-JUN-94 | HR_REP | 6500 | | 101 | 40 |
| 204 | Hermann | Baer | HBAER | 515.123.8888 | 07-JUN-94 | PR_REP | 10000 | | 101 | 70 |
| 205 | Shelley | Higgins | SHIGGINS | 515.123.8080 | 07-JUN-94 | AC_MGR | 12000 | | 101 | 110 |
| 206 | William | Gietz | WGIEZT | 515.123.8181 | 07-JUN-94 | AC_ACCOUNT | 8300 | | 205 | 110 |

107 rows selected.

Tabela JOB_HISTORY

```
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) |

```
SELECT * FROM job_history;
```

| EMPLOYEE_ID | START_DAT | END_DATE | JOB_ID | deptid |
|-------------|-----------|-----------|------------|--------|
| 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.

Criando Scripts Avançados

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Objetivos

Após concluir este apêndice, você será capaz de:

- **Descrever o tipo de problema solucionado com instruções SQL que geram código SQL**
- **Criar um script que gera um script de instruções DROP TABLE**
- **Criar um script que gera um script de instruções INSERT INTO**

ORACLE

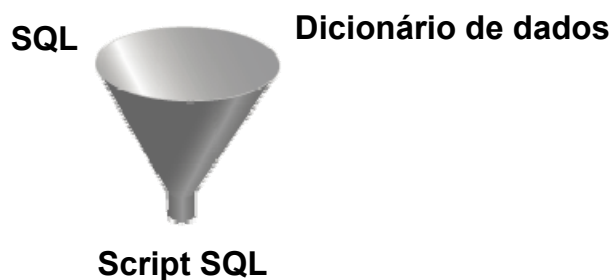
Copyright © 2004, Oracle. Todos os direitos reservados.

Objetivos

Neste apêndice, você aprenderá a criar e gerar um script SQL.

Usando SQL para Gerar SQL

- O SQL pode ser usado para gerar scripts no SQL
- O dicionário de dados:
 - é um conjunto de tabelas e views que contêm informações do banco de dados
 - é criado e mantido pelo servidor Oracle



ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Usando SQL para Gerar SQL

O SQL pode ser uma ferramenta avançada para gerar outras instruções SQL. Na maioria dos casos, isso envolve a criação de um arquivo de script. Você pode usar instruções SQL a partir de código SQL para:

- Evitar codificação repetitiva
- Acessar informações do dicionário de dados
- Eliminar ou recriar objetos do banco de dados
- Gerar predicados dinâmicos que contêm parâmetros de runtime

Os exemplos usados nesta lição envolvem a seleção de informações do dicionário de dados. O dicionário de dados é um conjunto de tabelas e views que contêm informações sobre o banco de dados. Esse conjunto é criado e mantido pelo servidor Oracle. Todas as tabelas do dicionário de dados pertencem ao usuário SYS. As informações armazenadas no dicionário de dados incluem os nomes dos usuários do servidor Oracle, os privilégios concedidos aos usuários, os nomes dos objetos do banco de dados, as constraints de tabelas e as informações de auditoria. Há quatro categorias de views de dicionário de dados. Cada categoria possui um prefixo distinto que reflete o uso pretendido.

| Prefixo | Description |
|---------|---|
| USER_ | Contém detalhes dos objetos pertencentes ao usuário |
| ALL_ | Contém detalhes dos objetos para os quais o usuário ganhou direitos de acesso e dos objetos pertencentes ao usuário |
| DBA_ | Contém detalhes dos usuários com privilégios de DBA para acessar qualquer objeto do banco de dados |
| V\$_ | Armazena informações sobre o desempenho e o bloqueio do servidor do banco de dados; disponível apenas para o DBA |

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 be accompanied with this e-mail. Contact your Oracle account manager for more information. If you have not received your personalized eKit, please contact your Oracle account manager.

Criando um Script Básico

```
SELECT 'CREATE TABLE ' || table_name ||  
       ' _test ' || 'AS SELECT * FROM '  
       || table_name || ' WHERE 1=2;'  
       AS "Create Table Script"  
FROM   user_tables;
```

| Create Table Script |
|---|
| CREATE TABLE COUNTRIES_test AS SELECT * FROM COUNTRIES WHERE 1=2; |
| CREATE TABLE DEPARTMENTS_test AS SELECT * FROM DEPARTMENTS WHERE 1=2; |
| CREATE TABLE EMPLOYEES_test AS SELECT * FROM EMPLOYEES WHERE 1=2; |
| CREATE TABLE JOBS_test AS SELECT * FROM JOBS WHERE 1=2; |
| CREATE TABLE JOB_GRADES_test AS SELECT * FROM JOB_GRADES WHERE 1=2; |
| CREATE TABLE JOB_HISTORY_test AS SELECT * FROM JOB_HISTORY WHERE 1=2; |
| CREATE TABLE LOCATIONS_test AS SELECT * FROM LOCATIONS WHERE 1=2; |
| CREATE TABLE REGIONS_test AS SELECT * FROM REGIONS WHERE 1=2; |

8 rows selected.

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Um Script Básico

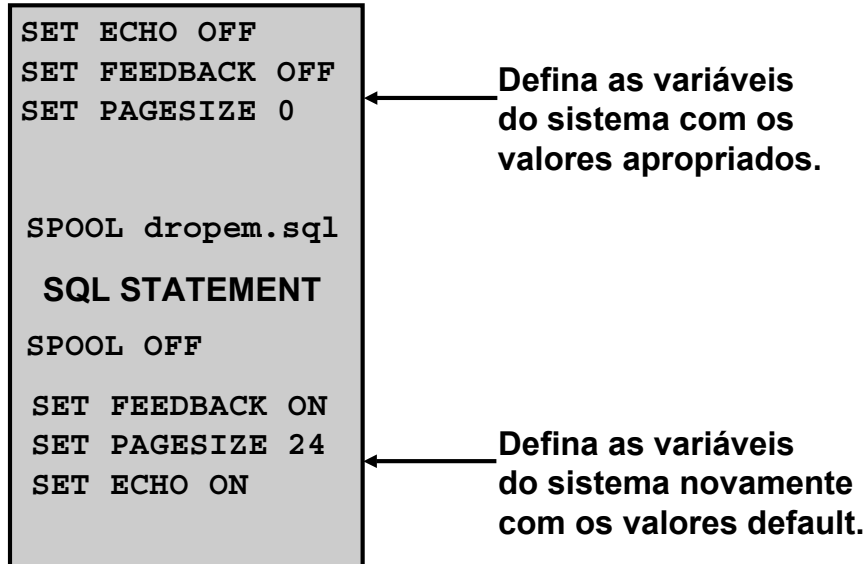
O exemplo do slide produz um relatório com instruções CREATE TABLE de todas as suas tabelas. Cada instrução CREATE TABLE produzida no relatório inclui a sintaxe para criar uma tabela usando o nome da tabela com um sufixo _test, apenas com a estrutura da respectiva tabela existente. O antigo nome da tabela é obtido na coluna TABLE_NAME da view de dicionário de dados USER_TABLES.

A próxima etapa é aprimorar o relatório para automatizar o processo.

Observação: Você pode consultar as tabelas do dicionário de dados para exibir os vários objetos do banco de dados que você possui. As views de dicionário de dados usadas frequentemente incluem:

- USER_TABLES Exibe descrição das tabelas do usuário
- USER_OBJECTS Exibe todos os objetos pertencentes ao usuário
- USER_TAB_PRIVS_MADE Exibe todas as concessões feitas aos objetos pertencentes ao usuário
- USER_COL_PRIVS_MADE Exibe todas as concessões feitas às colunas de objetos pertencentes ao usuário

Controlando o Ambiente



ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Controlando o Ambiente

A fim de executar as instruções SQL geradas, você deve capturá-las em um arquivo de spool que poderá, em seguida, ser executado. Também é preciso planejar a limpeza da saída gerada e certificar-se da supressão de elementos como cabeçalhos, mensagens de feedback, títulos superiores etc. Você pode executar tudo isso usando os comandos do *iSQL*Plus*.

O Panorama Completo

```
SET ECHO OFF
SET FEEDBACK OFF
SET PAGESIZE 0

SELECT 'DROP TABLE ' || object_name || ';'
FROM   user_objects
WHERE  object_type = 'TABLE'
/

SET FEEDBACK ON
SET PAGESIZE 24
SET ECHO ON
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

O Panorama Completo

A saída do comando do slide é salva em um arquivo denominado `dropem.sql`, usando a opção File Output do *iSQL*Plus*. Esse arquivo contém os dados a seguir. Agora, para iniciar o arquivo no *iSQL*Plus*, localize, carregue e execute o arquivo de script.

| 'DROPTABLE' OBJECT_NAME ';' |
|-------------------------------|
| DROP TABLE COUNTRIES; |
| DROP TABLE DEPARTMENTS; |
| DROP TABLE EMPLOYEES; |
| DROP TABLE JOBS; |
| DROP TABLE JOB_HISTORY; |
| DROP TABLE LOCATIONS; |
| DROP TABLE REGIONS; |

Observação: Por default, é feito o spool dos arquivos na pasta `ORACLE_HOME\ORANT\BIN` no Windows NT.

