

Oracle Database: Conceptos Fundamentales de PL/SQL

Guía del Alumno

D64254CS10

Edición 1.0

Febrero de 2010

D73736

ORACLE®

Autor

Brian Pottle

Colaboradores Técnicos y Responsables de Revisión

Tom Best

Christoph Burandt

Yanti Chang

Laszlo Czinkoczki

Ashita Dhir

Peter Driver

Gerlinde Frenzen

Nancy Greenberg

Chaitanya Kortamaddi

Tim Leblanc

Bryan Roberts

Abhishek X Singh

Puja Singh

Lex Van Der Werff

Hilda Simon

Manish Pawar

Diseñador Gráfico

Satish Bettegowda

Redactores

Vijayalakshmi Narasimhan

Daniel Milne

Editor

Veena Narasimhan

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Renuncia

Este documento contiene información propiedad de Oracle Corporation y se encuentra protegido por la legislación de derechos de autor y otras leyes sobre la propiedad intelectual. Usted sólo podrá realizar copias o imprimir este documento para uso exclusivo por usted en los cursos de formación de Oracle. Este documento no podrá ser modificado ni alterado en modo alguno. Salvo que la legislación de los derechos de autor lo considere un uso excusable o legal o "fair use", 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 USA. 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 advertencia:

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
- Objetivos del Curso I-3
- Esquema Human Resources (HR) para Este Curso I-4
- Agenda I-5
 - Información de las Cuentas de Clase I-6
 - Apéndices Utilizados en Este Curso I-7
 - Entornos de Desarrollo de PL/SQL I-8
 - ¿Qué Es Oracle SQL Developer? I-9
 - Codificación de PL/SQL en SQL*Plus I-10
 - Codificación de PL/SQL en Oracle JDeveloper I-11
 - Documentación sobre Oracle SQL y PL/SQL I-12
 - Resumen I-13
 - Visión General de la Práctica I: Introducción I-14

1 Introducción a PL/SQL

- Objetivos 1-2
- Agenda 1-3
- Acerca de PL/SQL 1-4
 - Arquitectura de Tiempo de Ejecución de PL/SQL 1-6
 - Ventajas de PL/SQL 1-7
 - Estructura de Bloque PL/SQL 1-10
 - Agenda 1-12
 - Tipos de Bloque 1-13
 - Construcciones de Programa 1-15
 - Examen de un Bloque Anónimo 1-17
 - Ejecución de un Bloque Anónimo 1-18
 - Agenda 1-19
 - Activación de la Salida de un Bloque PL/SQL 1-20
 - Visualización de la Salida de un Bloque PL/SQL 1-21
 - Prueba 1-22
 - Resumen 1-23
 - Práctica 1: Visión General 1-24

