

Oracle Database: Conceptos Fundamentales de SQL II

Volumen I • Guía del Alumno

D64260CS10

Edición 1.0

Enero de 2010

D73592

ORACLE®

Autores

Chaitanya Koratamaddi
 Brian Pottle
 Tulika Srivastava

Colaboradores y Revisores Técnicos

Claire Bennett
 Ken Cooper
 Yanti Chang
 Laszlo Czinkoczki
 Burt Demchick
 Gerlinde Frenzen
 Joel Goodman
 Laura Garza
 Richard Green
 Nancy Greenberg
 Akira Kinutani
 Wendy Lo
 Isabelle Marchand
 Timothy Mcglue
 Alan Paulson
 Manish Pawar
 Srinivas Putrevu
 Bryan Roberts
 Clinton Shaffer
 Hilda Simson
 Abhishek Singh
 Jenny Tsai Smith
 James Spiller
 Lori Tritz
 Lex van der Werff
 Marcie Young

Redactores

Amitha Narayan
 Daniel Milne
 Raj Kumar

Diseñador Gráfico

Satish Bettgowda

Editores

Veena Narasimhan
 Pavithran Adka

Copyright © 2010, Oracle. Todos los derechos reservados.

Renuncia

En este curso se ofrece una visión general de las funciones y mejoras planificadas en la versión 11g. Únicamente pretende ayudarle a evaluar las ventajas de negocio de actualizar a 11g y planificar los proyectos de TI.

Este curso, en cualquiera de sus formatos (incluidos los ejercicios prácticos del curso y el material impreso), contiene información de propiedad en exclusiva de Oracle. Este curso y la información que contiene no se podrán revelar, copiar, reproducir o distribuir a ninguna persona ajena a Oracle sin el consentimiento previo por escrito de Oracle. Este curso y su contenido no forman parte del acuerdo de licencia ni se pueden incorporar a ningún acuerdo contractual con Oracle, sus subsidiarias o filiales.

Este curso es sólo para fines informativos y únicamente pretende ayudarle a planificar la implantación y la actualización de las funciones del producto descritas. No existe ningún compromiso de entregar ningún material, código o funcionalidad, y no se confiará en él al tomar decisiones de compra. El desarrollo, la publicación y la distribución de las funciones o funcionalidades descritas en este documento seguirán siendo a entera discreción de Oracle.

Este documento contiene información propiedad de Oracle Corporation y se encuentra protegido por las leyes de copyright, así como por otras leyes de propiedad intelectual. El usuario podrá realizar copias o imprimir este documento para su uso exclusivo en los cursos de formación de Oracle. Este documento no podrá ser modificado ni alterado en modo alguno. Salvo que la legislación de copyright lo considere un uso legítimo, no podrá utilizar, compartir, descargar, cargar, copiar, imprimir, mostrar, representar, reproducir, publicar, conceder licencias, enviar, transmitir ni distribuir este documento total ni parcialmente sin autorización expresa por parte de Oracle.

La información contenida en este documento está sujeta a cambio sin previo aviso. Si detecta cualquier problema en el documento, le agradeceremos que nos lo comunique por escrito a: Oracle University, 500 Oracle Parkway, Redwood Shores, California 94065 EE. UU. Oracle Corporation no garantiza que este documento esté exento de errores.

Aviso sobre Restricción de Derechos

Si esta documentación se entrega al Gobierno de los EE. UU. o a cualquier entidad que la utilice en nombre del Gobierno de los EE. UU., se aplicará la siguiente disposición:

U.S. GOVERNMENT RIGHTS

The U.S. Government's rights to use, modify, reproduce, release, perform, display, or disclose these training materials are restricted by the terms of the applicable Oracle license agreement and/or the applicable U.S. Government contract.

Aviso de Marca Registrada

Oracle es una marca comercial registrada de Oracle Corporation y/o sus filiales. Todos los demás nombres pueden ser marcas comerciales de sus respectivos propietarios.

Contenido

I Introducción

Objetivos I-2

Agenda I-3

Objetivos del Curso I-4

Requisitos del Curso I-5

Agenda I-6

Tablas Utilizadas en este Curso I-8

Apéndices Utilizados en este Curso I-9

Entornos de Desarrollo I-10

Agenda I-11

Revisión de la Restricción de Datos I-12

Revisión de Ordenación de Datos I-13

Revisión de Funciones SQL I-14

Revisión de Funciones de Una Sola Fila I-15

Revisión de Tipos de Funciones de Grupo I-16

Revisión de Subconsultas I-17

Revisión de Manipulación de Datos I-18

Agenda I-19

Documentación sobre SQL de Oracle Database I-20

Recursos Adicionales I-21

Resumen I-22

Práctica I: Visión General I-23

1 Control del Acceso de los Usuarios

Objetivos 1-2

Agenda 1-3

Control del Acceso de los Usuarios 1-4

Privilegios 1-5

Privilegios del Sistema 1-6

Creación de un Usuario 1-7

Privilegios del Sistema de Usuario 1-8

Asignación de Privilegios del Sistema 1-9

Agenda 1-10

¿Qué es un Rol? 1-11

Creación y Asignación de Privilegios a un Rol 1-12

Cambio de Contraseña 1-13

Agenda 1-14
 Privilegios de Objeto 1-15
 Asignación de Privilegios de Objeto 1-17
 Transferencia de Privilegios 1-18
 Confirmación de Privilegios Otorgados 1-19
 Agenda 1-20
 Revocación de Privilegios de Objeto 1-21
 Prueba 1-23
 Resumen 1-24
 Práctica 1: Visión General 1-25

2 Gestión de Objetos de Esquema

Objetivos 2-2
 Agenda 2-3
 Sentencia `ALTER TABLE` 2-4
 Adición de Columnas 2-6
 Modificación de Columnas 2-7
 Borrado de Columnas 2-8
 Opción `SET UNUSED` 2-9
 Agenda 2-11
 Adición de Sintaxis de Restricción 2-12
 Adición de Restricciones 2-13
 Cláusula `ON DELETE` 2-14
 Restricciones Diferidas 2-15
 Diferencia entre `INITIALLY DEFERRED` y `INITIALLY IMMEDIATE` 2-16
 Borrado de Restricciones 2-18
 Desactivación de Restricciones 2-19
 Activación de Restricciones 2-20
 Restricciones en Cascada 2-22
 Cambio de Nombre de Columnas de Tabla y Restricciones 2-24
 Agenda 2-25
 Visión General de Índices 2-26
`CREATE INDEX` con la Sentencia `CREATE TABLE` 2-27
 Índices Basados en Funciones 2-29
 Eliminación de Índices 2-30
`DROP TABLE ... PURGE` 2-31
 Agenda 2-32
 Sentencia `FLASHBACK TABLE` 2-33
 Uso de la Sentencia `FLASHBACK TABLE` 2-35
 Agenda 2-36

Tablas Temporales	2-37
Creación de una Tabla Temporal	2-38
Agenda	2-39
Tablas Externas	2-40
Creación de un Directorio para la Tabla Externa	2-41
Creación de Tablas Externas	2-43
Creación de una Tabla Externa mediante <code>ORACLE_LOADER</code>	2-45
Consulta de Tablas Externas	2-47
Creación de una Tabla Externa mediante <code>ORACLE_DATAPUMP</code> : Ejemplo	2-48
Prueba	2-49
Resumen	2-51
Práctica 2: Visión General	2-52

3 Gestión de Objetos con Vistas de Diccionario de Datos

Objetivos	3-2
Agenda	3-3
Diccionario de Datos	3-4
Estructura del Diccionario de Datos	3-5
Uso de las Vistas de Diccionario	3-7
Vistas <code>USER_OBJECTS</code> y <code>ALL_OBJECTS</code>	3-8
Vista <code>USER_OBJECTS</code>	3-9
Agenda	3-10
Información sobre Tablas	3-11
Información sobre Columnas	3-12
Información sobre Restricciones	3-14
<code>USER_CONSTRAINTS</code> : Ejemplo	3-15
Consulta de <code>USER_CONS_COLUMNS</code>	3-16
Agenda	3-17
Información sobre Vistas	3-18
Información sobre Secuencias	3-19
Confirmación de Secuencias	3-20
Información sobre Índices	3-21
<code>USER_INDEXES</code> : Ejemplos	3-22
Consulta <code>USER_IND_COLUMNS</code>	3-23
Información sobre Sinónimos	3-24
Agenda	3-25
Adición de Comentarios a una Tabla	3-26
Prueba	3-27
Resumen	3-28
Práctica 3: Visión General	3-29

4 Manipulación de Juegos de Datos Grandes

Objetivos 4-2

Agenda 4-3

Uso de Subconsultas para Manipular Datos 4-4

Recuperación de Datos mediante una Subconsulta como Origen 4-5

Inserción mediante una Subconsulta como Destino 4-7

Uso de la Palabra Clave `WITH CHECK OPTION` en Sentencias DML 4-9

Agenda 4-11

Visión General de la Función por Defecto Explícita 4-12

Uso de Valores por Defecto Explícitos 4-13

Copia de Filas de Otra Tabla 4-14

Agenda 4-15

Visión General de Sentencias `INSERT` de Varias de Tablas 4-16Tipos de Sentencias `INSERT` de Varias Tablas 4-18Sentencias `INSERT` de Varias Tablas 4-19`INSERT ALL` Incondicional 4-21`INSERT ALL` Condicional: Ejemplo 4-23`INSERT ALL` Condicional 4-24`INSERT FIRST` Condicional: Ejemplo 4-26`INSERT FIRST` Condicional 4-27`INSERT` mediante giro 4-29

Agenda 4-32

Sentencia `MERGE` 4-33Sintaxis de Sentencias `MERGE` 4-34

Fusión de Filas: Ejemplo 4-35

Agenda 4-38

Seguimiento de Cambios en Datos 4-39

Ejemplo de Consulta de Flashback de Versiones 4-40

Cláusula `VERSIONS BETWEEN` 4-42

Prueba 4-43

Resumen 4-44

Práctica 4: Visión General 4-45

5 Gestión de Datos Situados en Distintas Zonas Horarias

Objetivos 5-2

Agenda 5-3

Zonas Horarias 5-4

Parámetro de Sesión `TIME_ZONE` 5-5`CURRENT_DATE`, `CURRENT_TIMESTAMP` y `LOCALTIMESTAMP` 5-6

Comparación de Fecha y Hora en la Zona Horaria de una Sesión 5-7

DBTIMEZONE y SESSIONTIMEZONE 5-9

Tipos de Dato TIMESTAMP 5-10

Campos TIMESTAMP 5-11

Diferencia ente DATE y TIMESTAMP 5-12

Comparación de Tipos de Dato TIMESTAMP 5-13

Agenda 5-14

Tipos de Dato INTERVAL 5-15

Campos INTERVAL 5-17

INTERVAL YEAR TO MONTH: Ejemplo 5-18

Tipo de Dato INTERVAL DAY TO SECOND: Ejemplo 5-20

Agenda 5-21

EXTRACT 5-22

TZ_OFFSET 5-23

FROM_TZ 5-25

TO_TIMESTAMP 5-26

TO_YMINTERVAL 5-27

TO_DSINTERVAL 5-28

Horario de Verano 5-29

Prueba 5-31

Resumen 5-32

Práctica 5: Visión General 5-33

6 Recuperación de Datos mediante Subconsultas

Objetivos 6-2

Agenda 6-3

Subconsultas de Varias Columnas 6-4

Comparaciones de Columnas 6-5

Subconsulta de Comparación Pairwise 6-6

Subconsulta de comparación No Pairwise 6-8

Agenda 6-10

Expresiones de Subconsultas Escalares 6-11

Subconsultas Escalares: Ejemplos 6-12

Agenda 6-14

Subconsultas Correlacionadas 6-15

Uso de Subconsultas Correlacionadas 6-17

Agenda 6-19

Uso del Operador EXISTS 6-20

Buscar Todos los Departamentos que No Tienen Empleados 6-22

Subconsulta Correlacionada UPDATE 6-23

Uso de la Subconsulta Correlacionada `UPDATE` 6-24
 Subconsulta Correlacionada `DELETE` 6-26
 Uso de una Subconsulta Correlacionada `DELETE` 6-27
 Agenda 6-28
 Cláusula `WITH` 6-29
 Cláusula `WITH`: Ejemplo 6-30
 Cláusula `WITH` Recursiva 6-32
 Cláusula `WITH` Recursiva: Ejemplo 6-33
 Prueba 6-34
 Resumen 6-35
 Práctica 6: Visión General 6-37

7 Soporte para Expresiones Normales

Objetivos 7-2
 Agenda 7-3
 ¿Qué Son Expresiones Regulares? 7-4
 Ventajas de Utilizar Expresiones Regulares 7-5
 Uso de Funciones y Condiciones de Expresiones Regulares en SQL y PL/SQL 7-6
 Agenda 7-7
 ¿Que Son Metacaracteres? 7-8
 Uso de Metacaracteres con Expresiones Normales 7-9
 Agenda 7-11
 Condiciones y Funciones de Expresiones Regulares: Sintaxis 7-12
 Realización de una Búsqueda Básica mediante la Condición `REGEXP_LIKE` 7-13
 Sustitución de Patrones mediante la Función `REGEXP_REPLACE` 7-14
 Búsqueda de Patrones mediante la Función `REGEXP_REPLACE` 7-15
 Búsqueda de Patrones mediante la Función `REGEXP_SUBSTR` 7-16
 Agenda 7-17
 Subexpresiones 7-18
 Uso de Subexpresiones con el Soporte para Expresiones Regulares 7-19
 ¿Por qué Acceder a la n Subexpresión? 7-20
`REGEXP_SUBSTR`: Ejemplo 7-21
 Agenda 7-22
 Uso de la Función `REGEXP_COUNT` 7-23
 Expresiones Regulares y Restricciones de Control: Ejemplos 7-24
 Prueba 7-25
 Resumen 7-26
 Práctica 7: Visión General 7-27

Apéndice A: Prácticas y Soluciones

Apéndice B: Descripciones de las Tablas**Apéndice C: Uso de SQL Developer**

Objetivos	C-2
¿Qué es Oracle SQL Developer?	C-3
Especificaciones de SQL Developer	C-4
Interfaz de SQL Developer 1.5	C-5
Creación de una Conexión a la Base Datos	C-7
Examen de Objetos de Bases de Datos	C-10
Visualización de la Estructura de la Tabla	C-11
Examen de Archivos	C-12
Creación de un Objeto de Esquema	C-13
Creación de una Nueva Tabla: Ejemplo	C-14
Uso de la Hoja de Trabajo de SQL	C-15
Ejecución de Sentencias SQL	C-18
Guardado de Scripts SQL	C-19
Ejecución de Archivos de Script Guardados: Método 1	C-20
Ejecución de Archivos de Script Guardados: Método 2	C-21
Formato del Código SQL	C-22
Uso de Fragmentos	C-23
Uso de Fragmentos: Ejemplo	C-24
Depuración de Procedimientos y Funciones	C-25
Informes de Bases de Datos	C-26
Creación de un Informe Definido por el Usuario	C-27
Motores de Búsqueda y Herramientas Externas	C-28
Definición de Preferencias	C-29
Restablecimiento del Diseño de SQL Developer	C-30
Resumen	C-31

Apéndice D: Uso de SQL*Plus

Objetivos	D-2
Interacción de SQL y SQL*Plus	D-3
Sentencias SQL frente a Comandos SQL*Plus	D-4
Visión General de SQL*Plus	D-5
Conexión a SQL*Plus	D-6
Visualización de la Estructura de la Tabla	D-7
Comandos de Edición SQL*Plus	D-9
Uso de LIST, n y APPEND	D-11
Uso del Comando CHANGE	D-12
Comandos de Archivos SQL*Plus	D-13
Uso de los Comandos SAVE y START	D-14

Comando <code>SERVEROUTPUT</code>	D-15
Uso del Comando <code>SQL*Plus SPOOL</code>	D-16
Uso del Comando <code>AUTOTRACE</code>	D-17
Resumen	D-18

Apéndice E: Uso de JDeveloper

Objetivos	E-2
Oracle JDeveloper	E-3
Database Navigator	E-4
Creación de una Conexión	E-5
Examen de Objetos de Bases de Datos	E-6
Ejecución de Sentencias SQL	E-7
Creación de Unidades de Programa	E-8
Compilación	E-9
Ejecución de una Unidad de Programa	E-10
Borrado de una Unidad de Programa	E-11
Ventana Structure	E-12
Ventana del Editor	E-13
Navegador de Aplicaciones	E-14
Despliegue de Procedimientos Java Almacenados	E-15
Publicación de Java en PL/SQL	E-16
¿Cómo Puedo Obtener más Información sobre JDeveloper 11g?	E-17
Resumen	E-18

Apéndice F: Generación de Informes Agrupando Datos Relacionados

Objetivos	F-2
Revisión de Funciones de Grupo	F-3
Revisión de la Cláusula <code>GROUP BY</code>	F-4
Revisión de la Cláusula <code>HAVING</code>	F-5
<code>GROUP BY</code> con los Operadores <code>ROLLUP</code> y <code>CUBE</code>	F-6
Operador <code>ROLLUP</code>	F-7
Operador <code>ROLLUP</code> : Ejemplo	F-8
Operador <code>CUBE</code>	F-9
Operador <code>CUBE</code> : Ejemplo	F-10
Función <code>GROUPING</code>	F-11
Función <code>GROUPING</code> : Ejemplo	F-12
<code>GROUPING SETS</code>	F-13
<code>GROUPING SETS</code> : Ejemplo	F-15
Columnas Compuestas	F-17

Columnas Compuestas: Ejemplo F-19
Agrupamientos Concatenados F-21
Agrupamientos Concatenados: Ejemplo F-22
Resumen F-23

Apéndice G: Recuperación Jerárquica

Objetivos G-2
Datos de Ejemplo de la Tabla `EMPLOYEES` G-3
Estructura de Árbol Natural G-4
Consultas Jerárquicas G-5
Recorrido por el Árbol G-6
Recorrido por el Árbol: De Abajo Arriba G-8
Recorrido por el Árbol: De Arriba Abajo G-9
Clasificación de Filas con la Pseudocolumna `LEVEL` G-10
Aplicación de Formato a Informes Jerárquicos con `LEVEL` y `LPAD` G-11
Eliminación de Ramas G-13
Resumen G-14

Apéndice H: Escritura de Archivos de Comandos Avanzados

Objetivos H-2
Uso de SQL para Generar SQL H-3
Creación de un Script Básico H-4
Control del Entorno H-5
Imagen Completa H-6
Volcado del Contenido de una Tabla en un Archivo H-7
Generación de un Predicado Dinámico H-9
Resumen H-11

Apéndice I: Componentes Arquitectónicos de Oracle Database

Objetivos I-2
Arquitectura de Oracle Database: Visión General I-3
Estructuras de Servidor de Oracle Database I-4
Conexión a la Base de Datos I-5
Interacción con Oracle Database I-6
Arquitectura de Memoria de Oracle I-8
Arquitectura de Proceso I-10
Proceso de Escritores de Base de Datos I-12
Proceso de Escritor de Log I-13
Proceso de Punto de Control I-14
Proceso de Supervisión del Sistema I-15

Proceso de Supervisión de Procesos	I-16
Arquitectura de Almacenamiento de Oracle Database	I-17
Estructuras de Bases de Datos Físicas y Lógicas	I-19
Procesamiento de Sentencias SQL	I-21
Procesamiento de Consultas	I-22
Pool Compartido	I-23
Caché de buffers de la base de datos	I-25
Área Global de Programa (PGA)	I-26
Procesamiento de una Sentencia DML	I-27
Buffer de Redo Log	I-29
Segmento de Rollback	I-30
Procesamiento COMMIT	I-31
Resumen de Arquitectura de Oracle Database	I-33

Prácticas y soluciones adicionales

Introducción

ORACLE®

Copyright © 2010, Oracle. Todos los derechos reservados.

Objetivos

Al finalizar esta lección, debería estar capacitado para lo siguiente:

- Describir los objetivos del curso
- Describir las tablas y el esquema de base de datos que se utilizan en el curso
- Identificar los entornos disponibles que se pueden utilizar para este curso
- Revisar algunos conceptos fundamentales de SQL

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Agenda

- **Objetivos del curso y agenda**
- Apéndices y esquema de base de datos utilizados en el curso y el entorno de desarrollo disponible en este curso
- Revisión de algunos conceptos fundamentales de SQL
- Documentación y recursos adicionales de Oracle Database 11g

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Objetivos del Curso

Al finalizar este curso, debería estar capacitado para lo siguiente:

- Controlar el acceso a la base de datos para objetos concretos
- Agregar nuevos usuarios con distintos niveles de privilegios de acceso
- Gestionar objetos de esquema
- Gestionar objetos con vistas de diccionario de datos
- Manipular grandes juegos de datos en Oracle Database mediante subconsultas
- Gestionar datos situados en distintas zonas horarias
- Escribir subconsultas de varias columnas
- Utilizar subconsultas escalares y correlacionadas
- Utilizar el soporte de expresiones regulares en SQL

The Oracle logo, consisting of the word "ORACLE" in a bold, sans-serif font, with a registered trademark symbol (®) to the upper right.

Copyright © 2010, Oracle. Todos los derechos reservados.

Requisitos del Curso

El curso *Oracle Database: Conceptos Fundamentales de SQL I* es un requisito previo para este curso.

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Requisitos del Curso

La preparación necesaria para este curso es *Oracle Database: Conceptos Fundamentales de SQL I*.

Este curso ofrece una introducción a la tecnología de Oracle Database. En este curso, aprenderá los conceptos básicos relativos a las bases de datos relacionales y al potente lenguaje de programación SQL. El curso proporciona los conocimientos esenciales de SQL que permiten escribir consultas en una o varias tablas, manipular datos en tablas, crear objetos de bases de datos y realizar consultas en los metadatos.

Agenda

- Primer día:
 - Introducción
 - Control del Acceso de los Usuarios
 - Gestión de Objetos de Esquema
 - Gestión de Objetos con Vistas de Diccionario de Datos
- Segundo día:
 - Manipulación de Juegos de Datos Grandes
 - Gestión de Datos Situados en Distintas Zonas Horarias
 - Recuperación de Datos mediante Subconsultas
 - Soporte para Expresiones Regulares

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

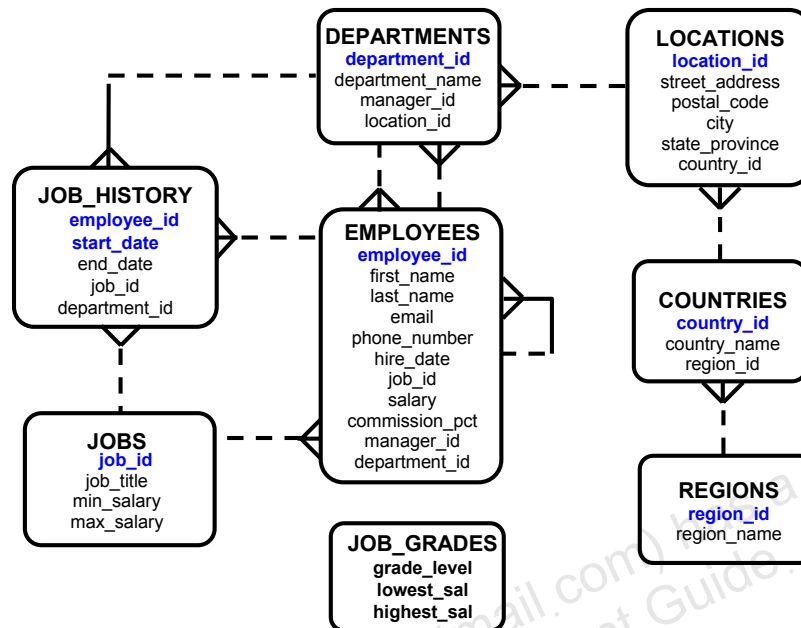
Agenda

- Objetivos del curso y agenda
- Apéndices y esquema de base de datos utilizados en el curso y el entorno de desarrollo disponible en este curso
- Revisión de Algunos Conceptos Fundamentales de SQL
- Documentación y recursos adicionales de Oracle Database 11g

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Tablas Utilizadas en este Curso



ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Descripción de las Tablas

Este curso utiliza datos de las siguientes tablas:

Descripciones de las Tablas

- La tabla **EMPLOYEES** contiene información sobre todos los empleados, como su nombre y apellido, ID de cargo, salarios, fechas de contratación, ID de departamento e ID de gestor. Se trata de una tabla secundaria de la tabla **DEPARTMENTS**.
- La tabla **DEPARTMENTS** contiene información, como el ID de departamento, el nombre de departamento, el ID de gestor y el ID de ubicación. Se trata de la tabla de clave primaria en la tabla **EMPLOYEES**.
- La tabla **LOCATIONS** contiene información sobre la ubicación de departamento. Contiene información sobre el ID de ubicación, calle, ciudad, provincia, código postal e ID de país. Se trata de la tabla de clave primaria en la tabla **DEPARTMENTS** y es secundaria de la tabla **COUNTRIES**.
- La tabla **COUNTRIES** contiene los nombres de país, ID de país e ID de región. Se trata de una tabla secundaria de la tabla **REGIONS**. Esta tabla es la tabla de clave primaria en la tabla **LOCATIONS**.
- La tabla **REGIONS** contiene los ID de región y los nombres de región de varios países. Se trata de una tabla de clave primaria en la tabla **COUNTRIES**.
- La tabla **JOB_GRADES** identifica un rango de salarios por grado de cargos. Los rangos de salarios no se solapan.
- La tabla **JOB_HISTORY** almacena el historial del cargo de los empleados.
- La tabla **JOBS** contiene los rangos de cargos y salarios.

Apéndices Utilizados en este Curso

- Apéndice A: Prácticas y Soluciones
- Apéndice B: Descripciones de las Tablas
- Apéndice C: Uso de SQL Developer
- Apéndice D: Uso de SQL*Plus
- Apéndice E: Uso de JDeveloper
- Apéndice F: Generación de Informes Agrupando Datos Relacionados
- Apéndice G: Recuperación Jerárquica
- Apéndice H: Escritura de Scripts Avanzados
- Apéndice I: Componentes Arquitectónicos de Oracle Database

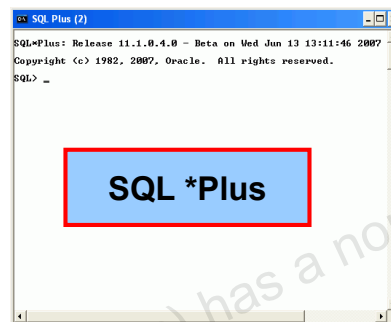
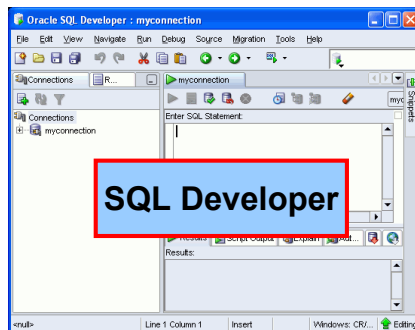
ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Entornos de Desarrollo

Existen dos entornos de desarrollo para este curso:

- La herramienta principal es Oracle SQL Developer.
- También puede utilizar la interfaz de línea de comandos de SQL*Plus.



ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Entornos de Desarrollo

SQL Developer

Este curso se desarrolla mediante Oracle SQL Developer como la herramienta para la ejecución de sentencias tratadas en los ejemplos de las diapositivas y prácticas.

- SQL Developer versión 1.5.4 se incluye con Oracle Database 11g y es la herramienta por defecto para esta clase.
- Además, SQL Developer versión 1.5.4 también está disponible en la máquina de la clase y se puede instalar para su uso. En el momento de la publicación de este curso, la versión 1.5.3 era la última versión SQL Developer.

SQL*Plus

También se puede utilizar el entorno de SQL*Plus para ejecutar todos los comandos SQL tratados en este curso.

Nota

- Consulte el apéndice C para obtener información sobre el uso de SQL Developer, que incluye instrucciones simples sobre la instalación de la versión 1.5.4.
- Consulte el apéndice D para obtener más información sobre el uso de SQL*Plus.

Agenda

- Objetivos del curso y agenda
- Apéndices y esquema de base de datos utilizados en el curso y el entorno de desarrollo disponible en este curso
- **Revisión de Algunos Conceptos Fundamentales de SQL**
- Documentación y recursos adicionales de Oracle Database 11g

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Agenda

Las siguientes diapositivas proporcionan una breve visión general de algunos de los conceptos que ha aprendido en el curso *Oracle Database: Conceptos Fundamentales de SQL I*.

Revisión de la Restricción de Datos

- Restringir las filas devueltas al utilizar la cláusula `WHERE`.
- Utilizar las condiciones de comparación para comparar una expresión con otro valor o expresión.

Operador	Significado
<code>BETWEEN ...AND...</code>	Entre dos valores (ambos incluidos)
<code>IN (set)</code>	Coincide con cualquiera de los valores de una lista
<code>LIKE</code>	Coincide con un patrón de caracteres

- Utilizar condiciones lógicas para combinar el resultado de dos condiciones de componentes y producir un único resultado según estas condiciones.

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Revisión de la Restricción de Datos

Puede restringir las filas que devuelve la consulta al utilizar la cláusula `WHERE`. Una cláusula `WHERE` contiene una condición que se debe cumplir e, inmediatamente después, le sigue la cláusula `FROM`.

La cláusula `WHERE` puede comparar valores en columnas, valores literales, expresiones aritméticas o funciones. Consta de tres elementos:

- Nombre de la columna
- Condición de comparación
- Nombre de la columna, constante o lista de valores

Utilice las condiciones de comparación en la cláusula `WHERE` en el siguiente formato:

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

Además de las condiciones mencionadas en la diapositiva, utilice otras condiciones de comparación como `=`, `<`, `>`, `<>`, `<=` y `>=`.

En SQL, están disponibles tres operadores lógicos:

- `AND`
- `OR`
- `NOT`

Revisión de Ordenación de Datos

- Ordenar las filas recuperadas con la cláusula `ORDER BY`:
 - `ASC`: orden ascendente, valor por defecto
 - `DESC`: orden descendente
- La cláusula `ORDER BY` es la última en una sentencia `SELECT`:

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

	LAST_NAME	JOB_ID	DEPARTMENT_ID	HIRE_DATE
1	King	AD_PRES	90	17-JUN-87
2	Whalen	AD_ASST	10	17-SEP-87
3	Kochhar	AD_VP	90	21-SEP-89
4	Hunold	IT_PROG	60	03-JAN-90
5	Ernst	IT_PROG	60	21-MAY-91
6	De Haan	AD_VP	90	13-JAN-93

...

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Revisión de Ordenación de Datos

El orden de las filas devueltas en un resultado de consulta no está definido. La cláusula `ORDER BY` se puede utilizar para ordenar las filas. Sin embargo, si utiliza la cláusula `ORDER BY`, debe ser la última cláusula de la sentencia SQL. Puede especificar una expresión, un alias o una posición de columna como la condición de ordenación.

Sintaxis

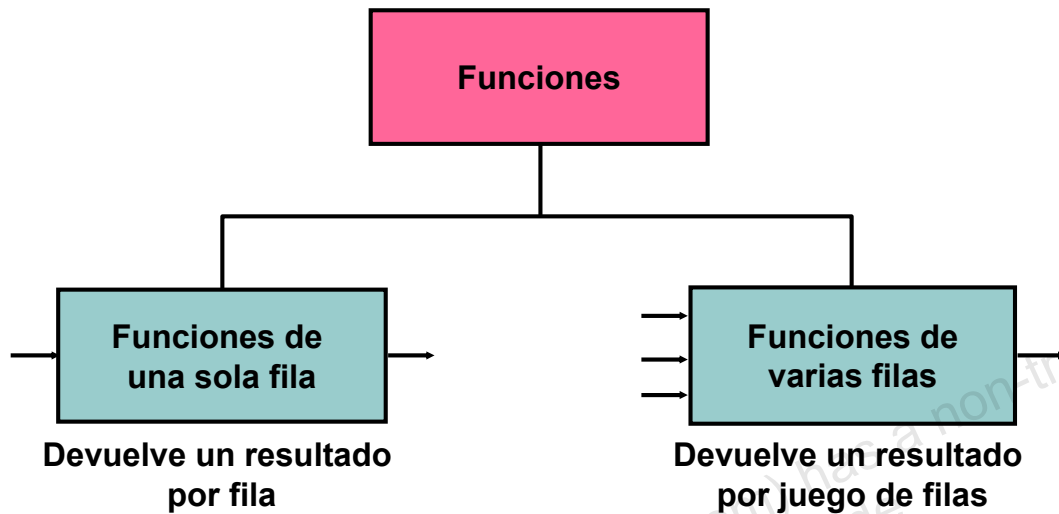
```
SELECT          expr
FROM            table
[WHERE          condition(s)]
[ORDER BY {column, expr, numeric_position} [ASC|DESC]];
```

En la sintaxis:

<code>ORDER BY</code>	especifica el orden en el que aparecen las filas recuperadas
<code>ASC</code>	ordena las filas en orden ascendente (orden por defecto)
<code>DESC</code>	ordena las filas en orden descendente

Si la cláusula `ORDER BY` no se utiliza, el orden no está definido y puede que el servidor de Oracle no recupere dos veces las filas en el mismo orden para la misma consulta. Utilice la cláusula `ORDER BY` para mostrar las filas en un orden específico.

Revisión de Funciones SQL



ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Revisión de Funciones SQL

Hay dos tipos de grupos de funciones:

- Funciones de una sola fila
- Funciones de varias filas

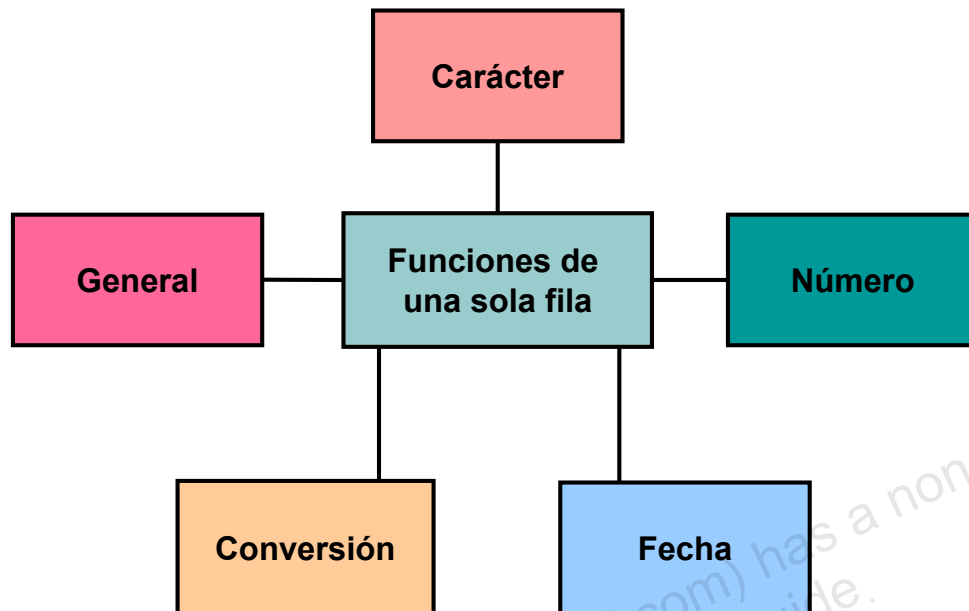
Funciones de Una Sola Fila

Estas funciones funcionan sólo en filas únicas y devuelven un resultado por fila. Existen distintos tipos de funciones de una sola fila como funciones de carácter, número, fecha, conversión y generales.

Funciones de Varias Filas

Las funciones pueden manipular grupos de filas para proporcionar un resultado por grupo de filas. Estas funciones también se conocen como *funciones de grupo*.

Revisión de Funciones de Una Sola Fila



ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

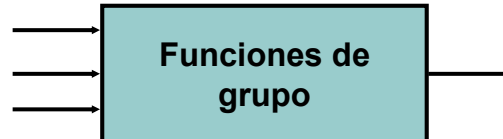
Revisión de Funciones de Una Sola Fila

A continuación se muestran los distintos tipos de funciones de una sola fila:

- **Funciones de carácter:** aceptan la entrada de caracteres y pueden devolver valores de número y de carácter.
- **Funciones numéricas:** aceptan valores de entrada y devuelven valores numéricos.
- **Funciones de fecha:** operan en valores del tipo de dato DATE. (Todas las funciones de fecha devuelven un valor de tipo de dato DATE excepto la función MONTHS_BETWEEN, que devuelve un número).
- **Funciones de conversión:** convierten un valor de un tipo de dato a otro.
- **Funciones generales:**
 - NVL
 - NVL2
 - NULLIF
 - COALESCE
 - CASE
 - DECODE

Revisión de Tipos de Funciones de Grupo

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



ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Revisión de Tipos de Funciones de Grupo

Cada una de las funciones acepta un argumento. La siguiente tabla identifica las opciones que se pueden utilizar en la sintaxis:

Función	Descripción
AVG ([DISTINCT <u>ALL</u>] <i>n</i>)	Valor medio de <i>n</i> ; ignora los valores nulos
COUNT ({ * [DISTINCT <u>ALL</u>] <i>expr</i> })	Número de filas donde <i>expr</i> evalúa otros valores que no son nulos (tiene en cuenta todas las filas seleccionadas con *, incluyendo duplicados y filas con valores nulos)
MAX ([DISTINCT <u>ALL</u>] <i>expr</i>)	Valor máximo de <i>expr</i> ; ignora los valores nulos
MIN ([DISTINCT <u>ALL</u>] <i>expr</i>)	Valor mínimo de <i>expr</i> ; ignora los valores nulos
STDDEV ([DISTINCT <u>ALL</u>] <i>n</i>)	Desviación estándar de <i>n</i> , ignora los valores nulos
SUM ([DISTINCT <u>ALL</u>] <i>n</i>)	Valores de suma de <i>n</i> , ignora los valores nulos
VARIANCE ([DISTINCT <u>ALL</u>] <i>n</i>)	Varianza de <i>n</i> ; ignora los valores nulos

Revisión de Subconsultas

- Una subconsulta es una sentencia `SELECT` que está anidada en la cláusula de otra sentencia `SELECT`.
- Sintaxis:

```
SELECT select_list
FROM   table
WHERE  expr operator
        (SELECT select_list
         FROM   table );
```

- Tipos de subconsultas:

Subconsulta de una sola fila	Subconsulta de varias filas
Devuelve una sola fila	Devuelve más de una fila
Utiliza operadores de comparación de una sola fila	Utiliza operadores de comparación de varias filas

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Revisión de Subconsultas

Puede crear sentencias potentes a partir de sentencias simples utilizando las subconsultas. Las subconsultas son útiles cuando una consulta está basada en criterios de búsqueda con valores intermedios desconocidos.

Puede colocar la subconsulta en diferentes cláusulas SQL, entre las que se incluyen las siguientes:

- Cláusula `WHERE`
- Cláusula `HAVING`
- Cláusula `FROM`

La subconsulta (consulta interna) se ejecuta una vez antes de la consulta principal (consulta externa). La consulta principal utiliza el resultado de la subconsulta.

Una subconsulta de una sola fila utiliza un operador de una sola fila `=`, `>`, `<`, `>=`, `<=`, y `<>`. Con una subconsulta de varias filas, utilice un operador de varias filas como `IN`, `ANY` y `ALL`.

Ejemplo: muestra los detalles de empleados cuyo salario es igual al salario mínimo.

```
SELECT last_name, salary, job_id
FROM   employees
WHERE  salary = (SELECT MIN(salary)
                 FROM   employees );
```

En el ejemplo, la función de grupo `MIN` devuelve un valor único a la consulta externa.

Nota: en este curso, aprenderá a utilizar varias subconsultas de varias columnas. Las subconsultas de varias columnas devuelven más de una columna de la sentencia interna `SELECT`.

Revisión de Manipulación de Datos

Las sentencias de lenguaje de manipulación de datos (DML) se ejecutan al:

- Agregar nuevas filas a una tabla
- Modificar filas existentes en una tabla
- Eliminar filas existentes de una tabla\

Función	Descripción
INSERT	Agrega una nueva fila a la tabla.
UPDATE	Modifica filas existentes en la tabla.
DELETE	Elimina filas existentes de la tabla.
MERGE	Actualiza, inserta o suprime una fila de forma condicional en/de una tabla.

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Revisión de Manipulación de Datos

Para agregar, actualizar o suprimir los datos de la base de datos, ejecute una sentencia DML. La recopilación de sentencias DML que forman una unidad lógica de trabajo se denomina transacción. Puede agregar nuevas filas a una tabla mediante la sentencia `INSERT`. Con la siguiente sintaxis, sólo se inserta una fila cada vez.

```
INSERT INTO table [(column [, column...])]
VALUES (value[, value...]);
```

Puede utilizar la sentencia `INSERT` para agregar filas a la tabla en la que se derivan los valores de las tablas existentes. En lugar de la cláusula `VALUES`, utilice una subconsulta. El número de columnas y sus tipos de dato de la lista de columnas de la cláusula `INSERT` deben coincidir con el número de valores y sus tipos de dato en la subconsulta.

La sentencia `UPDATE` modifica filas específicas si especifica la cláusula `WHERE`.

```
UPDATE table
SET column = value [, column = value, ...]
[WHERE condition];
```

Puede eliminar filas existentes mediante la sentencia `DELETE`. Puede suprimir filas concretas mediante la especificación de la cláusula `WHERE` en la sentencia `DELETE`.

```
DELETE [FROM] table
[WHERE condition];
```

Aprenderá acerca de la sentencia `MERGE` en la lección sobre manipulación de juegos de datos grandes.