Fazendo Dump do Conteúdo de uma Tabela para um Arquivo

```
SET HEADING OFF ECHO OFF FEEDBACK OFF
SET PAGESIZE 0

SELECT
  'INSERT INTO departments_test VALUES
  (' || department_id || ', ' || department_name ||
  ', ' || location_id || ');'
  AS "Insert Statements Script"
FROM   departments
/

SET PAGESIZE 24
SET HEADING ON ECHO ON FEEDBACK ON
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Fazendo Dump do Conteúdo da Tabela para um Arquivo

Às vezes, é útil ter os valores das linhas de uma tabela em um arquivo de texto no formato de uma instrução `INSERT INTO VALUES`. Esse script pode ser executado para preencher a tabela, caso ela tenha sido acidentalmente eliminada.

O exemplo do slide produz instruções `INSERT` para a tabela `DEPARTMENTS_TEST`, capturada no arquivo `data.sql` usando a opção `File Output` no `iSQL*Plus`.

O conteúdo do arquivo de script `data.sql` é o seguinte:

```
INSERT INTO departments_test VALUES
(10, 'Administration', 1700);
INSERT INTO departments_test VALUES
(20, 'Marketing', 1800);
INSERT INTO departments_test VALUES
(50, 'Shipping', 1500);
INSERT INTO departments_test VALUES
(60, 'IT', 1400);
...
```

Fazendo Dump do Conteúdo de uma Tabela para um Arquivo

| Source | Result |
|--------------------------------------|-------------------------------|
| <code>'''X'''</code> | <code>'X'</code> |
| <code>'''</code> | <code>'</code> |
| <code>''' department_name '''</code> | <code>'Administration'</code> |
| <code>','</code> | <code>','</code> |
| <code>')) ;</code> | <code>')) ;</code> |

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Fazendo Dump do Conteúdo da Tabela para um Arquivo (continuação)

Você deve ter reparado na grande quantidade de aspas simples no slide da página anterior. Um conjunto de quatro aspas simples produz uma aspa simples na instrução final. Lembre-se também de que valores de caracteres e de datas devem estar entre aspas.

Para exibir uma aspa simples em uma string, é preciso prefixá-la com outra aspa simples. Sendo assim, no quinto exemplo do slide, as aspas circundam a string inteira. A segunda aspa funciona como um prefixo para exibir a terceira aspa. Com isso, o resultado é uma aspa simples, seguida de parênteses e ponto-e-vírgula.

Gerando um Predicado Dinâmico

```
COLUMN my_col NEW_VALUE dyn_where_clause

SELECT DECODE('&deptno', null,
DECODE ('&&hiredate', null, ' ',
'WHERE hire_date=TO_DATE('' || '&&hiredate'', 'DD-MON-YYYY''))',
DECODE ('&&hiredate', null,
'WHERE department_id = ' || '&deptno',
'WHERE department_id = ' || '&deptno' ||
' AND hire_date = TO_DATE('' || '&&hiredate'', 'DD-MON-YYYY''))
AS my_col FROM dual;
```

```
SELECT last_name FROM employees &dyn_where_clause;
```

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Gerando um Predicado Dinâmico

O exemplo do slide gera uma instrução SELECT que recupera dados de todos os funcionários de um departamento que foram contratados em determinado dia. O script gera a cláusula WHERE dinamicamente.

Observação: Depois que a variável de usuário estiver no lugar, use o comando UNDEFINE para deletá-la.

A primeira instrução SELECT solicita a informação do número de departamento. Se você não informar nenhum número, a function DECODE tratará o número de departamento como nulo, e o usuário deverá informar a data de admissão. Caso você não informe nenhuma data, a function DECODE tratará a data de admissão como nula, e a cláusula WHERE dinâmica gerada também será nula, o que fará com que a segunda instrução SELECT recupere todas as linhas da tabela EMPLOYEES.

Observação: A variável NEW_V[ALUE] especifica uma variável para conter um valor de coluna. Você pode fazer referência à variável nos comandos TTITLE. Use NEW_VALUE para exibir os valores de coluna ou a data no título superior. Você deve incluir a coluna em um comando BREAK com a ação SKIP PAGE. O nome da variável não pode conter um símbolo de cerquilha (#). A variável NEW_VALUE é útil para relatórios detalhados/mestres nos quais há um novo registro mestre para cada página.

Gerando um Predicado Dinâmico (continuação)

Observação: Aqui, a data de admissão deve ser especificada no formato DD-MON-YYYY.

A instrução SELECT do slide anterior pode ser interpretada da seguinte maneira:

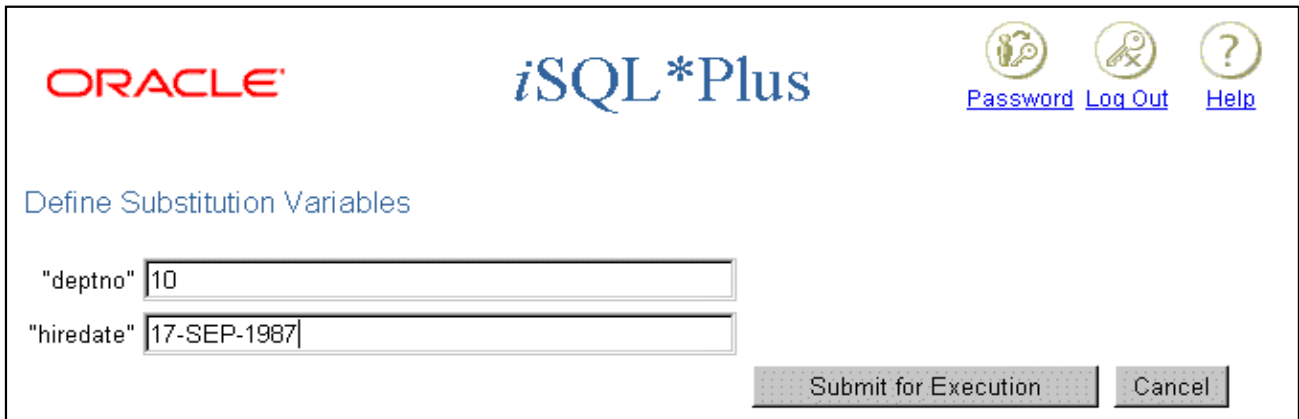
```