2 Declaración de Variables PL/SQL

Objetivos 2-2

Agenda 2-3

Uso de Variables 2-4

Requisitos para Nombres de Variable 2-5

Manejo de Variables en PL/SQL 2-6

Declaración e Inicialización de Variables PL/SQL 2-7

Delimitadores de Literales de Cadena 2-9

Agenda 2-10

Tipos de Variables 2-11

Instrucciones para Declaración e Inicialización de Variables PL/SQL 2-13

Instrucciones para Declaración de Variables PL/SQL 2-14

Reglas de Nomenclatura de Estructuras PL/SQL Utilizadas en este Curso 2-15

Tipos de Dato Escalar 2-16

Tipos de Dato Escalar Base 2-17

Declaración de Variables Escalares 2-21

Atributo %TYPE 2-22

Declaración de Variables con el Atributo %TYPE 2-24

Declaración de Variables Booleanas 2-25

Variables de Tipos de Dato LOB 2-26

Tipos de Dato Compuestos: Registros y Recopilaciones 2-27

Agenda 2-28

Variables de Enlace 2-29

Referencia a Variables de Enlace 2-31

Uso de AUTOPRINT con Variables de Enlace 2-32

Prueba 2-33

Resumen 2-34

Práctica 2: Visión General 2-35

3 Escritura de Sentencias Ejecutables

Objetivos 3-2

Agenda 3-3

Unidades Léxicas de los Bloques PL/SQL 3-4

Instrucciones y Sintaxis de Bloques PL/SQL 3-6

Comentario del Código 3-7

Funciones SQL en PL/SQL 3-8

Funciones SQL en PL/SQL: Ejemplos 3-9

Uso de Secuencias en Expresiones PL/SQL 3-10

Conversión del Tipo de Dato 3-11

Agenda 3-14

Bloques Anidados	3-15
Bloques Anidados: Ejemplo	3-16
Ámbito y Visibilidad de las Variables	3-17
Uso de Cualificadores en Bloques Anidados	3-19
Desafío: Determinación del Ámbito de las Variables	3-20
Agenda	3-22
Operadores de PL/SQL	3-23
Operadores de PL/SQL: Ejemplos	3-24
Instrucciones de Programación	3-25
Sangrado del Código	3-26
Prueba	3-27
Resumen	3-28
Práctica 3: Visión General	3-29
4 Interacción con Oracle Database Server: Sentencias SQL en Programas PL/SQL	
Objetivos	4-2
Agenda	4-3
Sentencias SQL en PL/SQL	4-4
Sentencias SELECT en PL/SQL	4-5
Recuperación de Datos en PL/SQL: Ejemplo	4-9
Recuperación de Datos en PL/SQL	4-10
Nomenclatura Ambigua	4-11
Reglas de Nomenclatura	4-12
Agenda	4-13
Uso de PL/SQL para Manipular Datos	4-14
Inserción de Datos: Ejemplo	4-15
Actualización de Datos: Ejemplo	4-16
Supresión de Datos: Ejemplo	4-17
Fusión de Filas	4-18
Agenda	4-20
Cursor SQL	4-21
Atributos de Cursor SQL para Cursos Implícitos	4-23
Prueba	4-25
Resumen	4-26
Práctica 4: Visión General	4-27
5 Escritura de las Estructuras de Control	
Objetivos	5-2
Control del Flujo de Ejecución	5-3
Agenda	5-4
Sentencia IF	5-5

Sentencia IF Simple 5-7
Sentencia IF THEN ELSE 5-8
Cláusula IF ELSIF ELSE 5-9
Valor NULL en Sentencia IF 5-10
Agenda 5-11
Expresiones CASE 5-12
Expresiones CASE: Ejemplo 5-13
Expresiones CASE de Búsqueda 5-14
Sentencia CASE 5-15
Manejo de Valores Nulos 5-16
Tablas Lógicas 5-17
¿Expresiones Booleanas o Expresiones Lógicas? 5-18
Agenda 5-19
Control Iterativo: Sentencias LOOP 5-20
Bucles Básicos 5-21
Bucle Básico: Ejemplo 5-22
Bucles WHILE 5-23
Bucles WHILE: Ejemplo 5-24
Bucles FOR 5-25
Bucles FOR: Ejemplo 5-27
Reglas del Bucle FOR 5-28
Uso Sugerido para los Bucles 5-29
Etiquetas y Bucles Anidados 5-30
NEtiquetas y Bucles Anidados: Ejemplo 5-31
Sentencia PL/SQL CONTINUE 5-32
Sentencia PL/SQL CONTINUE: Ejemplo 1 5-33
Sentencia PL/SQL CONTINUE: Ejemplo 2 5-34
Prueba 5-35
Resumen 5-36
Práctica 5: Visión General 5-37

6 Trabajar con Tipos de Dato Compuestos

Objetivos 6-2
Agenda 6-3
Tipos de Dato Compuestos 6-4
¿Registros o Recopilaciones PL/SQL? 6-5
Agenda 6-6
Registros PL/SQL 6-7
Creación de un Registro PL/SQL 6-8
Estructura de Registro PL/SQL 6-9