Agenda

- Objetivos del curso y agenda
- Apéndices y esquema de base de datos utilizados en el curso y el entorno de desarrollo disponible en este curso
- Revisión de algunos conceptos fundamentales de SQL
- Documentación y recursos adicionales de Oracle Database 11g

ORACLE®

Copyright © 2010, Oracle. Todos los derechos reservados.

Documentación sobre SQL de Oracle Database

- *Oracle Database New Features Guide*
- *Oracle Database Reference*
- *Oracle Database SQL Language Reference*
- *Oracle Database Concepts*
- *Oracle Database SQL Developer User's Guide Release 1.2*

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Documentación sobre SQL de Oracle Database

Navegue a <http://www.oracle.com/pls/db102/homepage> para acceder a la biblioteca de documentación de Oracle Database 10g.

Navegue a <http://www.oracle.com/pls/db112/homepage> para acceder a la biblioteca de documentación de Oracle Database 11g versión 2.

Recursos Adicionales

Para obtener más información sobre Oracle Database 11g SQL, consulte:

- *Oracle Database 11g: New Features eStudies*
- *Oracle by Example series (OBE): Oracle Database 11g*

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Resumen

En esta lección debe haber aprendido lo siguiente:

- Los objetivos del curso
- Las tablas de ejemplo utilizadas en el curso

ORACLE®

Copyright © 2010, Oracle. Todos los derechos reservados.

Práctica I: Visión General

En esta práctica se abordan los siguientes temas:

- Ejecución del tutorial en línea de SQL Developer
- Inicio de SQL Developer y creación de una nueva conexión a la base de datos y examen de las tablas
- Ejecución de sentencias SQL mediante la hoja de trabajo de SQL
- Revisión de los conceptos básicos de SQL

The Oracle logo, consisting of the word "ORACLE" in a bold, sans-serif font, with a registered trademark symbol (®) to its upper right.

Copyright © 2010, Oracle. Todos los derechos reservados.

Práctica I: Visión General

En esta práctica, utiliza SQL Developer para ejecutar sentencias SQL.

Nota: todas las prácticas escritas utilizan SQL Developer como entorno de desarrollo. Aunque se recomienda utilizar SQL Developer, también puede usar el entorno SQL*Plus disponible en este curso.

1

Control del Acceso de los Usuarios

ORACLE®

Copyright © 2010, Oracle. Todos los derechos reservados.

Rodrigo Esteves (rodrigo_@hotmail.com) has a non-transferable license to use this Student Guide.

Objetivos

Al finalizar esta lección, debería estar capacitado para lo siguiente:

- Diferenciar entre privilegios de sistema y privilegios de objeto
- Asignar privilegios en tablas
- Asignar roles
- Distinguir entre privilegios y roles

ORACLE®

Copyright © 2010, Oracle. Todos los derechos reservados.

Objetivos

En esta lección, aprenderá a controlar el acceso a la base de datos para especificar objetos y agregar nuevos usuarios con diferentes niveles de privilegios de acceso.

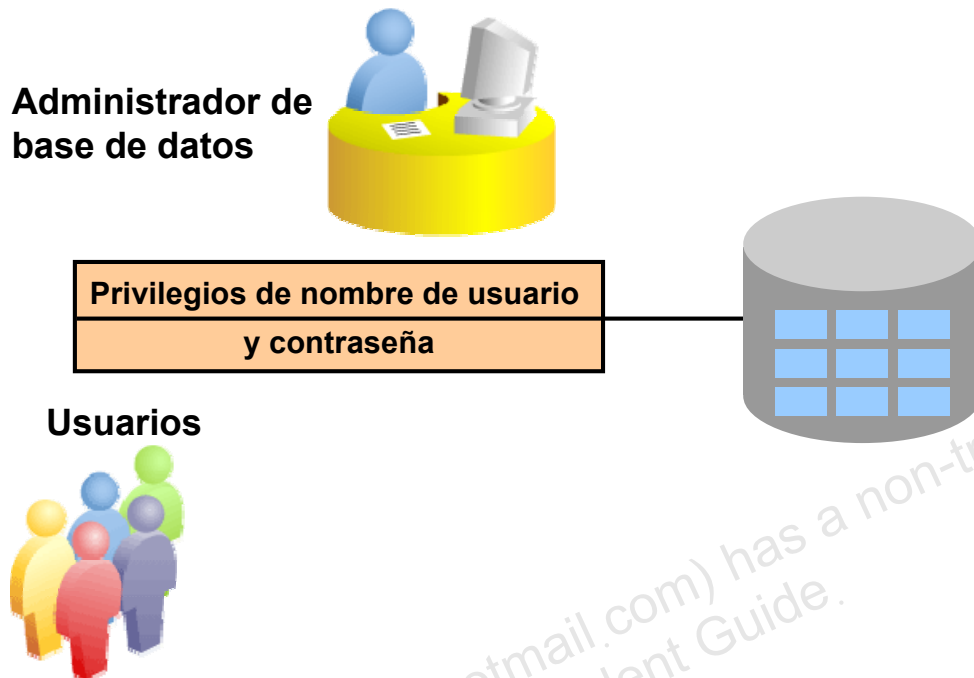
Agenda

- Privilegios del sistema
- Creación de roles
- Privilegios de objeto
- Revocación de privilegios de objeto

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Control del Acceso de los Usuarios



ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Control del Acceso de los Usuarios

En un entorno de varios usuarios, desea mantener la seguridad del uso y acceso a la base de datos. Con la seguridad de la base de datos del servidor de Oracle, puede realizar lo siguiente:

- Controlar el acceso a la base de datos.
- Proporcionar acceso a objetos específicos en la base de datos.
- Confirmar los privilegios asignados y recibidos con el diccionario de datos de Oracle.

La seguridad de la base de datos se puede clasificar en dos categorías: seguridad del sistema y de datos. La seguridad del sistema cubre el acceso y el uso de la base de datos a nivel del sistema, como nombre de usuario y contraseña, el espacio de disco asignado a los usuarios y las operaciones que los usuarios pueden realizar. La seguridad de la base de datos cubre el acceso y el uso de los objetos de la base de datos así como las acciones que dichos usuarios puedan realizar en los objetos.

Privilegios

- Seguridad de la base de datos:
 - Seguridad del sistema
 - Seguridad de datos
- Privilegios del sistema: realización de una acción determinada en la base de datos
- Privilegios de objeto: manipulación del contenido de los objetos de la base de datos
- Esquemas: recopilación de objetos como tablas, vistas y secuencias

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Privilegios

Un privilegio es el derecho a ejecutar sentencias SQL determinadas. El administrador de la base de datos (DBA) es un usuario de alto nivel con la capacidad de crear y otorgar acceso a los usuarios a la base de datos y sus objetos. Los usuarios necesitan los *privilegios del sistema* para acceder a la base de datos y los *privilegios de objeto* para manipular el contenido de los objetos en la base de datos. Los usuarios también proporcionan el privilegios para otorgar privilegios adicionales a otros usuarios o a los *roles*, que son grupos con nombres de privilegios relacionados.

Esquemas

Un *esquema* es una recopilación de objetos como tablas, vistas y secuencias. El esquema es propiedad de un usuario de base de datos y tiene el mismo nombre que el usuario.

Un privilegio del sistema es el derecho para realizar una acción determinada o para realizar una acción en objetos de esquema de un tipo determinado. Un privilegio de objeto proporciona al usuario la capacidad de realizar una acción determinada en un objeto de esquema específico.

Para obtener más información, consulte el manual de referencia *Oracle Database 2 Day DBA (DBA de 2 Días de Oracle Database)* para base de datos 10g u 11g.

Privilegios del Sistema

- Más de 100 privilegios disponibles.
- El administrador de la base de datos tiene privilegios del sistema de alto nivel para realizar tareas como:
 - Creación de nuevos usuarios
 - Eliminación de usuarios
 - Eliminación de tablas
 - Realización de copias de seguridad de tablas



Copyright © 2010, Oracle. Todos los derechos reservados.

Privilegios del Sistema

Hay más de 100 privilegios del sistema disponibles para usuarios y roles. Normalmente, el administrador de la base de datos (DBA) proporciona los privilegios del sistema.

Privilegios de DBA Típicos

Privilegio del Sistema	Operaciones Autorizadas
CREATE USER	El usuario con privilegios puede crear otros usuarios de Oracle.
DROP USER	El usuario con privilegios puede borrar otro usuario.
DROP ANY TABLE	El usuario con privilegios puede borrar una tabla en cualquier esquema.
BACKUP ANY TABLE	El usuario con privilegios puede realizar una copia de seguridad de tablas en cualquier esquema con la utilidad de exportación.
SELECT ANY TABLE	El usuario con privilegios puede consultar las tablas, vistas o vistas materializadas en cualquier esquema.
CREATE ANY TABLE	El usuario con privilegios puede crear una tabla en cualquier esquema.

Creación de un Usuario

El DBA crea el usuario con la sentencia `CREATE USER`.

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

```
CREATE USER demo
IDENTIFIED BY demo;
```

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Creación de un Usuario

El DBA crea el usuario mediante la ejecución de la sentencia `CREATE USER`. El usuario no tiene ningún privilegio en este punto. El DBA puede otorgar los privilegios a dicho usuario. Estos privilegios determinan lo que el usuario puede realizar a nivel de base de datos.

La diapositiva muestra la sintaxis abreviada para crear un usuario.

En la sintaxis:

user es el nombre del usuario que se va a crear
Password especifica que el usuario se debe conectar con esta contraseña

Para obtener más información, consulte *Oracle Database SQL Reference (Referencia de SQL de Oracle Database) para base de datos 10g u 11g*.

Nota: a partir de Oracle Database 11g, las contraseñas son sensibles a mayúsculas/minúsculas.

Privilegios del Sistema de Usuario

- Una vez creado un usuario, el DBA puede otorgar privilegios del sistema específicos a dicho usuario.

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

- Un desarrollador de aplicaciones, por ejemplo, puede tener los siguientes privilegios del sistema:
 - CREATE SESSION
 - CREATE TABLE
 - CREATE SEQUENCE
 - CREATE VIEW
 - CREATE PROCEDURE



Copyright © 2010, Oracle. Todos los derechos reservados.

Privilegios de Usuario Típicos

Una vez que el DBA crea un usuario, el DBA puede asignar privilegios a dicho usuario.

Privilegio del Sistema	Operaciones Autorizadas
CREATE SESSION	Conectar a la base de datos.
CREATE TABLE	Crear tablas en el esquema del usuario.
CREATE SEQUENCE	Crear una secuencia en el esquema del usuario.
CREATE VIEW	Crear una vista en el esquema del usuario.
CREATE PROCEDURE	Crear un paquete, función o procedimiento almacenado en el esquema del usuario.

En la sintaxis:

privilege

es el privilegio del sistema que se va a otorgar

user | *role* | *PUBLIC*

es el nombre del usuario, el nombre del rol o *PUBLIC*

(que designa que se asigne el privilegio a todos los usuarios)

Nota: los privilegios del sistema actuales están en la vista de diccionario *SESSION_PRIVS*.

El diccionario de datos es una recopilación de tablas y vistas que crea y mantiene el servidor de Oracle. Contienen información sobre la base de datos.

Asignación de Privilegios del Sistema

El DBA puede otorgar privilegios del sistema específicos a un usuario.

```
GRANT  create session, create table,  
        create sequence, create view  
TO      demo;
```

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Asignación de Privilegios del Sistema

El DBA utiliza la sentencia GRANT para asignar privilegios del sistema al usuario. Una vez que el usuario ha otorgado los privilegios, el usuario puede utilizar estos privilegios inmediatamente.

En el ejemplo de la diapositiva, al usuario demo se le ha asignado el privilegio para crear sesiones, tablas, secuencias y vistas.

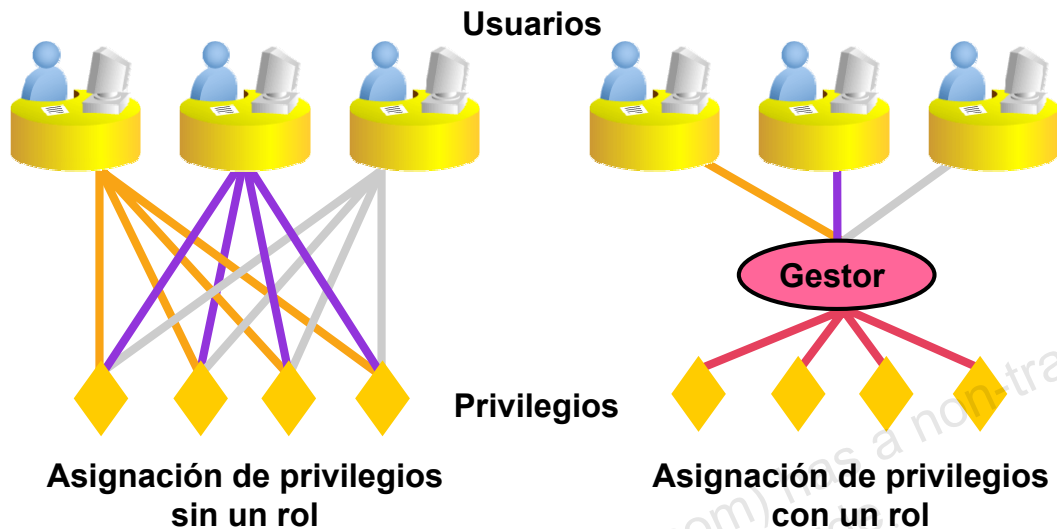
Agenda

- Privilegios del sistema
- **Creación de roles**
- Privilegios de objeto
- Revocación de privilegios de objeto

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

¿Qué es un Rol?



ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

¿Qué es un Rol?

Un rol es un grupo con nombre de privilegios relacionados que se pueden otorgar al usuario. Este método facilita la revocación y el mantenimiento de los privilegios.

Un usuario puede tener acceso a diferentes roles y el mismo rol se puede asignar a diferentes usuarios. Los roles normalmente se crean para una aplicación de base de datos.

Creación y Asignación de Roles

En primer lugar, el DBA debe crear el rol. A continuación, el DBA puede asignar privilegios al rol y asignar dicho rol a los usuarios.

Sintaxis

```
CREATE ROLE role;
```

En la sintaxis:

role es el nombre del rol que se va a crear

Después de crear el rol, el DBA puede utilizar la sentencia `GRANT` para asignar el rol a los usuarios, así como asignar privilegios al rol. Un rol no es un objeto de esquema; por lo tanto, cualquier usuario puede agregar privilegios a un rol.

Creación y Asignación de Privilegios a un Rol

- Crear un rol:

```
CREATE ROLE manager;
```

- Otorgar privilegios a un rol:

```
GRANT create table, create view  
TO manager;
```

- Otorgar un rol a los usuarios:

```
GRANT manager TO alice;
```

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Creación de un Rol

El ejemplo de la diapositiva crea un rol `manager` y, a continuación, permite al gestor crear tablas y vistas. Otorga al usuario `alice` el rol de un gestor. Ahora `alice` puede crear tablas y vistas.

Si los usuarios tiene varios roles otorgados, reciben todos los privilegios asociados a todos los roles.

Cambio de Contraseña

- El DBA crea la cuenta de usuario e inicializa la contraseña.
- Puede cambiar la contraseña mediante la sentencia ALTER USER.

```
ALTER USER demo
IDENTIFIED BY employ;
```

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Cambio de Contraseña

El DBA crea una cuenta e inicializa una contraseña para cada usuario. Puede cambiar la contraseña mediante la sentencia ALTER USER.

El ejemplo de la diapositiva muestra que el usuario demo cambia la contraseña mediante la sentencia ALTER USER.

Sintaxis

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

En la sintaxis:

<i>user</i>	es el nombre del usuario
<i>password</i>	especifica la nueva contraseña

Aunque esta sentencia se puede utilizar para cambiar la contraseña, hay otras muchas opciones. Debe tener el privilegio ALTER USER para cambiar cualquier otra opción.

Para obtener más información, consulte *Oracle Database SQL Reference (Referencia de SQL de Oracle Database) para base de datos 10g u 11g*.

Nota: SQL*Plus tiene un comando PASSWORD (PASSW) que se puede utilizar para cambiar la contraseña de un usuario al conectarse. Este comando no está disponible en SQL Developer.

Agenda

- Privilegios del sistema
- Creación de roles
- **Privilegios de objeto**
- Revocación de privilegios de objeto

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Privilegios de Objeto

Privilegio de objeto	Tabla	Vista	Secuencia
ALTER	✓		✓
DELETE	✓	✓	
INDEX	✓		
INSERT	✓	✓	
REFERENCES	✓		
SELECT	✓	✓	✓
UPDATE	✓	✓	

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Privilegios de Objeto

Un *privilegio de objeto* es un privilegio o derecho para realizar una acción determinada en una tabla, vista, secuencia o procedimiento específico. Cada objeto tiene un juego determinado de privilegios que se pueden otorgar. La tabla de la diapositiva muestra los privilegios para varios objetos. Tenga en cuenta que los únicos privilegios que se aplican a una secuencia son SELECT y ALTER. UPDATE, REFERENCES e INSERT se pueden restringir especificando un subjuego de columnas actualizables.

Un privilegio SELECT se puede restringir mediante la creación de una vista con un subjuego y otorgamiento del privilegio SELECT sólo en la vista. Un privilegio otorgado en un sinónimo se convierte en un privilegio en la tabla base a la que hace referencia el sinónimo.

Nota: con el privilegio REFERENCES, puede garantizar que otros usuarios pueden crear restricciones FOREIGN KEY que hacen referencia a la tabla.

Privilegios de Objeto

- Los privilegios de objeto varían de objeto a objeto.
- Un propietario tiene todos los privilegios sobre el objeto.
- Un propietario puede proporcionar privilegios específicos sobre el objeto de dicho propietario.

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

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Asignación de Privilegios de Objeto

Hay disponibles diferentes privilegios de objeto para distintos tipos de objetos de esquema. Un usuario automáticamente tiene todos los privilegios de objeto para los objetos de esquema que están en el esquema de usuario. Un usuario puede otorgar privilegios de objeto sobre cualquier objeto de esquema que el usuario posea a otro usuario o rol. Si la asignación incluye `WITH GRANT OPTION`, el usuario con privilegios puede otorgar el privilegio de objeto a otros usuarios; de lo contrario, el usuario con privilegios puede utilizar el privilegio pero no puede otorgarlo a otros usuarios.

En la sintaxis:

object_priv

ALL

columns

ON *object*

TO

PUBLIC

WITH GRANT OPTION

es un privilegio de objeto que se va a otorgar

especifica todos los privilegios de objeto

especifica la columna de una tabla o vista en la que

se otorgan los privilegios

es el objeto sobre el que se otorgan los privilegios

identifica a quién se le otorga el privilegio

otorga privilegios de objeto a todos los usuarios

permite al usuario con privilegios otorgar privilegios de

objeto a otros usuarios y roles

Nota: en la sintaxis, *schema* es el mismo nombre que el del propietario.

Asignación de Privilegios de Objeto

- Otorgar privilegios de consulta en la tabla `EMPLOYEES`:

```
GRANT  select
ON     employees
TO     demo;
```

- Otorgar privilegios para actualizar columnas específicas en los usuarios y roles:

```
GRANT  update (department_name, location_id)
ON     departments
TO     demo, manager;
```

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Instrucciones

- Para otorgar privilegios en un objeto, el objeto debe estar en el propio esquema o debe otorgar los privilegios de objeto `WITH GRANT OPTION`.
- Un propietario de objeto puede otorgar privilegios de objeto en el objeto a cualquier otro usuario o rol de la base de datos.
- El propietario de un objeto adquiere automáticamente todos los privilegios de objeto en dicho objeto.

En el primer ejemplo de la diapositiva, se otorga al usuario `demo` el privilegio para consultar la tabla `EMPLOYEES`. El segundo ejemplo, se otorgan los privilegios `UPDATE` sobre columnas específicas en la tabla `DEPARTMENTS` a `demo` y al rol `manager`.

Por ejemplo, si el esquema es `oraxx` y el usuario `demo`, ahora desea utilizar una sentencia `SELECT` para obtener los datos de la tabla `EMPLOYEES`, la sintaxis que debe utilizar es:

```
SELECT * FROM oraxx.employees;
```

Asimismo, el usuario `demo` puede crear un sinónimo para la tabla y ejecutar una sentencia `SELECT` a partir del sinónimo:

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

Nota: los DBA generalmente asignan privilegios del sistema; los usuarios que poseen un objeto pueden otorgar privilegios de objeto.

Transferencia de Privilegios

- Proporcionar una autoridad de usuario para transferir privilegios:

```
GRANT  select, insert
ON      departments
TO      demo
WITH    GRANT OPTION;
```

- Permitir a todos los usuarios del sistema consultar datos de la tabla DEPARTMENTS de Alice:

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

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Transferencia de Privilegios

Palabra clave WITH GRANT OPTION

El usuario con privilegios puede transferir un privilegio que se otorga con la cláusula WITH GRANT OPTION a otros usuarios y roles. Los privilegios de objeto otorgados con la cláusula WITH GRANT OPTION se revocan si se revoca el privilegio del otorgante.

El ejemplo de la diapositiva proporciona al usuario demo acceso a la tabla DEPARTMENTS con los privilegios para consultar la tabla y agregar filas a la tabla. El ejemplo también muestra que user1 puede proporcionar el resto de estos privilegios.

Palabra clave PUBLIC

Un propietario de una tabla puede otorgar acceso a todos los usuarios mediante la palabra clave PUBLIC.

El segundo ejemplo de la diapositiva permite a todos los usuarios del sistema consultar datos de la tabla DEPARTMENTS de Alice.

Confirmación de Privilegios Otorgados

Vista del Diccionario de Datos	Descripción
ROLE_SYS_PRIVS	Privilegios del sistema otorgados a roles
ROLE_TAB_PRIVS	Privilegios de tabla otorgados a roles
USER_ROLE_PRIVS	Roles a los que puede acceder el usuario
USER_SYS_PRIVS	Privilegios del sistema otorgados al usuario
USER_TAB_PRIVS_MADE	Privilegios de objeto otorgados a objetos del usuario
USER_TAB_PRIVS_RECD	Privilegios de objeto otorgados al usuario
USER_COL_PRIVS_MADE	Privilegios de objeto otorgados a columnas de objetos del usuario
USER_COL_PRIVS_RECD	Privilegios de objeto otorgados al usuario en columnas específicas

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Confirmación de Privilegios Otorgados

Si intenta realizar una operación no autorizada, como la supresión de una fila de una tabla para la que no tiene el privilegio `DELETE`, el servidor de Oracle no permite que se produzca la operación.

Si recibe el mensaje de error del servidor de Oracle “Table or view does not exist”, es porque ha realizado una de las siguientes acciones:

- Asignar un nombre a una tabla o vista que no existe
- Intentar realizar una operación en una tabla o vista para la que no tiene el privilegio adecuado

El diccionario de datos se organiza en tablas y vistas y contiene información sobre la base de datos. Puede acceder al diccionario de datos para ver los privilegios que tiene. La tabla de esta diapositiva describe varias vistas de diccionario de datos.

Dispone de más información sobre las vistas de diccionario de datos en la lección “Gestión de Objetos con Vistas de Diccionario de Datos”.

Nota: la vista de diccionario `ALL_TAB_PRIVS_MADE` describe todas las otorgaciones de objeto realizadas por el usuario en los objetos propiedad del usuario.

Agenda

- Privilegios del sistema
- Creación de roles
- Privilegios de objeto
- Revocación de privilegios de objeto

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Revocación de Privilegios de Objeto

- Utilice la sentencia `REVOKE` para revocar los privilegios otorgados a otros usuarios.
- Los privilegios otorgados a otros usuarios mediante la cláusula `WITH GRANT OPTION` también se revocan.

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



Copyright © 2010, Oracle. Todos los derechos reservados.

Revocación de Privilegios de Objeto

Puede eliminar los privilegios otorgados a otros usuarios mediante la sentencia `REVOKE`.

Al utilizar la sentencia `REVOKE`, los privilegios especificados se revocan de los usuarios que nombra y de cualquier otro usuario al que el usuario revocado otorgue dichos privilegios.

En la sintaxis:

`CASCADE` es necesario para eliminar las restricciones de integridad referencial realizadas en el objeto `CONSTRAINTS` mediante el privilegio `REFERENCES`

Para obtener más información, consulte *Oracle Database SQL Reference (Referencia de SQL de Oracle Database)* para base de datos 10g u 11g.

Nota: si un usuario deja la compañía y revoca sus privilegios, debe volver a otorgar los privilegios que este usuario podría haber otorgado a otros usuarios. Si borra la cuenta de usuario sin revocar los privilegios, los privilegios del sistema otorgados por este usuario a otros usuarios no se verán afectados por esta acción.

Revocación de Privilegios de Objeto

Revocar los privilegios `SELECT` e `INSERT` proporcionados al usuario `demo` en la tabla `DEPARTMENTS`.

```
REVOKE  select, insert
ON      departments
FROM    demo ;
```

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Revocación de Privilegios de Objeto (continuación)

El ejemplo de la diapositiva revoca los privilegios `SELECT` e `INSERT` proporcionados al usuario `demo` en la tabla `DEPARTMENTS`.

Nota: si a un usuario se le otorga un privilegio con la cláusula `WITH GRANT OPTION`, dicho usuario también puede otorgar el privilegio con la cláusula `WITH GRANT OPTION`, de modo que es posible crear una larga cadena de usuarios con privilegios, pero no se permiten permisos circulares (otorgado al antiguo otorgante). Si el propietario revoca un privilegio de un usuario que ha otorgado el privilegio a otros usuarios, la revocación tiene un efecto cascada en todos los privilegios otorgados.

Por ejemplo, si el usuario A otorga un privilegio `SELECT` en una tabla al usuario B incluyendo la cláusula `WITH GRANT OPTION`, el usuario B puede otorgar al usuario C el privilegio `SELECT` también con la cláusula `WITH GRANT OPTION` y el usuario C puede otorgar al usuario D el privilegio `SELECT`. Si el usuario A revoca los privilegios del usuario B, los privilegios otorgados a los usuarios C y D también se revocan.

Prueba

¿Cuáles de las siguientes afirmaciones son ciertas?

1. Una vez que el usuario crea un objeto, el usuario puede transferir los privilegios de objeto disponibles a otros usuarios mediante la sentencia GRANT.
2. Un usuario puede crear roles mediante la sentencia CREATE ROLE para transferir una recopilación de privilegios de objeto o de sistema a otros usuarios.
3. Los usuarios pueden cambiar sus propias contraseñas.
4. Los usuarios pueden ver los privilegios otorgados a los mismos y aquellos otorgados en sus objetos.

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Respuestas: 1, 3, 4

Resumen

En esta lección, debe haber aprendido lo siguiente:

- Diferenciar entre privilegios de sistema y privilegios de objeto
- Asignar privilegios en tablas
- Asignar roles
- Distinguir entre privilegios y roles

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Resumen

Los DBA establecen la seguridad de base de datos inicial de los usuarios asignando privilegios a los usuarios.

- El DBA crea usuarios que deben tener una contraseña. El DBA es responsable de establecer los privilegios del sistema inicial para un usuario.
- Una vez que el usuario crea un objeto, el usuario puede transferir los privilegios de objeto disponibles a otros usuarios mediante la sentencia `GRANT`.
- Un DBA puede crear roles mediante la sentencia `CREATE ROLE` para transferir una recopilación de privilegios de objeto o de sistema a varios usuarios. Los roles que se utilizan para otorgar y revocar privilegios son fáciles de mantener.
- Los usuarios pueden cambiar la contraseña mediante la sentencia `ALTER USER`.
- Puede eliminar privilegios de los usuarios mediante la sentencia `REVOKE`.
- Con las vistas de diccionario de datos, los usuarios pueden ver los privilegios otorgados a los mismos y aquellos otorgados en sus objetos.

Práctica 1: Visión General

En esta práctica se abordan los siguientes temas:

- Asignación de otros privilegios de usuarios a la tabla
- Modificación de otra tabla de usuario mediante los privilegios que se le han otorgado

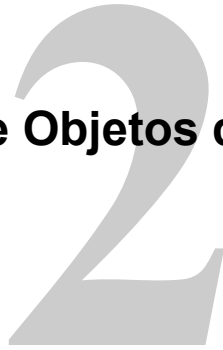
ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Práctica 1: Visión General

Trabaje en equipo con otros estudiantes en este ejercicio sobre el control de acceso a objetos de la base de datos.

Gestión de Objetos de Esquema



ORACLE®

Copyright © 2010, Oracle. Todos los derechos reservados.

Rodrigo Esteves (rodrigo_...@hotmail.com) has a non-transferable license to use this Student Guide.

Objetivos

Al finalizar esta lección, debería estar capacitado para lo siguiente:

- Agregar restricciones
- Crear índices
- Crear índices mediante la sentencia `CREATE TABLE`
- Crear índices basados en funciones
- Borrar columnas y definir columnas como `UNUSED`
- Realizar operaciones `FLASHBACK`
- Crear y utilizar tablas externas

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Objetivos

Esta lección contiene información sobre la creación de índices y restricciones, así como la modificación de objetos existentes. También aprenderá sobre las tablas externas y la provisión para asignar un nombre al índice en el momento de la creación de una restricción `PRIMARY KEY`.

Agenda

- **Uso de la sentencia `ALTER TABLE` para agregar, modificar y borrar una columna**
- **Gestión de restricciones:**
 - Adición y borrado de una restricción
 - Restricciones diferidas
 - Activación y desactivación de una restricción
- **Creación de índices:**
 - Uso de la sentencia `CREATE TABLE`
 - Creación de índices basados en funciones
 - Eliminación de índices
- **Realización de operaciones de flashback**
- **Creación y uso de tablas temporales**
- **Creación y uso de tablas externas**

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Sentencia ALTER TABLE

Utilizar la sentencia ALTER TABLE para:

- Agregar una nueva columna
- Modificar una columna existente
- Definir un valor por defecto para la nueva columna
- Borrar una columna

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Sentencia ALTER TABLE

Después de crear una tabla, puede que necesite cambiar la estructura de la tabla porque ha omitido una columna, se debe cambiar la definición de la columna o necesita eliminar columnas. Para ello, puede utilizar la sentencia ALTER TABLE.

Sentencia ALTER TABLE

Utilizar la sentencia ALTER TABLE para agregar, modificar y borrar columnas:

```
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 [, column] ...);
```

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Sentencia ALTER TABLE (continuación)

Puede agregar columnas a una tabla, modificarlas y borrarlas de la tabla mediante la sentencia ALTER TABLE.

En la sintaxis:

<i>table</i>	es el nombre de la tabla
ADD MODIFY DROP	es el tipo de modificación
<i>column</i>	es el nombre de la columna
<i>datatype</i>	es el tipo de dato y la longitud de la columna
DEFAULT <i>expr</i>	especifica el valor por defecto de una columna

Adición de Columnas

- Puede utilizar la cláusula `ADD` para agregar columnas:

```
ALTER TABLE dept80
ADD      (job_id VARCHAR2(9)) ;
```

```
ALTER TABLE dept80 succeeded.
```

- La nueva columna se convierte en la última:

	EMPLOYEE_ID	LAST_NAME	ANNSAL	HIRE_DATE	JOB_ID
1	145	Russell	14000	01-OCT-96	(null)
2	146	Partners	13500	05-JAN-97	(null)
3	147	Errazuriz	12000	10-MAR-97	(null)
4	148	Cambrault	11000	15-OCT-99	(null)
5	149	Zlotkey	10500	29-JAN-00	(null)

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Instrucciones para Agregar una Columna

- Puede agregar o modificar columnas.
- No puede especificar dónde debe aparecer la columna. La nueva columna se convierte en la última.

El ejemplo de la diapositiva agrega una columna con el nombre `JOB_ID` a la tabla `DEPT80`. La columna `JOB_ID` se convierte en la última columna de la tabla.

Nota: si una tabla ya contiene filas al agregar una columna, la nueva columna será inicialmente nula o utilizará el valor por defecto para todas las filas. Sólo puede agregar una columna `NOT NULL` obligatoria a una tabla que contenga datos en las demás columnas si especifica un valor por defecto. Puede agregar una columna `NOT NULL` a una tabla vacía sin el valor por defecto.

Modificación de Columnas

- Puede cambiar el tipo de dato, tamaño y valor por defecto de una columna.

```
ALTER TABLE dept80
MODIFY      (last_name VARCHAR2 (30) ) ;
```

```
ALTER TABLE dept80 succeeded.
```

- El cambio del valor por defecto sólo afecta a las inserciones posteriores en la tabla.



Copyright © 2010, Oracle. Todos los derechos reservados.

Modificación de Columnas

Para modificar una definición de columna, utilice la sentencia ALTER TABLE con la cláusula MODIFY. La modificación de la columna puede incluir cambios en el tipo de dato, tamaño y valor por defecto de una columna.

Instrucciones

- Puede aumentar el ancho o la precisión de una columna numérica.
- Puede aumentar el ancho de las columnas de caracteres.
- Puede reducir el ancho de una columna si:
 - La columna sólo contiene valores nulos
 - La tabla no tiene filas
 - La disminución del ancho de columna no es inferior a los valores existentes de dicha columna
- Puede cambiar el tipo de dato si la columna sólo contiene valores nulos. La única excepción son las conversiones de CHAR a VARCHAR2, que se pueden realizar con los datos de las columnas.
- Sólo puede convertir una columna CHAR al tipo de dato VARCHAR2 o una columna VARCHAR2 al tipo de dato CHAR si la columna contiene valores nulos o si no cambia el tamaño.
- El cambio del valor por defecto de una columna sólo afecta a las inserciones posteriores en la tabla.

Borrado de Columnas

Utilizar la cláusula `DROP COLUMN` para borrar columnas que ya no son necesarias de la tabla:

```
ALTER TABLE dept80
DROP COLUMN job_id;
```

```
ALTER TABLE dept80 succeeded.
```

	EMPLOYEE_ID	LAST_NAME	ANNSAL	HIRE_DATE
1	145	Russell	14000	01-OCT-96
2	146	Partners	13500	05-JAN-97
3	147	Errazuriz	12000	10-MAR-97
4	148	Cambrault	11000	15-OCT-99
5	149	Zlotkey	10500	29-JAN-00

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Borrado de Columnas

Para borrar una columna de la tabla, utilice la sentencia `ALTER TABLE` con la cláusula `DROP COLUMN`.

Instrucciones

- La columna puede o no contener datos.
- Con la sentencia `ALTER TABLE DROP COLUMN`, sólo se puede borrar una columna cada vez.
- Debe quedar al menos una columna en la tabla después de modificarla.
- Después de borrar una columna, no se puede recuperar.
- Una columna no se puede borrar si forma parte de una restricción o de una clave de índice a menos que se agregue la opción `Cascade`.
- El borrado de una columna puede tardar un rato si tiene muchos valores. En este caso, puede ser mejor definirla para que no se utilice y borrarla cuando haya menos usuarios en el sistema para evitar bloqueos ampliados.

Nota: no se pueden borrar determinadas columnas, como las columnas que forman parte de la clave de partición de una tabla particionada o las columnas que forman parte de la `CLAVE PRIMARIA` de una tabla organizada por índices. Para obtener más información sobre las tablas organizadas por índices y la tabla particionada, consulte *Oracle Database Concepts* (Conceptos de Oracle Database) y *Oracle Database Administrator's Guide* (Guía del Administrador de Oracle Database).

Opción SET UNUSED

- Puede utilizar la opción SET UNUSED para marcar una o más columnas como no utilizadas.
- Puede utilizar la opción DROP UNUSED COLUMNS para eliminar las columnas marcadas como no utilizadas.

```
ALTER TABLE <table name>
SET UNUSED (<column name> [ , <column name>]) ;

○

ALTER TABLE <table name>
SET UNUSED COLUMN <column name> [ , <column name>];

ALTER TABLE <table_name>
DROP UNUSED COLUMNS;
```

ORACLE®

Copyright © 2010, Oracle. Todos los derechos reservados.

Opción SET UNUSED

La opción SET UNUSED marca una o más columnas como no utilizadas para que se puedan borrar cuando la demanda de recursos del sistema sea menor. La especificación de esta cláusula no elimina realmente las columnas de destino de cada fila de la tabla (es decir, no restaura el espacio en disco utilizado por estas columnas). Por lo tanto, el tiempo de respuesta es más rápido que si se ejecutara la cláusula DROP. Las columnas no utilizadas se tratan como si se hubieran borrado, aunque sus datos de columna permanezcan en las filas de la tabla. Después de que una columna se haya marcado como no utilizada, no tendrá acceso a dicha columna. Las consultas SELECT * no recuperarán datos de las columnas marcadas como no utilizadas. Además, los nombres y tipos de columnas marcados como no utilizados no se mostrarán durante la sentencia DESCRIBE y podrá agregar a la tabla una nueva columna con el mismo nombre que la columna no utilizada. La información de SET UNUSED se almacena en la vista del diccionario de datos USER_UNUSED_COL_TABS.

Nota: las instrucciones para definir una columna como UNUSED son similares a las instrucciones para borrar una columna.

Opción SET UNUSED (continuación)**Opción DROP UNUSED COLUMNS**

DROP UNUSED COLUMNS elimina de la tabla todas las columnas marcadas actualmente como no utilizadas. Puede utilizar esta sentencia cuando desee reclamar el espacio en disco adicional de las columnas no utilizadas en la tabla. Si la tabla no contiene columnas no utilizadas, la sentencia no devuelve ningún error.

```
ALTER TABLE dept80
SET UNUSED (last_name);
```

```
ALTER TABLE succeeded
```

```
ALTER TABLE dept80
DROP UNUSED COLUMNS;
```

```
ALTER TABLE succeeded
```


Agenda

- Uso de la sentencia `ALTER TABLE` para agregar, modificar y borrar una columna
- Gestión de restricciones:
 - Adición y borrado de una restricción
 - Restricciones diferidas
 - Activación y desactivación de una restricción
- Creación de índices:
 - Uso de la sentencia `CREATE TABLE`
 - Creación de índices basados en funciones
 - Eliminación de índices
- Realización de operaciones de flashback
- Creación y uso de tablas temporales
- Creación y uso de tablas externas

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Adición de Sintaxis de Restricción

Utilizar la sentencia ALTER TABLE para:

- Agregar o borrar una restricción, pero sin modificar su estructura
- Activar o desactivar restricciones
- Agregar una restricción NOT NULL mediante la cláusula MODIFY

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

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Adición de Restricciones

Puede agregar una restricción para las tablas existentes mediante la sentencia ALTER TABLE con la cláusula ADD.

En la sintaxis:

<i>table</i>	es el nombre de la tabla
<i>constraint</i>	es el nombre de la restricción
<i>type</i>	es el tipo de restricción
<i>column</i>	es el nombre de la columna a la que afecta la restricción

La sintaxis de nombre de las restricciones es opcional, aunque se recomienda. Si no asigna un nombre a las restricciones, el sistema genera nombres de restricciones.

Instrucciones

- Puede agregar, borrar, activar o desactivar una restricción, pero no puede modificar su estructura.
- Puede agregar una restricción NOT NULL a una columna existente mediante la cláusula MODIFY de la sentencia ALTER TABLE.

Nota: sólo puede definir una columna NOT NULL si la tabla está vacía o si la columna tiene un valor para cada fila.

Adición de Restricciones

Agregar una restricción `FOREIGN KEY` a la tabla `EMP2` que indique que un gestor ya debe existir como empleado válido en la tabla `EMP2`.

```
ALTER TABLE emp2  
MODIFY employee_id PRIMARY KEY;
```

```
ALTER TABLE emp2 succeeded.
```

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

```
ALTER TABLE succeeded.
```

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Adición de Restricciones (continuación)

El primer ejemplo de la diapositiva modifica la tabla `EMP2` para agregar una restricción `PRIMARY KEY` a la columna `EMPLOYEE_ID`. Observe que como no se proporciona ningún nombre de restricción, el servidor de Oracle asigna automáticamente un nombre a la restricción. El segundo ejemplo de la diapositiva crea una restricción `FOREIGN KEY` en la tabla `EMP2`. La restricción garantiza que un gestor exista como empleado válido en la tabla `EMP2`.

Cláusula ON DELETE

- Utilizar la cláusula `ON DELETE CASCADE` para suprimir filas secundarias al suprimir una clave principal:

```
ALTER TABLE emp2 ADD CONSTRAINT emp_dt_fk
FOREIGN KEY (Department_id)
REFERENCES departments(department_id) ON DELETE CASCADE;
```

```
ALTER TABLE Emp2 succeeded.
```

- Utilizar la cláusula `ON DELETE SET NULL` para definir el valor de filas secundarias en nulo al suprimir una clave principal:

```
ALTER TABLE emp2 ADD CONSTRAINT emp_dt_fk
FOREIGN KEY (Department_id)
REFERENCES departments(department_id) ON DELETE SET NULL;
```

```
ALTER TABLE Emp2 succeeded.
```



Copyright © 2010, Oracle. Todos los derechos reservados.

ON DELETE

Mediante la cláusula `ON DELETE` puede determinar cómo maneja Oracle Database la integridad referencial si elimina un valor de clave primaria o única al que se hace referencia.

ON DELETE CASCADE

La acción `ON DELETE CASCADE` permite suprimir, pero no actualizar los datos de clave principal a los que se hace referencia desde la tabla secundaria. Al suprimir datos de la clave principal, todas las filas de la tabla secundaria que dependen de los valores de clave principal se suprimen también. Para especificar esta acción de referencial, incluya la opción `ON DELETE CASCADE` en la definición de la restricción `FOREIGN KEY`.