IF (<<deptno>> is not entered) THEN
  IF (<<hiredate>> is not entered) THEN
    return empty string
  ELSE
    return the string 'WHERE hire_date =
TO_DATE('<<hiredate>>', 'DD-MON-YYYY')'
ELSE
  IF (<<hiredate>> is not entered) THEN
    return the string 'WHERE department_id =
<<deptno>> entered'
  ELSE
    return the string 'WHERE department_id =
<<deptno>> entered
                                AND hire_date =
TO_DATE(' <<hiredate>>', 'DD-MON-YYYY')'
END IF

```

A string retornada passa a ser o valor da variável DYN_WHERE_CLAUSE, que será usada na segunda instrução SELECT.

Quando o primeiro exemplo do slide for executado, será solicitado que o usuário informe os valores de DEPTNO e HIREDATE:



The screenshot shows the Oracle iSQL*Plus interface. At the top, there are links for Password, Log Out, and Help. Below, the 'Define Substitution Variables' dialog is open. It has two input fields: 'deptno' with the value '10' and 'hiredate' with the value '17-SEP-1987'. At the bottom right of the dialog are two buttons: 'Submit for Execution' and 'Cancel'.

Será gerado o seguinte valor de MY_COL:

| MY_COL |
|--|
| WHERE department_id = 10AND hire_date = TO_DATE('17-SEP-1987','DD-MON-YYYY') |

Quando o segundo exemplo do slide for executado, será gerada a seguinte saída:

| LAST_NAME |
|-----------|
| Whalen |

Sumário

Neste apêndice, você deverá ter aprendido que:

- **É possível criar um script SQL para gerar outro script SQL.**
- **Os arquivos de script geralmente usam o dicionário de dados.**
- **É possível capturar a saída em um arquivo.**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Sumário

O SQL pode ser usado para gerar scripts SQL. Esses scripts podem ser usados para evitar codificação repetitiva, eliminar ou recriar objetos, obter ajuda do dicionário de dados e gerar predicados dinâmicos que contenham parâmetros de runtime.

Os comandos do *iSQL*Plus* podem ser usados para capturar os relatórios gerados pelas instruções SQL e limpar a saída gerada, como a supressão de cabeçalhos, mensagens de feedback etc.

Componentes da Arquitetura Oracle



ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Objetivos

Após concluir este apêndice, você será capaz de:

- **Descrever a arquitetura do servidor Oracle e seus principais componentes**
- **Listar as estruturas envolvidas na conexão de um usuário com uma instância Oracle**
- **Listar os estágios do processamento:**
 - **Consultas**
 - **Instruções DML**
 - **Commits**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Objetivos

Este apêndice apresenta a arquitetura do servidor Oracle, pois descreve as estruturas dos arquivos, processos e memória envolvidas no estabelecimento de uma conexão com o banco de dados e na execução de um comando SQL.

Arquitetura do Banco de Dados Oracle: Visão Geral

O banco de dados Oracle consiste em dois componentes principais:

- **O banco de dados ou as estruturas físicas**
- **A instância ou as estruturas da memória**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

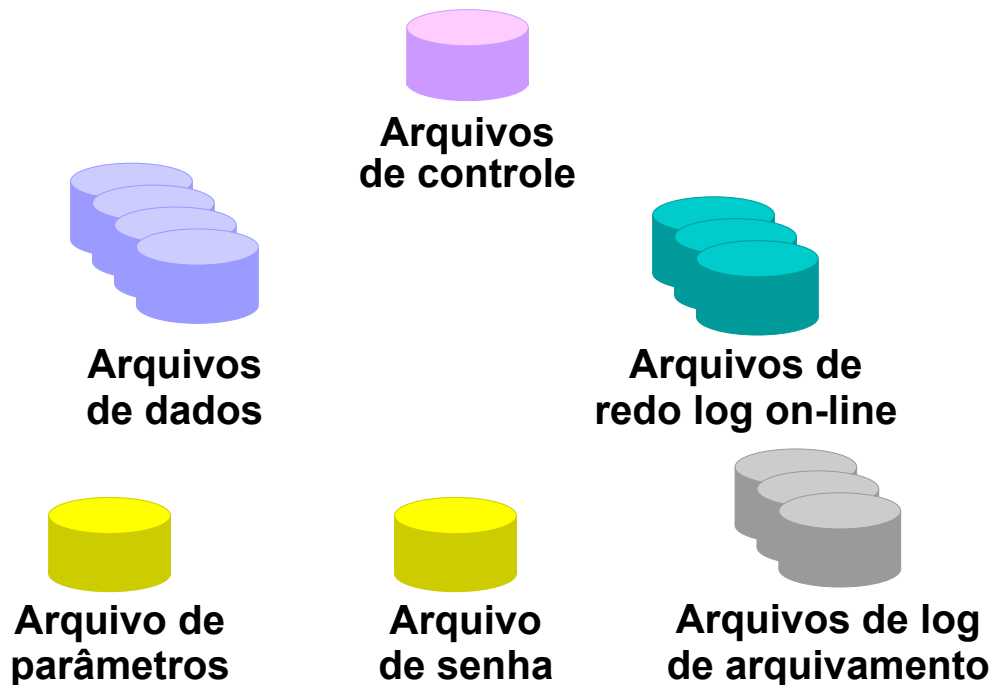
Oracle University and Impacta Tecnologia use only.

Arquitetura do Banco de Dados Oracle: Visão Geral

O banco de dados Oracle consiste em dois componentes principais: a instância e o banco de dados propriamente dito.

- O banco de dados consiste em arquivos físicos como:
 - O arquivo de controle no qual está armazenada a configuração do banco de dados
 - Os arquivos de redo log que contêm informações necessárias para a recuperação do banco de dados
 - Os arquivos de dados onde estão armazenados todos os dados
 - O arquivo de parâmetros que contém os parâmetros que controlam o tamanho e as propriedades de uma instância
 - O arquivo de senha que contém a senha de superusuário ou SYSDBA
- A instância consiste na SGA (System Global Area) e nos processos do servidor que executam tarefas no banco de dados.

Arquitetura Física do Banco de Dados



ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Arquitetura Física do Banco de Dados

Os arquivos que compõem um banco de dados Oracle encontram-se organizados da seguinte maneira:

- **Arquivos de controle:** Esses arquivos contêm dados sobre o banco de dados propriamente dito, denominados metadados. Eles são fundamentais para o banco de dados. Sem eles, não é possível abrir arquivos para acessar os dados do banco de dados.
- **Arquivos de dados:** Esses arquivos contêm os dados do banco de dados.
- **Arquivos de redo log on-line:** Esses arquivos permitem a recuperação da instância do banco de dados. Se o banco de dados falhar, mas não perder nenhum arquivo de dados, a instância poderá recuperar o banco de dados com as informações desses arquivos.

Há outros arquivos que oficialmente não fazem parte do banco de dados, mas que são importantes para seu bom funcionamento. São eles:

- **Arquivo de parâmetros:** O arquivo de parâmetros é usado para definir como a instância será configurada ao ser inicializada.
- **Arquivo de senha:** Este arquivo permite que os usuários se conectem remotamente ao banco de dados e executem tarefas administrativas.
- **Arquivos de log de arquivamento:** Estes arquivos contêm um histórico contínuo do redo gerado pela instância. Eles possibilitam a recuperação do banco de dados. Usando esses arquivos e um backup do banco de dados, é possível recuperar um arquivo de dados perdido.

Arquivos de Controle

- **Contêm informações da estrutura física do banco de dados**
- **Multiplexados para evitar perdas**
- **Leitura no estágio de montagem**



Arquivos de controle

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Arquivos de Controle

Quando você inicia a instância e monta o banco de dados, o arquivo de controle é lido. As entradas do arquivo de controle especificam os arquivos físicos que constituem o banco de dados.

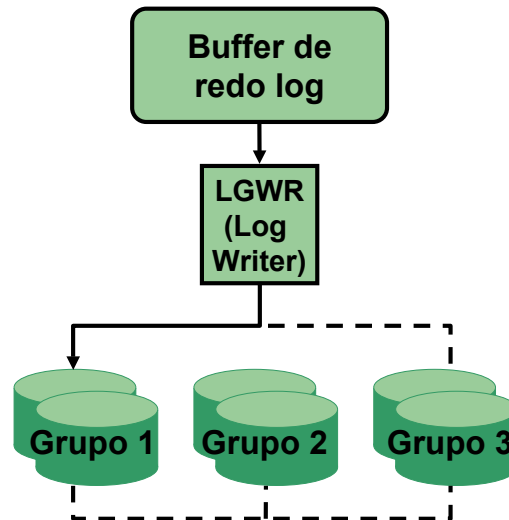
Quando você adiciona outros arquivos ao banco de dados, o arquivo de controle é automaticamente atualizado.

A localização dos arquivos de controle é especificada em um parâmetro de inicialização.

Para proteger-se de falha do banco de dados devido à perda do arquivo de controle, você deve multiplexar o arquivo de controle em, pelo menos, três dispositivos físicos diferentes. Ao especificar vários arquivos usando o parâmetro de inicialização, você permite que o servidor do banco de dados Oracle mantenha várias cópias do arquivo de controle.

Arquivos de Redo Log

- Alterações nos registros do banco de dados
- Multiplexados para evitar perdas



ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

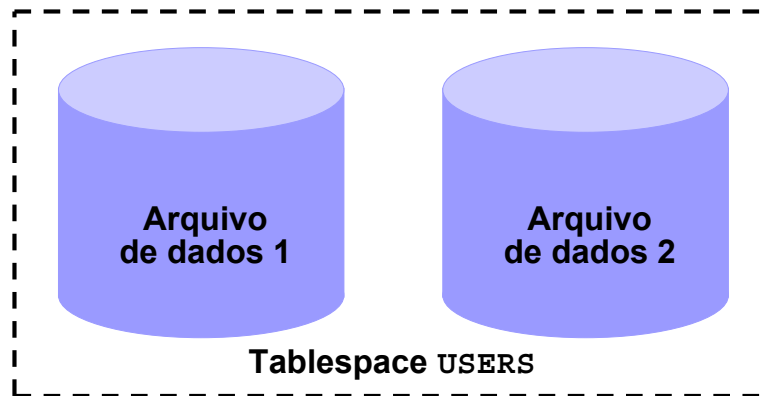
Arquivos de Redo Log

Os arquivos de redo log são usados para registrar alterações feitas no banco de dados como resultado de transações e de ações internas do servidor do banco de dados Oracle. Eles protegem o banco de dados da perda de integridade decorrente de falhas no sistema causadas por interrupções de energia, falhas de disco etc. Os arquivos de redo log devem ser multiplexados para garantir que as informações neles armazenadas não sejam perdidas em caso de falha de disco.