Atributo %ROWTYPE	6-10
Creación de un Registro PL/SQL: Ejemplo	6-12
Ventajas del Uso del Atributo %ROWTYPE	6-13
Otro Ejemplo de Atributo %ROWTYPE	6-14
Inserción de un Registro mediante %ROWTYPE	6-15
Actualización de una Fila en una Tabla mediante un Registro	6-16
Agenda	6-17
Matrices asociativas (tablas INDEX BY)	6-18
Estructura de la Matriz Asociativa	6-19
Pasos para Crear una Matriz Asociativa	6-20
Creación y Acceso a Matrices Asociativas	6-21
Uso de los Métodos de Tablas INDEX BY	6-22
Opción de Tabla de Registros INDEX BY	6-23
Opción de Tabla de Registros INDEX BY: Ejemplo 2	6-24
Tablas Anidadas	6-25
VARRAY	6-27
Resumen de los Tipos de Recopilación	6-28
Prueba	6-29
Resumen	6-30
Práctica 6: Visión General	6-31

7 Uso de Cursores Explícitos

Objetivos	7-2
Agenda	7-3
Cursores	7-4
Operaciones de Cursores Explícitos	7-5
Control de Cursores Explícitos	7-6
Agenda	7-8
Declaración del Cursor	7-9
Apertura del Cursor	7-11
Recuperación de Datos del Cursor	7-12
Cierre del Cursor	7-15
Cursores y Registros	7-16
Bucles FOR de Cursor	7-17
Atributos de Cursor Explícito	7-19
Atributo %ISOPEN	7-20
%ROWCOUNT y %NOTFOUND: Ejemplo	7-21
Bucles FOR de Cursor mediante Subconsultas	7-22
Agenda	7-23
Cursores con Parámetros	7-24

Agenda 7-26

Cláusula FOR UPDATE 7-27

Cláusula WHERE CURRENT OF 7-29

Prueba 7-30

Resumen 7-31

Práctica 7: Visión General 7-32

8 Manejo de Excepciones

Objetivos 8-2

Agenda 8-3

¿Qué Es una Excepción? 8-4

Manejo de Excepciones: Ejemplo 8-5

Descripción de Excepciones con PL/SQL 8-6

Manejo de Excepciones 8-7

Tipos de Excepción 8-8

Agenda 8-9

Sintaxis para Detectar Excepciones 8-10

Instrucciones para la Detección de Excepciones 8-12

Detección de Errores Predefinidos de Oracle Server 8-13

Detección de Errores No Predefinidos de Oracle Server 8-16

Detección de Errores No Predefinidos: Ejemplo 8-17

Funciones para la Detección de Excepciones 8-18

Detección de Excepciones Definidas por el Usuario 8-20

Propagación de Excepciones en un Subbloque 8-22

Procedimiento RAISE_APPLICATION_ERROR 8-23

Prueba 8-26

Resumen 8-27

Práctica 8: Visión General 8-28

9 Introducción a los Procedimientos y Funciones Almacenados

Objetivos 9-2

Agenda 9-3

Procedimientos y Funciones 9-4

Diferencias entre Bloques Anónimos y Subprogramas 9-5

Agenda 9-6

Procedimiento: Sintaxis 9-7

Creación de Procedimientos 9-8

Llamada de Procedimientos 9-10

Agenda 9-11

Función: Sintaxis 9-12

Creación de Funciones	9-13
Llamada de Funciones	9-14
Transferencia de un Parámetro a la Función	9-15
Llamada a Funciones con un Parámetro	9-16
Prueba	9-17
Resumen	9-18
Práctica 9: Visión General	9-19

A Prácticas y Soluciones

B Descripciones de las Tablas y Datos

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
Exploración de Objetos de Bases de Datos	C-10
Visualización de la Estructura de la Tabla	C-11
Exploración 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

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
- SQL*Plus: Visión General 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 SERVEROUTPUT D-15
- Uso del Comando SQL*Plus SPOOL D-16
- Uso del Comando AUTOTRACE D-17
- Resumen D-18

E Uso de JDeveloper

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

F Cursos REF

- Variables de Cursor F-2
- Uso de Variables de Cursor F-3
- Definición de Tipos REF CURSOR F-4
- Uso de las Sentencias OPEN-FOR, FETCH y CLOSE F-7
- Ejemplo de Recuperación F-10