ON DELETE SET NULL

Al suprimir datos de la clave principal, la acción `ON DELETE SET NULL` provoca que las filas de la tabla secundaria que dependen del valor de clave principal suprimido se conviertan en nulas.

Si omite esta cláusula, Oracle no le permitirá suprimir los valores de clave a los que se hace referencia en la tabla principal que tiene filas dependientes en la tabla secundaria.

Restricciones Diferidas

Las restricciones pueden tener los siguientes atributos:

- DEFERRABLE O NOT DEFERRABLE
- INITIALLY DEFERRED O INITIALLY IMMEDIATE

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

Restricción diferida en la creación

```
SET CONSTRAINTS dept2_id_pk IMMEDIATE
```

Cambio de un atributo de restricción específico

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

Cambio de todas las restricciones para una sesión

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Restricciones Diferidas

Puede diferir la comprobación de la validez de las restricciones hasta que termine la transacción. Una restricción es diferida si el sistema no comprueba si se ha satisfecho la restricción, hasta que se envía una sentencia COMMIT. Si se viola una restricción diferida, la base de datos devuelve un error y la transacción no se confirma y se realiza rollback de la misma. Si una restricción es inmediata (no diferida), se comprueba al final de cada sentencia. Si se viola, se realiza rollback de la sentencia inmediatamente. Si una restricción provoca una acción (por ejemplo, DELETE CASCADE), esta acción se realiza siempre como parte de la sentencia que la ha provocado, independientemente de si la restricción es diferida o inmediata. Utilice la sentencia SET CONSTRAINTS para especificar, para una determinada transacción, si una restricción diferible se debe comprobar después de cada sentencia de lenguaje de manipulación de datos (DML) o una vez confirmada la transacción. Para crear restricciones diferibles, debe crear índices no únicos para dicha restricción.

Puede definir restricciones como diferibles o no diferibles y como diferidas inicialmente o inmediatas inicialmente. Estos atributos pueden ser diferentes para cada restricción.

Ejemplo de uso: la política de la compañía determina que el departamento número 40 se debe cambiar a 45. El cambio de la columna DEPARTMENT_ID afecta a los empleados asignados a este departamento. Por lo tanto, convierte las claves PRIMARY KEY y FOREIGN KEY en diferibles y diferidas inicialmente. Debe actualizar tanto la información de departamento como la de empleado y, en el momento de la confirmación, se validan todas las filas.

Diferencia entre INITIALLY DEFERRED y INITIALLY IMMEDIATE

INITIALLY DEFERRED	Espera a comprobar la restricción hasta que termina la transacción
INITIALLY IMMEDIATE	Comprueba la restricción al final de la ejecución de la sentencia

```
CREATE TABLE emp_new_sal (salary NUMBER
CONSTRAINT sal_ck
CHECK (salary > 100)
DEFERRABLE INITIALLY IMMEDIATE,
bonus NUMBER
CONSTRAINT bonus_ck
CHECK (bonus > 0 )
DEFERRABLE INITIALLY DEFERRED );
```

```
create table succeeded.
```

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Diferencia entre INITIALLY DEFERRED y INITIALLY IMMEDIATE

Una restricción definida como diferible se puede especificar como INITIALLY DEFERRED o INITIALLY IMMEDIATE. La cláusula INITIALLY IMMEDIATE es el valor por defecto.

En el ejemplo de la diapositiva:

- La restricción sal_ck se crea como DEFERRABLE INITIALLY IMMEDIATE.
- La restricción bonus_ck se crea como DEFERRABLE INITIALLY DEFERRED.

Después de crear la tabla emp_new_sal como se muestra en la diapositiva, intente insertar valores en la tabla y observar los resultados. Cuando se satisfacen las restricciones sal_ck y bonus_ck, las filas se insertan sin errores.

Ejemplo 1: inserte una fila que viole la restricción sal_ck. En la sentencia CREATE TABLE, sal_ck se especifica como una restricción inmediata inicialmente. Esto significa que la restricción se verifica inmediatamente después de que tanto la sentencia INSERT como el usuario observen un error.

```
INSERT INTO emp_new_sal VALUES (90,5);
```

```
SQL Error: ORA-02290: check constraint (ORA21.SAL_CK) violated
02290. 00000 - "check constraint (%s.%s) violated"
```

Ejemplo 2: inserte una fila que viole la restricción bonus_ck. En la sentencia CREATE TABLE, bonus_ck se especifica como diferible y como diferida inicialmente. Por lo tanto, la restricción no se verifica hasta que el usuario vuelve a CONFIRMAR o definir el estado de la restricción como inmediata.

Diferencia entre INITIALLY DEFERRED y INITIALLY IMMEDIATE (continuación)

```
INSERT INTO emp_new_sal VALUES (110, -1);
```

```
1 rows inserted
```

La fila se inserta correctamente. Sin embargo, observa un error al confirmar la transacción.

```
COMMIT;
```

```
SQL Error: ORA-02091: transaction rolled back
ORA-02290: check constraint (ORA21.BONUS_CK) violated
02091. 00000 - "transaction rolled back"
```

La confirmación falla debido a una violación de la restricción. Por lo tanto, en este punto, la base de datos realiza un rollback de la transacción.

Ejemplo 3: defina el estado DEFERRED para todas las restricciones que se puedan diferir. Tenga en cuenta que también puede definir el estado DEFERRED para una única restricción si es necesario.

```
SET CONSTRAINTS ALL DEFERRED;
```

```
SET CONSTRAINTS succeeded.
```

Ahora, si intenta insertar una fila que viola la restricción sal_ck, la sentencia se ejecuta correctamente.

```
INSERT INTO emp_new_sal VALUES (90, 5);
```

```
1 rows inserted
```

Sin embargo, observa un error al confirmar la transacción. La transacción falla y se realiza un rollback de la misma. Esto se debe a que ambas restricciones se han comprobado después de CONFIRMAR.

```
COMMIT;
```

```
SQL Error: ORA-02091: transaction rolled back
ORA-02290: check constraint (ORA21.SAL_CK) violated
02091. 00000 - "transaction rolled back"
```

Ejemplo 4: defina el estado IMMEDIATE para las dos restricciones que se han definido como DEFERRED en el ejemplo anterior.

```
SET CONSTRAINTS ALL IMMEDIATE;
```

```
SET CONSTRAINTS succeeded.
```

Observa un error si intenta insertar una fila que viola sal_ck o bonus_ck.

```
INSERT INTO emp_new_sal VALUES (110, -1);
```

```
SQL Error: ORA-02290: check constraint (ORA21.BONUS_CK) violated
02290. 00000 - "check constraint (%s.%s) violated"
```

Nota: si crea una tabla sin especificar el nivel de diferida de una restricción, ésta se comprueba inmediatamente al final de cada sentencia. Por ejemplo, con la sentencia CREATE TABLE de la tabla newemp_details, si no especifica el nivel de diferida de la restricción newemp_det_pk, se comprueba inmediatamente.

```
CREATE TABLE newemp_details(emp_id NUMBER, emp_name
VARCHAR2(20),
CONSTRAINT newemp_det_pk PRIMARY KEY(emp_id));
```

Al intentar diferir la restricción newemp_det_pk que no es diferible, recibirá el siguiente error:

```
SET CONSTRAINT newemp_det_pk DEFERRED;
```

```
SQL Error: ORA-02447: cannot defer a constraint that is not deferrable
```

Borrado de Restricciones

- Eliminar la restricción de gestor de la tabla EMP2:

```
ALTER TABLE emp2
DROP CONSTRAINT emp_mgr_fk;
```

```
ALTER TABLE Emp2 succeeded.
```

- Eliminar la restricción PRIMARY KEY en la tabla DEPT2 y borrar la restricción FOREIGN KEY asociada en la columna EMP2.DEPARTMENT_ID:

```
ALTER TABLE dept2
DROP PRIMARY KEY CASCADE;
```

```
ALTER TABLE dept2 succeeded.
```

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Borrado de Restricciones

Para borrar una restricción, puede identificar el nombre de la restricción en las vistas USER_CONSTRAINTS y USER_CONS_COLUMNS del diccionario de datos. A continuación, utilice la sentencia ALTER TABLE con la cláusula DROP. La opción CASCADE de la cláusula DROP provoca que se borren también las restricciones dependientes.

Sintaxis

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

En la sintaxis:

<i>table</i>	es el nombre de la tabla
<i>column</i>	es el nombre de la columna a la que afecta la restricción
<i>constraint</i>	es el nombre de la restricción

Al borrar una restricción de integridad, el servidor ya no aplica esa restricción y deja de estar disponible en el diccionario de datos.

Desactivación de Restricciones

- Ejecutar la cláusula `DISABLE` de la sentencia `ALTER TABLE` para desactivar una restricción de integridad.
- Aplicar la opción `CASCADE` para desactivar las restricciones de integridad dependientes.

```
ALTER TABLE emp2
DISABLE CONSTRAINT emp_dt_fk;
```

```
ALTER TABLE Emp2 succeeded.
```



Copyright © 2010, Oracle. Todos los derechos reservados.

Desactivación de una Restricción

Puede desactivar una restricción sin borrarla o volver a crearla mediante la sentencia `ALTER TABLE` con la cláusula `DISABLE`.

Sintaxis

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

En la sintaxis:

`table` es el nombre de la tabla
`constraint` es el nombre de la restricción

Instrucciones

- Puede utilizar la cláusula `DISABLE` tanto en la sentencia `CREATE TABLE` como en `ALTER TABLE`.
- La cláusula `CASCADE` desactiva las restricciones de integridad dependientes.
- La desactivación de una restricción `UNIQUE` o `PRIMARY KEY` elimina el índice único.

Activación de Restricciones

- Activar una restricción de integridad desactivada actualmente en la definición de tabla mediante la cláusula

```
ALTER TABLE      emp2
ENABLE CONSTRAINT emp_dt_fk;
```

```
ALTER TABLE Emp2 succeeded.
```

- Se crea un índice `UNIQUE` automáticamente si activa una clave `UNIQUE` o una restricción `PRIMARY KEY`.

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Activación de una Restricción

Puede activar una restricción sin borrarla o volver a crearla mediante la sentencia `ALTER TABLE` con la cláusula `ENABLE`.

Sintaxis

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

En la sintaxis:

table es el nombre de la tabla
constraint es el nombre de la restricción

Instrucciones

- Si activa una restricción, se aplica a todos los datos de la tabla. Todos los datos de la tabla deben cumplir la restricción.
- Si activa una clave `UNIQUE` o una restricción `PRIMARY KEY`, se crea un índice `UNIQUE` o `PRIMARY KEY` automáticamente. Si ya existe un índice, lo pueden utilizar estas claves.
- Puede utilizar la cláusula `ENABLE` tanto en la sentencia `CREATE TABLE` como en `ALTER TABLE`.

Activación de una Restricción (continuación)

- La activación de una restricción PRIMARY KEY desactivada con la opción CASCADE no activa ninguna clave FOREIGN KEY dependiente de PRIMARY KEY.
- Para activar una restricción UNIQUE o PRIMARY KEY, debe tener los privilegios necesarios para crear un índice en la tabla.

Restricciones en Cascada

- La cláusula `CASCADE CONSTRAINTS` se utiliza junto con la cláusula `DROP COLUMN`.
- La cláusula `CASCADE CONSTRAINTS` borra todas las restricciones de integridad referencial que hacen referencia a las claves `PRIMARY` y `UNIQUE` definidas en las columnas borradas.
- La cláusula `CASCADE CONSTRAINTS` borra también todas las restricciones de varias columnas definidas en las columnas borradas.

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Restricciones en Cascada

Esta sentencia ilustra el uso de la cláusula `CASCADE CONSTRAINTS`. Supongamos que la cláusula `TEST1` se ha creado de la siguiente forma:

```
CREATE TABLE test1 (
  col1_pk NUMBER PRIMARY KEY,
  col2_fk NUMBER,
  col1 NUMBER,
  col2 NUMBER,
  CONSTRAINT fk_constraint FOREIGN KEY (col2_fk) REFERENCES
    test1,
  CONSTRAINT ck1 CHECK (col1_pk > 0 and col1 > 0),
  CONSTRAINT ck2 CHECK (col2_fk > 0));
```

Se devuelve un error para las siguientes sentencias:

```
ALTER TABLE test1 DROP (col1_pk); —col1_pk es una clave principal.
```

```
La restricción de varias columnas hace referencia a ALTER TABLE test1 DROP (col1);
—col1, ck1.
```

Restricciones en Cascada

Ejemplo:

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

```
ALTER TABLE Emp2 succeeded.
```

```
ALTER TABLE test1
DROP (col1_pk, col2_fk, col1) CASCADE CONSTRAINTS;
```

```
ALTER TABLE test1 succeeded.
```

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Restricciones en Cascada (continuación)

La ejecución de la siguiente sentencia borra la columna `EMPLOYEE_ID`, la restricción `PRIMARY KEY` y cualquier restricción `FOREIGN KEY` que haga referencia a la restricción `PRIMARY KEY` para la tabla `EMP2`:

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

Si las columnas a las que hacen referencia las restricciones definidas en las columnas borradas también se borran, `CASCADE CONSTRAINTS` no es necesario. Por ejemplo, si suponemos que ninguna otra restricción referencial de otras tablas hace referencia a la columna `COL1_PK`, es válido ejecutar la siguiente sentencia sin la cláusula `CASCADE CONSTRAINTS` para la tabla `TEST1` creada en la página anterior:

```
ALTER TABLE test1 DROP (col1_pk, col2_fk, col1);
```

Cambio de Nombre de Columnas de Tabla y Restricciones

Utilizar la cláusula `RENAME COLUMN` de la sentencia `ALTER TABLE` para cambiar el nombre de las columnas de la tabla.

a

```
ALTER TABLE marketing RENAME COLUMN team_id
TO id;
```

```
ALTER TABLE marketing succeeded.
```

Utilizar la cláusula `RENAME CONSTRAINT` de la sentencia `ALTER TABLE` para cambiar el nombre de cualquier restricción existente para la tabla.

b

```
ALTER TABLE marketing RENAME CONSTRAINT mktg_pk
TO new_mktg_pk;
```

```
ALTER TABLE marketing succeeded.
```

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Cambio de Nombre de Columnas de Tabla y Restricciones

Al cambiar el nombre de una columna de la tabla, el nuevo nombre no debe entrar en conflicto con el nombre de cualquier columna existente en la tabla. Puede utilizar cualquier otra cláusula junto con la cláusula `RENAME COLUMN`.

Los ejemplos de la diapositiva utilizan la tabla `marketing` con la CLAVE PRIMARIA `mktg_pk` definida en la columna `id`.

```
CREATE TABLE marketing (team_id NUMBER(10),
                        target VARCHAR2(50),
                        CONSTRAINT mktg_pk PRIMARY KEY(team_id));
```

```
CREATE TABLE succeeded.
```

El ejemplo **a** muestra que se ha cambiado el nombre de la columna `id` de la tabla de `marketing` a `mktg_id`. El ejemplo **b** muestra que el nombre de `mktg_pk` se ha cambiado a `new_mktg_pk`.

Al cambiar el nombre de una restricción existente para una tabla, el nuevo nombre no debe entrar en conflicto con ninguno de los nombres de las restricciones existentes. Puede utilizar la cláusula `RENAME CONSTRAINT` para cambiar el nombre de las restricciones generadas por el sistema.

Agenda

- Uso de la sentencia `ALTER TABLE` para agregar, modificar y borrar una columna
- Gestión de restricciones:
 - Adición y borrado de una restricción
 - Restricciones diferidas
 - Activación y desactivación de una restricción
- Creación de índices:
 - Uso de la sentencia `CREATE TABLE`
 - Creación de índices basados en funciones
 - Eliminación de índices
- Realización de operaciones de flashback
- Creación y uso de tablas temporales
- Creación y uso de tablas externas

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Visión General de Índices

Los índices se crean:

- Automáticamente
 - Creación de PRIMARY KEY
 - Creación de UNIQUE KEY
- Manualmente
 - Sentencia CREATE INDEX
 - Sentencia CREATE TABLE

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Visión General de Índices

Se pueden crear dos tipos de índices. Un tipo es un índice único. El servidor de Oracle crea automáticamente un índice único al definir una columna o un grupo de columnas en la tabla para tener una restricción PRIMARY KEY o UNIQUE. El nombre del índice es el nombre proporcionado a la restricción.

El otro tipo de índice es un índice no único, que un usuario puede crear. Por ejemplo, puede crear un índice para que se utilice una columna FOREIGN KEY en las uniones para mejorar la velocidad de recuperación.

Para crear un índice en una o más columnas, emita la sentencia CREATE INDEX.

Para obtener más información, consulte *Oracle Database SQL Reference (Referencia de SQL de Oracle Database)* para la base de datos 10g u 11g.

Nota: puede crear manualmente un índice único, pero se recomienda crear una restricción ÚNICA, que implícitamente crea el índice único.

CREATE INDEX con la Sentencia 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));
```

CREATE TABLE succeeded.

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

	INDEX_NAME	TABLE_NAME
1	EMP_ID_IDX	NEW_EMP

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

CREATE INDEX con la Sentencia CREATE TABLE

En el ejemplo de la diapositiva, la cláusula CREATE INDEX se utiliza con la sentencia CREATE TABLE para crear un índice PRIMARY KEY explícitamente. Puede asignar un nombre a los índices en el momento de la creación de PRIMARY KEY que sea diferente del nombre de la restricción PRIMARY KEY.

Puede consultar la vista del diccionario de datos USER_INDEXES para obtener más información sobre los índices.

Nota: aprenderá más sobre USER_INDEXES en la lección titulada “Gestión de Objetos con Vistas de Diccionario de Datos”.

El siguiente ejemplo ilustra el comportamiento de la base de datos si no se asigna un nombre al índice explícitamente:

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

CREATE TABLE succeeded.

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

	INDEX_NAME	TABLE_NAME
1	SYS_C0017294	EMP_UNNAMED_INDEX

CREATE INDEX con la Sentencia CREATE TABLE (continuación)

Observe que el servidor de Oracle proporciona un nombre genérico al índice creado para la columna PRIMARY KEY.

También puede utilizar un índice existente para la columna PRIMARY KEY, por ejemplo, si espera una gran carga de datos y desea acelerar la operación. Puede que desee desactivar las restricciones al realizar la carga y, a continuación, activarlas, en cuyo caso, el tener un índice único en PRIMARY KEY seguirá provocando que los datos se verifiquen durante la carga. Por lo tanto, en primer lugar puede crear un índice no único en la columna designada como PRIMARY KEY y, a continuación, crear la columna PRIMARY KEY y especificar que debe utilizar el índice existente. Los siguientes ejemplos ilustran este proceso:

Paso 1: crear la tabla:

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

Paso 2: crear el índice:

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

Paso 3: crear la CLAVE PRIMARIA:

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

Índices Basados en Funciones

- Un índice basado en funciones se basa en expresiones.
- La expresión de índice se genera a partir de las columnas de las tablas, restricciones, funciones SQL y funciones definidas por el usuario.

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

```
CREATE INDEX succeeded.
```

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

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Índices Basados en Funciones

Los índices basados en funciones definidos con las palabras clave `UPPER(column_name)` o `LOWER(column_name)` permiten realizar búsquedas no sensibles a mayúsculas y minúsculas.

Por ejemplo, considere el siguiente índice:

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

Esto facilita el procesamiento de consultas como:

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

El servidor de Oracle sólo utiliza el índice cuando se utiliza esta función concreta en una consulta. Por ejemplo, la siguiente sentencia puede utilizar el índice, pero sin la cláusula `WHERE`, el servidor de Oracle puede realizar una exploración de tabla completa:

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

Nota: el parámetro de inicialización `QUERY_REWRITE_ENABLED` se debe definir en `TRUE` para utilizar un índice basado en funciones.

El servidor de Oracle trata los índices con columnas marcadas como `DESC` como índices basados en funciones. Las columnas marcadas como `DESC` se ordenan en orden descendente.

Eliminación de Índices

- Eliminar un índice del diccionario de datos mediante el comando `DROP INDEX`:

```
DROP INDEX index;
```

- Eliminar el índice `UPPER_DEPT_NAME_IDX` del diccionario de datos:

```
DROP INDEX upper_dept_name_idx;
```

```
DROP INDEX upper_dept_name_idx succeeded.
```

- Para borrar un índice, debe ser el propietario del mismo o tener el privilegio `DROP ANY INDEX`.

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Eliminación de Índices

No puede modificar los índices. Para cambiar un índice, debe borrarlo y volver a crearlo. Eliminar una definición de índice del diccionario de datos mediante la emisión de la sentencia `DROP INDEX`. Para borrar un índice, debe ser el propietario del mismo o tener el privilegio `DROP ANY INDEX`.

En la sintaxis:

index es el nombre del índice.

Nota: si borra una tabla, los índices, restricciones y disparadores se borran automáticamente, pero permanecen las vistas y secuencias.

DROP TABLE ... PURGE

```
DROP TABLE dept80 PURGE ;
```

```
DROP TABLE dept80 succeeded.
```

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

DROP TABLE ... PURGE

Oracle Database proporciona una función para borrar tablas. Cuando se borra una tabla, la base de datos no libera inmediatamente el espacio asociado a la tabla. En su lugar, la base de datos cambia el nombre de la tabla y la coloca en una papelera de reciclaje, de donde se puede recuperar más adelante con la sentencia `FLASHBACK TABLE` si se da cuenta de que la ha borrado por error.

Si desea liberar inmediatamente el espacio asociado a la tabla en el momento de emitir la sentencia `DROP TABLE`, incluya la cláusula `PURGE` como se muestra en la sentencia de la diapositiva.

Especifique `PURGE` sólo si desea borrar la tabla y liberar el espacio asociado a ella en un solo paso. Si especifica `PURGE`, la base de datos no coloca la tabla y sus objetos dependientes en la papelera de reciclaje.

El uso de esta cláusula equivale a borrar la tabla en primer lugar y a depurarla a continuación de la papelera de reciclaje. Esta cláusula ahorra un paso del proceso. También proporciona una seguridad mejorada si desea evitar que aparezca material confidencial en la papelera de reciclaje.

Agenda

- Uso de la sentencia `ALTER TABLE` para agregar, modificar y borrar una columna
- Gestión de restricciones:
 - Adición y borrado de una restricción
 - Restricciones diferidas
 - Activación y desactivación de una restricción
- Creación de índices:
 - Uso de la sentencia `CREATE TABLE`
 - Creación de índices basados en funciones
 - Eliminación de índices
- **Realización de operaciones de flashback**
- Creación y uso de tablas temporales
- Creación y uso de tablas externas

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Sentencia FLASHBACK TABLE

- Permite recuperar tablas a un punto especificado en el tiempo con una única sentencia.
- Restaura los datos de la tabla junto con los índices y las restricciones.
- Permite revertir la tabla y su contenido a un determinado punto en el tiempo o a un número de cambio del sistema (SCN)



ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Sentencia FLASHBACK TABLE

Oracle Flashback Table permite recuperar tablas a un punto especificado en el tiempo con una única sentencia. Puede restaurar los datos de la tabla junto con los índices asociados y las restricciones mientras la base de datos está en línea, deshaciendo los cambios sólo en las tablas especificadas.

La función Flashback Table es similar a la herramienta de reparación de autoservicio. Por ejemplo, si un usuario suprime accidentalmente filas importantes de una tabla y, a continuación, desea recuperar las filas suprimidas, puede utilizar la sentencia FLASHBACK TABLE para restaurar la tabla al momento anterior a la supresión y ver las filas que faltan en la tabla.

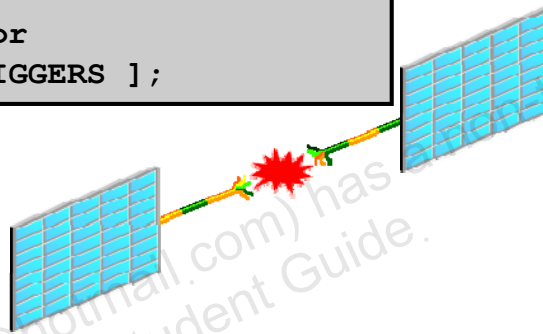
Si utiliza la sentencia FLASHBACK TABLE, puede revertir la tabla y su contenido a un determinado momento o SCN.

Nota: el SCN es un valor entero asociado a cada cambio de la base de datos. Es un número incremental único en la base de datos. Cada vez que confirma una transacción, se registra un nuevo SCN.

Sentencia FLASHBACK TABLE

- Herramienta de reparación de modificaciones accidentales de tabla
 - Restaura una tabla a un punto anterior en el tiempo.
 - Ventajas: facilidad de uso, disponibilidad y rápida ejecución.
 - Se ejecuta in-situ.
- Sintaxis:

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



ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Sentencia FLASHBACK TABLE (continuación)

Utilidad de Reparación de Autoservicio

Oracle Database proporciona un comando de lenguaje de definición de datos (DDL) SQL, **FLASHBACK TABLE**, para restaurar el estado de una tabla a un punto anterior en el tiempo en caso de que se haya suprimido o modificado involuntariamente. El comando **FLASHBACK TABLE** es una herramienta de reparación de autoservicio para restaurar datos de una tabla junto con los atributos asociados, como índices o vistas. Esto se realiza con la base de datos en línea, mediante una operación de rollback de los cambios posteriores efectuados en la tabla determinada. En comparación con los mecanismos tradicionales de recuperación, esta función ofrece ventajas significativas, como facilidad de uso, disponibilidad y mayor rapidez de restauración. Además, libera la sobrecarga del DBA para buscar y restaurar las propiedades específicas de la aplicación. La función de flashback de tabla no aborda la corrupción física provocada por un disco defectuoso.

Sintaxis

Puede llamar a una operación **FLASHBACK TABLE** en una o más tablas, incluso en tablas con diferentes esquemas. Puede especificar el punto en el tiempo al que desea revertir mediante un registro de hora válido. Por defecto, los disparadores de base de datos están desactivados durante la operación de flashback para todas las tablas implicadas. Puede sustituir este comportamiento mediante la especificación de la cláusula **ENABLE TRIGGERS**.

Nota: para obtener más información sobre la papelera de reciclaje y la semántica de flashback, consulte *Oracle Database Administrator's Guide (Guía del Administrador de Oracle Database)* para la base de datos 10g u 11g.

Uso de la Sentencia FLASHBACK TABLE

```
DROP TABLE emp2;
```

```
DROP TABLE emp2 succeeded.
```

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

ORIGINAL_NAME	OPERATION	DROPTIME
EMP2	DROP	2009-05-20:18:00:39

...

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

```
FLASHBACK TABLE succeeded.
```

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Uso de la Sentencia FLASHBACK TABLE

Sintaxis y Ejemplos

El ejemplo restaura la tabla EMP2 a un estado anterior a la sentencia DROP.

La papelera de reciclaje es realmente una tabla de diccionario de datos que contiene información sobre los objetos borrados. Las tablas borradas y los objetos asociados, como índices, restricciones, tablas anidadas, etc., no se eliminan y continúan ocupando espacio. Continúan contando en la cuotas de espacio del usuario hasta que se depuran específicamente de la papelera de reciclaje o hasta que la base de datos los tenga que depurar debido a las restricciones de espacio de tablespace.

Cada usuario se puede considerar propietario de una papelera de reciclaje, porque a menos que un usuario tenga el privilegio SYSDBA, los únicos objetos a los que tiene acceso el usuario en la papelera de reciclaje son aquéllos de los que es propietario. Un usuario puede ver sus objetos de la papelera de reciclaje mediante la siguiente sentencia:

```
SELECT * FROM RECYCLEBIN;
```

Al borrar un usuario, ninguno de los objetos que le pertenecen se colocan en la papelera de reciclaje y ninguno de los objetos que hay en la misma se depura.

Puede depurar la papelera de reciclaje con la siguiente sentencia:

```
PURGE RECYCLEBIN;
```

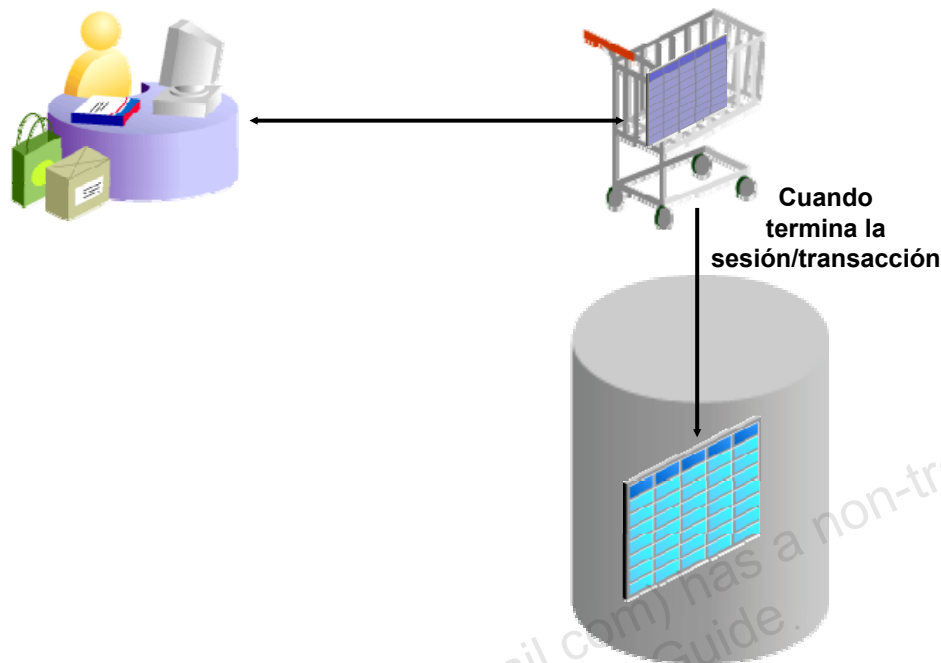
Agenda

- Uso de la sentencia `ALTER TABLE` para agregar, modificar y borrar una columna
- Gestión de restricciones:
 - Adición y borrado de una restricción
 - Restricciones diferidas
 - Activación y desactivación de una restricción
- Creación de índices:
 - Uso de la sentencia `CREATE TABLE`
 - Creación de índices basados en funciones
 - Eliminación de índices
- Realización de operaciones de flashback
- **Creación y uso de tablas temporales**
- Creación y uso de tablas externas

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Tablas Temporales



ORACLE®

Copyright © 2010, Oracle. Todos los derechos reservados.

Tablas Temporales

Una tabla temporal es una tabla que contiene datos que sólo existen durante la transacción o sesión. Los datos de una tabla temporal son privados para la sesión, lo que significa que cada sesión sólo puede ver y modificar sus propios datos.

Las tablas temporales son útiles en las aplicaciones en las que se debe almacenar en buffer un juego de resultados. Por ejemplo, el carro de la compra de una aplicación en línea puede ser una tabla temporal. Cada elemento está representado por una fila en la tabla temporal. Mientras compra en una tienda en línea, puede seguir agregando o eliminando elementos del gráfico. Durante la sesión, los datos de este gráfico son privados. Después de finalizar la compra y realizar los pagos, la aplicación mueve la fila del gráfico seleccionado a una tabla permanente. Al final de la sesión, los datos temporales se borran automáticamente.

Puesto que las tablas temporales se definen estadísticamente, puede crear índices para ellas. Los índices creados en tablas temporales son también temporales. Los datos del índice tienen el mismo ámbito de sesión o transacción que los datos de la tabla temporal. También puede crear una vista o un disparador en una tabla temporal.

Creación de una Tabla Temporal

```
CREATE GLOBAL TEMPORARY TABLE cart
ON COMMIT DELETE ROWS;
```

1

```
CREATE GLOBAL TEMPORARY TABLE today_sales
ON COMMIT PRESERVE ROWS AS
  SELECT * FROM orders
  WHERE order_date = SYSDATE;
```

2

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Creación de una Tabla Temporal

Para crear una tabla temporal, puede utilizar el siguiente comando:

```
CREATE GLOBAL TEMPORARY TABLE tablename
ON COMMIT [PRESERVE | DELETE] ROWS
```

Con la asociación de la siguiente configuración a la cláusula `ON COMMIT`, puede decidir si los datos de la tabla temporal son específicos de la transacción (por defecto) o de la sesión.

1. **DELETE ROWS:** como se muestra en el ejemplo 1 de la diapositiva, el valor `DELETE ROWS` crea una tabla temporal específica de la transacción. Una sesión se enlaza a la tabla temporal con la primera inserción de una transacción en la tabla. El enlace desaparece al final de la transacción. La base de datos trunca la tabla (suprime todas las filas) después de cada confirmación.
2. **PRESERVE ROWS:** como se muestra en el ejemplo 2 de la diapositiva, el valor `PRESERVE ROWS` crea una tabla temporal específica de la sesión. Cada sesión de vendedor puede almacenar sus propios datos de ventas para el día en la tabla. Cuando un vendedor realiza la primera inserción en la tabla `today_sales`, su sesión obtiene un enlace a la tabla `today_sales`. Este enlace desaparece al final de la sesión o mediante la emisión de `TRUNCATE` para la tabla en la sesión. La base de datos se trunca al terminar la sesión.

Al crear una tabla temporal en Oracle Database, se crea una definición de tabla estática. Al igual que las tablas permanentes, las tablas temporales se definen en el diccionario de datos. Sin embargo, las tablas temporales y sus índices no asignan automáticamente un segmento al crearlo. En su lugar, los segmentos temporales se asignan al insertar por primera vez los datos. Hasta que los datos se cargan en una sesión, la tabla aparece vacía.

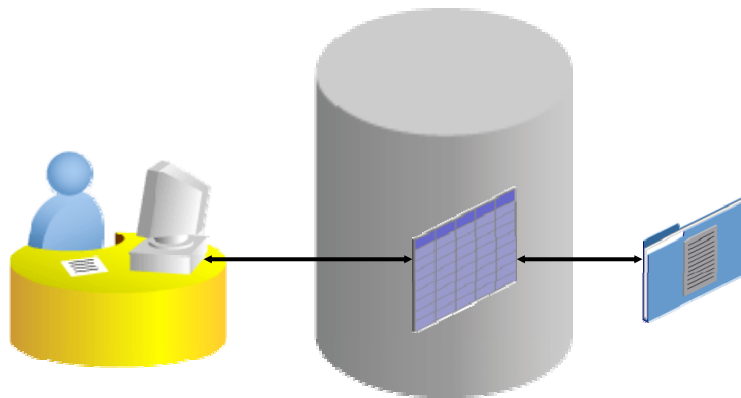
Agenda

- Uso de la sentencia `ALTER TABLE` para agregar, modificar y borrar una columna
- Gestión de restricciones:
 - Adición y borrado de una restricción
 - Restricciones diferidas
 - Activación y desactivación de una restricción
- Creación de índices:
 - Uso de la sentencia `CREATE TABLE`
 - Creación de índices basados en funciones
 - Eliminación de índices
- Realización de operaciones de flashback
- Creación y uso de tablas temporales
- **Creación y uso de tablas externas**

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Tablas Externas



ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Tablas Externas

Una tabla externa es una tabla de sólo lectura cuyos metadatos se almacenan en la base de datos, pero cuyos datos se almacenan fuera de la base de datos. Esta definición de tabla externa se puede considerar como una vista que se utiliza para ejecutar una consulta SQL en datos externos sin necesidad de que se carguen en primer lugar en la base de datos. Los datos de la tabla externa se pueden consultar y unir directamente y en paralelo sin necesidad de cargarlos en primer lugar en la base de datos. Puede utilizar SQL, PL/SQL y Java para consultar los datos de una tabla externa.

La principal diferencia entre las tablas externas y las normales es que las tablas organizadas de forma externa son de sólo lectura. No se puede realizar ninguna operación de lenguaje de manipulación de datos (DML) y no se puede crear ningún índice en ellas. Sin embargo, puede crear una tabla externa y, de este modo, descargar datos mediante el comando `CREATE TABLE AS SELECT`.

El servidor de Oracle Server proporciona dos controladores de acceso principal para las tablas externas. Uno, el controlador de acceso del cargador (o `ORACLE_LOADER`) se utiliza para leer datos de los archivos externos cuyo formato pueda interpretar `SQL*Loader`. Tenga en cuenta que toda la funcionalidad de `SQL*Loader` está soportada con las tablas externas. El controlador de acceso `ORACLE_DATAPUMP` se puede utilizar tanto para importar como para exportar datos con un formato independiente de la plataforma. El controlador de acceso `ORACLE_DATAPUMP` escribe filas de una sentencia `SELECT` para cargarlas en una tabla externa como parte de una sentencia `CREATE TABLE ... ORGANIZATION EXTERNAL ... AS SELECT`. A continuación, puede utilizar `SELECT` para leer los datos de dicho archivo de datos. También puede crear una definición de tabla externa en otro sistema y utilizar ese archivo de datos. Esto permite mover los datos entre diferentes Oracle Database.

Creación de un Directorio para la Tabla Externa

Crear un objeto `DIRECTORY` que corresponda al directorio en el sistema de archivos en el que reside el origen de datos externo.

```
CREATE OR REPLACE DIRECTORY emp_dir  
AS '/.../emp_dir';  
  
GRANT READ ON DIRECTORY emp_dir TO ora_21;
```

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Ejemplo de Creación de una Tabla Externa

Utilice la sentencia `CREATE DIRECTORY` para crear el objeto de directorio. Un objeto de directorio especifica un alias para un directorio del sistema de archivos del servidor en el que reside el origen de datos. Puede utilizar nombres de directorio al hacer referencia al origen de datos externo, en lugar de codificar el nombre de la ruta de acceso del sistema operativo, para una mayor flexibilidad de gestión de archivos.

Debe tener privilegios del sistema `CREATE ANY DIRECTORY` para crear directorios. Cuando crea un directorio, se le otorgan automáticamente los privilegios de objeto `READ` y `WRITE` y puede otorgar los privilegios `READ` y `WRITE` a otros usuarios y roles. El DBA también puede otorgar estos privilegios a otros usuarios y roles.

Un usuario necesita los privilegios `READ` para todos los directorios utilizados en las tablas externas a los que desee acceder y `WRITE` para las ubicaciones de los archivos log, de errores y de desechos que se van a utilizar.

Además, se necesita un privilegio `WRITE` si se utiliza el marco de tabla externa para cargar datos.

Oracle proporciona también el tipo `ORACLE_DATAPUMP`, con el que puede cargar datos (es decir, leer datos de una tabla de la base de datos e insertarla en una tabla externa) y, a continuación, volver a cargarla en Oracle Database. Se trata de una operación única que se puede realizar al crear la tabla. Después de realizar la creación y el relleno inicial, no puede actualizar, insertar o suprimir ninguna fila.

Ejemplo de Creación de una Tabla Externa (continuación)**Sintaxis**

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

En la sintaxis:

OR REPLACE	<p>Especifique OR REPLACE para volver a crear el objeto de base de datos de directorio si ya existe. Puede utilizar esta cláusula para cambiar la definición de un directorio existente sin borrar, volver a crear y volver a otorgar los privilegios de objeto de base de datos otorgados previamente en el directorio. Los usuarios a los que se han otorgado previamente privilegios en un directorio redefinido pueden seguir accediendo al directorio sin necesidad de volver a otorgar los privilegios.</p>
directory	<p>Especifique el nombre del objeto de directorio que desea crear. La longitud máxima del nombre de directorio es de 30 bytes. No puede cualificar un objeto de directorio con un nombre de esquema.</p>
'path_name'	<p>Especifique el nombre de la ruta de acceso completa del directorio del sistema de archivos al que desea acceder. El nombre de la ruta de acceso es sensible a mayúsculas/minúsculas.</p>

Creación de Tablas Externas

```
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 © 2010, Oracle. Todos los derechos reservados.

Creación de Tablas Externas

Puede crear tablas externas mediante la cláusula `ORGANIZATION EXTERNAL` de la sentencia `CREATE TABLE`. En realidad, no está creando una tabla. En su lugar, se crean metadatos en el diccionario de datos que puede utilizar para acceder a los datos externos. Puede utilizar la cláusula `ORGANIZATION` para especificar el orden en el que se deben almacenar las filas de datos de la tabla. Con la especificación `EXTERNAL` en la cláusula `ORGANIZATION`, se indica que la tabla es de sólo lectura y está ubicada fuera de la base de datos. Tenga en cuenta que los archivos externos ya deben existir fuera de la base de datos.

`TYPE <access_driver_type>` indica el controlador de acceso de la tabla externa.

El controlador de acceso es la interfaz de programación de aplicaciones (API) que interpreta los datos externos para la base de datos. Si no especifica `TYPE`, Oracle utiliza el controlador de acceso por defecto, `ORACLE_LOADER`. La otra opción es `ORACLE_DATAPUMP`.

Utilice la cláusula `DEFAULT DIRECTORY` para especificar uno o más objetos de directorio de Oracle Database que correspondan a directorios del sistema de archivos en el que pueden residir los orígenes de datos externos.

La cláusula `ACCESS PARAMETERS` opcional permite asignar valores a los parámetros del controlador de acceso específico para esta tabla externa.

Creación de Tablas Externas (continuación)

Utilice la cláusula `LOCATION` para especificar un localizador externo para cada origen de datos externo. Normalmente, `<location_specifier>` es un archivo, pero no es necesario que lo sea.

La cláusula `REJECT LIMIT` permite especificar el número de errores de conversión que se pueden producir durante una consulta de los datos externos antes de que se devuelva un error de Oracle y que se aborte la consulta. El valor por defecto es 0.