O redo log consiste em grupos de arquivos de redo log. Um grupo consiste em um arquivo de redo log e suas respectivas cópias multiplexadas. Cada cópia idêntica é considerada como um membro desse grupo, e cada grupo é identificado por um número. O processo LGWR (Log Writer) grava registros de redo do buffer de redo log para um grupo de redo logs até que o arquivo esteja preenchido, ou que seja solicitada uma operação de alternância de log. Em seguida, ele alterna e grava nos arquivos do próximo grupo. Os grupos de redo logs são usados de forma circular.

Tablespaces e Arquivos de Dados

- Os tablespaces consistem em um ou mais arquivos de dados.
- Os arquivos de dados pertencem a apenas um tablespace.



ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

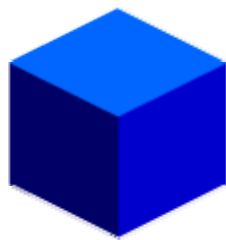
Tablespaces e Arquivos de Dados

Um banco de dados se divide em unidades de armazenamento lógicas denominadas tablespaces, que podem ser usadas para agrupar estruturas lógicas relacionadas. Cada banco de dados está logicamente dividido em um ou mais tablespaces. São explicitamente criados um ou mais arquivos de dados para cada tablespace, a fim de armazenar fisicamente os dados de todas as estruturas lógicas em um tablespace.

Observação: Também é possível criar tablespaces para arquivos grandes, que são os tablespaces com um arquivo de dados único, mas muito grande (até 4 blocos G). Os tablespaces para arquivos pequenos tradicionais (que são o default) podem conter vários arquivos de dados, mas os arquivos não podem ser tão grandes. Para obter mais informações sobre tablespaces para arquivos grandes, consulte o *Database Administrator's Guide*.

Segmentos, Extensões e Blocos

- Os segmentos ficam dentro de um tablespace.
- Os segmentos consistem em um conjunto de extensões.
- As extensões são um conjunto de blocos de dados.
- Os blocos de dados são mapeados para os blocos do sistema operacional.



Segmento



Extensões



Blocos de dados



Blocos do sistema operacional

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Segmentos, Extensões e Blocos

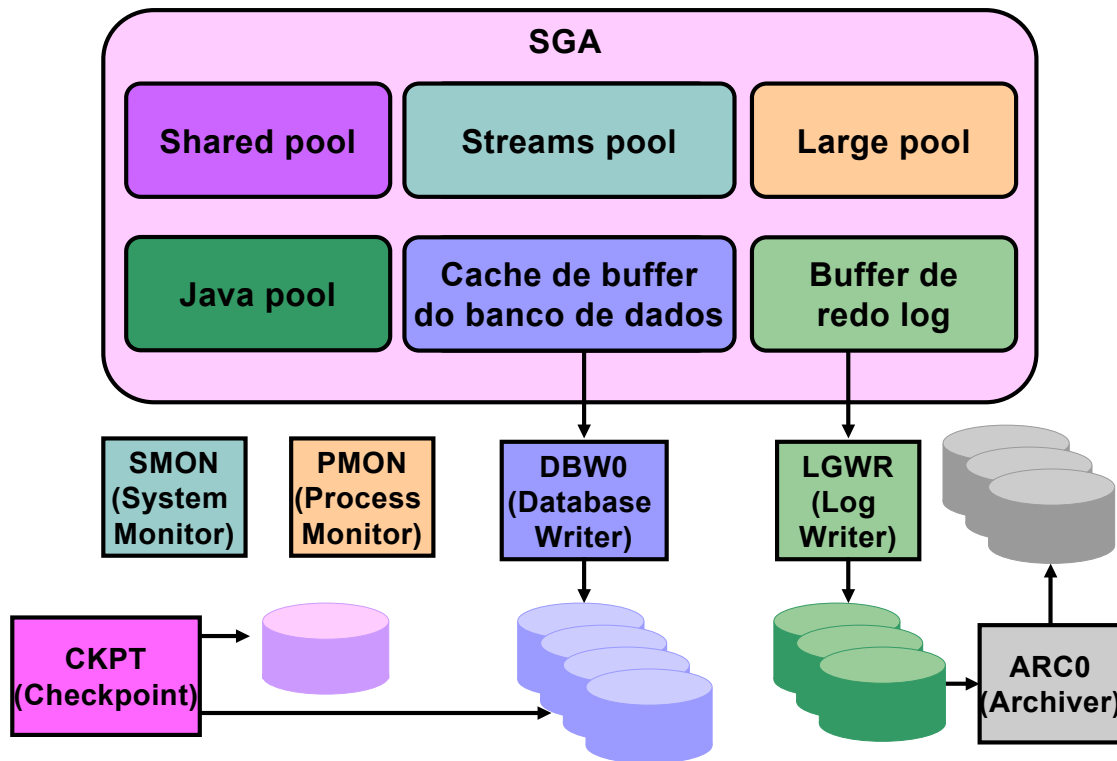
Objetos do banco de dados, como tabelas e índices, são armazenados em tablespaces como segmentos. Cada segmento contém uma ou mais extensões. Uma extensão consiste em blocos de dados contíguos, o que significa que cada extensão pode existir em apenas um arquivo de dados. Os blocos de dados são a menor unidade de entrada/saída do banco de dados.

Quando o banco de dados solicita ao sistema operacional um conjunto de blocos de dados, o sistema operacional mapeia essa solicitação para o bloco do sistema operacional no dispositivo de armazenamento. Sendo assim, não é necessário se preocupar com o endereço físico dos dados do banco de dados. Isso significa que um arquivo de dados pode ser distribuído e/ou espelhado em vários discos.

O tamanho do bloco de dados pode ser definido durante a criação do banco de dados. O tamanho default de 8 K é adequado para a maioria dos bancos de dados. Caso o banco de dados suporte uma aplicação de data warehouse com índices e tabelas grandes, um bloco maior poderá ser vantajoso. Se o banco de dados suportar uma aplicação transacional onde leituras e gravações são muito raras, poderá ser vantajoso bloco menor. O tamanho máximo do bloco depende do sistema operacional. O tamanho mínimo de bloco é 2 K e deve ser usado, se for usado, raramente.

É possível ter tablespaces com tamanhos de bloco diferentes. Geralmente, deve ser usado apenas para suportar tablespaces transportáveis. Para obter detalhes, consulte o *Database Administrator's Guide*.

Gerenciamento de Instâncias Oracle



Copyright © 2004, Oracle. Todos os direitos reservados.

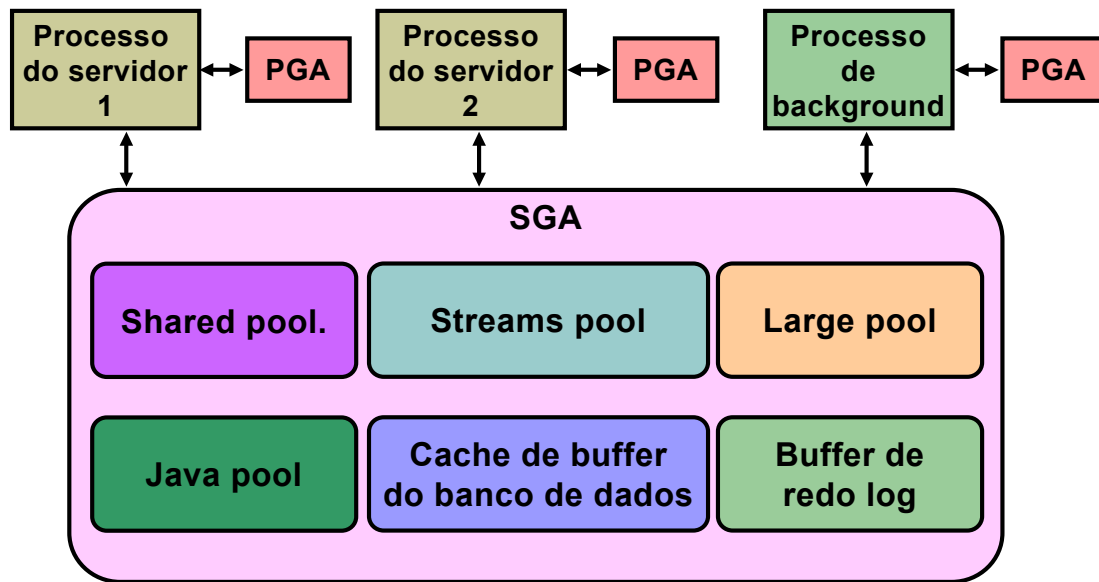
Gerenciamento de Instâncias Oracle

Um servidor do banco de dados Oracle consiste em um banco de dados Oracle e uma instância Oracle. Uma instância Oracle consiste em buffers de memória conhecidos como SGA (System Global Area) e em processos de background.

A instância está inativa (inexistente) até ser inicializada. Quando a instância é inicializada, é lido um arquivo de parâmetros de inicialização, e a instância é configurada de forma adequada.

Após a inicialização da instância e a abertura do banco de dados, os usuários podem acessar o banco de dados.

Estruturas de Memória Oracle



ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Estruturas de Memória Oracle

As estruturas de memória básicas associadas a uma instância Oracle incluem:

- **SGA (System Global Area):** Compartilhada por todos os processos do servidor e de background
- **PGA (Program Global Area):** Exclusivo a cada servidor e processo de background. Há um PGA para cada processo

A SGA (System Global Area) é uma área de memória compartilhada que contém dados e informações de controle para a instância.

A SGA consiste nas seguintes estruturas de dados:

- **Cache de buffer do banco de dados:** Armazena no cache blocos de dados recuperados do banco de dados
- **Buffer de redo log:** Armazena no cache informações sobre redo (usadas na recuperação de instância) até que elas possam ser gravadas nos arquivos físicos de redo log armazenados em disco
- **Shared pool:** Armazena no cache vários constructs que podem ser compartilhados pelos usuários
- **Large pool:** Área opcional usada para armazenar grandes solicitações de entrada/saída
- **Java pool:** Usado para todos os códigos e dados Java específicos à sessão dentro da JVM (Java virtual machine)
- **Streams pool:** Usado pelo Oracle Streams

Quando você iniciar a instância usando o Enterprise Manager ou o SQL*Plus, será exibida a memória alocada para a SGA.

Estruturas da Memória Oracle (continuação)

Com a infra-estrutura dinâmica da SGA, o tamanho do cache de buffer do banco de dados, do shared pool, do large pool, do Java pool e do Streams pool pode ser alterado sem necessidade de fazer o shutdown da instância.

O banco de dados pré-configurado foi ajustado previamente com as definições adequadas para os parâmetros de memória. No entanto, como a utilização do banco de dados é expansível, talvez seja necessário alterar as definições dos parâmetros de memória.

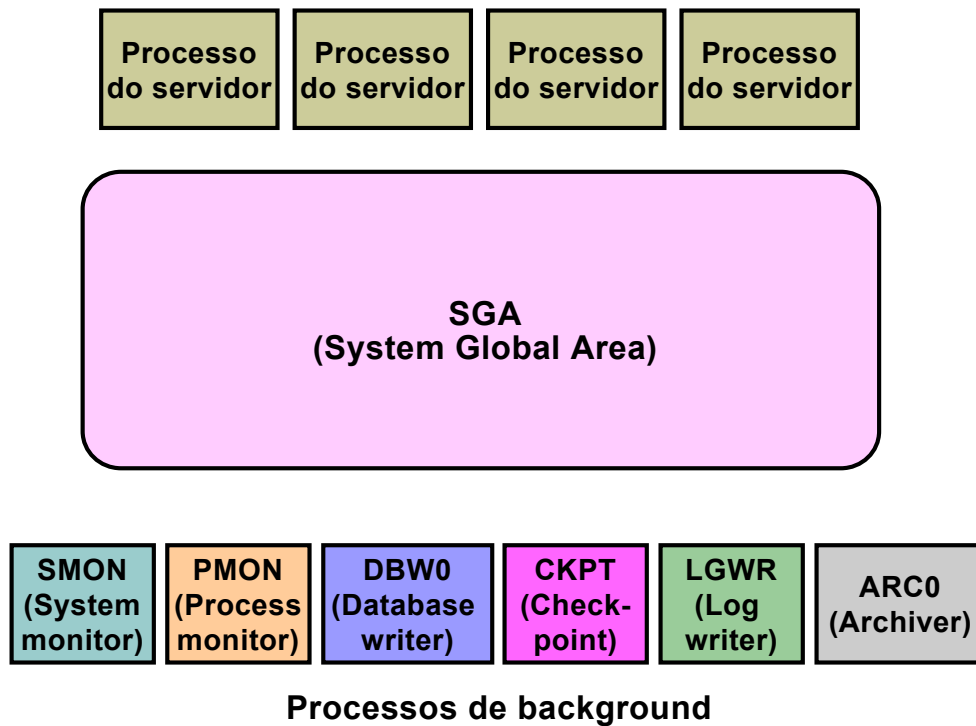
O Oracle fornece alertas e advisors para identificar problemas de dimensionamento de memória, com o objetivo de ajudá-lo a determinar os valores apropriados para os parâmetros de memória.

PGA (Program Global Area) é uma região da memória que contém dados e informações de controle para cada processo do servidor. Um processo do servidor atende às solicitações de um cliente. Cada processo do servidor tem a sua própria área PGA privada, criada quando o processo do servidor é iniciado. O acesso a ela é exclusivo a esse processo do servidor, e apenas códigos do Oracle que estiverem executando operações envolvendo a SGA poderão ler e gravar dados nessa área.

O volume de memória PGA usado e o seu conteúdo dependem de a instância ter sido configurada no modo de servidor compartilhado. Geralmente, a PGA contém o seguinte:

- **Área SQL privada:** Contém dados, como as informações de bind e as estruturas de memória de runtime. Cada sessão que executa uma instrução SQL tem uma área SQL privada.
- **Memória de sessão:** Memória alocada para reter as variáveis de uma sessão e outras informações relativas a essa sessão

Processos Oracle



ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Processos Oracle

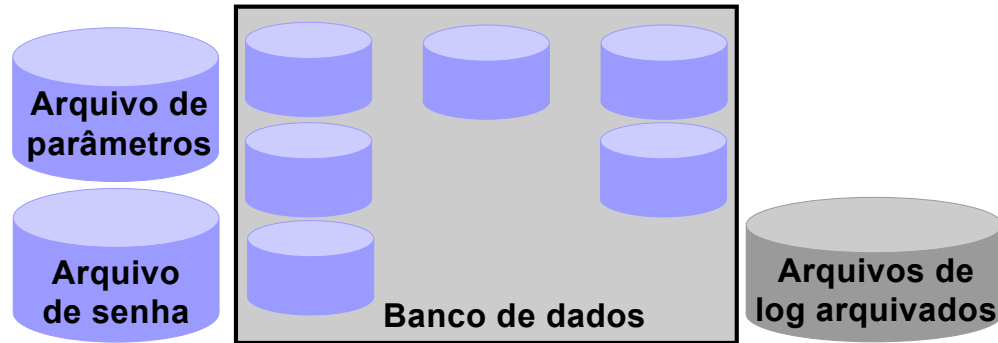
Quando você chama um programa de aplicação ou uma ferramenta Oracle como o Enterprise Manager, o servidor Oracle cria um processo do servidor para executar comandos da aplicação.

O Oracle também cria um conjunto de processos de background para uma instância, os quais interagem entre si e com o sistema operacional para gerenciar as estruturas de memória e executar entradas/saídas assincronamente para gravar dados em disco, além de fazer manutenção em geral.

Os processos de background que estarão presentes dependem dos recursos usados no banco de dados. Estes são os processos de background mais comuns:

- **SMON (System monitor):** Executa a recuperação de falha quando a instância é iniciada após uma falha
- **PMON (Process monitor):** Executa a limpeza do processo quando um processo do usuário falha
- **DBW_n (Database writer):** Grava blocos modificados do cache de buffer do banco de dados para os arquivos em disco
- **CKPT (Checkpoint):** Sinaliza para o DBW_n nos checkpoints e atualiza todos os arquivos de dados e de controle do banco de dados para indicar o checkpoint mais recente
- **LGWR (Log Writer):** Grava as entradas do redo log em disco
- **ARC_n (Archiver):** Copia os arquivos de redo log para armazenamento de arquivamento quando os arquivos de log estão cheios ou quando ocorre uma alternância de log

Outras Estruturas Físicas Importantes



ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Outros Arquivos Importantes

O servidor Oracle também usa outros arquivos que não fazem parte do banco de dados:

- O arquivo de parâmetros define as características de uma instância Oracle. Por exemplo, ele contém os parâmetros que dimensionam algumas das estruturas da memória na SGA.
- O arquivo de senha autentica os usuários com privilégios para inicializar e fazer shutdown de uma instância Oracle.
- Os arquivos de redo log arquivados são cópias off-line dos arquivos de redo log que podem ser necessários para a recuperação após falhas de mídia.

Processando uma Instrução SQL

- **Conecte-se a uma instância usando um:**
 - Processo do usuário
 - Processo do servidor
- **Os componentes do servidor Oracle usados dependem do tipo de instrução SQL:**
 - As consultas retornam linhas
 - As instruções DML registram alterações
 - O commit garante a recuperação da transação
- **Alguns componentes do servidor Oracle não participam do processamento da instrução SQL.**

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Componentes Usados para Processar SQL

Nem todos os componentes de uma instância Oracle são usados para processar instruções SQL. Os processos do usuário e do servidor são usados para conectar um usuário a uma instância Oracle. Esses processos não fazem parte da instância Oracle, mas são necessários para processar uma instrução SQL.

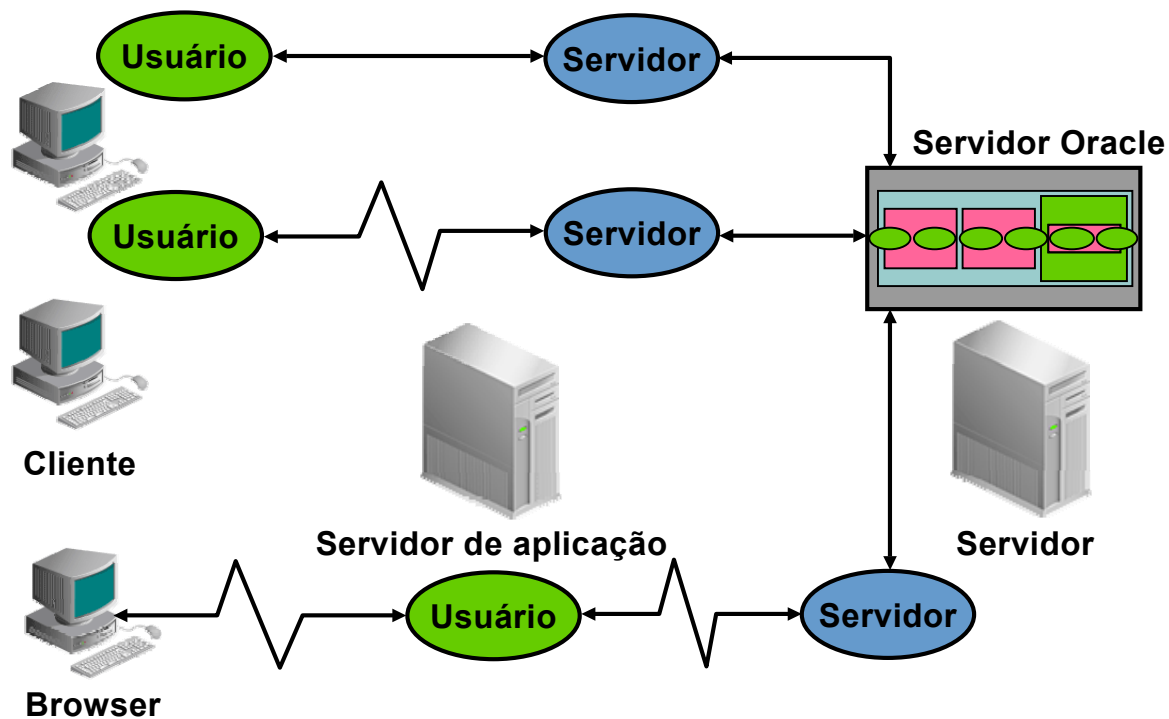
Alguns processos de background, estruturas da SGA e arquivos do banco de dados são usados para processar instruções SQL. Dependendo do tipo de instrução SQL, são usados componentes diferentes:

- As consultas exigem processamento adicional para retornar linhas ao usuário.
- As instruções DML (data manipulation language) exigem processamento adicional para registrar as alterações feitas nos dados.
- O processamento de commit garante que os dados modificados em uma transação poderão ser recuperados.

Alguns processos de background necessários não participam diretamente do processamento de uma instrução SQL, mas são usados para melhorar o desempenho e recuperar o banco de dados.

O processo de background opcional, ARC0, é usado para garantir que um banco de dados de produção possa ser recuperado.

Estabelecendo Conexão com uma Instância



ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Processos Usados para Conexão com uma Instância

Antes de submeterem instruções SQL a um servidor Oracle, os usuários devem se conectar a uma instância.

O usuário inicia uma ferramenta, como o *iSQL*Plus*, ou executa uma aplicação desenvolvida por meio de uma ferramenta, como o Oracle Forms. Essa ferramenta ou essa aplicação é executada em um *processo do usuário*.

Na maioria das configurações básicas, quando um usuário efetua login no servidor Oracle, é criado um processo no computador que executa esse servidor. Esse processo é chamado de processo do servidor. O processo do servidor se comunica com a instância Oracle em nome do processo do usuário executado no cliente. O processo do servidor executa instruções SQL em nome do usuário.

Conexão

Uma conexão é um caminho de comunicação entre um processo do usuário e um servidor Oracle. Um usuário do banco de dados pode se conectar a um servidor Oracle de três maneiras:

- O usuário efetua login no sistema operacional que está executando a instância Oracle e inicia uma aplicação ou uma ferramenta que acesse o banco de dados nesse sistema. O caminho de comunicação é estabelecido por meio dos mecanismos de comunicação entre os processos disponíveis no sistema operacional host.

Processos Usados para Conexão com uma Instância (continuação)

Conexão (continuação)

- O usuário inicia a aplicação ou a ferramenta em um computador local e se conecta por uma rede ao computador que está executando a instância Oracle. Nessa configuração, denominada cliente/servidor, o software de rede é usado para a comunicação entre o usuário e o servidor Oracle.
- Em uma conexão de três camadas, o computador do usuário se comunica pela rede com uma aplicação ou um servidor de rede, que, por sua vez, conecta-se por uma rede à máquina que está executando a instância Oracle. Por exemplo, o usuário executa um browser em um computador da rede para usar uma aplicação residente em um servidor NT que recupera os dados de um banco de dados Oracle em execução em um host UNIX.

Sessões

Uma sessão é uma conexão específica de um usuário com um servidor Oracle. A sessão começa quando o usuário é validado pelo servidor Oracle, e termina quando ele efetua logout ou quando ocorre um encerramento anormal. Pode haver várias sessões concorrentes para um usuário se ele efetuar logon a partir de várias ferramentas, aplicações ou terminais simultaneamente. Com exceção de algumas ferramentas especializadas de administração de banco de dados, o início de uma sessão de banco de dados exige que o servidor Oracle esteja disponível para uso.