La sintaxis para utilizar el controlador de acceso `ORACLE_DATAPUMP` es la siguiente:

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

Creación de una Tabla Externa mediante 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;
```

CREATE TABLE succeeded.

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Ejemplo de Creación de una tabla Externa mediante el Controlador de Acceso ORACLE_LOADER

Supongamos que hay un archivo plano que contiene registros con el siguiente formato:

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

Los registros están delimitados con nuevas líneas y todos los campos terminan con una coma (,).

El nombre del archivo es /emp_dir/emp.dat.

Para convertir este archivo en el origen de datos de una tabla externa, cuyos metadatos residirán en la base de datos, debe realizar los siguientes pasos:

1. Crear un objeto de directorio, emp_dir, del siguiente modo:
CREATE DIRECTORY emp_dir AS '/emp_dir' ;
2. Ejecutar el comando CREATE TABLE mostrado en la dispositiva.

El ejemplo de la diapositiva ilustra la especificación de tabla para crear una tabla externa para el archivo:

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

Ejemplo de Creación de una tabla Externa mediante el Controlador de Acceso ORACLE_LOADER (continuación)

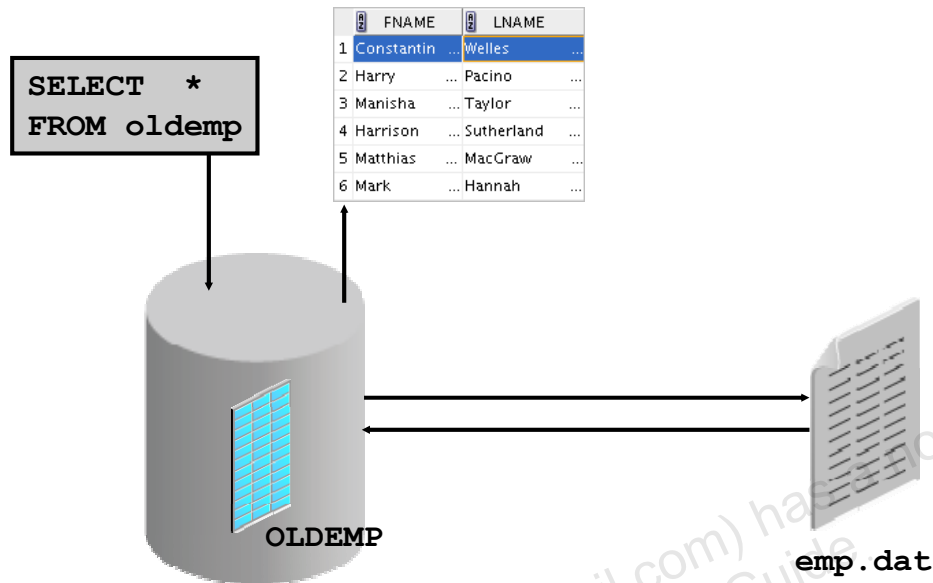
En el ejemplo, la especificación `TYPE` se proporciona sólo para ilustrar su uso. `ORACLE_LOADER` es el controlador de acceso por defecto si no se ha especificado ninguno. La opción `ACCESS PARAMETERS` proporciona valores para los parámetros del controlador de acceso específico, que interpreta el controlador y no el servidor de Oracle.

La cláusula `PARALLEL` permite que cinco servidores de ejecución en paralelo exploren simultáneamente los orígenes de datos externos (archivos) con la ejecución de la sentencia `INSERT INTO TABLE`. Por ejemplo, si se ha especificado `PARALLEL=5`, puede haber más de un servidor de ejecución en paralelo trabajando en un origen de datos. Ya que las tablas externas pueden ser muy grandes, por motivos de rendimiento, es aconsejable especificar la cláusula `PARALLEL` o una indicación paralela para la consulta.

La cláusula `REJECT LIMIT` especifica que si se producen más de 200 errores de conversión durante una consulta de los datos externos, la consulta se abortará y se devolverá un error. Estos errores de conversión se pueden producir cuando el controlador de acceso intenta transformar los datos del archivo de datos para que coincida con la definición de tabla externa.

Después de que el comando `CREATE TABLE` se ejecute correctamente, la tabla externa `OLDEMP` se puede describir y consultar de la misma forma que una tabla relacional.

Consulta de Tablas Externas



ORACLE®

Copyright © 2010, Oracle. Todos los derechos reservados.

Consulta de Tablas Externas

Una tabla externa no describe ningún dato almacenado en la base de datos. No describe cómo se almacenan los datos en el origen externo. En su lugar, describe cómo debe presentar el nivel de tabla externa los datos al servidor. Es responsabilidad del controlador de acceso y del nivel de tabla externa realizar las transformaciones necesarias en los archivos de datos para que coincidan con la definición de tabla externa.

Cuando el servidor de base de datos accede a los datos de un origen externo, llama al controlador de acceso adecuado para obtener los datos de un origen externo del modo en que espera el servidor de base de datos.

Es importante recordar que la descripción de los datos del origen de datos es independiente de la definición de la tabla externa. El archivo de origen puede contener más o menos campos que las columnas que hay en la tabla. Además, los tipos de dato de los campos del origen de datos pueden ser diferentes de las columnas de la tabla. El controlador de acceso se ocupa de garantizar que los datos del origen de datos se procesen de forma que coincidan con la definición de la tabla externa.

Creación de una Tabla Externa mediante ORACLE_DATAPUMP: Ejemplo

```
CREATE TABLE emp_ext
(employee_id, first_name, last_name)
ORGANIZATION EXTERNAL
(
  TYPE ORACLE_DATAPUMP
  DEFAULT DIRECTORY emp_dir
  LOCATION
    ('emp1.exp', 'emp2.exp')
)
PARALLEL
AS
SELECT employee_id, first_name, last_name
FROM employees;
```

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Creación de una Tabla Externa mediante ORACLE_DATAPUMP: ejemplo

Puede utilizar el controlador de acceso ORACLE_DATAPUMP para realizar las operaciones de carga y recarga con tablas externas mediante.

Nota: en el contexto de las tablas externas, la carga de datos se refiere a la lectura de los datos en una tabla externa y su carga en una tabla de la base de datos. La descarga de datos se refiere a la lectura de datos de una tabla y su inserción en una tabla externa.

El ejemplo de la diapositiva ilustra la especificación de tabla para crear una tabla externa mediante el controlador de acceso ORACLE_DATAPUMP: A continuación, los datos se rellenan en dos archivos: emp1.exp and emp2.exp.

Para rellenar los datos leídos de la tabla EMPLOYEES en una tabla externa, debe realizar los siguientes pasos:

1. Crear un objeto de directorio, emp_dir, del siguiente modo:
CREATE DIRECTORY emp_dir AS '/emp_dir' ;
2. Ejecutar el comando CREATE TABLE mostrado en la diapositiva.

Nota: el directorio emp_dir es el mismo que se ha creado en el ejemplo anterior de uso de ORACLE_LOADER.

Puede consultar la tabla externa mediante la ejecución del siguiente código:

```
SELECT * FROM emp_ext;
```

Prueba

Una restricción `FOREIGN KEY` fuerza la siguiente acción:
Al suprimir datos de la clave principal, todas las filas de la tabla secundaria que dependen de los valores de clave principal se suprimen también.

1. Verdadero
2. Falso

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Respuesta: 2

Prueba

En todos los casos, al ejecutar el comando `DROP TABLE`, la base de datos cambia el nombre de la tabla y la coloca en la papelera de reciclaje, de donde se puede recuperar posteriormente mediante la sentencia `FLASHBACK TABLE`.

1. Verdadero
2. Falso

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Respuesta: 2

Resumen

En esta lección, debe haber aprendido lo siguiente:

- Agregar Restricciones
- Crear índices
- Crear índices mediante la sentencia `CREATE TABLE`
- Crear índices basados en funciones
- Borrar columnas y definir columnas como `UNUSED`
- Realizar operaciones `FLASHBACK`
- Crear y utilizar tablas externas

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Resumen

En esta lección, aprenderá a realizar las siguientes tareas para la gestión de objetos de esquema:

- Modificar las tablas para agregar o modificar columnas o restricciones.
- Crear índices e índices basados en funciones mediante la sentencia `CREATE INDEX`.
- Borrar columnas no utilizadas.
- Utilizar mecanismos `FLASHBACK` para restaurar tablas.
- Utilizar la cláusula `ORGANIZATION EXTERNAL` de la sentencia `CREATE TABLE` para crear una tabla externa. Una tabla externa es una tabla de sólo lectura cuyos metadatos se almacenan en la base de datos, pero cuyos datos se almacenan fuera de la base de datos.
- Utilizar tablas externas para consultar datos sin cargarlos antes en la base de datos.
- Asignar nombres a los índices de columna `PRIMARY KEY` al crear la tabla con la sentencia `CREATE TABLE`.

Práctica 2: Visión General

En esta práctica se abordan los siguientes temas:

- Modificación de tablas
- Adición de columnas
- Borrado de columnas
- Creación de índices
- Creación de tablas externas

The Oracle logo, consisting of the word "ORACLE" in a bold, sans-serif font, with a registered trademark symbol (®) to its upper right.

Copyright © 2010, Oracle. Todos los derechos reservados.

Práctica 2: Visión General

En esta práctica, utilizará el comando `ALTER TABLE` para modificar columnas y agregar restricciones. Utilizará el comando `CREATE INDEX` para crear índices al crear una tabla, junto con el comando `CREATE TABLE`. Creará tablas externas.

Gestión de Objetos con Vistas de Diccionario de Datos

ORACLE®

Copyright © 2010, Oracle. Todos los derechos reservados.

Objetivos

Al finalizar esta lección, debería estar capacitado para lo siguiente:

- Utilizar las vistas del diccionario de datos para buscar datos en los objetos
- Consultar varias vistas del diccionario de datos

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Objetivos

En esta lección, se presentarán las vistas del diccionario de datos. Aprenderá que las vistas de diccionario se pueden utilizar para recuperar metadatos y crear informes sobre los objetos de esquema.

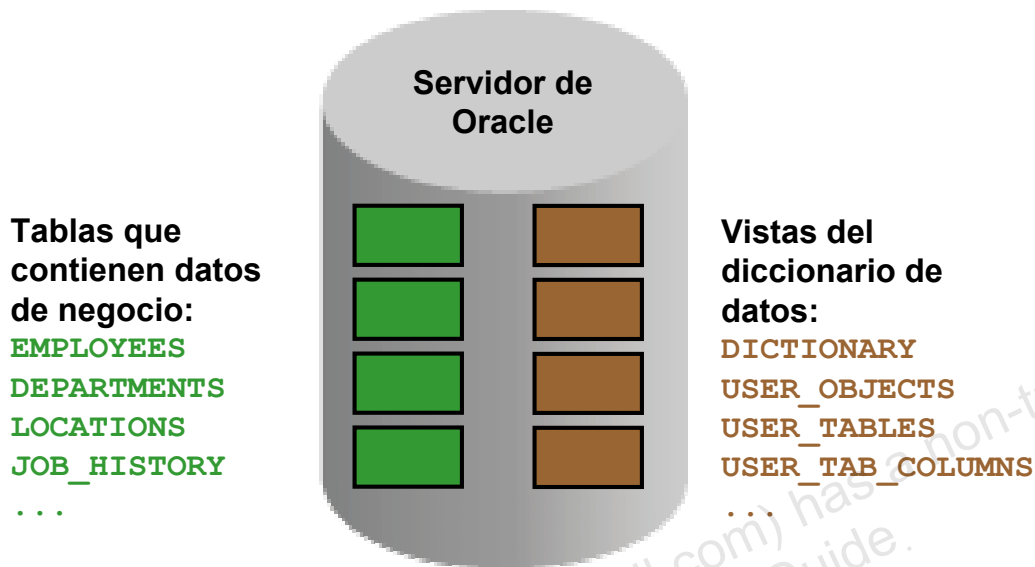
Agenda

- **Introducción al diccionario de datos**
- Consulta de las vistas de diccionario para obtener lo siguiente:
 - Información sobre tablas
 - Información sobre columnas
 - Información sobre restricciones
- Consulta de las vistas de diccionario para obtener lo siguiente:
 - Información sobre vistas
 - Información sobre secuencias
 - Información sobre sinónimos
 - Información sobre índices
- Adición de un comentario a una tabla y consulta de las vistas de diccionario para obtener información sobre comentarios

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Diccionario de Datos



ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Diccionario de Datos

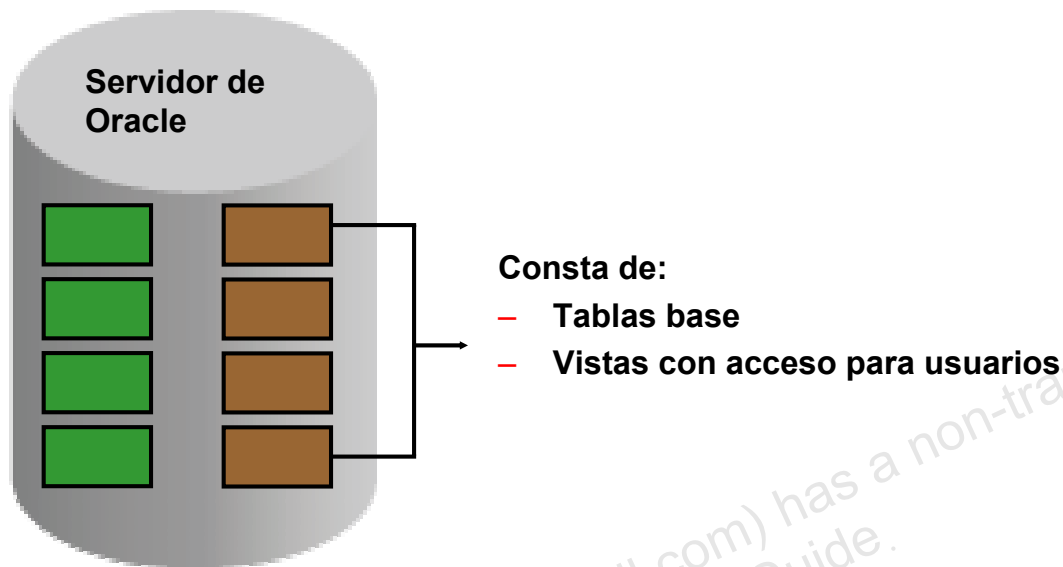
Las tablas de usuario son tablas creadas por el usuario y contienen datos de negocio, como EMPLOYEES. Existe otra recopilación de tablas y vistas en Oracle Database denominada *diccionario de datos*. Esta recopilación la crea y mantiene el servidor de Oracle y contiene información sobre la base de datos. El diccionario de datos está estructurado en tablas y vistas, como otros objetos de la base de datos. El diccionario de datos no sólo es común para todas las Oracle Database, sino que además es una herramienta importante para todos los usuarios, desde los usuarios finales a los diseñadores de aplicaciones y administradores de base de datos.

Puede utilizar sentencias SQL para acceder al diccionario de datos. Puesto que el diccionario de datos es de sólo lectura, sólo puede emitir consultas en sus tablas y vistas.

Puede consultar las vistas del diccionario que se basan en las tablas del diccionario para buscar información como:

- Definiciones de todos los objetos de esquema de la base de datos (tablas, vistas, índices, sinónimos, secuencias, procedimientos, funciones, paquetes, disparadores, etc.)
- Valores por defecto para columnas
- Información sobre restricciones de integridad
- Nombres de usuario de Oracle
- Privilegios y roles otorgados a cada usuario
- Información general de la base de datos adicional

Estructura del Diccionario de Datos



ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Estructura del Diccionario de Datos

Las tablas base subyacentes almacenan información sobre la base de datos asociada. Sólo el servidor de Oracle debe escribir y leer estas tablas. En raras ocasiones accederá a ellas directamente.

Existen varias vistas que resumen y muestran la información almacenada en las tablas base del diccionario de datos. Estas vistas descodifican los datos de la tabla base en información útil (como nombres de usuarios o tablas) mediante uniones y cláusulas `WHERE` para simplificar la unión.

A la mayoría de los usuarios se le proporciona acceso a las vistas en lugar de tablas base.

El usuario de Oracle `SYS` posee todas las tablas base y las vistas con acceso para usuarios del diccionario de datos. Ningún usuario de Oracle debe modificar *nunca* (`UPDATE`, `DELETE`, o `INSERT`) objetos de esquema o filas que estén en el esquema `SYS` porque dicha actividad puede comprometer la integridad de los datos.

Estructura del Diccionario de Datos

Ver reglas de nomenclatura:

Prefijo de Vista	Objetivo
USER	Vista del usuario (contenido del esquema; propiedad del usuario)
ALL	Vista del usuario ampliada (contenido al que puede acceder)
DBA	Vista del administrador de la base de datos (contenido de los esquemas de todos los usuarios)
V\$	Datos relacionados con el rendimiento

ORACLE®

Copyright © 2010, Oracle. Todos los derechos reservados.

Estructura del Diccionario de Datos (continuación)

El diccionario de datos consta de un juego de vistas. En muchos casos, un juego de vistas consta de tres vistas que contienen información similar y que se distingue de las demás por el prefijo.

Por ejemplo, hay una vista denominada USER_OBJECTS, otra ALL_OBJECTS y una tercera DBA_OBJECTS.

Estas tres vistas contienen información similar sobre los objetos de la base de datos, excepto en que el ámbito es diferente. USER_OBJECTS contiene información sobre los objetos que posee o que ha creado. ALL_OBJECTS contiene información sobre todos los objetos a los que tiene acceso.

DBA_OBJECTS contiene información sobre todos los objetos que poseen todos los usuarios. Para las vistas con el prefijo ALL o DBA, suele haber una columna adicional en la vista OWNER para identificar al propietario del objeto.

También hay un juego de vistas con el prefijo V\$. Estas vistas son dinámicas por naturaleza y contienen la información sobre el rendimiento. Las tablas de rendimiento dinámico no son auténticas tablas y la mayoría de los usuarios no pueden acceder a ellas. Sin embargo, los administradores de base de datos pueden consultar y crear vistas en las tablas y otorgar acceso a dichas vistas a otros usuarios. Este curso no profundiza en estas vistas.

Uso de las Vistas de Diccionario

Empezar en `DICTIONARY`. Contiene los nombres y las descripciones de las tablas y vistas del diccionario.

DESCRIBE DICTIONARY

Name	Null	Type
TABLE_NAME		VARCHAR2(30)
COMMENTS		VARCHAR2(4000)
2 rows selected		

```
SELECT *
FROM   dictionary
WHERE  table_name = 'USER_OBJECTS';
```

TABLE_NAME	COMMENTS
1 USER_OBJECTS	Objects owned by the user

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Uso de las Vistas de Diccionario

Para familiarizarse con las vistas del diccionario, puede utilizar la vista denominada `DICTIONARY`. Contiene el nombre y la descripción breve de cada vista del diccionario a la que tiene acceso.

Puede escribir consultas para buscar información sobre un nombre de vista concreto o buscar en la columna `COMMENTS` una palabra o un término. En el ejemplo mostrado, se describe la vista `DICTIONARY`. Tiene dos columnas. La sentencia `SELECT` recupera información sobre la vista del diccionario denominada `USER_OBJECTS`. La vista `USER_OBJECTS` contiene información sobre todos los objetos que posee.

Puede escribir consultas para buscar en la columna `COMMENTS` una palabra o un término.

Por ejemplo, la siguiente consulta devuelve los nombres de todas las vistas a las que tiene acceso en las que la columna `COMMENTS` contiene la palabra *columnas*:

```
SELECT table_name
FROM   dictionary
WHERE  LOWER(comments) LIKE '%columns%';
```

Nota: los nombres del diccionario de datos están en mayúsculas.

Vistas USER_OBJECTS y ALL_OBJECTS

USER_OBJECTS:

- Consulte USER_OBJECTS para ver todos los objetos que posee.
- Mediante USER_OBJECTS, puede obtener una lista de todos los nombres y tipos de objetos del esquema, además de la siguiente información:
 - Fecha de creación
 - Fecha de la última modificación
 - Estado (válido o no válido)

ALL_OBJECTS:

- Consulte ALL_OBJECTS para ver todos los objetos a los que tiene acceso.

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Vistas USER_OBJECTS y ALL_OBJECTS

Puede consultar la vista USER_OBJECTS para ver los nombres y tipos de todos los objetos del esquema. Esta vista tiene varias columnas:

- **OBJECT_NAME:** nombre del objeto
- **OBJECT_ID:** número de objeto de diccionario del objeto
- **OBJECT_TYPE:** tipo de objeto (como TABLE, VIEW, INDEX, SEQUENCE)
- **CREATED:** registro de hora de creación del objeto
- **LAST_DDL_TIME:** registro de hora de la última modificación del objeto resultante de un comando de lenguaje de definición de datos (DDL)
- **STATUS:** estado del objeto (VALID, INVALID o N/A)
- **GENERATED:** ¿ha generado el sistema el nombre de este objeto? (S | N)

Nota: ésta no es una lista completa de las columnas. Para obtener una lista completa, consulte “USER_OBJECTS” en *Oracle Database Reference* (Referencia de Oracle Database).

También puede consultar la vista ALL_OBJECTS para ver una lista de todos los objetos a los que tiene acceso.

Vista USER_OBJECTS

```
SELECT object_name, object_type, created, status
FROM   user_objects
ORDER BY object_type;
```

	OBJECT_NAME	OBJECT_TYPE	CREATED	STATUS
1	LOC_COUNTRY_IX	INDEX	19-MAY-09	VALID

...

53	EMPLOYEES2	TABLE	22-MAY-09	VALID
54	SECURE_EMPLOYEES	TRIGGER	19-MAY-09	VALID
55	UPDATE_JOB_HISTORY	TRIGGER	19-MAY-09	VALID
56	EMP_DETAILS_VIEW	VIEW	19-MAY-09	VALID

...

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Vista USER_OBJECTS

En el ejemplo se muestran los nombres, los tipos, las fechas de creación y los estados de todos los objetos propiedad de este usuario.

La columna OBJECT_TYPE contiene los valores de TABLE, VIEW, SEQUENCE, INDEX, PROCEDURE, FUNCTION, PACKAGE o TRIGGER.

La columna STATUS contiene un valor de VALID, INVALID o N/A. Aunque las tablas son siempre válidas, las vistas, los procedimientos, las funciones, los paquetes y los disparadores pueden no ser válidos.

Vista CAT

Para obtener una consulta y salida simplificada, puede consultar la vista CAT. Esta vista sólo contiene dos columnas: TABLE_NAME y TABLE_TYPE. Proporciona los nombres de todos los objetos INDEX, TABLE, CLUSTER, VIEW, SYNONYM, SEQUENCE o UNDEFINED.

Nota: CAT es un sinónimo para una vista USER_CATALOG que muestra las tablas, vistas, sinónimos y secuencias que posee el usuario.

Agenda

- Introducción al diccionario de datos
- Consulta de las vistas de diccionario para obtener lo siguiente:
 - Información sobre tablas
 - Información sobre columnas
 - Información sobre restricciones
- Consulta de las vistas de diccionario para obtener lo siguiente:
 - Información sobre vistas
 - Información sobre secuencias
 - Información sobre sinónimos
 - Información sobre índices
- Adición de un comentario a una tabla y consulta de las vistas de diccionario para obtener información sobre comentarios

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Información sobre Tablas

USER_TABLES:

```
DESCRIBE user_tables
```

Name	Null	Type
TABLE_NAME	NOT NULL	VARCHAR2(30)
TABLESPACE_NAME		VARCHAR2(30)
CLUSTER_NAME		VARCHAR2(30)
IOT_NAME		VARCHAR2(30)

...

```
SELECT table_name
FROM user_tables;
```

TABLE_NAME
1 REGIONS
2 LOCATIONS
3 DEPARTMENTS
4 JOBS
5 EMPLOYEES
6 JOB_HISTORY

...

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Información sobre Tablas

Puede utilizar la vista USER_TABLES para obtener los nombres de todas las tablas. La vista USER_TABLES contiene información sobre las tablas. Además de proporcionar el nombre de la tabla, contiene información detallada sobre el almacenamiento.

La vista TABS es un sinónimo de la vista USER_TABLES. Puede consultarla para ver una lista de las tablas que posee:

```
SELECT table_name
FROM tabs;
```

Nota: para obtener una lista completa de las columnas de la vista USER_TABLES, consulte “USER_TABLES” en *Oracle Database Reference* (Referencia de Oracle Database).

También puede consultar la vista ALL_TABLES para ver una lista de todas las tablas a las que tiene acceso.

Información sobre Columnas

USER_TAB_COLUMNS:

```
DESCRIBE user_tab_columns
```

Name	Null	Type
TABLE_NAME	NOT NULL	VARCHAR2(30)
COLUMN_NAME	NOT NULL	VARCHAR2(30)
DATA_TYPE		VARCHAR2(106)
DATA_TYPE_MOD		VARCHAR2(3)
DATA_TYPE_OWNER		VARCHAR2(30)
DATA_LENGTH	NOT NULL	NUMBER
DATA_PRECISION		NUMBER
DATA_SCALE		NUMBER
NULLABLE		VARCHAR2(1)

...

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Información sobre Columnas

Puede consultar la vista USER_TAB_COLUMNS para obtener información sobre las columnas de las tablas. Aunque la vista USER_TABLES proporciona información sobre los nombres de tabla y el almacenamiento, en la vista USER_TAB_COLUMNS encontrará información de columna.

Esta vista contiene información como:

- Nombres de columna
- Tipos de dato de columna
- Longitud de los tipos de dato
- Precisión y escala para las columnas NUMBER
- Si se permiten valores nulos (¿Hay alguna restricción NOT NULL en la columna?)
- Valor por defecto

Nota: para obtener una lista completa y descripción de las columnas de la vista USER_TAB_COLUMNS, consulte “USER_TAB_COLUMNS” en *Oracle Database Reference* (Referencia de Oracle Database).

Información sobre Columnas

```
SELECT column_name, data_type, data_length,
       data_precision, data_scale, nullable
FROM   user_tab_columns
WHERE  table_name = 'EMPLOYEES';
```

	COLUMN_NAME	DATA_TYPE	DATA_LENGTH	DATA_PRECISION
1	EMPLOYEE_ID	NUMBER	22	6
2	FIRST_NAME	VARCHAR2	20	(null)
3	LAST_NAME	VARCHAR2	25	(null)
4	EMAIL	VARCHAR2	25	(null)
5	PHONE_NUMBER	VARCHAR2	20	(null)
6	HIRE_DATE	DATE	7	(null)
7	JOB_ID	VARCHAR2	10	(null)
8	SALARY	NUMBER	22	8
9	COMMISSION_PCT	NUMBER	22	2
10	MANAGER_ID	NUMBER	22	6
11	DEPARTMENT_ID	NUMBER	22	4

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Información sobre Columnas (continuación)

Si consulta la tabla USER_TAB_COLUMNS, encontrará detalles sobre las columnas como los nombres, los tipos de dato, las longitudes de tipos de dato, las restricciones nulas y los valores por defecto de una columna.

En el ejemplo anterior se muestran las columnas, los tipos de dato, la longitud de datos y las restricciones nulas de la tabla EMPLOYEES. Tenga en cuenta que esta información es similar a la salida del comando DESCRIBE.

Para ver información sobre el juego de columnas como no utilizadas, puede utilizar la vista de diccionario USER_UNUSED_COL_TABS.

Nota: los nombres de los objetos en el diccionario de datos están en mayúscula.

Información sobre Restricciones

- `USER_CONSTRAINTS` describe las definiciones de restricciones en la tabla.
- `USER_CONS_COLUMNS` describe las columnas que posee y que se han especificado en las restricciones.

DESCRIBE user_constraints

Name	Null	Type
OWNER	NOT NULL	VARCHAR2(30)
CONSTRAINT_NAME	NOT NULL	VARCHAR2(30)
CONSTRAINT_TYPE		VARCHAR2(1)
TABLE_NAME	NOT NULL	VARCHAR2(30)
SEARCH_CONDITION		LONG()
R_OWNER		VARCHAR2(30)
R_CONSTRAINT_NAME		VARCHAR2(30)
DELETE_RULE		VARCHAR2(9)
STATUS		VARCHAR2(8)

...

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Información sobre Restricciones

Puede buscar los nombres de las restricciones, el tipo de restricción, el nombre de la tabla a la que se aplica la restricción, la condición de comprobación de las restricciones, información de restricciones de clave ajena, regla de supresión para las restricciones de clave ajena, el estado y otros muchos tipos de información sobre las restricciones.

Nota: para obtener una lista completa y descripción de las columnas de la vista `USER_CONSTRAINTS`, consulte “`USER_CONSTRAINTS`” en *Oracle Database Reference* (Referencia de Oracle Database).

USER_CONSTRAINTS: Ejemplo

```
SELECT constraint_name, constraint_type,
       search_condition, r_constraint_name,
       delete_rule, status
FROM   user_constraints
WHERE  table_name = 'EMPLOYEES';
```

	CONSTRAINT_NAME	C...	SEARCH_CONDITION	R_CONSTR...	DELET...	STATUS
1	EMP_LAST_NAME_NN	C	"LAST_NAME" IS NOT NULL	(null)	(null)	ENABLED
2	EMP_EMAIL_NN	C	"EMAIL" IS NOT NULL	(null)	(null)	ENABLED
3	EMP_HIRE_DATE_NN	C	"HIRE_DATE" IS NOT NULL	(null)	(null)	ENABLED
4	EMP_JOB_NN	C	"JOB_ID" IS NOT NULL	(null)	(null)	ENABLED
5	EMP_SALARY_MIN	C	salary > 0	(null)	(null)	ENABLED
6	EMP_EMAIL_UK	U	(null)	(null)	(null)	ENABLED
7	EMP_EMP_ID_PK	P	(null)	(null)	(null)	ENABLED
8	EMP_DEPT_FK	R	(null)	DEPT_ID_PK	NO ACTION	ENABLED
9	EMP_JOB_FK	R	(null)	JOB_ID_PK	NO ACTION	ENABLED
10	EMP_MANAGER_FK	R	(null)	EMP_EMP_ID_PK	NO ACTION	ENABLED

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

USER_CONSTRAINTS: Ejemplo

En el ejemplo mostrado, se consulta la vista USER_CONSTRAINTS para buscar los nombres, tipos, condiciones de comprobación, nombre de la restricción única a la que hace referencia la clave ajena, regla de supresión de una clave ajena y estado de las restricciones de la tabla EMPLOYEES.

CONSTRAINT_TYPE puede ser:

- C (restricción de control en una tabla) o NOT NULL)
- P (clave primaria)
- U (clave única)
- R (integridad referencial)
- V (con opción de control, en una vista)
- O (sólo lectura, en una vista)

DELETE_RULE puede ser:

- **CASCADE:** si se suprime el registro principal, se suprimen también los registros secundarios.
- **SET NULL:** si se suprime el registro principal, cambia el registro secundario respectivo a nulo.
- **NO ACTION:** un registro principal sólo se puede suprimir si no existen registros secundarios.

STATUS puede ser:

- **ENABLED:** la restricción está activa.
- **DISABLED:** no se ha activado la restricción.

Consulta de USER_CONS_COLUMNS

```
DESCRIBE user_cons_columns
```

Name	Null	Type
OWNER	NOT NULL	VARCHAR2(30)
CONSTRAINT_NAME	NOT NULL	VARCHAR2(30)
TABLE_NAME	NOT NULL	VARCHAR2(30)
COLUMN_NAME		VARCHAR2(4000)
POSITION		NUMBER

```
SELECT constraint_name, column_name
FROM   user_cons_columns
WHERE  table_name = 'EMPLOYEES';
```

	CONSTRAINT_NAME	COLUMN_NAME
1	EMP_LAST_NAME_NN	LAST_NAME
2	EMP_EMAIL_NN	EMAIL
3	EMP_HIRE_DATE_NN	HIRE_DATE
4	EMP_JOB_NN	JOB_ID
5	EMP_SALARY_MIN	SALARY
6	EMP_EMAIL_UK	EMAIL

...

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Consulta de USER_CONS_COLUMNS

Para buscar los nombres de las columnas a las que se aplica una restricción, consulte la vista de diccionario USER_CONS_COLUMNS. Esta vista indica el nombre del propietario de una restricción, el nombre de la restricción, la tabla en la que se aplica la restricción, los nombres de las columnas con la restricción y la posición original de la columna o el atributo en la definición del objeto.

Nota: una restricción se puede aplicar a más de una columna.

También puede escribir una condición de unión entre USER_CONSTRAINTS y USER_CONS_COLUMNS para crear una salida personalizada de ambas tablas.

Agenda

- Introducción al diccionario de datos
- Consulta de las vistas de diccionario para obtener los siguiente:
 - Información sobre tablas
 - Información sobre columnas
 - Información sobre restricciones
- Consulta de las vistas de diccionario para obtener lo siguiente:
 - Información sobre vistas
 - Información sobre secuencias
 - Información sobre sinónimos
 - Información sobre índices
- Adición de un comentario a una tabla y consulta de las vistas de diccionario para obtener información sobre comentarios

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Información sobre Vistas

1

```
DESCRIBE user_views
```

Name	Null	Type

VIEW_NAME	NOT NULL	VARCHAR2(30)
TEXT_LENGTH		NUMBER
TEXT		LONG()

2

```
SELECT view_name FROM user_views;
```

VIEW_NAME
1 EMP_DETAILS_VIEW

3

```
SELECT text FROM user_views
WHERE view_name = 'EMP_DETAILS_VIEW';
```

TEXT
1 SELECT e.employee_id, e.job_id, e.manager_id, e.department_id, d.location_id, l.co
...
AND c.region_id = r.region_id AND j.job_id = e.job_id WITH READ ONLY

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Información sobre Vistas

Después de crear la vista, puede consultar la vista del diccionario de datos denominada `USER_VIEWS` para ver el nombre y la definición de la vista. El texto de la sentencia `SELECT` que constituye la vista se almacena en una columna `LONG`. La columna `LENGTH` es el número de caracteres de la sentencia `SELECT`. Por defecto, al realizar una selección en una columna `LONG`, sólo se muestran los primeros 80 caracteres del valor de la columna. Para ver más de 80 caracteres en `SQL*Plus`, utilice el comando `SET LONG`:

```
SET LONG 1000
```

En los ejemplos de la diapositiva:

1. Se muestran las columnas `USER_VIEWS`. Tenga en cuenta que ésta es sólo una lista parcial.
2. Se recuperan los nombres de las vistas.
3. Se muestra la sentencia `SELECT` para `EMP_DETAILS_VIEW` desde el diccionario.

Acceso a los Datos a través de Vistas

Al acceder a los datos a través de una vista, el servidor de Oracle realiza las siguientes operaciones:

- Recupera la definición de la vista de la tabla `USER_VIEWS` del diccionario de datos.
- Comprueba los privilegios de acceso de la tabla base de la vista.
- Convierte la consulta de la vista en una operación equivalente en la tabla o tablas base subyacentes. Es decir, se recuperan los datos o se realiza una actualización de las tablas base.

Información sobre Secuencias

DESCRIBE user_sequences

Name	Null	Type
SEQUENCE_NAME	NOT NULL	VARCHAR2(30)
MIN_VALUE		NUMBER
MAX_VALUE		NUMBER
INCREMENT_BY	NOT NULL	NUMBER
CYCLE_FLAG		VARCHAR2(1)
ORDER_FLAG		VARCHAR2(1)
CACHE_SIZE	NOT NULL	NUMBER
LAST_NUMBER	NOT NULL	NUMBER

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Información sobre Secuencias

La vista `USER_SEQUENCES` describe todas las secuencias que posee. Al crear la secuencia, se especifican los criterios almacenados en la vista `USER_SEQUENCES`. Las columnas de esta vista son:

- **SEQUENCE_NAME:** nombre de la secuencia.
- **MIN_VALUE:** valor mínimo de la secuencia.
- **MAX_VALUE:** valor máximo de la secuencia.
- **INCREMENT_BY:** valor de incremento de la secuencia.
- **CYCLE_FLAG:** ¿está la secuencia a punto de alcanzar el límite?
- **ORDER_FLAG:** ¿se han generado los números de secuencia en orden?
- **CACHE_SIZE:** número de números de secuencia en caché.
- **LAST_NUMBER:** último número de secuencia escrito en el disco. Si una secuencia utiliza el almacenamiento en caché, el número escrito en el disco es el último número colocado en la caché de secuencia. Es probable que este número sea mayor que el último número de secuencia utilizado.

Confirmación de Secuencias

- Verificar los valores de secuencia en la tabla de diccionario de datos USER_SEQUENCES.

```
SELECT  sequence_name, min_value, max_value,
        increment_by, last_number
FROM    user_sequences;
```

	SEQUENCE_NAME	MIN_VALUE	MAX_VALUE	INCREMENT_BY	LAST_NUMBER
1	DEPARTMENTS_SEQ	1	9990	10	280
2	EMPLOYEES_SEQ	1	999999999999999...	1	207
3	LOCATIONS_SEQ	1	9900	100	3300

- La columna LAST_NUMBER muestra el siguiente número de secuencia disponible si no se especifica NOCACHE.

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Confirmación de Secuencias

Después de crear la secuencia, se registra en el diccionario de datos. Puesto que una secuencia es un objeto de base de datos, podrá identificarla en la tabla de diccionario de datos USER_OBJECTS.

También puede confirmar la configuración de la secuencia mediante su selección en la vista de diccionario de datos USER_SEQUENCES.

Visualización del Siguiente Valor de Secuencia Disponible sin Aumentarlo

Si la secuencia se ha creado con NOCACHE, es posible ver el siguiente valor de secuencia disponible sin aumentarlo con la consulta de la tabla USER_SEQUENCES.

Información sobre Índices

- `USER_INDEXES` proporciona información sobre los índices.
- `USER_IND_COLUMNS` describe las columnas que comprenden los índices y las columnas de los índices de las tablas.