Observação: O tipo de conexão explicada aqui, na qual existe uma correspondência exata entre um processo do usuário e um processo do servidor, denomina-se conexão de servidor dedicado.

Processando uma Consulta

- **Parse:**
 - Pesquisa uma instrução idêntica
 - Verifica a sintaxe, os nomes dos objetos e os privilégios
 - Bloqueia os objetos usados durante o parse
 - Cria e armazena planos de execução
- **Execute:** Identifica as linhas selecionadas
- **Fetch:** Retorna linhas para o processo do usuário

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

Etapas do Processamento de Consultas

As consultas diferem de outros tipos de instruções SQL porque, se forem bem-sucedidas, retornarão dados. Enquanto outras instruções apenas retornam sucesso ou falha, uma consulta pode retornar uma linha ou milhares de linhas.

Há três estágios principais no processamento de uma consulta:

- Parse
- Execute
- Fetch

Durante o estágio de *parse*, a instrução SQL passa do processo do usuário para o processo do servidor, e uma representação da instrução SQL analisada por parse é carregada em uma área SQL compartilhada.

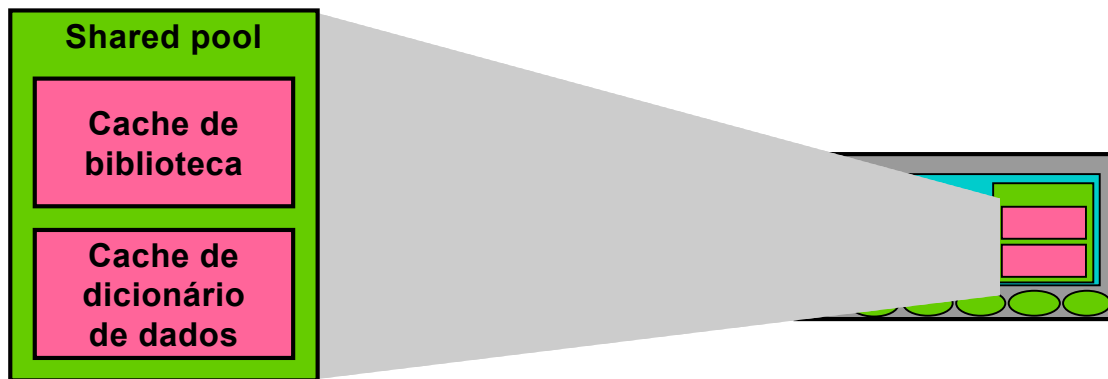
Durante o parse, o processo do servidor executa as seguintes funções:

- Procura uma cópia da instrução SQL no shared pool
- Valida a instrução SQL verificando a sintaxe
- Faz consultas ao dicionário de dados para validar as definições das tabelas e das colunas

O comando `execute fetch` executa a instrução usando a melhor abordagem do otimizador, e o comando `fetch` recupera as linhas para o usuário.

O Shared Pool

- O cache de biblioteca contém o texto da instrução SQL, o código analisado por parse e o plano de execução.
- O cache do dicionário de dados contém definições e privilégios de tabelas, colunas e outros objetos.
- O tamanho do shared pool é definido por `SHARED_POOL_SIZE`.



ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Componentes do Shared Pool

Durante o estágio de parse, o processo do servidor usa a área da SGA conhecida como shared pool para compilar a instrução SQL. O shared pool possui dois componentes principais:

- Cache de biblioteca
- Cache de dicionário de dados

Cache de Biblioteca

O cache de biblioteca armazena informações sobre as instruções SQL usadas mais recentemente em uma estrutura de memória denominada área SQL compartilhada. A área SQL compartilhada contém:

- O texto da instrução SQL
- A árvore de parse: Uma versão compilada da instrução
- O plano de execução: As etapas a serem seguidas ao executar a instrução

Otimizador é a function do servidor Oracle que determina o plano de execução ideal.

Se uma instrução SQL for executada novamente, e uma área SQL compartilhada já contiver o plano de execução da instrução, o processo do servidor não precisará efetuar o parse da instrução. O cache de biblioteca melhora o desempenho das aplicações que reutilizam instruções SQL, reduzindo o tempo de análise e os requisitos de memória. Se não for reutilizada, a instrução SQL será eliminada do cache de biblioteca.

Componentes do Shared Pool (continuação)

Cache de Dicionário de Dados

O cache de dicionário de dados, também conhecido como cache de dicionário ou cache de linhas, é um conjunto das definições mais usadas recentemente no banco de dados. Inclui informações sobre arquivos do banco de dados, tabelas, índices, colunas, usuários, privilégios e outros objetos do banco de dados.

Durante a fase de parse, o processo do servidor procura as informações no cache de dicionário para resolver (converter) os nomes de objetos especificados na instrução SQL e validar os privilégios de acesso. Se necessário, o processo do servidor iniciará a carga dessas informações a partir dos arquivos de dados.

Dimensionando o Shared Pool

O tamanho do shared pool é especificado pelo parâmetro de inicialização `SHARED_POOL_SIZE`.

Cache de Buffer do Banco de Dados.

- Armazena os blocos mais usados recentemente
- Tamanho de um buffer baseado em `DB_BLOCK_SIZE`
- Número de buffers definido por `DB_BLOCK_BUFFERS`



ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Function do Cache de Buffer do Banco de Dados

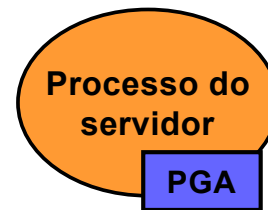
Quando uma consulta é processada, o processo do servidor consulta os blocos necessários no cache de buffer do banco de dados. Se o bloco não é encontrado nesse cache, o processo do servidor lê o bloco no arquivo de dados e coloca uma cópia no cache de buffer. Como as solicitações subseqüentes do mesmo bloco podem encontrá-lo na memória, talvez elas não precisem de leituras físicas. O servidor Oracle usa um algoritmo LRU para defasar os buffers não acessados recentemente, de modo a criar espaço para novos blocos no cache de buffer.

Dimensionando o Cache de Buffer do Banco de Dados

O tamanho de cada buffer do cache de buffer é igual ao tamanho de um bloco Oracle e é especificado pelo parâmetro `DB_BLOCK_SIZE`. O número de buffers é igual ao valor do parâmetro `DB_BLOCK_BUFFERS`.

PGA (Program Global Area)

- **Não compartilhada**
- **Gravável apenas pelo processo do servidor**
- **Contém:**
 - **Área de classificação**
 - **Informações da sessão**
 - **Estado do cursor**
 - **Espaço da pilha**



ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Componentes da PGA

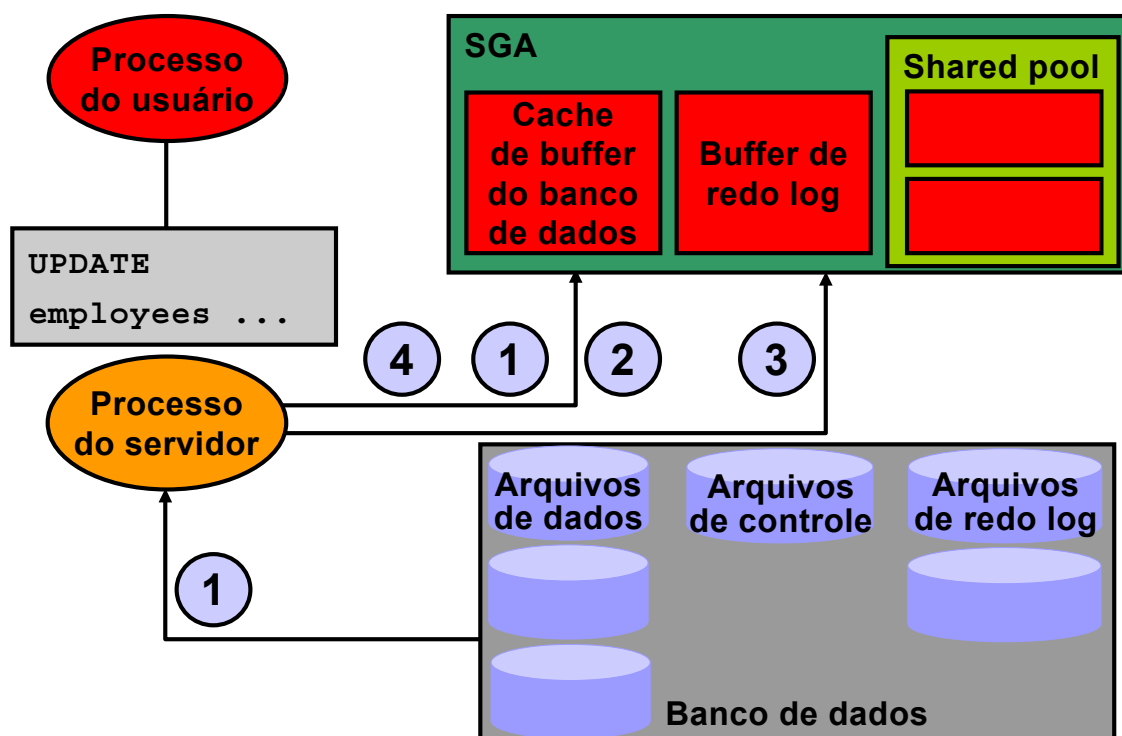
Uma PGA (Program Global Area) é uma região da memória que contém informações sobre controle e dados para um processo do servidor. Trata-se de uma memória não-compartilhada criada pelo Oracle quando um processo do servidor é iniciado. O acesso a essa memória é exclusividade desse processo do servidor, além de ser lido e gravado apenas pelo código do servidor Oracle que está atuando em seu nome. A memória PGA alocada por cada processo do servidor associado a uma instância Oracle é chamada de memória PGA agregada alocada pela instância.

Em uma configuração de servidor dedicado, a PGA do servidor inclui os seguintes componentes:

- **Área de classificação:** Usada para várias classificações que podem ser necessárias para processar a instrução SQL
- **Informações de sessão:** Incluem os privilégios de usuário e as estatísticas de desempenho para a sessão
- **Estado do cursor:** Indica o estágio de processamento das instruções SQL que a sessão está usando no momento
- **Espaço da pilha:** Contém outras variáveis da sessão

A PGA é alocada quando um processo é criado e desalocada quando o processo é encerrado.

Processando uma Instrução DML



ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Etapas de Processamento de Instruções DML

Uma instrução DML (data manipulation language) necessita de apenas duas fases de processamento:

- A fase de parse é igual à usada para processar uma consulta.
- O comando de execução requer processamento adicional para alterar os dados.

Fase de Execução das Instruções DML

Para executar uma instrução DML:

- Se os blocos de rollback e dados ainda não estiverem no cache de buffer, o processo do servidor lerá esses blocos nos arquivos de dados contidos no cache de buffer.
- O processo do servidor impõe bloqueios nas linhas a serem modificadas.
- No buffer de redo log, o processo do servidor registra as alterações a serem feitas no rollback e nos dados.
- As alterações do bloco de rollback registram os valores dos dados antes da modificação. O bloco de rollback é usado para armazenar a imagem original dos dados, de modo que seja possível efetuar rollback das instruções DML, se necessário.
- As alterações dos blocos de dados registram os novos valores dos dados.

Etapas de Processamento de Instruções DML (continuação)

Fase de Execução das Instruções DML (continuação)

O processo do servidor registra a imagem original no bloco de rollback e atualiza o bloco de dados. Essas duas alterações são feitas no cache de buffer do banco de dados. Os blocos alterados no cache de buffer são marcados como buffers sujos, ou seja, buffers diferentes dos blocos correspondentes no disco.

O processamento de um comando DELETE ou INSERT usa etapas semelhantes. A imagem original de um comando DELETE contém os valores das colunas da linha deletada, e a imagem original de um comando INSERT contém as informações de localização da linha.

Como as alterações feitas nos blocos são registradas apenas nas estruturas de memória e não são gravadas imediatamente no disco, uma falha no computador que cause a perda da SGA também poderá ocasionar a perda dessas alterações.

Buffer de Redo Log

- Tem seu tamanho definido por LOG_BUFFER
- Alterações nos registros feitas por meio da instância
- É usado seqüencialmente
- É um buffer circular



ORACLE

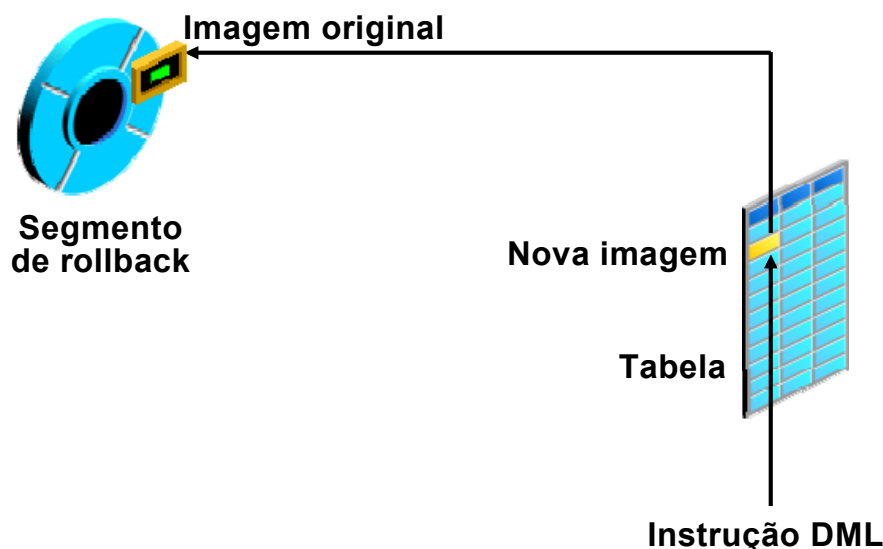
Copyright © 2004, Oracle. Todos os direitos reservados.

Características do Buffer de Redo Log

O processo do servidor registra a maioria das alterações feitas nos blocos de arquivos de dados no buffer de redo log, que é uma parte da SGA. O buffer de redo log tem as seguintes características:

- O tamanho em bytes é definido pelo parâmetro LOG_BUFFER.
- Ele registra o bloco que foi alterado, o local da alteração e o novo valor de uma entrada de redo. Para uma entrada de redo, não faz diferença o tipo de bloco alterado. Ela apenas registra os bytes que foram alterados no bloco.
- O buffer de redo log é usado seqüencialmente, e as alterações feitas por uma transação podem ser intercaladas com as alterações feitas por outras transações.
- Trata-se de um buffer circular que é reutilizado após o preenchimento, mas somente depois de todas as entradas antigas de redo terem sido registradas nos arquivos de redo log.

Segmento de Rollback



ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Segmento de Rollback

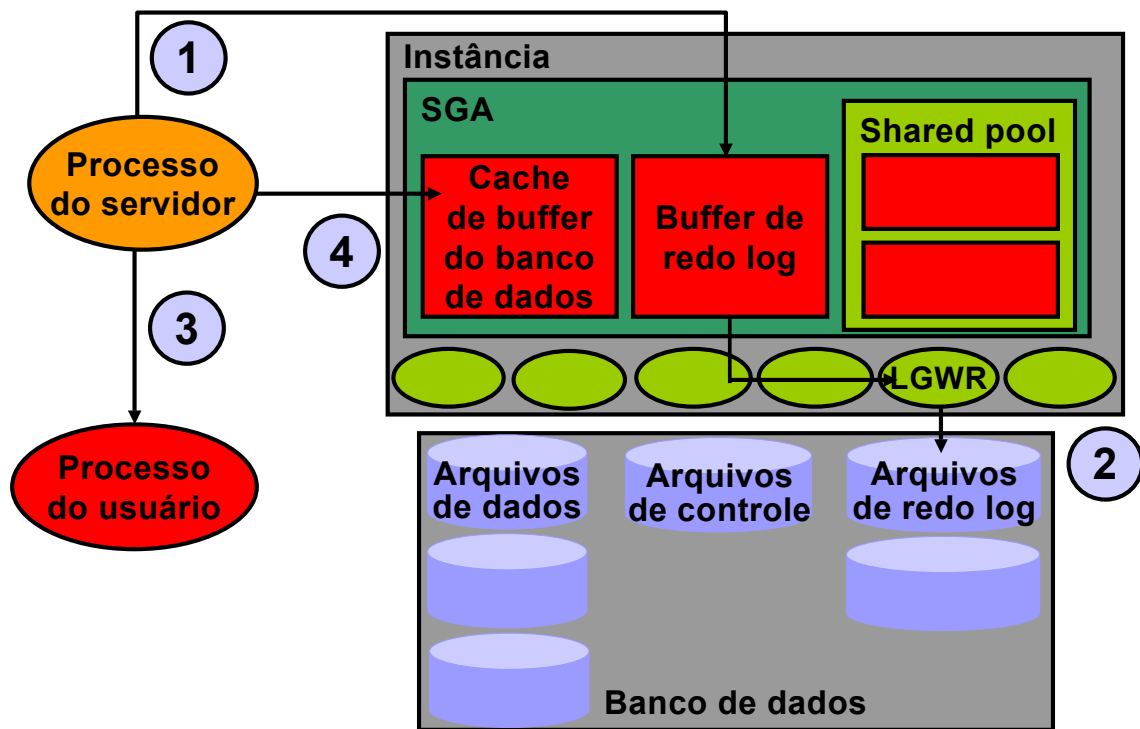
Antes de fazer uma alteração, o processo do servidor salva o valor de dados antigo em um segmento de rollback. Essa imagem original é usada para:

- Desfazer as alterações, se a transação for submetida a um rollback
- Manter a consistência de leitura, garantindo que outras transações não vejam as alterações não submetidas a commit feitas pela instrução DML
- Recuperar o banco de dados até um estado consistente em caso de falhas

Os segmentos de rollback, como tabelas e índices, existem em arquivos de dados, e os blocos de rollback são trazidos para o cache de buffer do banco de dados, conforme necessário. Os segmentos de rollback são criados pelo DBA.

As alterações feitas nos segmentos de rollback são registradas no buffer de redo log.

Processamento de COMMIT



ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

COMMIT rápido

O servidor Oracle usa um mecanismo de commit rápido que garante que as alterações submetidas a commit poderão ser recuperadas em caso de falha na instância.

SCN (System Change Number)

Sempre que uma transação é submetida a commit, o servidor Oracle atribui um SCN (system change number) à transação. O SCN aumenta de maneira uniforme e é exclusivo no banco de dados. Ele é usado pelo servidor Oracle como um timestamp interno para sincronizar dados e para garantir consistência de leitura quando os dados forem recuperados dos arquivos de dados. O uso do SCN permite que o servidor Oracle execute verificações de consistência independentemente da data e a hora do sistema operacional.

Etapas no Processamento de COMMITs

Quando um COMMIT é executado, ocorrem as seguintes etapas:

- O processo do servidor coloca um registro de commit no buffer de redo log, juntamente com o SCN.
- O processo LGWR executa uma gravação contígua de todas as entradas do buffer de redo log até o registro de commit dos arquivos de redo log. Depois deste ponto, o servidor Oracle pode garantir que as alterações não serão perdidas, mesmo que ocorra uma falha na instância.

COMMIT rápido (continuação)

Etapas no Processamento de COMMITs (continuação)

- O usuário é informado de que o COMMIT está concluído.
- O processo do servidor registra informações para indicar que a transação foi concluída e que os bloqueios de recursos podem ser liberados.

A descarga dos buffers sujos no arquivo de dados é executada de forma independente por DBW0, e pode ocorrer antes ou depois do commit.

Vantagens do COMMIT Rápido

O mecanismo de commit rápido garante a recuperação dos dados ao gravar as alterações no buffer de redo log, em vez de fazê-lo nos arquivos de dados. Ele apresenta as seguintes vantagens:

- As gravações sequenciais nos arquivos de log são mais rápidas do que a gravação em blocos diferentes no arquivo de dados.
- São gravadas nos arquivos de log apenas as informações mínimas necessárias para registrar as alterações, enquanto nos arquivos de dados, seria necessária a gravação de blocos de dados inteiros.
- Se diversas transações solicitarem commit ao mesmo tempo, a instância incluirá os registros de redo log por meio de piggyback em uma única gravação.
- A menos que o buffer de redo log esteja extremamente cheio, será necessária apenas uma gravação síncrona por transação. Se ocorrer o piggyback, poderá haver menos de uma gravação síncrona por transação.
- Como o buffer de redo log pode ser descarregado antes do COMMIT, o tamanho da transação não afeta o tempo necessário para uma operação COMMIT real.

Observação: O rollback de uma transação não faz com que o LGWR seja acionado para gravar no disco. O servidor Oracle sempre faz rollback nas alterações não submetidas a commit, nas recuperações de falhas. Se ocorrer uma falha após um rollback, antes do registro das entradas de rollback no disco, a ausência de um registro de commit será suficiente para garantir que as alterações feitas pela transação sofrerão rollback.

Sumário

Neste apêndice, você aprendeu a:

- **Identificar arquivos do banco de dados:** arquivos de dados, arquivos de controle e redo logs on-line
- **Descrever as estruturas da memória SGA:** cache de buffer do banco de dados, pool SQL compartilhado e buffer de redo log