```
DESCRIBE user_indexes
```

Name	Null	Type
INDEX_NAME	NOT NULL	VARCHAR2(30)
INDEX_TYPE		VARCHAR2(27)
TABLE_OWNER	NOT NULL	VARCHAR2(30)
TABLE_NAME	NOT NULL	VARCHAR2(30)
TABLE_TYPE		VARCHAR2(11)
UNIQUENESS		VARCHAR2(9)

...

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Información sobre Índices

Consulte la vista `USER_INDEXES` para obtener los nombres de los índices, el nombre de tabla en la que se crea el índice y si el índice es único.

Nota: para obtener una lista completa y descripción de las columnas de la vista `USER_INDEXES`, consulte “`USER_INDEXES`” en *Oracle Database Reference (Referencia de Oracle Database)* para 10g u 11g.

USER_INDEXES: Ejemplos

a `SELECT index_name, table_name, uniqueness
FROM user_indexes
WHERE table_name = 'EMPLOYEES';`

ID	INDEX_NAME	ID	TABLE_NAME	ID	UNIQUENESS
1	EMP_EMAIL_UK		EMPLOYEES		UNIQUE
2	EMP_EMP_ID_PK		EMPLOYEES		UNIQUE
3	EMP_DEPARTMENT_IX		EMPLOYEES		NONUNIQUE
4	EMP_JOB_IX		EMPLOYEES		NONUNIQUE
5	EMP_MANAGER_IX		EMPLOYEES		NONUNIQUE
6	EMP_NAME_IX		EMPLOYEES		NONUNIQUE

b `SELECT index_name, table_name
FROM user_indexes
WHERE table_name = 'emp_lib';`

ID	INDEX_NAME	ID	TABLE_NAME
1	SYS_C0011777		EMP_LIB

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

USER_INDEXES: Ejemplo

En el ejemplo de la diapositiva **a**, la vista USER_INDEXES se consulta para obtener el nombre del índice, el nombre de la tabla en la que se crea el índice y si el índice es único.

En el ejemplo de la diapositiva **b**, observe que el servidor de Oracle proporciona un nombre genérico al índice creado para la columna PRIMARY KEY. La tabla EMP_LIB se crea mediante el siguiente código:

```
CREATE TABLE EMP_LIB
  (book_id NUMBER(6) PRIMARY KEY ,
   title VARCHAR2(25),
   category VARCHAR2(20));
```

```
CREATE TABLE succeeded.
```


Consulta USER_IND_COLUMNS

```
DESCRIBE user_ind_columns
```

Name	Null	Type
INDEX_NAME		VARCHAR2(30)
TABLE_NAME		VARCHAR2(30)
COLUMN_NAME		VARCHAR2(4000)
COLUMN_POSITION		NUMBER
COLUMN_LENGTH		NUMBER
CHAR_LENGTH		NUMBER
DESCEND		VARCHAR2(4)

```
SELECT index_name, column_name, table_name
FROM   user_ind_columns
WHERE  index_name = 'lname_idx';
```

	INDEX_NAME	COLUMN_NAME	TABLE_NAME
1	LNAME_IDX	LAST_NAME	EMP_TEST

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Consulta USER_IND_COLUMNS

La vista de diccionario USER_IND_COLUMNS proporciona información, como el nombre del índice, nombre de la tabla con índices, nombre de una columna del índice y la posición de la columna en el índice.

En el ejemplo de la diapositiva, la tabla emp_test y el índice LNAME_IDX se crean mediante el siguiente código:

```
CREATE TABLE emp_test AS SELECT * FROM employees;
CREATE INDEX LNAME_IDX ON emp_test (Last_Name);
```

Información sobre Sinónimos

```
DESCRIBE user_synonyms
```

Name	Null	Type
SYNONYM_NAME	NOT NULL	VARCHAR2(30)
TABLE_OWNER		VARCHAR2(30)
TABLE_NAME	NOT NULL	VARCHAR2(30)
DB_LINK		VARCHAR2(128)

```
SELECT *
FROM user_synonyms;
```

	SYNONYM_NAME	TABLE_OWNER	TABLE_NAME	DB_LINK
1	TEAM2	ORA22	DEPARTMENTS	(null)



Copyright © 2010, Oracle. Todos los derechos reservados.

Información sobre Sinónimos

La vista USER_SYNONYMS del diccionario describe los sinónimos privados (sinónimos que sólo posee el usuario).

Puede consultar esta vista para buscar los sinónimos. Puede consultar ALL_SYNONYMS para buscar el nombre de todos los sinónimos disponibles para el usuario y los objetos a los que se aplican estos sinónimos.

Las columnas de esta vista son:

- **SYNONYM_NAME:** nombre del sinónimo
- **TABLE_OWNER:** propietario del objeto al que hace referencia el sinónimo
- **TABLE_NAME:** nombre de la tabla o vista a la que hace referencia el sinónimo
- **DB_LINK:** nombre de la referencia del enlace de base de datos (si lo hay)

Agenda

- Introducción al diccionario de datos
- Consulta de las vistas de diccionario para obtener los siguiente:
 - Información sobre tablas
 - Información sobre columnas
 - Información sobre restricciones
- Consulta de las vistas de diccionario para obtener lo siguiente:
 - Información sobre vistas
 - Información sobre secuencias
 - Información sobre sinónimos
 - Información sobre índices
- **Adición de un comentario a una tabla y consulta de las vistas de diccionario para obtener información sobre comentarios**

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Adición de Comentarios a una Tabla

- Puede agregar comentarios a una tabla o columna mediante la sentencia `COMMENT`:

```
COMMENT ON TABLE employees
IS 'Employee Information';
```

```
COMMENT ON COLUMN employees.first_name
IS 'First name of the employee';
```

- Los comentarios se pueden visualizar a través de las vistas del diccionario de datos:
 - `ALL_COL_COMMENTS`
 - `USER_COL_COMMENTS`
 - `ALL_TAB_COMMENTS`
 - `USER_TAB_COMMENTS`



Copyright © 2010, Oracle. Todos los derechos reservados.

Adición de Comentarios a una Tabla

Puede agregar un comentario de hasta 4.000 bytes sobre una columna, tabla, vista o instantánea mediante la sentencia `COMMENT`. El comentario se almacena en el diccionario de datos y se puede visualizar en una de las siguientes vistas del diccionario de datos en el columna `COMMENTS`:

- `ALL_COL_COMMENTS`
- `USER_COL_COMMENTS`
- `ALL_TAB_COMMENTS`
- `USER_TAB_COMMENTS`

Sintaxis

```
COMMENT ON {TABLE table | COLUMN table.column}
IS 'text';
```

En la sintaxis:

table es el nombre de la tabla
column es el nombre de la columna de una tabla
text es el texto de un comentario

Para borrar un comentario de la base de datos, defínalo en una cadena vacía (''):

```
COMMENT ON TABLE employees IS '';
```

Prueba

Las vistas de diccionario que se basan en tablas del diccionario contienen información como:

1. Definiciones de todos los objetos de esquema en la base de datos
2. Valores por defecto de las columnas
3. Información sobre restricciones de integridad
4. Privilegios y roles otorgados a cada usuario
5. Todo lo anterior

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Respuesta: 5

Resumen

En esta lección, debe haber aprendido a buscar información sobre los objetos a través de las siguientes vistas de diccionario:

- `DICTIONARY`
- `USER_OBJECTS`
- `USER_TABLES`
- `USER_TAB_COLUMNS`
- `USER_CONSTRAINTS`
- `USER_CONS_COLUMNS`
- `USER_VIEWS`
- `USER_SEQUENCES`
- `USER_INDEXES`
- `USER_SYNONYMS`

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Resumen

En esta lección, aprenderá acerca de algunas de las vistas de diccionario disponibles. Puede utilizar estas vistas de diccionario para buscar información sobre las tablas, las restricciones, las vistas, las secuencias y los sinónimos.

Práctica 3: Visión General

En esta práctica se abordan los siguientes temas:

- Consulta de las vistas de diccionario para obtener información sobre tablas y columnas
- Consulta de las vistas de diccionario para obtener información sobre restricciones
- Consulta de las vistas de diccionario para obtener información sobre vistas
- Consulta de las vistas de diccionario para obtener información sobre secuencias
- Consulta de las vistas de diccionario para obtener información sobre sinónimos
- Consulta de las vistas de diccionario para obtener información sobre índices
- Adición de un comentario a una tabla y consulta de las vistas de diccionario para obtener información sobre comentarios

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Práctica 3: Visión General

En esta práctica, consultará las vistas de diccionario para obtener información sobre el esquema.

4

Manipulación de Juegos de Datos Grandes

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Rodrigo Esteves (rodrigo_@hotmail.com) has a non-transferable license to use this Student Guide.

Objetivos

Al finalizar esta lección, debería estar capacitado para lo siguiente:

- Manipular datos mediante subconsultas
- Especificar valores por defecto explícitos en las sentencias `INSERT` y `UPDATE`
- Describir las funciones de sentencias `INSERTs`
- Utilizar los siguientes tipos sentencias `INSERTs` de varias tablas:
 - `INSERT` incondicional
 - `INSERT` mediante giro
 - `INSERT ALL` condicional
 - `INSERT FIRST` condicional
- Fusión de Filas en una Tabla
- Realizar un seguimiento de los cambios en los datos a lo largo de un período de tiempo

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Objetivos

En esta lección, aprenderá a manipular datos en Oracle Database mediante subconsultas. Aprenderá a utilizar la palabra clave `DEFAULT` en sentencias `INSERT` y `UPDATE` para identificar un valor de columna por defecto. También aprenderá sobre sentencias `INSERT` de varias tablas, la sentencia `MERGE` y el seguimiento de cambios en la base de datos.

Agenda

- Manipulación de datos mediante subconsultas
- Especificación de valores por defecto explícitos en sentencias `INSERT` y `UPDATE`
- Uso de los siguientes tipos de sentencias `INSERTs` de varias tablas:
 - `INSERT` incondicional
 - `INSERT` mediante giro
 - `INSERT ALL` condicional
 - `INSERT FIRST` condicional
- Fusión de Filas en una Tabla
- Seguimiento de los cambios en los datos a lo largo de un período de tiempo

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Uso de Subconsultas para Manipular Datos

Puede utilizar subconsultas en sentencias de lenguaje de manipulación de datos (DML) para:

- Recuperar datos mediante una vista en línea
- Copiar datos de una tabla a otra
- Actualizar datos en tablas según los valores de otra tabla
- Suprimir filas de una tabla según las filas de otra tabla

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Uso de Subconsultas para Manipular Datos

Las subconsultas se pueden utilizar para recuperar datos de una tabla que puede utilizar como entrada para una sentencia `INSERT` en una tabla diferente. De esta forma, puede copiar de forma sencilla grandes volúmenes de datos de una tabla a otra con una única sentencia `SELECT`. Igualmente, puede utilizar las subconsultas para realizar actualizaciones y supresiones masivas, utilizándolas en la cláusula `WHERE` de las sentencias `UPDATE` y `DELETE`. También puede utilizar subconsultas en la cláusula `FROM` de una sentencia `SELECT`. Se denomina una vista en línea.

Nota: ha aprendido cómo actualizar y suprimir filas según otra tabla en el curso titulado *Oracle Database: Conceptos Fundamentales de SQL I*.

Recuperación de Datos mediante una Subconsulta como Origen

```
SELECT department_name, city
FROM departments
NATURAL JOIN (SELECT l.location_id, l.city, l.country_id
FROM loc l
JOIN countries c
ON (l.country_id = c.country_id)
JOIN regions USING(region_id)
WHERE region_name = 'Europe');
```

	DEPARTMENT_NAME	CITY
1	Human Resources	London
2	Sales	Oxford
3	Public Relations	Munich

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Recuperación de Datos mediante una Subconsulta como Origen

Puede utilizar una subconsulta en la cláusula FROM de una sentencia SELECT, que es muy similar a la forma en la que se usan las vistas. Una subconsulta en la cláusula FROM de una sentencia SELECT también se denominada una vista *en línea*. Una subconsulta en la cláusula FROM de una sentencia SELECT define un origen de datos para una sentencia SELECT determinada, sólo para dicha sentencia SELECT. Como ocurre con una vista de base de datos, la sentencia SELECT de la subconsulta puede ser tan simple o compleja como desee.

Al crear una vista de base de datos, la sentencia SELECT se almacena en el diccionario de datos.

En las situaciones en las que no tiene los privilegios necesarios para crear vistas de base de datos o si desea probar la idoneidad de una sentencia SELECT para convertirla en una vista, puede utilizar una vista en línea.

Con las vistas en línea, puede tener todo el código necesario para soportar la consulta en un sólo lugar. Esto significa que puede evitar la complejidad de la creación de una vista de base de datos independiente. El ejemplo de la diapositiva muestra cómo utilizar una vista en línea para mostrar el nombre de departamento y la ciudad en Europa. La subconsulta en la cláusula FROM recupera el ID de ubicación, nombre de ciudad y el país mediante la unión de tres tablas diferentes. La salida de la consulta interna se considera como una tabla para la consulta externa. La consulta interna es similar a la de una vista de base de datos, pero no tiene ningún nombre físico.

En el ejemplo de la diapositiva, la tabla loc se crea mediante la ejecución de la siguiente sentencia:

```
CREATE TABLE loc AS SELECT * FROM locations;
```

Recuperación de Datos mediante una Subconsulta como Origen (continuación)

Puede mostrar la misma salida que en el ejemplo de la diapositiva mediante los siguientes dos pasos:

1. Cree una vista de base de datos:

```
CREATE OR REPLACE VIEW european_cities
AS
SELECT l.location_id, l.city, l.country_id
FROM   loc l
JOIN   countries c
ON(l.country_id = c.country_id)
JOIN regions USING(region_id)
WHERE region_name = 'Europe';
```

2. Una la vista EUROPEAN_CITIES con la tabla DEPARTMENTS:

```
SELECT department_name, city
FROM   departments
NATURAL JOIN european_cities;
```

Nota: ha aprendido cómo crear vistas de base de datos en el curso titulado *Oracle Database: Conceptos Fundamentales de SQL I*.

Inserción mediante una Subconsulta como Destino

```
INSERT INTO (SELECT l.location_id, l.city, l.country_id
              FROM   locations l
              JOIN   countries c
              ON(l.country_id = c.country_id)
              JOIN   regions USING(region_id)
              WHERE  region_name = 'Europe')
VALUES (3300, 'Cardiff', 'UK');
```

1 rows inserted

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Inserción mediante una Subconsulta como Destino

Puede utilizar una subconsulta en lugar del nombre de la tabla en la cláusula INTO de la sentencia INSERT. La lista SELECT de esta subconsulta debe tener el mismo número de columnas que la lista de columnas de la cláusula VALUES. Si la sentencia INSERT no funciona correctamente, se deben seguir las reglas de las columnas de la tabla base. Por ejemplo, no puede introducir un ID de ubicación duplicado u omitir un valor obligatorio de la columna NOT NULL.

Este uso de subconsultas evita tener que crear una vista sólo para realizar una operación con INSERT.

El ejemplo de la diapositiva utiliza una subconsulta en lugar de LOC para crear un registro para una nueva ciudad europea.

Nota: también puede realizar la operación INSERT en la vista EUROPEAN_CITIES mediante el siguiente código:

```
INSERT INTO european_cities
VALUES (3300, 'Cardiff', 'UK');
```

Inserción mediante una Subconsulta como Destino

Comprobar los resultados.

```
SELECT location_id, city, country_id
FROM   loc
```

	LOCATION_ID	CITY	COUNTRY_ID
20	2900	Geneva	CH
21	3000	Bern	CH
22	3100	Utrecht	NL
23	3200	Mexico City	MX
24	3300	Cardiff	UK

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Inserción mediante una Subconsulta como Destino (continuación)

El ejemplo de la diapositiva muestra que la inserción mediante la vista en línea ha creado un nuevo registro en la tabla base LOC.

En el siguiente ejemplo se muestran los resultados de la subconsulta utilizada para identificar la tabla para la sentencia INSERT.

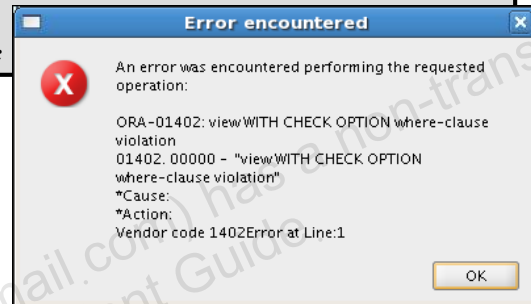
```
SELECT l.location_id, l.city, l.country_id
FROM   loc l
JOIN   countries c
ON(l.country_id = c.country_id)
JOIN   regions USING(region_id)
WHERE  region_name = 'Europe'
```

	LOCATION_ID	CITY	COUNTRY_ID
6	2700	Munich	DE
7	2900	Geneva	CH
8	3000	Bern	CH
9	3100	Utrecht	NL
10	3300	Cardiff	UK

Uso de la Palabra Clave WITH CHECK OPTION en Sentencias DML

La palabra clave WITH CHECK OPTION prohíbe cambiar filas que no estén en la subconsulta.

```
INSERT INTO ( SELECT location_id, city, country_id
              FROM   loc
              WHERE  country_id IN
                    (SELECT country_id
                     FROM countries
                     NATURAL JOIN regions
                     WHERE region name = 'Europe')
              WITH CHECK OPTION )
VALUES (3600, 'Washington', 'US');
```



ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Uso de la Palabra Clave WITH CHECK OPTION en Sentencias DML

Especifique la palabra clave WITH CHECK OPTION para indicar que si se utiliza la subconsulta en lugar de una tabla en una sentencia INSERT, UPDATE o DELETE, ninguno de los cambios producidos por las filas no incluidas en la subconsulta se permitirán en dicha tabla.

El ejemplo de la diapositiva muestra cómo utilizar una vista en línea con WITH CHECK OPTION.

La sentencia INSERT impide la creación de registros en la tabla LOC de una ciudad que no esté en Europa.

El siguiente ejemplo se ejecuta correctamente debido a los cambios en la lista VALUES.

```
INSERT INTO ( SELECT location_id, city, country_id
              FROM   loc
              WHERE  country_id IN
                    (SELECT country_id
                     FROM countries
                     NATURAL JOIN regions
                     WHERE region_name = 'Europe')
              WITH CHECK OPTION )
VALUES (3500, 'Berlin', 'DE');
```

Uso de la Palabra Clave **WITH CHECK OPTION** en Sentencias DML (continuación)

El uso de una vista en línea con **WITH CHECK OPTION** proporciona un método sencillo para evitar cambios en la tabla.

Para evitar la creación de una ciudad que no sea europea, también se puede utilizar una vista de base de datos realizando los siguientes pasos:

1. Cree una vista de base de datos:

```
CREATE OR REPLACE VIEW european_cities
AS
SELECT location_id, city, country_id
FROM   locations
WHERE  country_id in
      (SELECT country_id
       FROM countries
       NATURAL JOIN regions
       WHERE region_name = 'Europe')
WITH CHECK OPTION;
```

2. Verifique los resultados insertando los datos:

```
INSERT INTO european_cities
VALUES (3400, 'New York', 'US');
```

El segundo paso produce el mismo error como muestra la diapositiva.

Agenda

- Manipulación de datos mediante subconsultas
- **Especificación de valores por defecto explícitos en sentencias INSERT y UPDATE**
- Uso de los siguientes tipos de INSERTs de varias tablas:
 - INSERT incondicional
 - INSERT mediante giro
 - INSERT ALL condicional
 - INSERT FIRST condicional
- Fusión de Filas en una Tabla
- Seguimiento de los cambios en los datos a lo largo de un período de tiempo

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Visión General de la Función por Defecto Explícita

- Utilice la palabra clave `DEFAULT` como un valor de columna donde el valor de columna por defecto es el deseado.
- Esto permite al usuario controlar dónde y cuándo se debe aplicar el valor por defecto a los datos.
- Se pueden utilizar valores por defecto explícitos en las sentencias `INSERT` y `UPDATE`.

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Valores por Defecto Explícitos

Se puede utilizar la palabra clave `DEFAULT` en sentencias `INSERT` y `UPDATE` para identificar un valor de columna por defecto. Si no existe ningún valor por defecto, se utiliza un valor nulo.

La opción `DEFAULT` le evita tener que codificar el valor por defecto en los programas o consultar el diccionario para buscarlo, como se hizo antes de que se introdujera esta función.

La codificación del valor por defecto supone un problema si cambia el valor por defecto, ya que el código, por lo tanto, debe cambiar. Normalmente, no se suele realizar el acceso al diccionario en una aplicación; por lo tanto, es una función muy importante.

Uso de Valores por Defecto Explícitos

- DEFAULT con INSERT:

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

- DEFAULT con UPDATE:

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

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Uso de Valores por Defecto Explícitos

Especifique DEFAULT para definir la columna con el valor especificado anteriormente como el valor por defecto para la columna. Si no se ha especificado ningún valor por defecto para la columna correspondiente, el servidor de Oracle define la columna en nula.

En el primer ejemplo de la diapositiva, la sentencia INSERT utiliza un valor por defecto para la columna MANAGER_ID. Si no se ha definido ningún valor por defecto para la columna, en su lugar se inserta un valor nulo.

El segundo ejemplo utiliza la sentencia UPDATE para definir la columna MANAGER_ID en un valor por defecto para el departamento 10. Si no se define ningún valor por defecto para la columna, el valor se cambia a nulo.

Nota: al crear una tabla, puede especificar un valor por defecto para una columna. Esto se tratará en *Conceptos Fundamentales de SQL I*.

Copia de Filas de Otra Tabla

- Escribir la sentencia `INSERT` con una 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%';
```

33 rows inserted

- No utilizar la cláusula `VALUES`.
- Hacer coincidir el número de columnas de la cláusula `INSERT` con el de la subconsulta.

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Copia de Filas de Otra Tabla

Puede utilizar la sentencia `INSERT` para agregar filas a la tabla en la que se derivan los valores de las tablas existentes. En lugar de la cláusula `VALUES`, utilice una subconsulta.

Sintaxis

```
INSERT INTO table [ column (, column) ] subquery;
```

En la sintaxis:

`table` es el nombre de la tabla
`column` es el nombre de la columna de la tabla que se debe rellenar
`subquery` es la subconsulta que devuelve filas en la tabla

El número de columnas y sus tipos de dato de la lista de columnas de la cláusula `INSERT` deben coincidir con el número de valores y sus tipos de dato en la subconsulta. Para crear una copia de la filas de una tabla, utilice `SELECT *` en la subconsulta.

```
INSERT INTO EMPL3
SELECT *
FROM employees;
```

Nota: utilice la cláusula `LOG ERRORS` en la sentencia DML para facilitar que termine la operación DML independientemente de los errores. Oracle escribe los detalles del mensaje de error en una tabla de registro de errores que ha creado. Para obtener más información, consulte *Oracle Database SQL Reference (Referencia de SQL de Oracle Database)* para base de datos 10g u 11g.

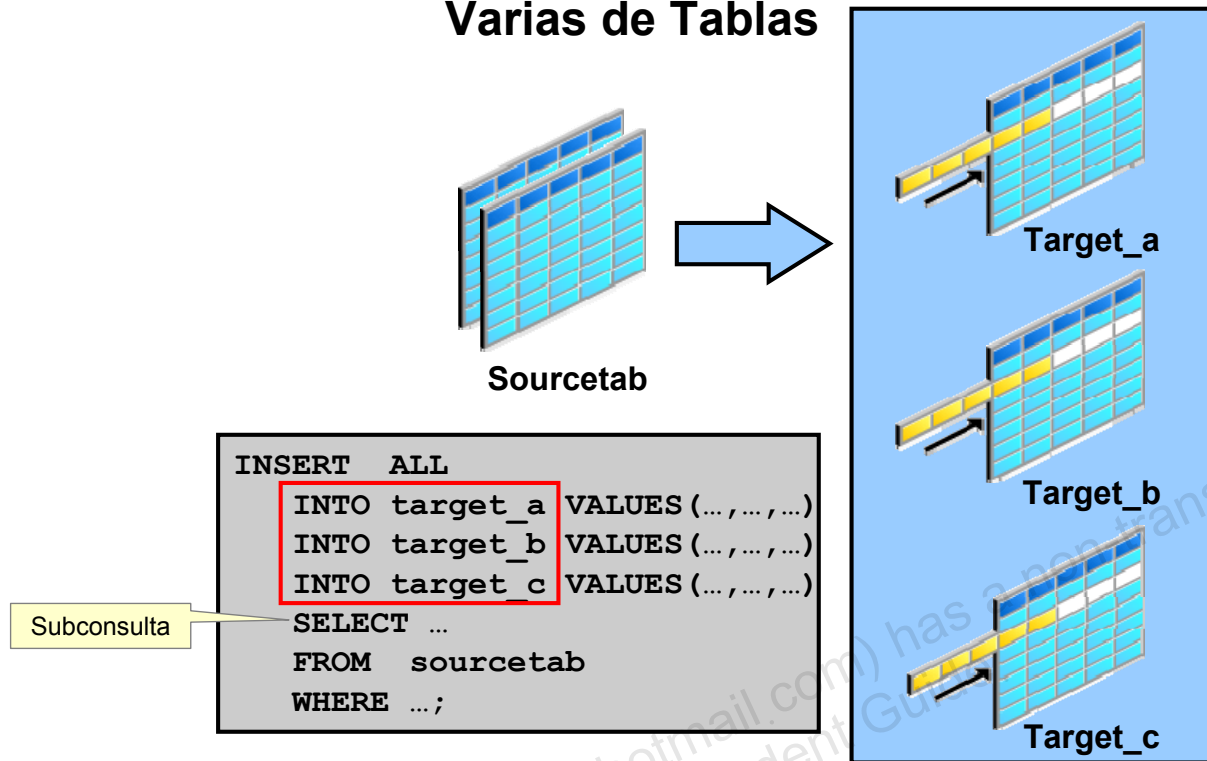
Agenda

- Manipulación de datos mediante subconsultas
- Especificación de valores por defecto explícitos en sentencias `INSERT` y `UPDATE`
- **Uso de los siguientes tipos de `INSERTs` de varias tablas:**
 - `INSERT` incondicional
 - `INSERT` mediante giro
 - `INSERT ALL` condicional
 - `INSERT FIRST` condicional
- Fusión de Filas en una Tabla
- Seguimiento de los cambios en los datos a lo largo de un período de tiempo

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Visión General de Sentencias INSERT de Varias de Tablas



ORACLE®

Copyright © 2010, Oracle. Todos los derechos reservados.

Visión General de Sentencias INSERT de Varias de Tablas

En una sentencia INSERT de varias tablas, inserte las filas calculadas derivadas de las filas devueltas de la evaluación de una subconsulta en una o más tablas.

Las sentencias INSERT de varias tablas son útiles en un escenario de almacén de datos. Debe cargar el almacén de datos regularmente para que pueda cumplir con el propósito de facilitar el análisis de negocio. Para ello, los datos de uno o más sistemas operativos se deben extraer y copiar en el almacén. El proceso de extracción de datos del sistema de origen y colocación en el almacén de datos comúnmente se denomina ETL, que significa extracción, transformación y carga.

Durante la extracción, los datos deseados se deben identificar y extraer de muchos orígenes diferentes, como sistemas de base de datos y aplicaciones. Después de la extracción, los datos se deben transportar físicamente al sistema de destino o a un sistema intermedio para futuros procesamiento. Según los medios de transporte seleccionados, se pueden realizar algunas transformaciones durante este proceso. Por ejemplo, una sentencia SQL que accede directamente a un destino remoto a través de un gateway puede concatenar dos columnas como parte de la sentencia SELECT.

Después de cargar los datos en Oracle Database, se pueden ejecutar las transformaciones de datos mediante operaciones SQL. Una sentencia INSERT de varias tablas es una de las técnicas utilizadas para implantar las transformaciones de datos de SQL.

Visión General de Sentencias INSERT de Varias de Tablas

- También puede utilizar la sentencia `INSERT...SELECT` para insertar filas en varias tablas como parte de una única sentencia DML.
- Las sentencias `INSERT` de varias tablas se utilizan en sistemas de almacenamiento de datos para transferir datos de uno o más orígenes operativos a un juego de tablas de destino.
- Proporcionan una mejora significativa del rendimiento en:
 - Sentencia única DML frente a diversas sentencias `INSERT...SELECT`
 - Una única sentencia DML frente a un procedimiento para realizar varias inserciones mediante la sintaxis `IF . . . THEN`



Copyright © 2010, Oracle. Todos los derechos reservados.

Visión General de Sentencias INSERT de Varias de Tablas (continuación)

Las sentencias `INSERT` de varias tablas ofrecen la ventaja de la sentencia `INSERT . . . SELECT` al llamar varias tablas implicadas como destinos. Sin la sentencia `INSERT` de varias tablas, ha tenía que utilizar n sentencias `INSERT . . . SELECT` independientes, procesando así los mismos datos de origen n veces y aumentando la carga de trabajo de transformación n veces.

Al igual que sucede con la sentencia `INSERT . . . SELECT` existente, la nueva sentencia se puede ejecutar en paralelo y utilizar con el mecanismo de carga directa para obtener un rendimiento más rápido.

Cada registro de flujos de entrada, como una tabla de base de datos no relacional, ahora se puede convertir en varios registros para un entorno de tabla de base de datos más relacional. Asimismo, para implantar esta funcionalidad, debe escribir varias sentencias `INSERT`.

Tipos de Sentencias INSERT de Varias Tablas

Los diferentes tipos de sentencias INSERT de varias tablas son:

- INSERT incondicional
- INSERT ALL condicional
- INSERT mediante giro
- INSERT FIRST condicional



Copyright © 2010, Oracle. Todos los derechos reservados.

Tipos de Sentencias INSERT de Varias Tablas

Utilice las diferentes cláusulas para indicar el tipo de sentencia INSERT que se va a ejecutar.

Los tipos de sentencias INSERT de varias tablas son:

- **INSERT incondicional:** por cada fila devuelta por la subconsulta, se inserta una fila en cada una de las tablas de destino.
- **INSERT ALL condicional:** por cada fila devuelta por la subconsulta, se inserta una fila en cada tabla de destino si se cumple la condición especificada.
- **INSERT mediante giro:** se trata de un caso especial de INSERT ALL incondicional.
- **INSERT FIRST condicional:** por cada fila devuelta por la subconsulta, se inserta una fila en la primera tabla de destino en la que se cumple la condición.

Sentencias INSERT de Varias Tablas

- Sintaxis para INSERT de varias tablas:

```
INSERT [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 © 2010, Oracle. Todos los derechos reservados.

Sentencias INSERT de Varias Tablas

La diapositiva muestra el formato genérico para sentencias INSERT de varias tablas.

INSERT incondicional: ALL into_clause

Especifique ALL seguido de varias insert_into_clauses para realizar una sentencia INSERT de varias tablas incondicional. El servidor de Oracle ejecuta cada insert_into_clause una vez para cada fila devuelta por la subconsulta.

INSERT condicional: conditional_insert_clause

Especifique conditional_insert_clause para realizar una sentencia INSERT de varias tablas condicional. El servidor de Oracle filtra cada insert_into_clause mediante la condición WHEN correspondiente, que determina si dicha insert_into_clause se ejecuta. Una única sentencia INSERT de varias tablas puede contener hasta 127 cláusulas WHEN.

INSERT condicional: ALL

Si especifica ALL, el servidor de Oracle evalúa cada cláusula WHEN independientemente de los resultados de la evaluación de cualquier otra cláusula WHEN. Para cada cláusula WHEN cuya condición se evalúe como True, el servidor de Oracle ejecuta la lista de cláusulas INTO correspondientes.

Sentencias **INSERT** de Varias Tablas (continuación)

INSERT condicional: FIRST

Si especifica **FIRST**, el servidor de Oracle evalúa cada cláusula **WHEN** en el orden en el que aparece en la sentencia. Si la primera cláusula **WHEN** se evalúa como **True**, el servidor de Oracle ejecuta la cláusula **INTO** correspondiente y omite las cláusulas **WHEN** posteriores de la fila determinada.

INSERT condicional: Cláusula ELSE

Para una fila determinada, si ninguna cláusula **WHEN** se evalúa como **True**:

- Si ha especificado una cláusula **ELSE**, el servidor de Oracle ejecuta la lista de cláusulas **INTO** asociadas a la cláusula **ELSE**.
- Si no especifica una cláusula **ELSE**, el servidor de Oracle no realiza ninguna acción para dicha fila.

Restricciones en Sentencias INSERT de Varias Tablas

- Puede realizar sentencias **INSERT** de varias tablas sólo en tablas y no en vistas o vistas materializadas.
- No puede realizar una sentencia **INSERT** de varias tablas en una tabla remota.
- No puede especificar una expresión de recopilación de tablas al realizar una sentencia **INSERT** de varias tablas.
- En una sentencia **INSERT** de varias tablas, todas las cláusulas `insert_into_clauses` no se pueden combinar para especificar más de 999 columnas de destino.

INSERT ALL Incondicional

- Seleccione los valores `EMPLOYEE_ID`, `HIRE_DATE`, `SALARY` y `MANAGER_ID` de la tabla `EMPLOYEES` de aquellos empleados cuyos `EMPLOYEE_ID` sean superiores a 200.
- Inserte estos valores en las tablas `SAL_HISTORY` y `MGR_HISTORY` mediante una sentencia `INSERT` de varias tablas.

```
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;
```

12 rows inserted

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

INSERT ALL Incondicional

En el ejemplo de la diapositiva se insertan filas en las tablas `SAL_HISTORY` y `MGR_HISTORY`.

La sentencia `SELECT` recupera los detalles como el ID de empleado, la fecha de contratación, el salario y el ID de gestor de aquellos empleados cuyo ID de empleado sea superior a 200 de la tabla `EMPLOYEES`. Los detalles como el ID de empleado, la fecha de contratación y el salario se insertan en la tabla `SAL_HISTORY`. Los detalles como el ID de empleado, el ID de gestor y el salario se insertan en la tabla `MGR_HISTORY`.

Esta sentencia `INSERT` se denomina una sentencia `INSERT` incondicional porque no se aplican más restricciones a las filas que las recuperadas por la sentencia `SELECT`. Todas las filas recuperadas por la sentencia `SELECT` se insertan en dos tablas: `SAL_HISTORY` y `MGR_HISTORY`. La cláusula `VALUES` en las sentencias `INSERT` especifica las columnas de la sentencia `SELECT` que se deben insertar en cada una de las tablas. Cada fila devuelta por la sentencia `SELECT` da como resultado dos inserciones: una para la tabla `SAL_HISTORY` y otra para la tabla `MGR_HISTORY`.

INSERT ALL Incondicional (continuación)

Se han seleccionado un total de 12 filas:

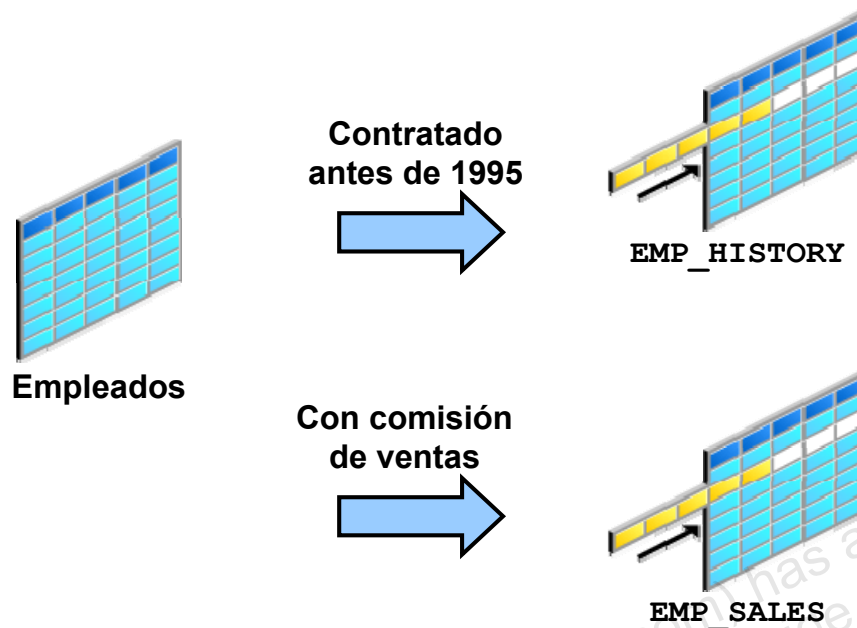
```
SELECT COUNT(*) total_in_sal FROM sal_history;
```

TOTAL_IN_SAL	
1	6

```
SELECT COUNT(*) total_in_mgr FROM mgr_history;
```

TOTAL_IN_MGR	
1	6

INSERT ALL Condicional: Ejemplo



ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

INSERT ALL Condicional: Ejemplo

Para todos los empleados de las tablas de empleados, si los empleados se han contratado antes de 1995, inserte dicho registro de empleado en el historial de empleados. Si el empleado gana una comisión de ventas, inserte la información de registro en la tabla **EMP_SALES**. En la página siguiente se muestra la sentencia SQL.

INSERT ALL Condicional

```

INSERT ALL
  WHEN HIREDATE < '01-JAN-95' THEN
    INTO emp_history VALUES (EMPID, HIREDATE, SAL)
  WHEN COMM IS NOT NULL THEN
    INTO emp_sales VALUES (EMPID, COMM, SAL)
  SELECT employee_id EMPID, hire_date HIREDATE,
         salary SAL, commission_pct COMM
  FROM employees
  
```

48 rows inserted

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

INSERT ALL Condicional

El ejemplo de la diapositiva es similar al que se muestra en la diapositiva anterior porque inserta filas en las tablas EMP_HISTORY EMP_SALES. La sentencia SELECT recupera detalles como ID de empleado, fecha de contratación, salario y porcentaje de comisión de todos los empleados de la tabla EMPLOYEES. Los detalles como el ID de empleado, fecha de contratación y salario se insertan en la tabla EMP_HISTORY. Los detalles como el ID de empleado, fecha de contratación y salario se insertan en la tabla EMP_SALES.

Esta sentencia INSERT se denomina INSERT condicional porque no se aplican más restricciones a las filas que las recuperadas por la sentencia SELECT. De las filas recuperadas por la sentencia SELECT, sólo aquellas filas en las que la fecha de contratación sea anterior a 1995 se insertan en la tabla EMP_HISTORY. Del mismo modo, sólo aquellas filas en las que el valor del porcentaje de comisión no sea nulo se insertan en la tabla EMP_SALES.

```
SELECT count(*) FROM emp_history;
```

	COUNT(*)
1	13

```
SELECT count(*) FROM emp_sales;
```

	COUNT(*)
1	35

INSERT ALL Condicional (continuación)

También puede utilizar la cláusula ELSE con la sentencia INSERT ALL.

Ejemplo:

```
INSERT ALL
  WHEN job_id IN
    (select job_id FROM jobs WHERE job_title LIKE '%Manager%')
  THEN
    INTO managers2(last_name,job_id,SALARY)
  VALUES (last_name,job_id,SALARY)
  WHEN SALARY>10000 THEN
    INTO richpeople(last_name,job_id,SALARY)
  VALUES (last_name,job_id,SALARY)
  ELSE
    INTO poorpeople VALUES (last_name,job_id,SALARY)
  SELECT * FROM employees;
```

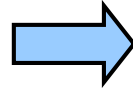
Resultado:

116 rows inserted

INSERT FIRST Condicional: Ejemplo

Supuesto: si el salario de un empleado es 2.000, el registro se inserta sólo en la tabla SAL_LOW.

Salario > 5.000



SAL_LOW

EMPLOYEES

5000 <= Salario
<= 10.000



SAL_MID

De lo contrario



SAL_HIGH

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

INSERT FIRST Condicional: Ejemplo

Para todos los empleados en la tabla EMPLOYEES, inserte la información de empleado en la primera tabla de destino que cumpla la condición. En el ejemplo, si un empleado tiene un salario de 2.000, el registro se inserta sólo en la tabla SAL_LOW. En la página siguiente se muestra la sentencia SQL.

INSERT FIRST Condicional

```

INSERT FIRST
  WHEN salary < 5000 THEN
    INTO sal_low VALUES (employee_id, last_name, salary)
  WHEN salary between 5000 and 10000 THEN
    INTO sal_mid VALUES (employee_id, last_name, salary)
  ELSE
    INTO sal_high VALUES (employee_id, last_name, salary)
SELECT employee_id, last_name, salary
FROM employees

```

107 rows inserted

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

INSERT FIRST Condicional

La sentencia SELECT recupera los detalles como ID de empleado, apellido y salario de cada empleado en la tabla EMPLOYEES. Para cada registro de empleado, se inserta en la primera tabla de destino que cumpla la condición.

Esta sentencia INSERT se denomina INSERT FIRST condicional. La condición WHEN salary < 5000 se evalúa en primer lugar. Si la primera cláusula WHEN se evalúa como True, el servidor de Oracle ejecuta la cláusula INTO correspondiente e inserta el registro en la tabla SAL_LOW. Omite las cláusulas WHEN posteriores para esta fila.

Si la fila no cumple la primera condición WHEN (WHEN salary < 5000), se evalúa la siguiente condición (WHEN salary between 5000 and 10000). Si esta condición se evalúa como True, el registro se inserta en la tabla SAL_MID y se omite la última condición.

Si no se evalúan como True la primera condición (WHEN salary < 5000) ni la segunda (WHEN salary between 5000 and 10000), el servidor de Oracle ejecuta la cláusula INTO correspondiente para la cláusula ELSE.

INSERT FIRST Condicional (continuación)

Se han insertado un total de 20 filas:

```
SELECT count(*) low FROM sal_low;
```

	A Z	LOW
1		49

```
SELECT count(*) mid FROM sal_mid;
```

	A Z	MID
1		43

```
SELECT count(*) high FROM sal_high;
```

	A Z	HIGH
1		15

INSERT mediante giro

Convertir el juego de registros de ventas de la tabla de base de datos no relacional a un formato relacional.

Emp_ID	Week_ID	MON	TUES	WED	THUR	FRI
176	6	2000	3000	4000	5000	6000



Employee_ID	WEEK	SALES
176	6	2000
176	6	3000
176	6	4000
176	6	5000
176	6	6000

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

INSERT mediante giro

Giro es una operación en la que debe crear una transformación de forma que cada registro de cualquier flujo de entrada, como una tabla de base datos no relacional, se deba convertir en varios registros para un entorno de tabla de base de datos más relacional.

Suponga que recibe un juego de registros de ventas de una tabla de base de datos no relacional:

SALES_SOURCE_DATA, en el siguiente formato:

EMPLOYEE_ID, WEEK_ID, SALES_MON, SALES_TUE, SALES_WED,
SALES_THUR, SALES_FRI

Desea almacenar estos registros en la tabla SALES_INFO con un formato relacional más típico:

EMPLOYEE_ID, WEEK, SALES

Para resolver este problema, debe crear una transformación de forma que cada registro de la tabla de base de datos no relacional original, SALES_SOURCE_DATA, se convierta en cinco registros para la tabla SALES_INFO del almacén. Esta operación normalmente se denomina *giro*.

En la página siguiente se muestra la solución a este problema.

INSERT mediante giro

```

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 inserted

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

INSERT mediante giro (continuación)

En el ejemplo de la diapositiva, los datos de ventas se reciben de la tabla de base de datos no relacional SALES_SOURCE_DATA, que son los detalles de las ventas realizadas por un vendedor cada día de una semana, durante una semana con un ID de semana determinado.

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 mediante giro (continuación)

```
SELECT * FROM SALES_SOURCE_DATA;
```

	EMPLOYEE_ID	WEEK_ID	SALES_MON	SALES_TUE	SALES_WED	SALES_THUR	SALES_FRI
1	178	6	1750	2200	1500	1500	3000

```
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
1	178	6	1750
2	178	6	2200
3	178	6	1500
4	178	6	1500
5	178	6	3000

Observe que en el ejemplo anterior al utilizar INSERT mediante giro, una fila de la tabla SALES_SOURCE_DATA se convierte en cinco registros de la tabla relacional, SALES_INFO.

Unauthorized reproduction or distribution prohibited. Copyright© 2014, Oracle and/or its affiliates.

Agenda

- Manipulación de datos mediante subconsultas
- Especificación de valores por defecto explícitos en sentencias `INSERT` y `UPDATE`
- Uso de los siguientes tipos de sentencias `INSERTs` de varias tablas:
 - `INSERT` incondicional
 - `INSERT` mediante giro
 - `INSERT ALL` condicional
 - `INSERT FIRST` condicional
- **Fusión de Filas en una Tabla**
- Seguimiento de los cambios en los datos a lo largo de un período de tiempo

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Sentencia MERGE

- Proporciona la capacidad de actualizar, insertar o suprimir datos de forma condicional en una tabla de base de datos.
- Realiza una sentencia `UPDATE` si la fila existe y una sentencia `INSERT` si es una fila nueva:
 - Evita actualizaciones independientes
 - Aumenta el rendimiento y es fácil de usar
 - Es útil en aplicaciones de almacenamiento de datos

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Sentencia MERGE

El servidor de Oracle soporta la sentencia `MERGE` para operaciones `INSERT`, `UPDATE` y `DELETE`. Mediante esta sentencia, puede actualizar, insertar o suprimir una fila de forma condicional en una tabla y, por lo tanto, evitar varias sentencias DML. La decisión sobre si se actualiza, se inserta o se suprime en la tabla destino se basa en una condición de la cláusula `ON`.

Debe tener los privilegios de objeto `INSERT` y `UPDATE` en la tabla de destino y el privilegio de objeto `SELECT` en la tabla de origen. Para especificar la cláusula `DELETE` de `merge_update_clause`, además debe tener el privilegio de objeto `DELETE` en la tabla de destino.

La sentencia `MERGE` es determinista. No puede actualizar la misma fila de la tabla de destino varias veces en la misma sentencia `MERGE`.

Un enfoque alternativo es utilizar bucles PL/SQL y varias sentencias DML. La sentencia `MERGE`, sin embargo, es fácil de utilizar y se expresa de una forma más simple como una única sentencia SQL.

La sentencia `MERGE` es adecuada en diferentes aplicaciones de almacenamiento de datos.

Por ejemplo, en una aplicación de almacenamiento de datos, puede que necesite trabajar con datos que provengan de varios orígenes, algunos de ellos pueden estar duplicados. Con la sentencia `MERGE`, puede agregar o modificar filas de forma condicional.

Sintaxis de Sentencias MERGE

Puede insertar, actualizar, suprimir filas de forma condicional en una tabla mediante la sentencia MERGE.

```
MERGE INTO table_name table_alias
  USING (table|view|sub_query) alias
  ON (join condition)
  WHEN MATCHED THEN
    UPDATE SET
      col1 = col1_val,
      col2 = col2_val
  WHEN NOT MATCHED THEN
    INSERT (column_list)
    VALUES (column_values);
```

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Fusión de Filas

Puede actualizar filas existentes e insertar filas nuevas de forma condicional mediante la sentencia MERGE. Mediante la sentencia MERGE, puede suprimir filas obsoletas al mismo tiempo que actualiza las filas de una tabla. Para ello, incluya una cláusula DELETE con su propia cláusula WHERE en la sintaxis de la sentencia MERGE.

En la sintaxis:

Cláusula INTO	especifica la tabla de destino que está actualizando o en la que está insertando
Cláusula USING	identifica el origen de los datos que se van a actualizar o insertar; puede ser una tabla, vista o subconsulta
Cláusula ON	la condición en la que la operación MERGE actualiza o inserta
WHEN MATCHED	indica al servidor cómo responder a los resultados de la condición de unión
WHEN NOT MATCHED	

Nota: para obtener más información, consulte *Oracle Database SQL Reference (Referencia de SQL de Oracle Database)* para base de datos 10g u 11g.

Fusión de Filas: Ejemplo

Insertar o actualizar filas en la tabla `COPY_EMP3` para que coincidan con la tabla `EMPLOYEES`.

```

MERGE INTO copy_emp3 c
USING (SELECT * FROM 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,
...
DELETE WHERE (E.COMMISSION_PCT IS NOT NULL)
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 © 2010, Oracle. Todos los derechos reservados.

Fusión de Filas: Ejemplo

```

MERGE INTO copy_emp3 c
USING (SELECT * FROM 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*2,
c.commission_pct  = e.commission_pct,
c.manager_id      = e.manager_id,
c.department_id   = e.department_id
DELETE WHERE (E.COMMISSION_PCT IS NOT NULL)
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);

```

Fusión de Filas: Ejemplo (continuación)

La tabla COPY_EMP3 se crea mediante el siguiente código:

```
CREATE TABLE COPY_EMP3 AS SELECT * FROM EMPLOYEES
WHERE SALARY < 10000;
```

A continuación, consulte la tabla COPY_EMP3.

```
SELECT employee_id, salary, commission_pct FROM COPY_EMP3;
```

	EMPLOYEE_ID	SALARY	COMMISSION_PCT
1	198	5200	(null)
2	199	5200	(null)
3	200	8800	(null)
4	202	12000	(null)
5	203	13000	(null)

...

64	197	6000	(null)
65	162	10500	0.25
66	146	13500	0.3
67	150	10000	0.3

...

Observe que hay algunos empleados con SALARY < 10000 y hay dos empleados con COMMISSION_PCT.

El ejemplo de la diapositiva coincide el EMPLOYEE_ID en la tabla COPY_EMP3 con el EMPLOYEE_ID de la tabla EMPLOYEES. Si se encuentra una coincidencia, la fila de la tabla COPY_EMP3 se actualiza para que la fila coincida en la tabla EMPLOYEES y el salario de empleado se dobla. Se suprimen los registros de los dos empleados con valores en la columna COMMISSION_PCT. Si no se encuentra ninguna coincidencia, se insertan las filas en la tabla COPY_EMP3.

Fusión de Filas: Ejemplo

```
TRUNCATE TABLE copy_emp3;
SELECT * FROM copy_emp3;
0 rows selected
```

```
MERGE INTO copy_emp3 c
USING (SELECT * FROM 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,
...
DELETE WHERE (E.COMMISSION_PCT IS NOT NULL)
WHEN NOT MATCHED THEN
INSERT VALUES (e.employee_id, e.first_name, ...)
```

```
SELECT * FROM copy_emp3;
107 rows selected.
```

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Fusión de Filas: Ejemplo (continuación)

Los ejemplos de la diapositiva muestran que la tabla COPY_EMP3 está vacía. Se evalúa la condición `c.employee_id = e.employee_id`. La condición devuelve false, no hay coincidencias.

La lógica está en la cláusula `WHEN NOT MATCHED`, y el comando `MERGE` inserta las filas de la tabla `EMPLOYEES` en la tabla `COPY_EMP3`. Esto significa que la tabla `COPY_EMP3` ahora tiene exactamente los mismos datos que la tabla `EMPLOYEES`.

```
SELECT employee_id, salary, commission_pct from copy_emp3;
```

R	EMPLOYEE_ID	R	SALARY	R	COMMISSION_PCT
1	144		2500		(null)
2	143		2600		(null)
3	202		6000		(null)
4	141		3500		(null)
5	174		11000		0.3
...					
15	149		10500		0.2
16	206		8300		(null)
17	176		8600		0.2
18	124		5800		(null)
19	205		12000		(null)
20	178		7000		0.15

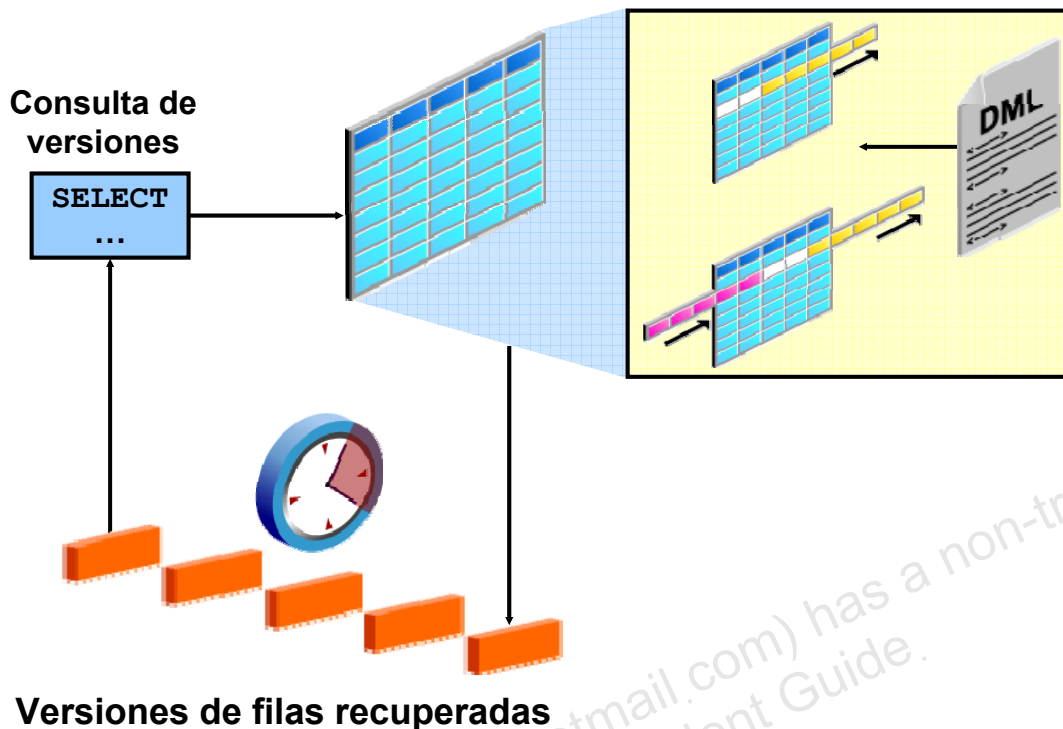
Agenda

- Manipulación de datos mediante subconsultas
- Especificación de valores por defecto explícitos en sentencias `INSERT` y `UPDATE`
- Uso de los siguientes tipos de sentencias `INSERTs` de varias tablas:
 - `INSERT` incondicional
 - `INSERT` mediante giro
 - `INSERT ALL` condicional
 - `INSERT FIRST` condicional
- Fusión de Filas en una Tabla
- Seguimiento de los cambios en los datos a lo largo de un período de tiempo

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Seguimiento de Cambios en Datos



ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Seguimiento de Cambios en Datos

Puede que detecte que, de algún modo, los datos en una tabla se han cambiado de forma incorrecta. Para averiguarlo, puede utilizar varias consultas de flashback para ver los datos de fila en puntos específicos en el tiempo. Para una mayor eficacia, puede utilizar la función de consulta de flashback de versiones para ver todos los cambios realizados en una fila en un período de tiempo. Esta función permite agregar una cláusula `VERSIONS` a una sentencia `SELECT`, que especifica un número de cambio del sistema (SCN) o el rango de registro de hora en el que desea ver los cambios en los valores de fila. La consulta también puede devolver metadatos asociados, como la transacción responsable del cambio.

Además, después de identificar la transacción errónea, puede utilizar la función de consulta de flashback de transacciones para identificar otros cambios realizados por la transacción.

A continuación, tendrá la opción de utilizar la función de flashback de tabla para restaurar la tabla a un estado anterior a los cambios realizados.

Puede utilizar una consulta en una tabla con la cláusula `VERSIONS` para producir todas las versiones de todas las filas que existen o que han existido entre la hora de ejecución de la consulta y los segundos de `undo_retention` antes de la hora actual. `undo_retention` es un parámetro de inicialización, parámetro de ajuste automático. Se hace referencia a una consulta que incluye una cláusula `VERSIONS` como una consulta de versiones. Los resultados de una consulta de versiones se comportan como si se hubiera aplicado la cláusula `WHERE` a las versiones de las filas. La consulta de versiones devuelve versiones de las filas sólo entre transacciones.

Número de cambio del sistema (SCN): el servidor de Oracle asigna un SCN para identificar los registro de redo de cada transacción confirmada.

Ejemplo de Consulta de Flashback de Versiones

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

1

```
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

1

	SALARY
1	4200

3

	SALARY
1	5460
2	4200

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Ejemplo de Consulta de Flashback de Versiones

En el ejemplo de la diapositiva, se recupera el salario para el empleado 107 (1). El salario del empleado 107 aumenta el 30 por ciento y se confirma este cambio (2). Se muestran las diferentes versiones del salario (3).

La cláusula `VERSIONS` no cambia el plan de la consulta. Por ejemplo, si ejecuta una consulta en una tabla que utiliza el método de acceso a índices, la misma consulta en la misma tabla con la cláusula `VERSIONS` continúa utilizando el método de acceso a índices. Las versiones de las filas devueltas por la consulta de versiones son versiones de filas entre transacciones. La cláusula `VERSIONS` no tiene ningún efecto en el comportamiento transaccional de una consulta. Esto significa que una consulta en una tabla con una cláusula `VERSIONS` hereda el entorno de consulta de la transacción en curso.

Se puede especificar la cláusula `VERSIONS` por defecto como `VERSIONS BETWEEN {SCN|TIMESTAMP} MINVALUE AND MAXVALUE`.

La cláusula `VERSIONS` es una extensión SQL sólo para consultar. Puede tener operaciones DML y DDL que utilizan una cláusula `VERSIONS` con subconsultas. La consulta de versiones de fila recupera todas las versiones confirmadas de las filas seleccionadas. No se devuelven los cambios realizados por la transacción activa actual. La consulta de versiones recupera todas las encarnaciones de la filas. Esto fundamentalmente significa que las versiones devueltas incluyen las versiones suprimidas y posteriormente insertadas de dichas filas.

Ejemplo de Consulta de Flashback de Versiones (continuación)

El acceso a filas de una consulta de versiones se puede definir en una de las dos categorías siguientes:

- **Acceso a filas según ROWID:** en el caso de un acceso basado en ROWID, se devuelven todas las versiones del ROWID especificado sin tener en cuenta el contenido de fila. Esto fundamentalmente significa que se devuelven todas las versiones del espacio del bloque que indica ROWID.
- **Resto de acceso a filas:** para el resto de acceso a filas, se devuelven todas las versiones de las filas.

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
1	18-JUN-09 05.07.10.000000000 PM	(null)	5460
2	(null)	18-JUN-09 05.07.10.000000000 PM	4200

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Cláusula VERSIONS BETWEEN

Puede utilizar la cláusula VERSIONS BETWEEN para recuperar todas las versiones de la filas que existen o han existido entre la hora de ejecución de la consulta y un punto anterior en el tiempo.

Si el tiempo de retención de deshacer es menor que el tiempo de límite inferior o el SCN de la cláusula BETWEEN, la consulta recupera las versiones sólo hasta el tiempo de retención de deshacer. El intervalo de tiempo de la cláusula BETWEEN se puede especificar como un intervalo de SCN o un intervalo de tiempo real. Este intervalo de hora se cierra tanto en el límite inferior como en el superior.

En el ejemplo, se recuperan los cambios del salario de Lorentz. El valor NULL de END_DATE para la primera versión indica que se trata de la versión existente en el momento de la consulta. El valor NULL para START_DATE de la última versión indica que la versión se ha creado en un momento anterior a la hora de retención de deshacer.

Prueba

Al utilizar el comando `INSERT` o `UPDATE`, la palabra clave `DEFAULT` le evita tener que codificar el valor por defecto en los programas o consultar el diccionario para buscarlo.

1. Verdadero
2. Falso

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Respuesta: 1

Resumen

En esta lección, debe haber aprendido lo siguiente:

- Utilizar sentencias DML y transacciones de control
- Describir las funciones de sentencias `INSERTs`
- Utilizar los siguientes tipos sentencias `INSERTs` de varias tablas:
 - `INSERT` incondicional
 - `INSERT` mediante giro
 - `INSERT ALL` condicional
 - `INSERT FIRST` condicional
- Fusionar filas en una tabla
- Manipular datos mediante subconsultas
- Realizar un seguimiento de los cambios en los datos a lo largo de un período de tiempo

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Resumen

En esta lección, debe haber aprendido cómo manipular los datos en Oracle Database mediante subconsultas. También debe haber aprendido sobre sentencias `INSERT` de varias tablas, la sentencia `MERGE` y a realizar un seguimiento en los cambios en la base de datos.

Práctica 4: Visión General

En esta práctica se abordan los siguientes temas:

- Realización de `INSERTs` de varias tablas
- Realización de operaciones `MERGE`
- Seguimiento de versiones de fila

ORACLE®

Copyright © 2010, Oracle. Todos los derechos reservados.

Gestión de Datos Situados en Distintas Zonas Horarias

ORACLE®

Copyright © 2010, Oracle. Todos los derechos reservados.

Objetivos

Al finalizar esta lección, debería estar capacitado para lo siguiente:

- Utilizar tipos de dato similares a `DATE` que almacenan segundos fraccionarios y realizan un seguimiento de zonas horarias
- Utilizar tipos de dato que almacenan la diferencia entre dos valores de fecha/hora
- Utilizar las siguientes funciones de fecha/hora:
 - `CURRENT_DATE`
 - `CURRENT_TIMESTAMP`
 - `LOCALTIMESTAMP`
 - `DBTIMEZONE`
 - `SESSIONTIMEZONE`
 - `EXTRACT`
 - `TZ_OFFSET`
 - `FROM_TZ`
 - `TO_TIMESTAMP`
 - `TO_YMINTERVAL`
 - `TO_DSINTERVAL`

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Objetivos

En esta lección, aprenderá a utilizar tipos de dato similares a `DATE` que almacenan segundos fraccionarios y realizan un seguimiento de zonas horarias. Esta lección trata algunas de las funciones de fecha/hora disponibles en Oracle Database.

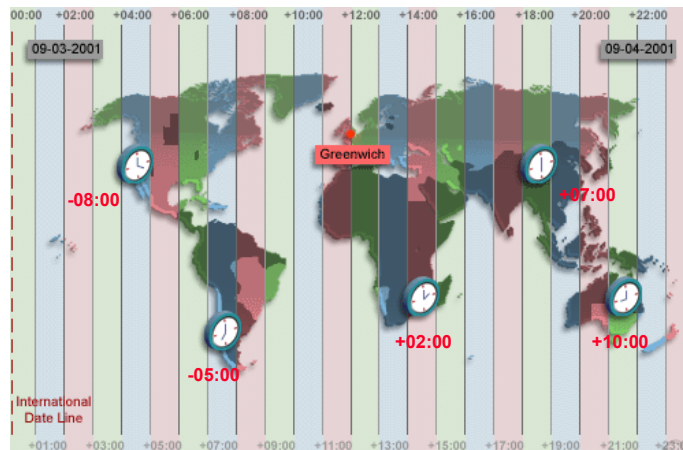
Agenda

- `CURRENT_DATE`, `CURRENT_TIMESTAMP`
y `LOCALTIMESTAMP`
- Tipo de dato `INTERVAL`
- Uso de las siguientes funciones:
 - `EXTRACT`
 - `TZ_OFFSET`
 - `FROM_TZ`
 - `TO_TIMESTAMP`
 - `TO_YMINTERVAL`
 - `TO_DSINTERVAL`

ORACLE®

Copyright © 2010, Oracle. Todos los derechos reservados.

Zonas Horarias



La imagen representa la hora de cada zona horaria cuando la hora de Greenwich es las 12:00.

ORACLE®

Copyright © 2010, Oracle. Todos los derechos reservados.

Zonas Horarias

Las horas del día se miden por el giro de la tierra. La hora del día en un determinado momento depende de su ubicación. Cuando es mediodía en Greenwich, Inglaterra, será medianoche en la Línea Internacional de Fecha. La tierra se divide en 24 zonas horarias, una para cada hora del día. La hora a lo largo de la línea de meridiano en Greenwich, Inglaterra, se denomina Hora Media de Greenwich (GMT). GMT se conoce ahora como Hora Universal Coordinada (UTC). UTC es la hora estándar que toman como referencia el resto de las zonas horarias del mundo. Es la misma durante todo el año y no le afecta el horario de verano. La línea de meridiano es una línea imaginaria que a del polo norte al polo sur. Se denomina longitud cero y es la línea respecto a la que se miden las demás líneas de longitud. Todas las horas se miden respecto a UTC y todos los lugares tienen una latitud (su distancia hacia el norte o hacia el sur del ecuador) y una longitud (su distancia hacia el este o hacia el oeste del meridiano de Greenwich).

Parámetro de Sesión `TIME_ZONE`

`TIME_ZONE` se puede definir en:

- Un desplazamiento absoluto
- Zona horaria de la base de datos
- Zona horaria local del sistema operativo
- Una determinada región

```
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 © 2010, Oracle. Todos los derechos reservados.

Parámetro de Sesión `TIME_ZONE`

Oracle Database soporta el almacenamiento de la zona horaria en los datos de fecha y hora, así como de los segundos fraccionarios. El comando `ALTER SESSION` se puede utilizar para cambiar los valores de zona horaria en una sesión del usuario. Los valores de zona horaria se pueden definir en un desplazamiento absoluto, una determinada zona horaria, una zona horaria de base de datos o una zona horaria local.

CURRENT_DATE, CURRENT_TIMESTAMP y LOCALTIMESTAMP

- **CURRENT_DATE:**
 - Devuelve la fecha actual de la sesión de usuario.
 - Tiene un tipo de dato `DATE`.
- **CURRENT_TIMESTAMP:**
 - Devuelve la fecha y hora actual de la sesión de usuario.
 - Tiene el tipo de dato `TIMESTAMP WITH TIME ZONE`.
- **LOCALTIMESTAMP:**
 - Devuelve la fecha y hora actual de la sesión de usuario.
 - Tiene un tipo de dato `TIMESTAMP`.

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

CURRENT_DATE, CURRENT_TIMESTAMP y LOCALTIMESTAMP

Las funciones `CURRENT_DATE` y `CURRENT_TIMESTAMP` devuelven la fecha actual y el registro de hora actual, respectivamente. El tipo de dato de `CURRENT_DATE` es `DATE`. El tipo de dato de `CURRENT_TIMESTAMP` es `TIMESTAMP WITH TIME ZONE`. Los valores devueltos muestran el desplazamiento de zona horaria de la sesión SQL que ejecuta las funciones. El desplazamiento de zona horaria es la diferencia (en horas y minutos) entre la hora local y UTC. El tipo de dato `TIMESTAMP WITH TIME ZONE` tiene el formato:

`TIMESTAMP [(fractional_seconds_precision)] WITH TIME ZONE`

donde `fractional_seconds_precision` especifica opcionalmente el número de dígitos de la parte fraccionaria del campo de fecha/hora `SECOND` y puede ser un número del intervalo 0 a 9.

El valor por defecto es 6.

La función `LOCALTIMESTAMP` devuelve la fecha y hora actuales en la zona horaria de la sesión. La diferencia entre `LOCALTIMESTAMP` y `CURRENT_TIMESTAMP` es que `LOCALTIMESTAMP` devuelve un valor `TIMESTAMP`, mientras que `CURRENT_TIMESTAMP` devuelve un valor `TIMESTAMP WITH TIME ZONE`.

Estas funciones son sensibles al soporte de idioma nacional (NLS), es decir, los resultados tendrán los formatos de calendario y fecha/hora de NLS.

Nota: la función `SYSDATE` devuelve la fecha y hora actuales como un tipo de dato `DATE`. Ha aprendido cómo utilizar la función `SYSDATE` en el curso titulado *Oracle Database: Conceptos Fundamentales de SQL I*.

Comparación de Fecha y Hora en la Zona Horaria de una Sesión

El parámetro `TIME_ZONE` se define en `-5:00` y, a continuación, se ejecutan las sentencias `SELECT` para cada fecha y hora para comparar las diferencias.

```
ALTER SESSION
SET NLS_DATE_FORMAT = 'DD-MON-YYYY HH24:MI:SS';
ALTER SESSION SET TIME_ZONE = '-5:00';

SELECT SESSIONTIMEZONE, CURRENT_DATE FROM DUAL; 1

SELECT SESSIONTIMEZONE, CURRENT_TIMESTAMP FROM DUAL; 2

SELECT SESSIONTIMEZONE, LOCALTIMESTAMP FROM DUAL; 3
```

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Comparación de Fecha y Hora en la Zona Horaria de una Sesión

El comando `ALTER SESSION` define el formato de fecha de la sesión en `'DD-MON-YYYY HH24:MI:SS'`, es decir, días del mes (1–31)–nombre abreviado del mes–hora del día del año de 4 dígitos (0–23):minutos (0–59):segundos (0–59).

El ejemplo de la diapositiva ilustra que la sesión se modifica para definir el parámetro `TIME_ZONE` en `-5:00`. A continuación, se ejecuta la sentencia `SELECT` para `CURRENT_DATE`, `CURRENT_TIMESTAMP` y `LOCALTIMESTAMP` para observar las diferencias.

Nota: el parámetro `TIME_ZONE` especifica el desplazamiento de zona horaria local por defecto para la sesión SQL actual. `TIME_ZONE` es sólo un parámetro de sesión, no un parámetro de inicialización. El parámetro `TIME_ZONE` se define de la siguiente forma:

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

La máscara de formato `([+ | -] hh:mm)` indica las horas y minutos antes o después de UTC.

Comparación de Fecha y Hora en la Zona Horaria de una Sesión

Resultados de las consultas:

ALTER SESSION succeeded.

SESSIONTIMEZONE	CURRENT_DATE
1 -05:00	23-JUN-2009 01:34:52

1

SESSIONTIMEZONE	CURRENT_TIMESTAMP
1 -05:00	23-JUN-09 01.35.26.239882000 AM -05:00

2

SESSIONTIMEZONE	LOCALTIMESTAMP
1 -05:00	23-JUN-09 01.36.21.811798000 AM

3

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Comparación de Fecha y Hora en la Zona Horaria de una Sesión (continuación)

En este caso, la función `CURRENT_DATE` devuelve la fecha actual en la zona horaria de la sesión, la función `CURRENT_TIMESTAMP` devuelve la fecha y hora actuales en la zona horaria de la sesión como un valor del tipo de dato `TIMESTAMP WITH TIME ZONE` y la función `LOCALTIMESTAMP` devuelve la fecha y hora actuales en la zona horaria de la sesión.

DBTIMEZONE Y SESSIONTIMEZONE

- Mostrar el valor de la zona horaria de la base de datos:

```
SELECT DBTIMEZONE FROM DUAL;
```

DBTIMEZONE
1 +00:00

- Mostrar el valor de la zona horaria de la sesión:

```
SELECT SESSIONTIMEZONE FROM DUAL;
```

SESSIONTIMEZONE
1 -05:00

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

DBTIMEZONE Y SESSIONTIMEZONE

El DBA define la zona horaria por defecto de la base de datos mediante la especificación de la cláusula `SET TIME_ZONE` de la sentencia `CREATE DATABASE`. Si se omite, la zona horaria por defecto de la base de datos es la zona horaria del sistema operativo. La zona horaria de la base de datos no se puede cambiar para una sesión con la sentencia `ALTER SESSION`.

La función `DBTIMEZONE` devuelve el valor de la zona horaria de la base de datos. El tipo de retorno es un desplazamiento de zona horaria (un tipo de carácter con el formato: `'[+|-] TZh:TzM'`) o un nombre de región de zona horaria, en función de cómo haya especificado el usuario el valor de zona horaria de la base de datos en la sentencia `CREATE DATABASE` o `ALTER DATABASE` más reciente. El ejemplo de la diapositiva muestra que la zona horaria de la base de datos se ha definido en “-05:00”, ya que el parámetro `TIME_ZONE` tiene el formato:

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

La función `SESSIONTIMEZONE` devuelve el valor de la zona horaria de la sesión actual. El tipo de retorno es un desplazamiento de zona horaria (un tipo de carácter en formato `'[+|-] TZh:TzM'`) o un nombre de región de zona horaria, en función de cómo haya especificado el usuario el valor de zona horaria de la sesión en la sentencia `ALTER SESSION` más reciente. El ejemplo de la diapositiva muestra que la zona horaria de la sesión es el desplazamiento respecto a UTC de -8 horas. Observe que la zona horaria de la base de datos es diferente de la zona horaria de la sesión actual.

Tipos de Dato **TIMESTAMP**

Tipo de Dato	Campos
TIMESTAMP	Year, Month, Day, Hour, Minute, Second with fractional seconds
TIMESTAMP WITH TIME ZONE	Igual que el tipo de dato TIMESTAMP ; también incluye: TIMEZONE_HOUR y TIMEZONE_MINUTE o TIMEZONE_REGION
TIMESTAMP WITH LOCAL TIME ZONE	Igual que el tipo de dato TIMESTAMP ; también incluye un desplazamiento de zona horaria en su valor.

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Tipos de Dato **TIMESTAMP**

El tipo de dato **TIMESTAMP** es una extensión del tipo de dato **DATE**.

TIMESTAMP (fractional_seconds_precision)

Este tipo de dato contiene los valores de año, mes y día, así como los valores de hora, minutos y segundos, donde la precisión de segundos fraccionarios significativa es el número de dígitos en la parte fraccionaria del campo de fecha/hora **SECOND**. Los valores aceptados de **fractional_seconds_precision** significativa son de 0 a 9. el valor por defecto es 6.

TIMESTAMP (fractional_seconds_precision) WITH TIME ZONE

Este tipo de dato contiene todos los valores de **TIMESTAMP** así como el valor de desplazamiento de zona horaria.

TIMESTAMP (fractional_seconds_precision) WITH LOCAL TIME ZONE

Este tipo de dato contiene todos los valores de **TIMESTAMP**, con las siguientes excepciones:

- Los datos se normalizan en la zona horaria de la base de datos al almacenarlo en la base de datos.
- Al recuperar los datos, los usuarios los ven en la zona horaria de la sesión.

Campos **TIMESTAMP**

Campo Datetime	Valores Válidos
YEAR	de -4712 a 9999 (excepto el año 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), donde 9(N) es la precisión
TIMEZONE_HOUR	de -12 a 14
TIMEZONE_MINUTE	de 00 a 59

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Campos **TIMESTAMP**

Cada tipo de dato de fecha/hora está compuesto de varios de estos campos. Los tipos de dato de fecha/hora sólo se pueden comparar y asignar mutuamente si tienen los mismos campos de fecha/hora.

Diferencia entre DATE y TIMESTAMP

A

```
-- when hire_date is
of type DATE
```

```
SELECT hire_date
FROM employees;
```

	HIRE_DATE
1	21-JUN-99
2	13-JAN-00
3	17-SEP-87
4	17-FEB-96
5	17-AUG-97
6	07-JUN-94
7	07-JUN-94
8	07-JUN-94

B

```
ALTER TABLE employees
MODIFY hire_date TIMESTAMP;
```

```
SELECT hire_date
FROM employees;
```

	HIRE_DATE
1	21-JUN-99 12.00.00.000000000 AM
2	13-JAN-00 12.00.00.000000000 AM
3	17-SEP-87 12.00.00.000000000 AM
4	17-FEB-96 12.00.00.000000000 AM
5	17-AUG-97 12.00.00.000000000 AM
6	07-JUN-94 12.00.00.000000000 AM
7	07-JUN-94 12.00.00.000000000 AM
8	07-JUN-94 12.00.00.000000000 AM

...

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Tipo de Dato TIMESTAMP: Ejemplo

En la diapositiva, el ejemplo A muestra los datos de la columna `hire_date` de la tabla `EMPLOYEES` cuando el tipo de dato de la columna es `DATE`. En el ejemplo B, la tabla se ha modificado y el tipo de dato de la columna `hire_date` se ha convertido en `TIMESTAMP`. La salida muestra las diferencias de visualización. Puede convertir de `DATE` a `TIMESTAMP` cuando la columna tiene datos, pero no puede convertir de `DATE` o `TIMESTAMP` a `TIMESTAMP WITH TIME ZONE` a menos que la columna esté vacía.

Puede especificar la precisión de segundos fraccionarios para el registro de hora. Si no se especifica ninguna, como se muestra en ejemplo, se define por defecto en 6.

Por ejemplo, la siguiente sentencia define la precisión de segundos fraccionarios en 7:

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

Nota: el tipo de dato de fecha de Oracle aparece por defecto como se muestra en este ejemplo. Sin embargo, el tipo de dato de fecha contiene también información adicional como las horas, minutos, segundos, AM y PM. Para obtener la fecha en este formato, puede aplicar una máscara de formato o una función al valor de fecha.

Comparación de Tipos de Dato **TIMESTAMP**

```
CREATE TABLE web_orders
(order_date TIMESTAMP WITH TIME ZONE,
delivery_time TIMESTAMP WITH LOCAL TIME ZONE);
```

```
INSERT INTO web_orders values
(current_date, current_timestamp + 2);
```

```
SELECT * FROM web_orders;
```

ORDER_DATE	DELIVERY_TIME
1 23-JUN-09 01.56.39.000000000 AM -05:00	25-JUN-09 01.56.39.000000000 AM

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Comparación de Tipos de Dato **TIMESTAMP**

En el ejemplo de la diapositiva, se crea una nueva tabla `web_orders` con una columna del tipo de dato `TIMESTAMP WITH TIME ZONE` y una columna del tipo de dato `TIMESTAMP WITH LOCAL TIME ZONE`. Esta tabla se rellena cuando se establece `web_order`. El registro de hora y la zona horaria del usuario cuando se establece el orden se inserta según el valor `CURRENT_DATE`.

El registro de hora y la zona horaria local se rellenan insertando dos días del valor `CURRENT_TIMESTAMP` en el registro cada vez que se establece un orden. Cuando una compañía basada en Web garantiza el envío, puede estimar su hora de entrega según la zona horaria de la persona que establece el orden.

Agenda

- CURRENT_DATE, CURRENT_TIMESTAMP y LOCALTIMESTAMP
- **Tipos de dato** INTERVAL
- **Uso de las siguientes funciones:**
 - EXTRACT
 - TZ_OFFSET
 - FROM_TZ
 - TO_TIMESTAMP
 - TO_YMINTERVAL
 - TO_DSINTERVAL

ORACLE®

Copyright © 2010, Oracle. Todos los derechos reservados.

Tipos de Dato INTERVAL

- Los tipos de dato `INTERVAL` se utilizan para almacenar la diferencia entre dos valores de fecha/hora.
- Hay dos clases de intervalos:
 - Año-mes
 - Día-hora
- La precisión del intervalo:
 - Es el subjuego real de campos que constituye un intervalo.
 - Se especifica en el cualificador de intervalo.

Tipo de Dato	Campos
<code>INTERVAL YEAR TO MONTH</code>	Year, Month
<code>INTERVAL DAY TO SECOND</code>	Days, Hour, Minute, Second with fractional seconds

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Tipos de Dato INTERVAL

Los tipos de dato `INTERVAL` se utilizan para almacenar la diferencia entre dos valores de fecha/hora. Hay dos clases de intervalos: intervalos año-mes y día-hora. El tipo de intervalo año-mes consta de un subjuego contiguo de campos de `YEAR` y `MONTH`, mientras que el intervalo de día-hora consta de un subjuego contiguo de campos que consta de `DAY`, `HOURL`, `MINUTE` y `SECOND`.

El subjuego de campos real que constituye un intervalo se denomina precisión del intervalo y se especifica en el cualificador de intervalo. Puesto que el número de días de un año depende del calendario, el intervalo año-mes depende de `NLS`, mientras que el intervalo día-hora es independiente de `NLS`.

El cualificador de intervalo puede especificar también la precisión del campo inicial, que es el número de dígitos del campo inicial o único, y en caso de que el campo final sea `SECOND`, puede especificar también la precisión de segundos fraccionarios, que es el número de dígitos en la parte fraccional del valor `SECOND`. Si no se especifica, el valor por defecto para la precisión de campo inicial es de 2 dígitos y el valor por defecto para la precisión de segundos fraccionarios es de 6 dígitos.

Tipos de Dato INTERVAL (continuación)**INTERVAL YEAR (year_precision) TO MONTH**

Este tipo de dato almacena un período de tiempo en años y meses, donde `year_precision` es el número de dígitos del campo de fecha/hora `YEAR`. Los valores aceptados son de 0 a 9. El valor por defecto es 6.

INTERVAL DAY (day_precision) TO SECOND (fractional_seconds_precision)

Este tipo de dato almacena un período de tiempo en días, horas, minutos y segundos, donde `day_precision` es el número máximo de dígitos del campo de fecha/hora `DAY` (los valores aceptados son de 0 a 9; el valor por defecto es 2) y `fractional_seconds_precision` es el número de dígitos de la parte fraccionaria del campo `SECOND`. Los valores aceptados son de 0 a 9. El valor por defecto es 6.

Campos INTERVAL

Campo INTERVAL	Valores Válidos para el Intervalo
YEAR	Cualquier valor positivo o negativo
MONTH	de 00 a 11
DAY	Cualquier valor positivo o negativo
HOURL	de 00 a 23
MINUTE	de 00 a 59
SECOND	de 00 a 59.9(N), donde 9(N) es la precisión

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Campos INTERVAL

INTERVAL YEAR TO MONTH puede tener los campos YEAR y MONTH.

INTERVAL DAY TO SECOND puede tener los campos DAY, HOUR, MINUTE y SECOND.

El subjuego real de campos que constituye un elemento de cualquier tipo lo define un cualificador de intervalo y este subjuego se denomina precisión del elemento.

Los intervalos año-mes sólo se pueden comparar y asignar mutuamente con otros intervalos año-mes y los intervalos día-hora sólo se pueden comparar y asignar mutuamente con otros intervalos día-hora.

INTERVAL YEAR TO MONTH: Ejemplo

```
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
1	123	0-8
2	155	200-0
3	678	200-11

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Tipo de Dato INTERVAL YEAR TO MONTH

INTERVAL YEAR TO MONTH almacena un período de tiempo mediante los campos de fecha/hora YEAR y MONTH. Especifique INTERVAL YEAR TO MONTH como se muestra a continuación:

INTERVAL YEAR [(year_precision)] TO MONTH

donde year_precision es el número de dígitos del campo de fecha/hora YEAR. El valor por defecto de year_precision es 2.

Restricción: el campo inicial debe ser más significativo que el final. Por ejemplo, INTERVAL '0-1' MONTH TO YEAR no es válido.

Ejemplos

- INTERVAL '123-2' YEAR(3) TO MONTH
Indica un intervalo de 123 años y 2 meses.
- INTERVAL '123' YEAR(3)
Indica un intervalo de 123 años y 0 meses.
- INTERVAL '300' MONTH(3)
Indica un intervalo de 300 meses.
- INTERVAL '123' YEAR
Devuelve un error porque la precisión por defecto es 2 y 123 tiene tres dígitos.

Tipo de Dato INTERVAL YEAR TO MONTH (continuación)

Oracle Database soporta dos tipos de dato de intervalo: INTERVAL YEAR TO MONTH y INTERVAL DAY TO SECOND; el tipo de columna, el argumento PL/SQL, la variable y el tipo de retorno debe ser uno de los dos. Sin embargo, para los literales de intervalo, el sistema reconoce otros tipos de intervalo ANSI (American National Standards Institute) como INTERVAL '2' YEAR o INTERVAL '10' HOUR. En estos casos, cada intervalo se convierte en uno de los dos tipos soportados.

En el ejemplo de la diapositiva, se crea una tabla WARRANTY, que contiene una columna warranty_time que utiliza el tipo de dato INTERVAL YEAR (3) TO MONTH. En ella se insertan diferentes valores para indicar los años y meses para diferentes productos. Cuando estas filas se recuperen de la tabla, verá el valor de año separado del valor de mes con un carácter (-).

Tipo de Dato INTERVAL DAY TO SECOND: Ejemplo

```
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
1	100012	90 0:0:0.0
2	56098	6 3:30:16.0

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Tipo de Dato INTERVAL DAY TO SECOND: Ejemplo

En el ejemplo de la diapositiva, se crea una tabla `lab` con una columna `test_time` del tipo de dato `INTERVAL DAY TO SECOND`. A continuación, se inserta en la tabla el valor `'90 00:00:00'` para indicar 90 días y 0 horas, 0 minutos y 0 segundos e `INTERVAL '6 03:30:16' DAY TO SECOND` para indicar 6 días, 3 horas, 30 minutos y 16 segundos. La sentencia `SELECT` indica cómo se muestran estos datos en la base de datos.

Agenda

- `CURRENT_DATE`, `CURRENT_TIMESTAMP`
y `LOCALTIMESTAMP`
- Tipos de dato `INTERVAL`
- **Uso de las siguientes funciones:**
 - `EXTRACT`
 - `TZ_OFFSET`
 - `FROM_TZ`
 - `TO_TIMESTAMP`
 - `TO_YMINTERVAL`
 - `TO_DSINTERVAL`

ORACLE®

Copyright © 2010, Oracle. Todos los derechos reservados.

EXTRACT

- Mostrar el componente YEAR de SYSDATE.

```
SELECT EXTRACT (YEAR FROM SYSDATE) FROM DUAL;
```

EXTRACT(YEARFROMSYSDATE)
2009

- Mostrar el componente MONTH de HIRE_DATE para aquellos empleados cuyo MANAGER_ID sea 100.

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

	LAST_NAME	HIRE_DATE	EXTRACT(MONTHFROMHIRE_DATE)
1	Hartstein	17-FEB-1996 00:00:00	2
2	Kochhar	21-SEP-1989 00:00:00	9
3	De Haan	13-JAN-1993 00:00:00	1
4	Raphaely	07-DEC-1994 00:00:00	12
5	Weiss	18-JUL-1996 00:00:00	7

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

EXTRACT

La expresión EXTRACT extrae y devuelve el valor de un campo de fecha/hora especificado o una expresión de valores de intervalo. Puede extraer cualquiera de los componentes mencionados en la siguiente sintaxis mediante la función EXTRACT. La sintaxis de la función EXTRACT es:

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

Al extraer TIMEZONE_REGION o TIMEZONE_ABBR (abreviatura), el valor devuelto es una cadena que contiene el nombre de la zona horaria adecuada correspondiente o su abreviatura. Al extraer cualquiera de los demás valores, el valor devuelto es una fecha del calendario gregoriano. Cuando se extrae de un campo de fecha/hora con un valor de zona horaria, el valor devuelto está en UTC.

En el primer ejemplo de la diapositiva, se utiliza la función EXTRACT para extraer YEAR de SYSDATE. En el segundo ejemplo de la diapositiva, se utiliza la función EXTRACT para extraer MONTH de la columna HIRE_DATE de la tabla EMPLOYEES para aquellos empleados que dependen del gestor cuyo EMPLOYEE_ID es 100.