- **Explicar os principais processos de background:** DBW0, LGWR, CKPT, PMON, SMON e ARC0
- **Listar as etapas de processamento de instruções SQL:** parse, execute, fetch

ORACLE

Copyright © 2004, Oracle. Todos os direitos reservados.

Oracle University and Impacta Tecnologia use only.

Sumário

Arquivos do Banco de Dados Oracle

O banco de dados Oracle inclui os seguintes arquivos:

- **Arquivos de controle:** Contêm informações necessárias para verificar a integridade do banco de dados, inclusive os nomes dos outros arquivos do banco de dados. (Os arquivos de controle geralmente são espelhados.)
- **Arquivos de dados:** Contêm os dados do banco de dados, inclusive tabelas, índices, segmentos de rollback e segmentos temporários
- **Redo logs on-line:** Contêm as alterações feitas nos arquivos de dados. (Os redo logs on-line são usados para recuperação e geralmente são espelhados.)

Outros arquivos geralmente usados com o banco de dados incluem:

- **Arquivo de parâmetros:** Define as características de uma instância Oracle
- **Arquivo de senha:** Autentica os usuários privilegiados do banco de dados
- **Redo logs arquivados:** São backups dos redo logs on-line

Sumário (continuação)

Estruturas da Memória SGA

A SGA (System Global Area) tem três estruturas principais:

- **Shared pool:** Armazena as instruções SQL mais executadas recentemente e os dados do dicionário de dados mais usados recentemente
- **Cache de buffer do banco de dados:** Armazena os dados mais usados recentemente
- **Buffer de redo log:** Registra as alterações feitas no banco de dados usando a instância

Processos de Background

Uma instância Oracle de produção inclui os seguintes processos:

- **DBW0 (Database writer):** Grava os dados alterados nos arquivos de dados
- **LGWR (Log Writer):** Registra as alterações dos arquivos de dados nos arquivos de redo log on-line
- **SMON (System monitor):** Verifica a consistência e inicia a recuperação do banco de dados quando ele é aberto
- **PMON (Process monitor):** Limpa os recursos em caso de falha de um dos processos
- **CKPT (Checkpoint process):** Atualiza as informações de status do banco de dados após um checkpoint
- **ARC0 (Archiver):** Faz backup do redo log on-line para garantir a recuperação após uma falha de mídia. (Este processo é opcional, mas geralmente é incluído em uma instância de produção.)

Dependendo da configuração, a instância também pode incluir outros processos.

Etapas de Processamento de Instruções SQL

As etapas usadas para processar uma instrução SQL incluem:

- **Parse:** Compila a instrução SQL
- **Execute:** Identifica linhas selecionadas ou aplica alterações de DML aos dados
- **Fetch:** Retorna as linhas consultadas por uma instrução SELECT

Índice

A

Adiando Constraints 02-13
 Adicionando uma coluna 02-05
 Agrupamento Concatenado 04-21,04-22,04-23
 ALL INSERT 03-02,03-16,03-21,03-35
 ALTER 01-12
 ALTER SESSION 05-04,..., 05-09,05-13,05-17
 ALTER TABLE 02-03,...,02-19, 06-16, 08-13
 ALTER USER 01-11, 01-19
 Arquitetura do Banco de Dados D-1...D-30
 Ativando Constraints 02-15, 02-16

C

Classificando Linhas 07-10
 Cláusula WITH 06-22, 06-23, 06-24
 Coluna Composta 04-17,04-19,04-23
 Comparações Emparelhadas 06-04, 06-25
 Comparações Não Emparelhadas 06-04
 CONNECT BY 06-07,07-05...07-14
 CONNECT BY PRIOR 07-05 ... 07-14
 Constraint 01-17,02-07, 02-10...02-20, 06-07, 08-1, C-03
 Constraints de Verificação 08-13
 Constraints em Cascata 02-18, 02-19
 Consulta de Versões 03-32
 Consulta Externa 03-07, 06-08, 06-10, 06-12, 06-13,06-14
 Consulta Interna 06-08, 06-10, 06-10, 06-12, 06-14, 06-15
 Consultas Hierárquicas 03-05
 Controlando Alterações 02-31
 Correlação 06-13
 Criar Banco de Dados 05-09, C-03
 Criar Índice 02-20,02-21,02-23,02-38,02-39
 CUBE 04-06,...,04-24
 CURRENT_DATE 05-05, 05-06, 05-06, 05-15, 05-34,05-35
 CURRENT_TIMESTAMP 05-05, 05-07, 05-08, 05-17, 05-34, 05-35

D

Data/Horário 05-05...05-35
 DBTIMEZONE 05-04, 05-09, 05-26, 05-34
 DEFAULT DIRECTORY 02-33,02-35
 Desativando Constraints 02-15
 DROP TABLE 02-26,02-28, C-02,C-06

E

Eliminando uma coluna 02-07

EXIST 06-14,...,06-16,06-20, 06-21,06-24,06-26,06-27

Expurgar 02-26,02-28

EXTRACT 05-25

F

FIRST INSERT 03-16,03-22,03-23

Flashback de Consulta de Versão 02-32

FLASHBACK TABLE 02-26, 02-27, 02-28

FROM_TZ 05-28

Funções de conversão 05-28...05-31

Funções de Data/Horário 05-34,05-35

Fuso Horário 05-01... 05-34

G

Greenwich Mean Time 05-03,05-06,05-14

GROUP BY 03-07, 03-23, 04-03 ,..., 04-23, 06-07, 06-20, 06-24, 06-25

GROUP BY ROLLUP 04-12, 04-17, 04-19

GROUPING 04-03,...,04-23

GROUPING SET 04-13, 04-15, 04-17, 04-21, 04-23

H

HAVING 03-05, 04-03, 04-05, 04-07, 04-09, 06-20, C-04

Horário de Verão 05-03,05-28,05-32,

I

Índices baseados em Função 02-23, 02-24

INSERT 01-12, 01-15, 01-18, 02-06, 02-31,03-03,..., 03-06, 03-11,..., 03-30

INSERT ALL Incondicional 02-19

INSERT Condicional 02-20 ...02-23

Instrução UPDATE Correlacionada 06-17,06-18

INTERVAL 05-18-05-24

INTERVAL DAY TO SECOND 05-22, -05-24

INTERVAL YEAR TO MONTH 05-21...05-22

L

LEVEL 07-05, 07-10, 07-11, 07-14,

Limite de Rejeição 02-33,02-35

Linhas de Tabelas de Referência Cruzada 04-06

Linhas Superagregadas 04-06,04-08,04-09

LOCALTIMESTAMP 05-05, 05-08, 05-08, 05-34, 05-35

LPAD 07-11

M

MERGE 03-27...03-30
Metacaracteres 08-04...08-06
Modificando uma coluna 02-06

N

NLS_Date_Language 05-29
Nó Filho 07-10
Nó-Raiz 07-10
NOT EXIST 01-16, 06-14,06-16
NOT IN 03-11,06-05, 06-06, 06-16

O

ON DELETE CASCADE 02-12
ORACLE_LOADER 02-35
ORDER BY 02-23, 04-03, 04-04, 04-05, 04-07, 04-09, 06-08,
ORGANIZATION EXTERNAL 02-33,02-35

P

Parâmetro de Acesso 02-33,02-35
Percorrendo a Árvore 07-07...07-09
PIVOTING 03-16, 03-24, 03-25
Pseudocoluna 07-05, 07-10, 07-14,07-15

R

Reduzir a Árvore 07-13, 07-14
REGEXP_INSTR 8-7
REGEXP_LIKE 8-7
REGEXP_REPLACE 8-7
REGEXP_SUBSTR 8-7
Relacionamento Pai/Filho 07-04
Relatório Estruturado em Árvore 07-02
Relatórios de Tabelas de Referência Cruzada 04-09
ROLLUP 04-06 ... 04-23

S

SESSIONTIMEZONE 05-06 ,...,05-09, 05-26
SET TIME_ZONE 05-04, 05-06 ,..., 05-09,05-17
SET UNUSED 02-08
START WITH 07-05, 07-06, 07-11,07-14,08-10
Subconsulta 03-04... 03-11, 03-14,03-17,06-03... 06_26, 07-06, 07-07
Subconsulta Correlacionada 06-10 ...06-13
Subconsulta Correlacionada para Atualizar Linha 06-17,06-18
Subconsulta Correlacionada para Deletar 06-20,06-21
Subconsulta Escalar 06-07...06-09
Subconsultas de Várias Colunas 06-03, 06-07
Substituindo Padrões 08-12
Suporte a Expressões Comuns 8-1 ,..., 8-13

T

Tabela Externa 02-29,02-31,02-35,02-37

TIMESTAMP 05-10...05-17

TIMESTAMP WITH LOCAL TIMEZONE 05-16, 05-17

TIMESTAMP WITH TIME ZONE 05-15

TO_DSINTERVAL 05-30

TO_YMINTERVAL 05-31

TZ_OFFSET 05-26, 05-34

U

UNION 04-07,04-10, 04-13, 04-17, 04-23

UNION ALL 04-07, 04-10,04-13, 04-13,04-17,04-23

V

V\$TIMEZONE_NAME 05-28

Valores de Tabelas de Referência Cruzada 04-09,04-23

VERSIONS BETWEEN 03-34

Versões de Linhas 03-36