TZ_OFFSET

Mostrar el desplazamiento de zona horaria para las zonas horarias 'US/Eastern', 'Canada/Yukon' y 'Europe/London':

```
SELECT TZ_OFFSET('US/Eastern'),
       TZ_OFFSET('Canada/Yukon'),
       TZ_OFFSET('Europe/London')
FROM   DUAL;
```

TZ_OFFSET('US/EASTERN')	TZ_OFFSET('CANADA/YUKON')	TZ_OFFSET('EUROPE/LONDON')
1 -04:00	-07:00	+01:00

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

TZ_OFFSET

La función `TZ_OFFSET` devuelve el desplazamiento de zona horaria correspondiente al valor introducido. El valor de retorno depende de la fecha en la que se ejecute la sentencia. Por ejemplo, se la función `TZ_OFFSET` devuelve un valor `-08:00`, este valor indica que la zona horaria en la que se ha ejecutado el comando es de ocho horas por debajo de la hora UTC. Puede introducir un nombre de zona horaria válido, un desplazamiento de zona horaria a partir de UTC (que simplemente se devuelve a sí mismo) o la palabra clave `SESSIONTIMEZONE` o `DBTIMEZONE`. La sintaxis de la función `TZ_OFFSET` es:

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

La compañía Fold Motor tiene su sede en Michigan, EE.UU., que es la zona horaria de EE.UU./Oriental. El presidente de la compañía, el señor Fold, desea realizar una llamada a conferencia con los vicepresidentes de operaciones en Canadá y Europa, que se encuentran las zonas horarias de Canadá/Yukón y Europa/Londres, respectivamente. El señor Fold desea averiguar la hora en cada uno de estos lugares para asegurarse de que su equipo de gestión senior esté disponible para asistir a la reunión. Su secretario, el señor Scott, le ayuda mediante la emisión de las consultas que se muestran en el ejemplo y obtiene los siguientes resultados:

- La zona horaria 'US/Eastern' es de cuatro horas por debajo de la hora UTC.
- La zona horaria 'Canada/Yukon' es de siete horas por debajo de la hora UTC.
- La zona horaria 'Europe/London' es de una hora por encima de la hora UTC.

TZ_OFFSET (continuación)

Para obtener una lista de los valores de nombre de las zonas horarias válidas, puede consultar la vista de rendimiento dinámico V\$TIMEZONE_NAMES.

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

	TZNAME	TZABBREV
1	Africa/Abidjan	LMT
2	Africa/Abidjan	GMT
3	Africa/Accra	LMT
4	Africa/Accra	GMT
5	Africa/Accra	GHST

...

Unauthorized reproduction or distribution prohibited. Copyright© 2014, Oracle and/or its affiliates.

Rodrigo Esteves (rodrigo_ush@hotmail.com) has a non-transferable license to use this Student Guide.

FROM_TZ

Mostrar el valor de `TIMESTAMP '2000-03-28 08:00:00'` como un valor de `TIMESTAMP WITH TIME ZONE` para la región de zona horaria `'Australia/North'`.

```
SELECT FROM_TZ(TIMESTAMP
                '2000-07-12 08:00:00', 'Australia/North')
FROM DUAL;
```

FROM_TZ(TIMESTAMP'2000-07-1208:00:00','AUSTRALIA/NORTH')
1 12-JUL-00 08.00.00.000000000 AM AUSTRALIA/NORTH

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

FROM_TZ

La función `FROM_TZ` convierte un valor de `TIMESTAMP` en un valor de `TIMESTAMP WITH TIME ZONE`.

La sintaxis de la función `FROM_TZ` es:

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

donde `time_zone_value` es una cadena de caracteres con el formato `'TZR:TZM'` o una expresión de caracteres que devuelve una cadena `TZR` (región de zona horaria) con un formato `TZD` opcional. `TZD` es una abreviatura de la cadena de zona horaria con la información de horario de verano. `TZR` representa la región de zona horaria en las cadenas de entrada de fecha/hora. Los ejemplos son `'Australia/North'`, `'PST'` para la hora oficial de EE.UU./Pacífico, `'PDT'` para el horario de verano de EE.UU./Pacífico, etc.

El ejemplo de la diapositiva convierte un valor de `TIMESTAMP` en un valor de `TIMESTAMP WITH TIME ZONE`.

Nota: para ver una lista de los valores válidos para los elementos de formato `TZR` y `TZD`, consulte la vista de rendimiento dinámico `V$TIMEZONE_NAMES`.

TO_TIMESTAMP

Mostrar la cadena de caracteres '2007-03-06 11:00:00' como un valor de `TIMESTAMP`:

```
SELECT TO_TIMESTAMP ('2007-03-06 11:00:00',
                     'YYYY-MM-DD HH:MI:SS')
FROM DUAL;
```

```
TO_TIMESTAMP('2007-03-0611:00:00','YYYY-MM-DDHH:MI:SS')
06-MAR-07 11.00.00.000000000
```

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

TO_TIMESTAMP

La función `TO_TIMESTAMP` convierte una cadena del tipo de dato `CHAR`, `VARCHAR2`, `NCHAR` o `NVARCHAR2` en un valor del tipo de dato `TIMESTAMP`. La sintaxis de la función `TO_TIMESTAMP` es:

```
TO_TIMESTAMP (char, [fmt], ['nlsparam'])
```

El `fmt` opcional especifica el formato de `char`. Si omite `fmt`, a cadena debe tener el formato por defecto del tipo de dato `TIMESTAMP`. El parámetro `nlsparam` opcional especifica el idioma en el que se devuelven las abreviaturas y los nombres del mes y del día. Este argumento puede tener este formato:

```
'NLS_DATE_LANGUAGE = language'
```

Si omite `nlsparams`, esta función utiliza el lenguaje de fecha por defecto para la sesión.

El ejemplo de la diapositiva convierte una cadena de caracteres en un valor de `TIMESTAMP`.

Nota: utilice la función `TO_TIMESTAMP_TZ` para convertir una cadena del tipo de dato `CHAR`, `VARCHAR2`, `NCHAR` o `NVARCHAR2` en un valor del tipo de dato `TIMESTAMP WITH TIME ZONE`. Para obtener más información, consulte *Oracle Database SQL Language Reference (Referencia sobre Lenguaje SQL de Oracle Database)* para la base de datos 10g u 11g.

TO_YMINTERVAL

Mostrar una fecha que sea un año y dos meses posterior a la fecha de contratación de los empleados que trabajan en el departamento con el `DEPARTMENT_ID` 20.

```
SELECT hire_date,
       hire_date + TO_YMINTERVAL('01-02') AS
       HIRE_DATE_YMININTERVAL
FROM   employees
WHERE  department_id = 20;
```

	HIRE_DATE	HIRE_DATE_YMININTERVAL
1	17-FEB-1996 00:00:00	17-APR-1997 00:00:00
2	17-AUG-1997 00:00:00	17-OCT-1998 00:00:00

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

TO_YMINTERVAL

La función `TO_YMINTERVAL` convierte una cadena de caracteres del tipo de dato `CHAR`, `VARCHAR2`, `NCHAR` o `NVARCHAR2` en un tipo de dato `INTERVAL YEAR TO MONTH`. El tipo de dato `INTERVAL YEAR TO MONTH` almacena un período de tiempo mediante los campos de fecha/hora `YEAR` y `MONTH`. El formato de `INTERVAL YEAR TO MONTH` es el siguiente:

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

donde `year_precision` es el número de dígitos del campo de fecha/hora `YEAR`. El valor por defecto de `year_precision` es 2.

La sintaxis de la función `TO_YMINTERVAL` es:

```
TO_YMINTERVAL (char)
```

donde `char` es la cadena de caracteres que se va a convertir.

El ejemplo de la diapositiva calcula una fecha un año y dos meses posterior a la fecha de contratación de los empleados del departamento 20 de la tabla `EMPLOYEES`.

TO_DSINTERVAL

Mostrar una fecha que sea 100 días y 10 horas posterior a la fecha de contratación de todos los empleados.

```
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
1	OConnell	06-21-99:12:00:00	09-29-99:10:00:00
2	Grant	01-13-00:12:00:00	04-22-00:10:00:00
3	Whalen	09-17-87:12:00:00	12-26-87:10:00:00
4	Hartstein	02-17-96:12:00:00	05-27-96:10:00:00
5	Fay	08-17-97:12:00:00	11-25-97:10:00:00
6	Mavris	06-07-94:12:00:00	09-15-94:10:00:00
7	Baer	06-07-94:12:00:00	09-15-94:10:00:00
8	Higgins	06-07-94:12:00:00	09-15-94:10:00:00

...

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

TO_DSINTERVAL

La función `TO_DSINTERVAL` convierte una cadena de caracteres del tipo de dato `CHAR`, `VARCHAR2`, `NCHAR` o `NVARCHAR2` en un tipo de dato `INTERVAL DAY TO SECOND`.

En el ejemplo de la diapositiva, se obtiene la fecha 100 días y 10 horas posterior a la fecha de contratación.

Horario de Verano

- Primer Domingo de Abril
 - La hora pasa de 01:59:59 AM a 03:00:00 AM.
 - Los valores de 02:00:00 AM a 02:59:59 AM no son válidos.
- Último Domingo de Octubre
 - La hora pasa de 02:00:00 AM a 01:00:01 AM.
 - Los valores de 01:00:01 AM a 02:00:00 AM son ambiguos porque se visitan dos veces.

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Horario de Verano (DST)

La mayoría de los países occidentales adelantan el reloj una hora durante los meses de verano. Este período se denomina horario de verano. El horario de verano abarca desde el primer domingo de abril hasta el último domingo de octubre en la mayoría de los Estado Unidos, México y Canadá. Los países de la Unión Europea tienen en cuenta el horario de verano, pero lo llaman período de verano. El horario de verano en Europa empieza una semana antes que para sus homólogos norteamericanos, pero finaliza al mismo tiempo.

Oracle Database determina automáticamente, para cualquier región de zona horaria, si el horario de verano está en vigor y devuelve los valores de hora local correspondientes. El valor de fecha/hora es suficiente para Oracle Database para determinar si el horario de verano está en vigor para una región determinada en todos los casos, excepto los de límite. Un caso de límite se produce durante el período en el que entra en vigor o termina el horario de verano. Por ejemplo, en la región de EE.UU./Oriental, cuando entra en vigor el horario de verano, la hora cambia de 01:59:59 AM a 03:00:00 AM. El intervalo de una hora entre 02:00:00 AM y 02:59:59 AM. no existe. Cuando el horario de verano termina, la hora cambia de 02:00:00 AM a 01:00:01 AM y el intervalo de una hora entre 01:00:01 AM y 02:00:00 AM se repite.

Horario de Verano (DST) (continuación)

ERROR_ON_OVERLAP_TIME

ERROR_ON_OVERLAP_TIME es un parámetro de sesión que notifica al sistema que debe emitir un error cuando encuentre un valor de fecha/hora que tenga lugar en el período solapado y no haya especificado ninguna abreviatura de zona horaria para distinguir el período.

Por ejemplo, el horario de verano finaliza el 31 de octubre a las 02:00:01 AM. El período solapado es:

- de 10/31/2004 01:00:01 AM a 10/31/2004 02:00:00 AM (EDT)
- de 10/31/2004 01:00:01 AM a 10/31/2004 02:00:00 AM (EST)

Si introduce una cadena de fecha/hora que tiene lugar en uno de estos dos períodos, debe especificar la abreviatura de la zona horaria (por ejemplo, EDT o EST) en la cadena de entrada para que el sistema determine el período. Sin esta abreviatura de fecha/hora, el sistema hace lo siguiente:

Si el parámetro ERROR_ON_OVERLAP_TIME es FALSE, asume que la hora de entrada es la hora oficial (por ejemplo, EST). De lo contrario, se produce un error .

Prueba

El parámetro de sesión `TIME_ZONE` se puede definir en:

1. Un desplazamiento relativo
2. Zona horaria de la base de datos
3. Zona horaria local del sistema operativo
4. Una determinada región

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Respuestas: 2, 3, 4

Resumen

En esta lección debe haber aprendido a utilizar las siguientes funciones:

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

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Resumen

Esta lección trata algunas de las funciones de fecha/hora disponibles en Oracle Database.

Práctica 5: Visión General

En esta práctica se aborda el uso de las funciones de fecha/hora.

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Práctica 5: Visión General

En esta práctica, mostrará los desplazamientos de zona horaria, `CURRENT_DATE`, `CURRENT_TIMESTAMP` y `LOCALTIMESTAMP`. También definirá las zonas horarias y utilizará la función `EXTRACT`.

Recuperación de Datos mediante Subconsultas



ORACLE®

Copyright © 2010, Oracle. Todos los derechos reservados.

Rodrigo Esteves (rodrigo_...@hotmail.com) has a non-transferable license to use this Student Guide.

Objetivos

Al finalizar esta lección, debería estar capacitado para lo siguiente:

- Escribir una subconsulta de varias columnas
- Utilizar subconsultas escalares en SQL
- Solucionar problemas con subconsultas correlacionadas
- Actualizar y suprimir filas mediante subconsultas correlacionadas
- Utilizar los operadores `EXISTS` y `NOT EXISTS`
- Utilizar la cláusula `WITH`

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Objetivos

En esta lección, aprenderá a escribir subconsultas y subconsultas de varias columnas en la cláusula `FROM` de una sentencia `SELECT`. También aprenderá a solucionar problemas mediante subconsultas escalares, correlacionadas y la cláusula `WITH`.

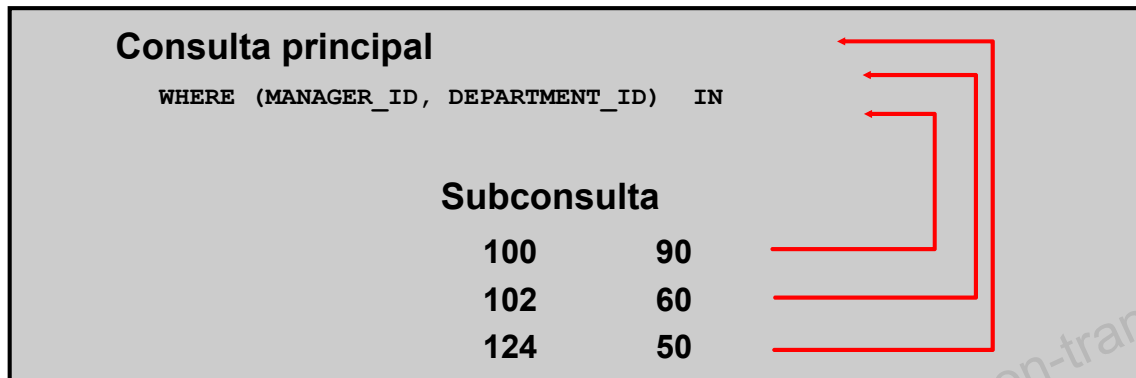
Agenda

- Escritura de una subconsulta de varias columnas
- Uso de subconsultas escalares en SQL
- Solución de problemas con subconsultas correlacionadas
- Uso de los operadores `EXISTS` y `NOT EXISTS`
- Uso de la cláusula `WITH`

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Subconsultas de Varias Columnas



Cada fila de una consulta principal se compara con los valores de una subconsulta de varias filas y varias columnas.

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Subconsultas de Varias Columnas

Hasta ahora ha escrito subconsultas de una sola fila y subconsultas de varias filas en las que sólo se devuelve una columna mediante la sentencia interna `SELECT` y ésta se utiliza para evaluar la expresión de la sentencia `SELECT` principal. Si desea comparar dos o más columnas, debe escribir una cláusula `WHERE` compuesta con operadores lógicos. Con las subconsultas de varias columnas, puede combinar condiciones `WHERE` duplicadas en una única cláusula `WHERE`.

Sintaxis

```
SELECT      column, column, ...
FROM table
WHERE (column, column, ...) IN
      (SELECT column, column, ...
       FROM   table
       WHERE  condition);
```

El gráfico de la diapositiva ilustra que los valores de `MANAGER_ID` y `DEPARTMENT_ID` de la consulta principal se van a comparar con los valores de `MANAGER_ID` y `DEPARTMENT_ID` recuperados por la subconsulta. Puesto que el número de columnas que se van a comparar es más de una, el ejemplo cualifica una subconsulta de varias columnas.

Nota: antes de ejecutar los ejemplos de siguientes diapositivas, debe crear la tabla `empl_demo` y rellenarla con datos mediante el archivo `lab_06_insert_empdata.sql`.

Comparaciones de Columnas

La comparaciones de varias columnas con subconsultas pueden ser:

- Comparaciones no pairwise
- Comparaciones pairwise

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Comparaciones Pairwise frente a No Pairwise

Las comparaciones de varias columnas con subconsultas pueden ser no pairwise o pairwise. Si tiene en cuenta el ejemplo “¿Mostrar los detalles de los empleados que trabajan en el mismo departamento y tienen el mismo gerente, como 'Daniel'?” obtendrá el resultado correcto con la siguiente sentencia:

```
SELECT first_name, last_name, manager_id, department_id
FROM empl_demo
WHERE manager_id IN (SELECT manager_id
                     FROM empl_demo
                     WHERE first_name = 'Daniel')
AND department_id IN (SELECT department_id
                     FROM empl_demo
                     WHERE first_name = 'Daniel');
```

Hay sólo un “Daniel” en la tabla EMPL_DEMO (Daniel Faviet, que gestiona el empleado 108 y trabaja en el departamento 100). Sin embargo, si las subconsultas devuelven más de una fila, el resultado puede no ser correcto. Por ejemplo, si ejecuta la misma consulta pero sustituye “John” por “Daniel,” obtendrá un resultado incorrecto. Esto se debe a que la combinación de department_id y manager_id es importante. Para obtener el resultado correcto para esta consulta, necesita una comparación pairwise.

Subconsulta de Comparación Pairwise

Mostrar los detalles de los empleados que gestiona el mismo gestor y que trabajan en el mismo departamento que los empleados con el nombre the “John”.

```
SELECT employee_id, manager_id, department_id
FROM   empl_demo
WHERE  (manager_id, department_id) IN
      (SELECT manager_id, department_id
       FROM empl_demo
       WHERE first_name = 'John')
AND first_name <> 'John';
```

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Subconsulta de Comparación Pairwise

El ejemplo de la diapositiva compara la combinación de valores de la columna `MANAGER_ID` y la columna `DEPARTMENT_ID` de cada fila de la tabla `EMPL_DEMO` con los valores de la columna `MANAGER_ID` y la columna `DEPARTMENT_ID` para los empleados de con un valor de `FIRST_NAME` de “John.” En primer lugar, se ejecuta la subconsulta para recuperar los valores de `MANAGER_ID` y `DEPARTMENT_ID` para los empleados con un valor de `FIRST_NAME` de “John”. Esta subconsulta devuelve lo siguiente:

	MANAGER_ID	DEPARTMENT_ID
1	108	100
2	123	50
3	100	80

Subconsulta de Comparación Pairwise (continuación)

Estos valores se comparan con las columnas `MANAGER_ID` y `DEPARTMENT_ID` de cada fila de la tabla `EMPL_DEMO`. Si coincide la combinación, se muestra la fila. En la salida, los registros de los empleados con un valor de `FIRST_NAME` de “John” no se mostrarán. A continuación, se muestra la salida de la consulta de la diapositiva:

	EMPLOYEE_ID	MANAGER_ID	DEPARTMENT_ID
1	113	108	100
2	112	108	100
3	111	108	100
4	109	108	100
5	195	123	50
6	194	123	50
7	193	123	50
8	192	123	50
9	140	123	50
10	138	123	50
11	137	123	50
12	149	100	80
13	148	100	80
14	147	100	80
15	146	100	80

Subconsulta de comparación No Pairwise

Mostrar los detalles de los empleados que gestiona el mismo gestor que los empleados con el nombre “John” y que trabajan en el mismo departamento que los empleados con el nombre “John”.

```
SELECT employee_id, manager_id, department_id
FROM empl_demo
WHERE manager_id IN
    (SELECT manager_id
     FROM empl_demo
     WHERE first_name = 'John')
AND department_id IN
    (SELECT department_id
     FROM empl_demo
     WHERE first_name = 'John')
AND first_name <> 'John';
```

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Subconsulta de comparación No Pairwise

El ejemplo muestra una comparación no pairwise de columnas. En primer lugar, se ejecuta la subconsulta para recuperar los valores de `MANAGER_ID` para los empleados con un valor de `FIRST_NAME` de “John”. Del mismo modo, se ejecuta la segunda subconsulta para recuperar los valores de `DEPARTMENT_ID` para los empleados con un valor de `FIRST_NAME` de “John”. Los valores recuperados de las columnas `MANAGER_ID` y `DEPARTMENT_ID` se comparan con las columnas `MANAGER_ID` y `DEPARTMENT_ID` de cada fila de la tabla `EMPL_DEMO`. Si la columna `MANAGER_ID` de la fila `EMPL_DEMO` coincide con alguno de los valores de `MANAGER_ID` recuperados por la subconsulta interna y si la columna `DEPARTMENT_ID` de la fila de la tabla `EMPL_DEMO` coincide con alguno de los valores de `DEPARTMENT_ID` recuperados por la segunda subconsulta, se muestra el registro.

Subconsulta de Comparación No Pairwise (continuación)

A continuación, se muestra la salida de la consulta de la diapositiva anterior:

	EMPLOYEE_ID	MANAGER_ID	DEPARTMENT_ID
1	109	108	100
2	111	108	100
3	112	108	100
4	113	108	100
5	120	100	50
6	121	100	50
7	122	100	50
8	123	100	50
9	124	100	50
10	137	123	50
11	138	123	50
12	140	123	50
13	192	123	50
14	193	123	50
15	194	123	50
16	195	123	50
17	146	100	80
18	147	100	80
19	148	100	80
20	149	100	80

Esta consulta recupera filas adicionales a la comparación pairwise (aquellas con la combinación de `manager_id=100` y `department_id=50` o `80`, aunque ningún empleado con el nombre “John” tenga dicha combinación).

Agenda

- Escritura de una subconsulta de varias columnas
- **Uso de subconsultas escalares en SQL**
- Solución de problemas con subconsultas correlacionadas
- Uso de los operadores `EXISTS` y `NOT EXISTS`
- Uso de la cláusula `WITH`

ORACLE®

Copyright © 2010, Oracle. Todos los derechos reservados.

Expresiones de Subconsultas Escalares

- Una expresión de subconsulta escalar es una subconsulta que devuelve exactamente un valor de columna de una fila.
- Las subconsultas escalares se pueden utilizar en:
 - La parte de condición y expresión de `DECODE` y `CASE`
 - Todas las cláusulas de `SELECT` excepto `GROUP BY`
 - La cláusula `SET` y `WHERE` de una sentencia `UPDATE`

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Subconsultas Escalares en SQL

Una subconsulta que devuelve exactamente un valor de columna de una fila también se denomina subconsulta escalar. Las subconsultas de varias columnas que se escriben para comparar dos o más columnas, mediante una cláusula `WHERE` compuesta y operadores lógicos, no se cualifican como subconsultas escalares.

El valor de la expresión de subconsulta escalar es el valor del elemento de la lista de selección de la subconsulta. Si la subconsulta devuelve 0 filas, el valor de la expresión de subconsulta escalar es `NULL`. Si la subconsulta devuelve más de una fila, el servidor de Oracle devuelve un error. El servidor de Oracle ha soportado siempre el uso de una subconsulta escalar en una sentencia `SELECT`.

Puede utilizar subconsultas escalares en:

- La parte de condición y expresión de `DECODE` y `CASE`
- Todas las cláusulas de `SELECT` excepto `GROUP BY`
- Las cláusulas `SET` y `WHERE` de una sentencia `UPDATE`

Sin embargo, las subconsultas escalares no son expresiones válidas en los siguientes casos:

- Como valores por defecto para columnas y expresiones hash para clusters
- En la cláusula `RETURNING` de las sentencias de lenguaje de manipulación de datos (DML)
- Como base de un índice basado en funciones
- En cláusulas `GROUP BY`, restricciones `CHECK` y condiciones `WHEN`
- En cláusulas `CONNECT BY`
- En sentencias no relacionadas con consultas, como `CREATE PROFILE`

Subconsultas Escalares: Ejemplos

- Subconsultas escalares en expresiones 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 en la cláusula 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 © 2010, Oracle. Todos los derechos reservados.

Subconsultas Escalares: Ejemplos

El primer ejemplo de la diapositiva demuestra que las subconsultas escalares se pueden utilizar en expresiones CASE. La consulta interna devuelve el valor 20, que es el ID de departamento del departamento cuyo ID de ubicación es 1800. La expresión CASE de la consulta externa utiliza el resultado de la consulta interna para mostrar el ID de empleado, el apellido y un valor de Canada o USA, en función de si el ID de departamento del registro recuperado por la consulta externa es 20 o no.



A continuación, se muestra el resultado del primer ejemplo de la diapositiva:

...

	EMPLOYEE_ID	LAST_NAME	LOCATION
1	198	OConnell	USA
2	199	Grant	USA
3	200	Whalen	USA
4	201	Hartstein	Canada
5	202	Fay	Canada
6	203	Mavris	USA

Subconsultas Escalares: Ejemplos (continuación)

El segundo ejemplo de la diapositiva demuestra que las subconsultas escalares se pueden utilizar en la cláusula `ORDER BY`. El ejemplo ordena la salida según el valor de `DEPARTMENT_NAME` haciendo coincidir `DEPARTMENT_ID` de la tabla `EMPLOYEES` con el de `DEPARTMENT_ID` de la tabla `DEPARTMENTS`. Esta comparación se realiza en una subconsulta escalar en la cláusula `ORDER BY`. A continuación, se muestra el resultado del segundo ejemplo:

 2	EMPLOYEE_ID	 2	LAST_NAME
1	205		Higgins
2	206		Gietz
3	200		Whalen
4	100		King
5	101		Kochhar
6	102		De Haan
7	112		Urman
8	108		Greenberg
9	109		Faviet

...

El segundo ejemplo utiliza una subconsulta correlacionada. En una subconsulta correlacionada, la subconsulta hace referencia a una columna de una tabla a la que se hace referencia en la sentencia principal. Las subconsultas correlacionadas se explican más adelante en esta lección.

Agenda

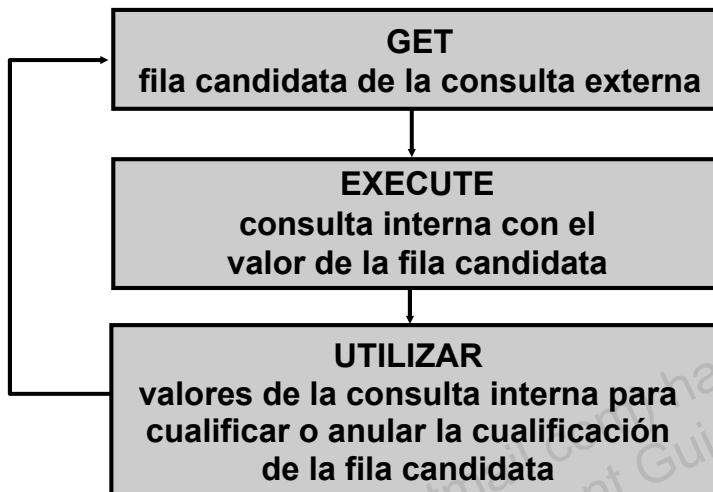
- Escritura de una subconsulta de varias columnas
- Uso de subconsultas escalares en SQL
- **Solución de problemas con subconsultas correlacionadas**
- Uso de los operadores `EXISTS` y `NOT EXISTS`
- Uso de la cláusula `WITH`

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Subconsultas Correlacionadas

Las subconsultas correlacionadas se utilizan para el procesamiento fila por fila. Cada subconsulta se ejecuta una vez para cada fila de la consulta externa.



ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Subconsultas Correlacionadas

El servidor de Oracle realiza una subconsulta correlacionada cuando la subconsulta hace referencia a una columna de una tabla a la que se hace referencia en la sentencia principal. Una subconsulta correlacionada se evalúa una vez para cada fila procesada por la sentencia principal. La sentencia principal puede ser una sentencia SELECT, UPDATE o DELETE.

Subconsultas Anidadas frente a Subconsultas Correlacionadas

Con una subconsulta anidada normal, la consulta SELECT interna se ejecuta en primer lugar y una vez, devolviendo valores que utilizará la consulta principal. Una subconsulta correlacionada, sin embargo, se ejecuta una vez para cada fila candidata considerada por la consulta externa. Es decir, la consulta interna está controlada por la consulta externa.

Ejecución de Subconsulta Anidada

- La consulta interna se ejecuta en primer lugar y busca un valor.
- La consulta externa se ejecuta una vez, con el valor de la consulta interna.

Ejecución de Subconsulta Correlacionada

- Obtenga una fila candidata (recuperada por la consulta externa).
- Ejecute la consulta interna con el valor de la fila candidata.
- Utilice los valores resultantes de la consulta interna para cualificar o anular la cualificación del candidato.
- Repita este procedimiento hasta que no quede ninguna fila candidata.

Subconsultas Correlacionadas

La subconsulta hace referencia a una columna de una tabla de la consulta principal.

```
SELECT column1, column2, ...  
FROM   table1 Outer_table  
WHERE  column1 operator  
        (SELECT column1, column2  
         FROM   table2  
         WHERE  expr1 =  
                Outer_table.expr2) ;
```

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Subconsultas Correlacionadas (continuación)

Una subconsulta correlacionada es una forma de leer todas las filas de una tabla y comparar los valores de cada una de ellas con los datos relacionados. Se utiliza cuando una subconsulta debe devolver un resultado diferente o un juego de resultados para cada fila candidata considerada por la consulta principal. Es decir, debe utilizar una subconsulta correlacionada para responder a una consulta de varias partes cuya respuesta depende del valor de cada fila procesada por la sentencia principal.

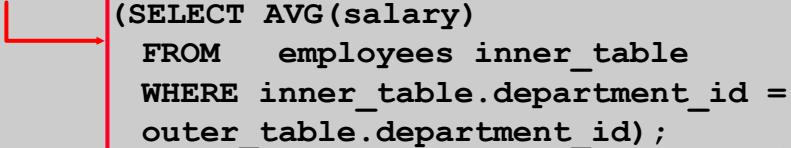
El servidor de Oracle realiza una subconsulta correlacionada cuando la subconsulta hace referencia a una columna de una tabla de la consulta principal.

Nota: puede utilizar los operadores ANY y ALL en una subconsulta correlacionada.

Uso de Subconsultas Correlacionadas

Buscar todos los empleados que ganan más del salario medio de su departamento.

```
SELECT last_name, salary, department_id
FROM   employees outer_table
WHERE  salary >
      (SELECT AVG(salary)
       FROM   employees inner_table
       WHERE  inner_table.department_id =
            outer_table.department_id);
```



Cada que una fila de la consulta externa se procesa, la consulta interna se evalúa.

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Uso de Subconsultas Correlacionadas

El ejemplo de la diapositiva determina qué empleados ganan más del salario medio de su departamento. En este caso, la subconsulta correlacionada calcula específicamente el salario medio de cada departamento.

Puesto que tanto la consulta interna como la externa utilizan la tabla `EMPLOYEES` en la cláusula `FROM`, se proporciona un alias a `EMPLOYEES` en la sentencia `SELECT` externa para una mayor claridad. El alias hace toda la sentencia `SELECT` más legible. Sin el alias, la consulta no funcionaría correctamente porque la sentencia no podría distinguir entre la columna de la tabla interna y la columna de la tabla externa.

Uso de Subconsultas Correlacionadas

Mostrar los detalles de aquellos empleados que han cambiado de cargo al menos dos veces.

```
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
1	200	Whalen	AD_ASST
2	101	Kochhar	AD_VP
3	176	Taylor	SA_REP

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Uso de Subconsultas Correlacionadas (continuación)

El ejemplo de la diapositiva muestra los detalles de aquellos empleados que han cambiado de cargo al menos dos veces. El servidor de Oracle evalúa la subconsulta correlacionada de la siguiente forma:

1. Seleccione una fila de la tabla especificada en la consulta externa. Ésta será la fila candidata actual.
2. Almacene el valor de la columna a la que se hace referencia en la subconsulta desde la fila candidata. (En el ejemplo de la diapositiva, la columna a la que se hace referencia en la subconsulta es E.EMPLOYEE_ID.)
3. Ejecute la subconsulta en la que su condición hace referencia al valor de la fila candidata de la consulta externa. (En el ejemplo de la diapositiva, la función de grupo COUNT(*) se evalúa según el valor de la columna E.EMPLOYEE_ID obtenido en el paso 2.)
4. Evalúe la cláusula WHERE de la consulta externa según los resultados de la subconsulta ejecutada en el paso 3. Esto determina si se selecciona la fila candidata para la salida. (En el ejemplo, el número de veces que un empleado ha cambiado de cargo, evaluado por la subconsulta, se compara con 2 en la cláusula WHERE de la consulta externa. Si se cumple la condición, se muestra el registro de ese empleado.)
5. Repita el procedimiento para la siguiente fila candidata de la tabla, y así sucesivamente, hasta que se hayan procesado todas las filas de la tabla.

La correlación se establece a través de un elemento de la consulta externa en la subconsulta. En este ejemplo, se compara EMPLOYEE_ID de la tabla de la subconsulta con EMPLOYEE_ID de la tabla de la consulta externa.

Agenda

- Escritura de una subconsulta de varias columnas
- Uso de subconsultas escalares en SQL
- Solución de problemas con subconsultas correlacionadas
- **Uso de los operadores EXISTS y NOT EXISTS**
- Uso de la cláusula WITH

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Uso del Operador EXISTS

- El operador EXISTS prueba la existencia de filas en el juego de resultados de la subconsulta.
- Si se encuentra un valor de fila de subconsulta:
 - La búsqueda no continúa en la consulta interna
 - La condición se marca como TRUE
- Si no se encuentra un valor de fila de subconsulta:
 - La condición se marca como FALSE
 - La búsqueda continúa en la consulta interna

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Operador EXISTS

Con las sentencias SELECT de anidamiento, todos los operadores lógicos son válidos. Además, puede utilizar el operador EXISTS. Este operador se suele utilizar con subconsultas correlacionadas para probar si un valor recuperado por la consulta externa existe en el juego de resultados de los valores recuperados por la consulta interna. Si la subconsulta devuelve al menos una fila, el operador devuelve TRUE. Si el valor no existe, se devuelve FALSE. En consecuencia, NOT EXISTS prueba si un valor recuperado por la consulta externa no forma parte del juego de resultados de los valores recuperados por la consulta interna.

Uso del Operador EXISTS

```
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
1	201	Hartstein	MK_MAN	20
2	205	Higgins	AC_MGR	110
3	100	King	AD_PRES	90
4	101	Kochhar	AD_VP	90
5	102	De Haan	AD_VP	90
6	103	Hunold	IT_PROG	60
7	108	Greenberg	FL_MGR	100
8	114	Raphaely	PU_MAN	30

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Uso del Operador EXISTS

El operador EXISTS garantiza que la búsqueda en la consulta interna no continúe cuando la condición encuentre al menos una coincidencia de gestor y número de empleado:

```
WHERE manager_id = outer.employee_id.
```

Observe que la consulta SELECT interna no tiene que devolver un valor concreto, por lo que se puede seleccionar una constante.

Buscar Todos los Departamentos que No Tienen Empleados

```
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
1	120 Treasury
2	130 Corporate Tax
3	140 Control And Credit
4	150 Shareholder Services
5	160 Benefits
6	170 Manufacturing
7	180 Construction

...

All Rows Fetched: 16

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Uso del Operador NOT EXISTS

Solución Alternativa

Se puede utilizar una construcción NOT IN como alternativa a un operador NOT EXISTS, como se muestra en el siguiente ejemplo:

```
SELECT department_id, department_name
FROM departments
WHERE department_id NOT IN (SELECT department_id
                           FROM employees);
```

All Rows Fetched: 0

Sin embargo, NOT IN se evalúa como FALSE si alguno de los miembros del juego es un valor nulo NULL. Por lo tanto, la consulta no devolverá ninguna fila aunque haya filas en la tabla departments que cumplan la condición WHERE.

Subconsulta Correlacionada UPDATE

Utilizar una subconsulta correlacionada para actualizar las filas de una tabla según las filas de otra tabla.

```
UPDATE table1 alias1
SET    column = (SELECT expression
                   FROM    table2 alias2
                   WHERE   alias1.column =
                           alias2.column);
```

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Subconsulta Correlacionada UPDATE

En el caso de la sentencia UPDATE, puede utilizar una subconsulta correlacionada para actualizar las filas de una tabla según las filas de otra tabla.

Uso de la Subconsulta Correlacionada UPDATE

- Desnormalizar la tabla EMPL6 mediante la adición de una columna para almacenar el nombre de departamento.
- Rellenar la tabla mediante una actualización 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 © 2010, Oracle. Todos los derechos reservados.

Subconsulta Correlacionada UPDATE (continuación)

El ejemplo de la diapositiva desnormaliza la tabla EMPL6 mediante la adición de una columna para almacenar el nombre de departamento y, a continuación, rellena la tabla mediante una actualización correlacionada.

A continuación, se muestra otro ejemplo de una actualización correlacionada.

Sentencia con Problemas

La tabla REWARDS contiene una lista de empleados que han excedido sus expectativas de rendimiento. Utilice una subconsulta correlacionada para actualizar las filas de la tabla EMPL6 según las filas de la tabla REWARDS:

```
UPDATE empl6
SET    salary = (SELECT empl6.salary + rewards.pay_raise
                  FROM    rewards
                  WHERE   employee_id =
                        empl6.employee_id
                  AND    payraise_date =
                        (SELECT MAX(payraise_date)
                         FROM    rewards
                         WHERE   employee_id = empl6.employee_id))
WHERE  empl6.employee_id
IN      (SELECT employee_id FROM rewards);
```


Subconsulta Correlacionada UPDATE (continuación)

En este ejemplo se utiliza la tabla REWARDS. La tabla REWARDS tiene las siguientes columnas: EMPLOYEE_ID, PAY_RAISE y PAYRAISE_DATE. Cada vez que un empleado obtiene un aumento de sueldo, se inserta un registro con detalles como el ID de empleado, el importe del aumento de sueldo y la fecha de recepción del aumento de sueldo en la tabla REWARDS. La tabla REWARDS puede contener más de un registro para un empleado. La columna PAYRAISE_DATE se utiliza para identificar el aumento de sueldo más reciente recibido por un empleado.

En el ejemplo, la columna SALARY de la tabla EMPL6 se actualiza para reflejar el último aumento de sueldo recibido por un empleado. Para ello, se agrega el salario actual del empleado con el correspondiente aumento de sueldo desde la tabla REWARDS.

Subconsulta Correlacionada DELETE

Utilizar una subconsulta correlacionada para suprimir filas de una tabla según las filas de otra tabla.

```
DELETE FROM table1 alias1
WHERE column operator
      (SELECT expression
       FROM table2 alias2
       WHERE alias1.column = alias2.column);
```

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Subconsulta Correlacionada DELETE

En el caso de una sentencia DELETE, puede utilizar una subconsulta correlacionada para suprimir sólo aquellas filas que sólo existen en la otra tabla. Si decide que sólo mantendrá los cuatro últimos registros del historial del cargo en la tabla JOB_HISTORY, cuando un empleado se transfiera a un quinto cargo, tendrá que suprimir la fila JOB_HISTORY más antigua consultando en la tabla JOB_HISTORY el registro MIN(START_DATE) del empleado. El siguiente código ilustra cómo se puede realizar la operación anterior mediante una subconsulta correlacionada DELETE:

```
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));
```

Uso de una Subconsulta Correlacionada DELETE

Utilizar una subconsulta correlacionada para suprimir sólo aquellas filas de la tabla `EMPL6` que también existen en la tabla `EMP_HISTORY`.

```
DELETE FROM empl6 E
WHERE employee_id =
      (SELECT employee_id
       FROM   emp_history
       WHERE  employee_id = E.employee_id);
```

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Subconsulta Correlacionada DELETE (continuación)

Ejemplo

En este ejemplo se utilizan dos tablas: Son las siguientes:

- La tabla `EMPL6`, que proporciona detalles de todos los empleados actuales.
- La tabla `EMP_HISTORY`, que proporciona detalles de los empleados anteriores.

`EMP_HISTORY` contiene datos relativos a los empleados anteriores, por lo que sería un error que el mismo registro del empleado existiera en las tablas `EMPL6` y `EMP_HISTORY`. Puede suprimir dichos registros erróneos mediante la subconsulta relativa mostrada en la diapositiva.

Agenda

- Escritura de una subconsulta de varias columnas
- Uso de subconsultas escalares en SQL
- Solución de problemas con subconsultas correlacionadas
- Uso de los operadores `EXISTS` y `NOT EXISTS`
- **Uso de la cláusula `WITH`**

ORACLE®

Copyright © 2010, Oracle. Todos los derechos reservados.

Cláusula WITH

- Con la cláusula `WITH`, puede utilizar el mismo bloque de consulta en una sentencia `SELECT` cuando se produce más de una vez en una consulta compleja.
- La cláusula `WITH` recupera los resultados de un bloque de consulta y lo almacena en el tablespace temporal del usuario.
- La cláusula `WITH` puede mejorar el rendimiento.

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Cláusula WITH

Con la cláusula `WITH`, puede definir un bloque de consulta antes de utilizarlo en una consulta. La cláusula `WITH` (denominada anteriormente `subquery_factoring_clause`) permite reutilizar el mismo bloque de consulta en una sentencia `SELECT` cuando se produce más de una vez en una consulta compleja. Esto es especialmente útil cuando una consulta tiene muchas referencias al mismo bloque de consulta y existen uniones y agregaciones.

Con la cláusula `WITH`, puede reutilizar la misma consulta cuando es muy costoso evaluar el bloque de consulta y se produce más de una vez en una consulta compleja. Con la cláusula `WITH`, el servidor de Oracle recupera los resultados de un bloque de consulta y lo almacena en el tablespace temporal del usuario. Esto puede mejorar el rendimiento.

Ventajas de la Cláusula WITH

- Facilita la lectura de la consulta.
- Evalúa una cláusula una sola vez, incluso si aparece varias veces en la consulta.
- En la mayoría de los casos, puede mejorar el rendimiento de grandes consultas.

Cláusula WITH: Ejemplo

Con la cláusula `WITH`, escriba una consulta para mostrar el nombre de departamento y el total de los salarios de aquellos departamentos cuyo salario total sea superior al salario medio de los departamentos.

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Cláusula WITH: Ejemplo

El problema que se muestra en la diapositiva podría necesitar cálculos intermedios:

1. Calcule el salario total de cada departamento y almacene el resultado mediante la cláusula `WITH`.
2. Calcule el salario medio de los departamentos y almacene el resultado mediante una cláusula `WITH`.
3. Compare el salario total calculado en el primer paso con el salario medio calculado en el segundo paso. Si el salario total de un determinado departamento es superior al salario medio de todos los departamentos, muestre el nombre de departamento y el salario total del departamento.

En la página siguiente se proporciona la solución a este problema.

Cláusula WITH: Ejemplo

```

WITH
dept_costs AS (
  SELECT d.department_name, SUM(e.salary) AS dept_total
  FROM   employees e JOIN departments d
  ON     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 © 2010, Oracle. Todos los derechos reservados.

Cláusula WITH: Ejemplo (continuación)

El código SQL de la diapositiva es un ejemplo de una situación en la que puede mejorar el rendimiento y escribir SQL más simplemente mediante la cláusula WITH. La consulta crea los nombres de consulta DEPT_COSTS y AVG_COST y, a continuación, los utiliza en el cuerpo de la consulta principal. Internamente, la cláusula WITH se resuelve como una vista en línea o una tabla temporal. El optimizador selecciona la resolución adecuada en función del costo o beneficio del almacenamiento temporal de los resultados de la cláusula WITH.

La salida generada por el código SQL de la diapositiva es el siguiente:

	DEPARTMENT_NAME	DEPT_TOTAL
1	Sales	304500
2	Shipping	156400

Notas de Uso de la Cláusula WITH

- Se utiliza sólo con las sentencias SELECT.
- El nombre de la consulta es visible para todos los bloques de consulta de elementos WITH (incluidos sus bloques de subconsulta) definidos después de ella y del propio bloque de consulta principal (incluidos sus bloques de subconsulta).
- Cuando el nombre de consulta es el mismo que el de una tabla existente, el analizador busca desde adentro afuera y el nombre del bloque de consulta tiene prioridad sobre el nombre de tabla.
- La cláusula WITH puede contener más de una consulta. Cada consulta se separa a continuación por una coma.

Cláusula WITH Recursiva

La cláusula WITH recursiva

- Permite la formulación de consultas recursivas.
- Crea una consulta con un nombre, denominado nombre de elemento de WITH recursiva.
- Contiene dos tipos de miembro de bloques de consulta: fijación y recursivo.
- Es compatible con ANSI.

**ORACLE**

Copyright © 2010, Oracle. Todos los derechos reservados.

Cláusula WITH Recursiva

En Oracle Database 11g Versión 2, la cláusula WITH se ha ampliado para permitir la formulación de consultas recursivas.

La cláusula WITH recursiva define una consulta recursiva con un nombre, el *nombre de elemento de WITH recursiva*. La definición de elemento de WITH recursiva debe contener al menos dos bloques de consulta: un miembro de fijación y un miembro recursivo. Puede haber varios miembros de fijación, pero sólo uno recursivo.

La cláusula WITH recursiva, Oracle Database 11g Versión 2, cumple *parcialmente* el estándar ANSI (American National Standards Institute). La cláusula WITH recursiva se puede utilizar para consultar datos jerárquicos como los gráficos de organización.

Cláusula WITH Recursiva: Ejemplo

FLIGHTS Table

	SOURCE	DESTIN	FLIGHT_TIME
1	San Jose	Los Angeles	1.3
2	New York	Boston	1.1
3	Los Angeles	New York	5.8

1

```
WITH Reachable_From (Source, Destin, TotalFlightTime) AS
(
    SELECT Source, Destin, Flight_time
    FROM Flights
    UNION ALL
    SELECT incoming.Source, outgoing.Destin,
           incoming.TotalFlightTime+outgoing.Flight_time
    FROM Reachable_From incoming, Flights outgoing
    WHERE incoming.Destin = outgoing.Source
)
SELECT Source, Destin, TotalFlightTime
FROM Reachable_From;
```

2

3

	SOURCE	DESTIN	TOTALFLIGHTTIME
1	San Jose	Los Angeles	1.3
2	New York	Boston	1.1
3	Los Angeles	New York	5.8
4	San Jose	New York	7.1
5	Los Angeles	Boston	6.9
6	San Jose	Boston	8.2

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Cláusula WITH Recursiva: Ejemplo

El ejemplo 1 de la diapositiva muestra los registros de una tabla **FLIGHTS** que describe los vuelos entre dos ciudades. Este ejemplo es específico de 11g versión R2 de Oracle Database.

Con la consulta del ejemplo 2, puede consultar la tabla **FLIGHTS** para mostrar el tiempo total del vuelo entre cualquier origen y el destino. La cláusula **WITH** de la consulta, que se denomina **Reachable From**, tiene una consulta **UNION ALL** con dos ramas. La primera rama es la rama de *fijación*, que selecciona todas las filas de la tabla **Flights**. La segunda rama es la rama recursiva. Une el contenido de **Reachable From** con la tabla **Flights** para buscar otras ciudades a las que se pueda viajar y las agrega al contenido de **Reachable From**. La operación terminará cuando la rama recursiva no encuentre más filas.

El ejemplo 3 muestra el resultado de la consulta que selecciona todo el contenido del elemento de la cláusula **WITH Reachable From**.

Para obtener más información, consulte:

- *Oracle Database SQL Language Reference 11g Release 2.0*
- *Oracle Database Data Warehousing Guide 11g Release 2.0*

Prueba

Con una subconsulta correlacionada, la sentencia `SELECT` interna controla la sentencia `SELECT` externa.

1. Verdadero
2. Falso

ORACLE®

Copyright © 2010, Oracle. Todos los derechos reservados.

Respuesta: 2

Resumen

En esta lección, debe haber aprendido lo siguiente:

- Una subconsulta de varias columnas devuelve más de una columna.
- Las comparaciones de varias columnas pueden ser pairwise o no pairwise.
- Una subconsulta de varias columnas se puede utilizar también en la cláusula `FROM` de una sentencia `SELECT`.

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Resumen

Puede utilizar las subconsultas de varias columnas para combinar varias condiciones `WHERE` en una única cláusula `WHERE`. Las comparaciones de columnas de una subconsulta de varias columnas pueden ser comparaciones pairwise o no pairwise.

Puede utilizar una subconsulta para definir una tabla en la que se podrá operar mediante una consulta de contenido.

Las subconsultas escalares se pueden utilizar en:

- La parte de condición y expresión de `DECODE` y `CASE`
- Todas las cláusulas de `SELECT` excepto `GROUP BY`
- Una cláusula `SET` y `WHERE` de la sentencia `UPDATE`

Resumen

- Las subconsultas correlacionadas son útiles cuando una subconsulta debe devolver un resultado diferente para cada fila candidata.
- El operador `EXISTS` es un operador booleano que prueba la presencia de un valor.
- Las subconsultas correlacionadas se pueden utilizar con sentencias `SELECT`, `UPDATE` y `DELETE`.
- Puede utilizar la cláusula `WITH` para usar el mismo bloque de consulta en una sentencia `SELECT` que se produce más de una vez.

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Resumen (continuación)

El servidor de Oracle realiza una subconsulta correlacionada cuando la subconsulta hace referencia a una columna de una tabla a la que se hace referencia en la sentencia principal. Una subconsulta correlacionada se evalúa una vez para cada fila procesada por la sentencia principal. La sentencia principal puede ser una sentencia `SELECT`, `UPDATE` o `DELETE`. Con la cláusula `WITH`, puede reutilizar la misma consulta cuando es muy costoso volver a evaluar el bloque de consulta y se produce más de una vez en una consulta compleja.

Práctica 6: Visión General

En esta práctica se abordan los siguientes temas:

- Creación de subconsultas de varias columnas
- Escritura de subconsultas correlacionadas
- Uso del operador `EXISTS`
- Uso de subconsultas escalares
- Uso de la cláusula `WITH`

The Oracle logo, consisting of the word "ORACLE" in a bold, sans-serif font, with a registered trademark symbol (®) to its upper right.

Copyright © 2010, Oracle. Todos los derechos reservados.

Práctica 6: Visión General

En esta práctica, escribirá subconsultas de varias columnas y subconsultas escalares y correlacionadas. También resolverá problemas escribiendo la cláusula `WITH`.

Soporte para Expresiones Normales

ORACLE®

Copyright © 2010, Oracle. Todos los derechos reservados.

Objetivos

Al finalizar esta lección, debería estar capacitado para lo siguiente:

- Mostrar las ventajas del uso de expresiones regulares
- Utilizar expresiones regulares para buscar, hacer coincidir y sustituir cadenas

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Objetivos

En esta lección, aprenderá a utilizar la función de soporte de expresiones regulares. El soporte de expresiones regulares está disponible en SQL y PL/SQL.

Agenda

- **Introducción a expresiones regulares**
- Uso de metacaracteres con expresiones regulares
- Uso de funciones de expresiones regulares:
 - REGEXP_LIKE
 - REGEXP_REPLACE
 - REGEXP_INSTR
 - REGEXP_SUBSTR
- Acceso a subexpresiones
- Uso de la función REGEXP_COUNT
- Expresiones regulares y restricciones de control

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

¿Qué Son Expresiones Regulares?

- Utilice las expresiones regulares para buscar (y manipular) patrones simples y complejos en datos de cadena mediante convenciones de sintaxis estándar.
- Utilice un juego de condiciones y funciones SQL para buscar y manipular cadenas en SQL y PL/SQL.
- Especifique una expresión regular mediante:
 - Metacaracteres, que son operadores que especifican los algoritmos de búsqueda
 - Literales, que son los caracteres que está buscando

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

¿Qué Son Expresiones Regulares?

Oracle Database proporciona soporte para expresiones regulares. La implantación cumple con el estándar POSIX (Portable Operating System for UNIX), controlado por IEEE (Institute of Electrical and Electronics Engineers), para la sintaxis y semántica de coincidencia de datos ASCII. Las capacidades multilingües de Oracle amplían las capacidades de coincidencia de los operadores más allá del estándar POSIX. Las expresiones normales son un método de descripción de patrones tanto simples como complejos para la búsqueda y la manipulación.

La manipulación y la búsqueda de cadenas contribuyen a un amplio porcentaje de la lógica dentro de una aplicación basada en web. Su uso va desde una tarea simple, como la búsqueda de la palabra "San Francisco" en un texto especificado, la tarea compleja de extracción de todas las URL del texto hasta la tarea más compleja de búsqueda de todas las palabras cuyo segundo carácter sea una vocal.

Junto con SQL nativo, el uso de expresiones regulares permite operaciones muy potentes de búsqueda y de manipulación en datos almacenados en Oracle Database. Puede utilizar esta función para resolver problemas de forma sencilla que de otra manera implicaría una programación compleja.

Ventajas de Utilizar Expresiones Regulares

Las expresiones regulares permiten implantar la lógica de coincidencia compleja en la base de datos con las siguientes ventajas:

- Al centralizar la lógica de coincidencia en Oracle Database, evita el procesamiento complejo de cadenas de juegos de resultados de SQL mediante aplicaciones de nivel medio.
- Mediante el uso de expresiones regulares del servidor para aplicar restricciones, elimina la necesidad de codificar la lógica de validación de datos en el cliente.
- Las condiciones y funciones de expresiones regulares PL/SQL y SQL incorporadas hacen que las manipulaciones de cadenas sean más sencillas y potentes que en las versiones anteriores de Oracle Database 11g.



Copyright © 2010, Oracle. Todos los derechos reservados.

Ventajas de Utilizar Expresiones Regulares

Las expresiones regulares son un componente de procesamiento de texto potente de lenguajes de programación como PERL y Java. Por ejemplo, un script de PERL puede procesar cada archivo HTML en un directorio, leer su contenido en una variable escalar como una única cadena y, a continuación, utilizar expresiones regulares para buscar URL en la cadena. Uno de los motivos por el que muchos desarrolladores escriben en PERL, es que se trata de una funcionalidad de coincidencia de patrones sólida. El soporte de Oracle de expresiones regulares permite a los desarrolladores implantar la lógica de coincidencia compleja en la base de datos. Las expresiones regulares se introdujeron en Oracle Database 10g.

Uso de Funciones y Condiciones de Expresiones Regulares en SQL y PL/SQL

Nombre de Función o Condición	Descripción
REGEXP_LIKE	Es similar al operador LIKE, pero utiliza la coincidencia de expresiones regulares en lugar de la coincidencia de patrones simple (condición).
REGEXP_REPLACE	Busca un patrón de expresión normal y lo reemplaza por una cadena de sustitución.
REGEXP_INSTR	Busca una cadena para un patrón de expresión regular y devuelve la posición de la coincidencia encontrada.
REGEXP_SUBSTR	Busca un patrón de expresión regular dentro de una cadena dada y extrae la subcadena coincidente.
REGEXP_COUNT	Devuelve el número de veces que se ha encontrado una coincidencia de patrón en una cadena de entrada.

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Uso de Funciones y Condiciones de Expresiones Regulares en SQL y PL/SQL

Oracle Database proporciona un juego de funciones SQL que puede utilizar para buscar y manipular cadenas mediante expresiones regulares. Utilice estas funciones en un literal de texto, variable de enlace o cualquier columna que contenga datos de caracteres como CHAR, NCHAR, CLOB, NCLOB, NVARCHAR2 y VARCHAR2 (pero no LONG). Una expresión regular debe ir entre comillas simples. Esto garantiza que la función SQL interpreta la expresión completa y puede mejorar la legibilidad del código.

- **REGEXP_LIKE:** Esta condición busca una columna con caracteres para un patrón. Utilice esta condición en la cláusula WHERE de una consulta para devolver las filas coincidentes con la expresión regular especificada.
- **REGEXP_REPLACE:** Esta función busca un patrón en una columna de caracteres y sustituye cada incidencia de dicho patrón con el patrón especificado.
- **REGEXP_INSTR:** Esta función busca una cadena para una determinada incidencia de un patrón de expresión regular. Especifique las incidencias que desea buscar y la posición inicial desde la que buscar. Esta función devuelve un entero que indica la posición en la cadena en la que se encuentra la coincidencia.
- **REGEXP_SUBSTR:** Esta función devuelve la coincidencia de subcadena actual con el patrón de expresión regular especificado.
- **REGEXP_COUNT:** Esta función, introducida en 11g versión 2, devuelve el número de veces que se encuentra una coincidencia de patrón en la cadena de entrada.

Agenda

- Introducción a expresiones regulares
- **Uso de metacaracteres con expresiones regulares**
- Uso de funciones de expresiones regulares:
 - REGEXP_LIKE
 - REGEXP_REPLACE
 - REGEXP_INSTR
 - REGEXP_SUBSTR
- Acceso a subexpresiones
- Uso de la función REGEXP_COUNT

ORACLE®

Copyright © 2010, Oracle. Todos los derechos reservados.

¿Que Son Metacaracteres?

- Los metacaracteres son caracteres especiales con un significado especial, como un comodín, un carácter repetitivo, un carácter no coincidente o un rango de caracteres.
- Puede utilizar varios símbolos de metacaracteres predefinidos en la coincidencia de patrones.
- Por ejemplo, la expresión regular `^(f|ht)tps?:$` busca lo siguiente desde el principio de la cadena:
 - Los literales `f` o `ht`
 - El literal `t`
 - El literal `p`, seguido opcionalmente por el literal `s`
 - El literal dos puntos “:” al final de la cadena.

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

¿Que Son Metacaracteres?

La expresión regular de la diapositiva coincide con las cadenas `http:`, `https:`, `ftp:` y `ftp:` strings.

Nota: para obtener una lista de metacaracteres de expresiones regulares, consulte *Oracle Database Advanced Application Developer's Guide for 10g or 11g (Guía del Desarrollador de Aplicaciones Avanzadas de Oracle Database para 10g u 11g)*.

Uso de Metacaracteres con Expresiones Normales

Sintaxis	Descripción
.	Coincide con cualquier carácter en el juego de caracteres soportados, excepto NULL.
+	Coincide con una o más incidencias.
?	Coincide con ninguna o una ocurrencia.
*	Coincide con ninguna o más incidencias de una subexpresión anterior.
{ <i>m</i> }	Coincide exactamente con <i>m</i> incidencias de la expresión anterior.
{ <i>m</i> , }	Coincide al menos con <i>m</i> incidencias de la subexpresión anterior.
{ <i>m</i> , <i>n</i> }	Coincide al menos con <i>m</i> , pero no más de <i>n</i> , incidencias de la subexpresión anterior.
[...]	Coincide con cualquier carácter único de la lista entre corchetes.
	Coincide con una de las alternativas.
(. . .)	Trata la expresión entre paréntesis como una unidad. La subexpresión puede ser una cadena de literales o una expresión compleja con operadores.

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Uso de Metacaracteres en Funciones de Expresiones Regulares

Cualquier carácter, “.”: **a.b** coincide con las cadenas **abb**, **acb** y **adb**, pero no con **acc**.

Uno o más, “+”: **a+** coincide con las cadenas **a**, **aa** y **aaa**, pero no coincide con **bbb**.

Cero o uno, “?”: **ab?c** coincide con las cadenas **abc** y **ac**, pero no coincide con **abbc**.

Cero o más, “*”: **ab*c** coincide con las cadenas **ac**, **abc** y **abbc**, pero no coincide con **abb**.

Recuento exacto “{*m*}”: **a{3}** coincide con las cadenas **aaa**, pero no coincide con **aa**.

Al menos el recuento, “{*m*,}”: **a{3,}** coincide con las cadenas **aaa** y **aaaa**, pero no con **aa**.

Entre recuento, “{*m*,*n*}”: **a{3,5}** coinciden con las cadenas **aaa**, **aaaa** y **aaaaa**, pero no con **aa**.

Lista de caracteres coincidentes, “[...]”: **[abc]** coincide con el primer carácter en las cadenas **all**, **bill** y **cold**, pero no coincide con los caracteres en **doll**.

O, “|”: **a|b** coincide con el carácter **a** o **b**.

Subexpresión, “(...)”: **(abc)?def** coincide con la cadena opcional **abc**, seguida de **def**. La expresión coincide con **abcdefghi** y **def**, pero no coincide con **ghi**. La subexpresión puede ser una cadena de literales o una expresión compleja con operadores.

Uso de Metacaracteres con Expresiones Normales

Sintaxis	Descripción
<code>^</code>	Coincide con el principio de una cadena.
<code>\$</code>	Coincide con el final de una cadena.
<code>\</code>	Trata el metacarácter posterior en la expresión como un literal.
<code>\n</code>	Coincide con la <i>n</i> (1–9) subexpresión anterior de lo introducido entre paréntesis. Los paréntesis hacen que se recuerde una expresión; una referencia hacia atrás hace referencia a la misma.
<code>\d</code>	Carácter dígito.
<code>[:class:]</code>	Coincide con los caracteres que pertenecen a la clase de caracteres POSIX especificada.
<code>[^:class:]</code>	Coincide con cualquier carácter único que <i>no</i> esté en la lista entre corchetes.

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Uso de Metacaracteres en Funciones de Expresiones Regulares (continuación)

Principio/fin de la fijación de línea, “`^`” y “`$`”: `^def` coincide con `def` en la cadena `defghi` pero no coincide con `def` en `abcdef`. `def$` coincide con `def` en la cadena `abcdef` pero no coincide con `def` en la cadena `defghi`.

Carácter de escape “`\`”: `\+` busca un signo `+`. Coincide con el carácter de signo más en la cadena `abc+def`, pero no coincide con `Abcdef`.

Referencia hacia atrás, “`\n`”: `(abc|def)xy\1` coincide con las cadenas `abcxyabc` y `defxydef`, pero no coincide con `abcxydef` ni `abcxy`. Una referencia hacia atrás permite buscar una cadena repetida sin conocer la cadena real con anterioridad. Por ejemplo, la expresión `^(.*)\1$` coincide con una línea con dos instancias adyacentes de la misma cadena.

Carácter de dígito, “`\d`”: La expresión `^[\d{3}\] \d{3}-\d{4}$` coincide con `[650] 555-1212` pero no con `650-555-1212`.

Clase de caracteres, “`[:class:]`”: `[:upper:]+` busca uno o más caracteres en mayúscula consecutivos. Coincide con `DEF` en la cadena `abcDEFghi` pero no con la cadena `abcdefghijkl`.

Lista de caracteres no coincidentes (o clase), “`[^...]`”: `[^abc]` coincide con el carácter `d` en la cadena `abcdef`, pero no con `a`, `b` ni `c`.

Agenda

- Introducción a expresiones regulares
- Uso de metacaracteres con expresiones regulares
- **Uso de funciones de expresiones regulares:**
 - REGEXP_LIKE
 - REGEXP_REPLACE
 - REGEXP_INSTR
 - REGEXP_SUBSTR
- Acceso a subexpresiones
- Uso de la función REGEXP_COUNT

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Condiciones y Funciones de Expresiones Regulares: Sintaxis

```
REGEXP_LIKE (source_char, pattern [,match_option]
```

```
REGEXP_INSTR (source_char, pattern [, position  
[, occurrence [, return_option  
[, match_option [, subexpr]]]])
```

```
REGEXP_SUBSTR (source_char, pattern [, position  
[, occurrence [, match_option  
[, subexpr]]]])
```

```
REGEXP_REPLACE (source_char, pattern [,replacestr  
[, position [, occurrence  
[, match_option]]]])
```

```
REGEXP_COUNT (source_char, pattern [, position  
[, occurrence [, match_option]]]])
```

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Condiciones y Funciones de Expresiones Regulares: Sintaxis

La sintaxis de las condiciones y funciones de expresiones regulares es la siguiente:

- **source_char**: expresión de carácter que sirve como valor de búsqueda.
- **pattern**: expresión regular, literal de texto.
- **occurrence**: entero positivo que indica la incidencia de patrón que se debe buscar en el servidor de Oracle **source_char**. El valor por defecto es 1.
- **position**: entero positivo que indica el carácter de **source_char** dónde debe empezar a buscar el servidor de Oracle. El valor por defecto es 1.
- **return_option**:
 - 0: devuelve la posición del primer carácter de la incidencia (por defecto).
 - 1: devuelve la posición del carácter que sigue a la incidencia.
- **Replacestr**: cadena de carácter que sustituye el patrón.
- **match_parameter**:
 - “c”: utiliza una coincidencia sensible a mayúsculas y minúsculas (por defecto).
 - “i”: utiliza una coincidencia no sensible a mayúsculas y minúsculas.
 - “n”: permite el operador de coincidencia de cualquier carácter.
 - “m”: trata la cadena origen como cadena de varias líneas.
- **subexpr**: fragmento del patrón entre paréntesis. Las subexpresiones se explican más adelante en esta misma lección.

Realización de una Búsqueda Básica mediante la Condición REGEXP_LIKE

```
REGEXP_LIKE(source_char, pattern [, match_parameter ])
```

```
SELECT first_name, last_name
FROM employees
WHERE REGEXP_LIKE (first_name, '^Ste(v|ph)en$');
```

	FIRST_NAME	LAST_NAME
1	Steven	King
2	Steven	Markle
3	Stephen	Stiles

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Realización de una Búsqueda Básica mediante la Condición REGEXP_LIKE

REGEXP_LIKE es similar a la condición LIKE, excepto que REGEXP_LIKE realiza coincidencias de expresiones regulares en lugar de las coincidencias de patrones simples que realiza LIKE. Esta condición evalúa cadenas mediante caracteres como define el juego de caracteres de entrada.

Ejemplo de REGEXP_LIKE

En esta consulta, al contrario que la tabla EMPLOYEES, se muestran todos los empleados cuyos nombres contengan Steven o Stephen. En la expresión utilizada '^Ste(v|ph)en\$':

- ^ indica el principio de la expresión
- \$ indica el final de la expresión
- | indica cualquiera/o

Sustitución de Patrones mediante la Función REGEXP_REPLACE

```
REGEXP_REPLACE(source_char, pattern [,replacestr  
[, position [, occurrence [, match_option]]]])
```

```
SELECT REGEXP_REPLACE(phone_number, '\.', '-') AS phone  
FROM employees;
```

Original

	LAST_NAME	PHONE
1	OConnell	650.507.9833
2	Grant	650.507.9844
3	Whalen	515.123.4444
4	Hartstein	515.123.5555

Resultados parciales

	LAST_NAME	PHONE
1	OConnell	650-507-9833
2	Grant	650-507-9844
3	Whalen	515-123-4444
4	Hartstein	515-123-5555

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Sustitución de Patrones mediante la Función REGEXP_REPLACE

Con la función REGEXP_REPLACE, vuelva a formatear el número de teléfono para sustituir el delimitador de punto (.) por un delimitador de guión (-). A continuación se muestra una explicación de cada uno de los elementos utilizados en el ejemplo de la expresión regular:

- phone_number es la columna de origen.
- '\.' es el patrón de búsqueda.
 - Utilice comillas simples (' ') para buscar el punto (.) del carácter literal.
 - Utilice una barra invertida (\) para buscar un carácter que normalmente se trata como metacarácter.
- '-' es la cadena de sustitución.

Búsqueda de Patrones mediante la Función REGEXP_REPLACE

```
REGEXP_INSTR (source_char, pattern [, position [,
occurrence [, return_option [, match_option]]]])
```

```
SELECT street_address,
REGEXP_INSTR(street_address,'[[:alpha:]]') AS
  First_Alpha_Position
FROM locations;
```

	STREET_ADDRESS	FIRST_ALPHA_POSITION
1	1297 Via Cola di Rie	6
2	93091 Calle della Testa	7
3	2017 Shinjuku-ku	6
4	9450 Kamiya-cho	6

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Búsqueda de Patrones mediante la Función REGEXP_REPLACE

En este ejemplo, la función REGEXP_INSTR se utiliza para buscar la dirección de la ubicación del primer carácter alfabético, independientemente de si está en mayúscula o minúscula. Tenga en cuenta que `[<class>:]` implica una clase de caracteres y coincide con cualquier carácter en dicha clase; `[[:alpha:]]` coincide con cualquier carácter alfabético. Se muestran los resultados parciales.

En la expresión utilizada en la consulta `'[[:alpha:]]'`:

- `[` inicia la expresión
- `[[:alpha:]]` indica la clase de caracteres alfabéticos
- `]` finaliza la expresión

Nota: el operador de la clase de caracteres POSIX permite buscar una expresión en una lista de caracteres que sea miembro de una clase de caracteres POSIX. Puede utilizar este operador para buscar formato específico como caracteres en mayúscula o puede buscar caracteres especiales como caracteres de dígitos o puntuación. Está soportado el juego completo de las clases de caracteres POSIX. Utilice la sintaxis `[[:class:]]`, donde `class` es el nombre de la clase de caracteres POSIX que buscar. La siguiente expresión regular busca uno o más caracteres en mayúscula consecutivos: `[[:upper:]]+`.

Búsqueda de Patrones mediante la Función REGEXP_SUBSTR

```
REGEXP_SUBSTR (source_char, pattern [, position
                [, occurrence [, match_option]])
```

```
SELECT REGEXP_SUBSTR(street_address , ' [^ ]+ ') AS Road
FROM locations;
```

	ROAD
1	Via
2	Calle
3	(null)
4	(null)
5	Jabberwocky

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Búsqueda de Patrones mediante la Función REGEXP_SUBSTR

En este ejemplo, se extraen los nombres de vías de la tabla LOCATIONS. Para ello, se devuelve el contenido en la columna STREET_ADDRESS que se encuentra después del primer espacio mediante la función REGEXP_SUBSTR. En la expresión utilizada en la consulta ' [^]+ ':

- [inicia la expresión
- ^ indica NO
- indica espacio
-] finaliza la expresión
- + indica 1 o más
- indica espacio

Agenda

- Introducción a expresiones regulares
- Uso de metacaracteres con expresiones regulares
- Uso de funciones de expresiones regulares:
 - REGEXP_LIKE
 - REGEXP_REPLACE
 - REGEXP_INSTR
 - REGEXP_SUBSTR
- **Acceso a subexpresiones**
- Uso de la función REGEXP_COUNT

ORACLE

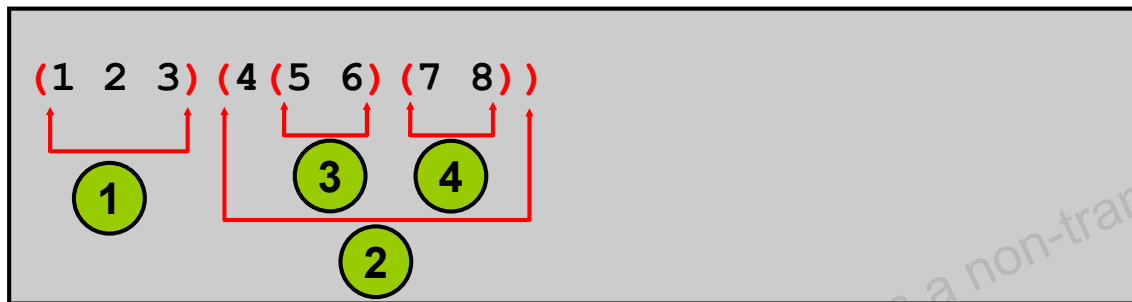
Copyright © 2010, Oracle. Todos los derechos reservados.

Subexpresiones

Examine esta expresión:

```
(1 2 3) (4 (5 6) (7 8))
```

Las subexpresiones son:



ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Subexpresiones

Oracle Database 11g proporciona el parámetro de soporte de expresiones regulares para acceder a una subexpresión. En el ejemplo de la diapositiva, se muestra una cadena de dígitos. Los paréntesis identifican las subexpresiones en la cadena de dígitos. La lectura de izquierda a derecha y desde los paréntesis externos a los internos, las subexpresiones en la cadena de dígitos son:

1. 123
2. 45678
3. 56
4. 78

Puede buscar aquellas subexpresiones con las funciones REGEXP_INSTR y REGEXP_SUBSTR.

Uso de Subexpresiones con el Soporte para Expresiones Regulares

```

SELECT
  REGEXP_INSTR
① ('0123456789',      -- source char or search value
② '(123)(4(56)(78))', -- regular expression patterns
③ 1,                  -- position to start searching
④ 1,                  -- occurrence
⑤ 0,                  -- return option
⑥ 'i',                -- match option (case insensitive)
⑦ 1)                  -- sub-expression on which to search
  "Position"
FROM dual;

```

Position
1
2

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Uso de Subexpresiones con el Soporte para Expresiones Normales

REGEXP_INSTR y REGEXP_SUBSTR tienen un parámetro opcional SUBEXPR que permite dirigirse a una subcadena determinada de la expresión regular que se está evaluando.

En el ejemplo que muestra la diapositiva, puede que desee buscar el primer patrón de subexpresión de la lista de subexpresiones. El ejemplo mostrado identificar varios parámetros para la función REGEXP_INSTR. Este ejemplo es específico de 11g versión R2 de Oracle Database.

1. Se identifica la cadena que está buscando.
2. Se identifican las subexpresiones. La primera subexpresión es 123. La segunda subexpresión es 45678, la tercera 56 y la cuarta 78.
3. El tercer parámetro identifica desde qué posición iniciar la búsqueda.
4. El cuarto parámetro identifica la incidencia del patrón que desea buscar. 1 significa buscar la primera incidencia.
5. El quinto parámetro es la opción de devolución. Se trata la posición del primer carácter de la incidencia. (Si especifica 1, se devuelve la posición del carácter que sigue a la incidencia).
6. El sexto parámetro identifica si la búsqueda debe ser sensible a mayúscula y minúsculas.
7. El último parámetro es el parámetro agregado en Oracle Database 11g. Este parámetro especifica la subexpresión que desea buscar. En el ejemplo mostrado, está buscando la primera subexpresión, que es 123.

¿Por qué Acceder a la n Subexpresión?

- Un uso más realista: secuenciación de ADN
- Es posible que necesite buscar un subpatrón que identifique una proteína necesaria para la inmunidad en el ADN del ratón.

```
SELECT
  REGEXP_INSTR('ccacctttccctccactcctcacgttctcacctgtaaagcgccctc
cctcatcccatgcccccttaccctgcagggtagagtaggctagaaccagagagctccaagc
tccatctgtggagaggtgccatccttgggctgcagagagaggagaatttgcccaaagctgcc
tgcagagcttcaccacccttagtctcacaaagccttgagttcatagcatttcttgagtttca
ccctgccagcaggacactgcagcacccaaagggctccaggagtaggggtgccctcaagag
gctcttgggtctgatggccacatcctggaattgttttcaagttgatggtcacagccctgagggc
atgtaggggctggggatgcgctctgctctgctctcctcctgaaccctgaaccctctggc
taccacagagcacttagagccag',
    '(gtc(tcac)(aaag))',
    1, 1, 0, 'i',
    1) "Position"
FROM dual;
```

Position
1 195

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

¿Por qué Acceder a la n Subexpresión?

En ciencias de la vida, es posible que necesite extraer los desplazamientos de las coincidencias de subexpresiones de una secuencia de ADN para un procesamiento posterior. Por ejemplo, puede que tenga que buscar una secuencia de proteínas específica, como el desplazamiento inicial de la secuencia de ADN precedida por `gtc` y seguida por `tcac` y por `aaag`. Para cumplir este objetivo, puede utilizar la función `REGEXP_INSTR`, que devuelve la posición en la que se encuentra una coincidencia.

En el ejemplo de la diapositiva, se devuelve la posición de la primera subexpresión (`gtc`). `gtc` aparece a partir de la posición 195 de la cadena de ADN. Este ejemplo es específico de 11g R2 versión de Oracle Database.

Si modifica el ejemplo de la diapositiva para buscar la segunda subexpresión (`tcac`), la consulta da como resultado la siguiente salida. `tcac` aparece a partir de la posición 198 de la cadena de ADN.

Position
1 198

Si modifica el ejemplo de la diapositiva para buscar la tercera subexpresión (`aaag`), la consulta da como resultado la siguiente salida. `aaag` aparece a partir de la posición 202 de la cadena de ADN.

Position
1 202

REGEXP_SUBSTR: Ejemplo

```

SELECT
  REGEXP_SUBSTR
  ① ('acgctgcactgca', -- source char or search value
    ② 'acg(.*)gca',   -- regular expression pattern
    ③ 1,              -- position to start searching
    ④ 1,              -- occurrence
    ⑤ 'i',            -- match option (case
    ⑥ insensitive)    -- insensitive
    1)                -- sub-expression
  "Value"
FROM dual;

```

Value
1 ctgcact

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

REGEXP_SUBSTR: Ejemplo

En el ejemplo de la diapositiva:

1. acgctgcactgca es el origen que se va a buscar.
2. acg(.*)gca es el patrón que se va a buscar. Busque acg seguido de gca con los caracteres potenciales entre acg y gca.
3. Inicie la búsqueda en el primer carácter del origen.
4. Busque la primera incidencia del patrón.
5. Utilice una coincidencia no sensible a mayúsculas y minúsculas en el origen.
6. Utilice un valor entero no negativo que identifique la *n* subexpresión a la que se va a dirigir. Se trata de un parámetro de subexpresión. En este ejemplo, 1 indica la primera subexpresión. Puede utilizar un valor de 0-9. Un cero significa que no se ha dirigido ninguna subexpresión. El valor por defecto de este parámetro es 0.

Este ejemplo es específico de 11g versión R2 de Oracle Database.

Agenda

- Introducción a expresiones regulares
- Uso de metacaracteres con expresiones regulares
- Uso de funciones de expresiones regulares:
 - REGEXP_LIKE
 - REGEXP_REPLACE
 - REGEXP_INSTR
 - REGEXP_SUBSTR
- Acceso a subexpresiones
- **Uso de la función REGEXP_COUNT**

ORACLE®

Copyright © 2010, Oracle. Todos los derechos reservados.

Uso de la Función REGEXP_COUNT

```
REGEXP_COUNT (source_char, pattern [, position
              [, occurrence [, match_option]])
```

```
SELECT REGEXP_COUNT (
  'ccacctttccctccactcctcacgtttctcacctgttaaagcgtccctccctcatcccatgcccccttaccctgcag
  ggtagagtaggctagaaaccagagagctccaagctccatctgtggagaggtgccatccttgggctgcagagagaggag
  aatttgcccaaagctgcctgcagagcttcaccacccttagtctcacaagccttgagttcatagcatttcttgagtt
  ttcacctgcccagcaggacactgcagcaccacaaagggcttccagagtaggggtgccctcaagaggctcttgggtc
  tgatggccacatcctggaattgtttcaagttgatgggtcacagccctgaggcatgtagggcggtgggagtgcgctctg
  ctctgctctcctcctgaacccctgaacccctctggctacccagagcacttagagccag' ,
  'gtc') AS Count
FROM dual;
```

COUNT
4

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Uso de la Función REGEXP_COUNT

La función REGEXP_COUNT evalúa cadenas mediante caracteres como define el juego de caracteres de entrada. Devuelve un entero que indica el número de incidencias del patrón. Si no se encuentra ninguna coincidencia, la función devuelve 0.

En el ejemplo de la diapositiva, el número de incidencias de una subcadena de ADN se determina mediante la función REGEXP_COUNT. Este ejemplo es específico de 11g versión R2 de Oracle Database.

El siguiente ejemplo muestra que el número de veces que se produce el patrón 123 en la cadena 123123123123 es tres veces. La búsqueda se inicia a partir de la segunda posición de la cadena.

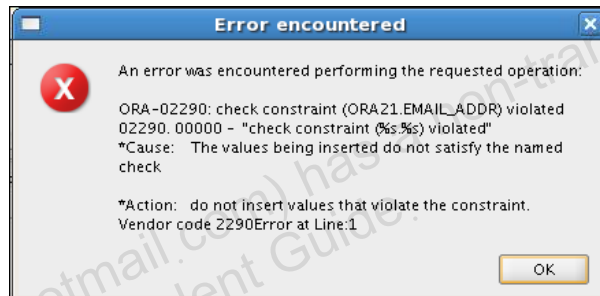
```
SELECT REGEXP_COUNT
  ('123123123123', -- source char or search value
   '123',          -- regular expression pattern
   2,              -- position where the search should start
   'i')            -- match option (case insensitive)
  AS Count
FROM dual;
```

COUNT
3

Expresiones Regulares y Restricciones de Control: Ejemplos

```
ALTER TABLE emp8
ADD CONSTRAINT email_addr
CHECK (REGEXP_LIKE(email, '@')) NOVALIDATE;
```

```
INSERT INTO emp8 VALUES
(500, 'Christian', 'Patel', 'ChrisP2creme.com',
1234567890, '12-Jan-2004', 'HR_REP', 2000, null, 102, 40);
```



ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Expresiones Regulares y Restricciones de Control: Ejemplos

Las expresiones regulares también se pueden utilizar en restricciones CHECK. En este ejemplo, se agrega una restricción CHECK en la columna EMAIL de la tabla EMPLOYEES. Esto garantiza que sólo se aceptan las cadenas que contengan el símbolo “@”. Se prueba la restricción. Se viola la restricción CHECK porque la dirección de correo electrónico no contiene el símbolo necesario. La cláusula NOVALIDATE garantiza que no se comprueban los datos existentes.

En el ejemplo de la diapositiva, la tabla emp8 se crea mediante el siguiente código:

```
CREATE TABLE emp8 AS SELECT * FROM employees;
```

Nota: el ejemplo de la diapositiva se ejecuta mediante la opción “Execute Statement” en SQL Developer. El formato de salida es diferente si utiliza la opción “Run Script”.

Prueba

Con el uso de las expresiones regulares en SQL y PL/SQL, puede:

1. Evitar el procesamiento complejo de cadenas del juego de resultados de SQL mediante aplicaciones de nivel medio
2. Evitar la lógica de validación de datos en el cliente
3. Aplicar restricciones en el servidor

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Respuestas: 1, 2, 3

Resumen

En esta lección, debe haber aprendido a utilizar expresiones regulares para buscar, hacer coincidir y sustituir cadenas.

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Resumen

En esta lección, aprenderá a utilizar las funciones de soporte de expresiones regulares. El soporte de expresiones regulares está disponible en SQL y PL/SQL.

Práctica 7: Visión General

Esta práctica trata el uso de funciones de expresiones regulares para realizar las siguientes tareas:

- Buscar, sustituir y manipular datos
- Crear una nueva tabla `CONTACTS` y agregar una restricción `CHECK` a la columna `p_number` para garantizar que se introducen los números de teléfono en la base de datos en un formato estándar específico
- Probar la adición de algunos números de teléfono en la columna `p_number` mediante varios formatos

ORACLE

Copyright © 2010, Oracle. Todos los derechos reservados.

Práctica 7: Visión General

En esta práctica, utilice funciones de expresiones regulares para buscar, sustituir y manipular datos. Cree también una nueva tabla `CONTACTS` y agregue una restricción `CHECK` a la columna `p_number` para garantizar que se introducen los números de teléfono en la base de datos en un formato estándar específico.