Prácticas y Soluciones adicionales

I

Introducción

ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Objetivos

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

- Describir los objetivos del curso
- Describir el esquema de base de datos HR que se utiliza en el curso
- Identificar los entornos de interfaz de usuario disponibles que se pueden utilizar en este curso
- Consultar los apéndices, la documentación y otros recursos disponibles



Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Objetivos

En esta lección se proporciona una visión general de alto nivel del curso y su flujo. Conocerá el esquema de base de datos y las tablas que se utilizan en el curso. En el curso se presentan componentes como SQL, PL/SQL, funciones de compilación y herramientas como SQL Developer utilizadas en este curso.

Objetivos del Curso

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

- Identificar las extensiones de programación que PL/SQL proporciona para SQL
- Escribir código PL/SQL para interactuar con la base de datos
- Diseñar bloques PL/SQL anónimos que se ejecuten de forma eficaz
- Utilizar construcciones de programación PL/SQL y sentencias de control condicional
- Manejar errores en tiempo de ejecución
- Describir funciones y procedimientos almacenados

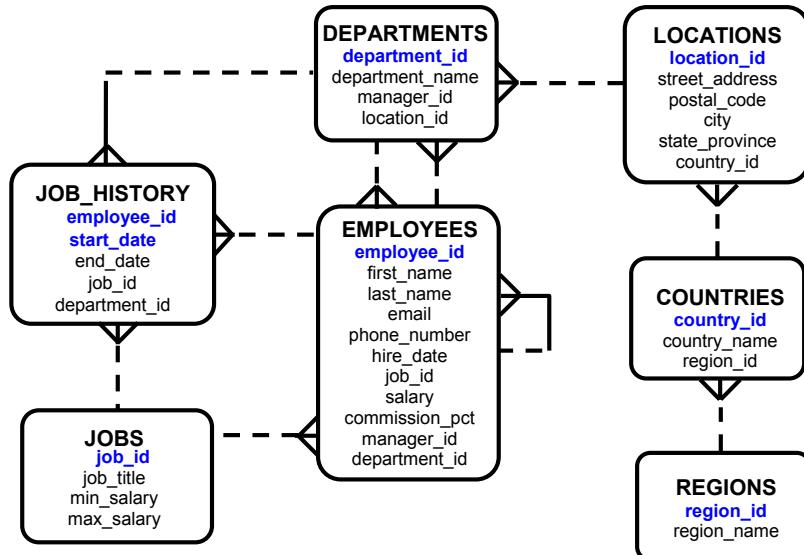
ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Objetivos del Curso

En este curso se presentan los conceptos básicos de PL/SQL. Conocerá la sintaxis, los bloques y las construcciones de programación PL/SQL y podrá conocer las ventajas de la integración de SQL con dichas construcciones. Aprenderá a escribir unidades de programa PL/SQL y ejecutarlas de forma eficaz. Además, aprenderá a utilizar SQL Developer como entorno de desarrollo para PL/SQL. También aprenderá a diseñar unidades de programa reutilizables como, por ejemplo, procedimientos y funciones.

Esquema Human Resources (HR) para Este Curso



ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Esquema Human Resources (HR) para Este Curso

El esquema Human Resources (HR) forma parte de los esquemas de ejemplo de Oracle que se pueden instalar en una base de datos Oracle. Las sesiones prácticas de este curso utilizan datos del esquema HR.

Descripciones de las Tablas

- REGIONS contiene filas que representan una región, como América o Asia.
- COUNTRIES contiene filas para países, que están asociados a una región.
- LOCATIONS contiene la dirección concreta de una oficina, almacén o fábrica de una compañía en un país determinado.
- DEPARTMENTS muestra detalles de los departamentos en los que trabajan los empleados. Cada departamento puede tener una relación que represente al superior del departamento en la tabla EMPLOYEES.
- EMPLOYEES contiene detalles sobre cada empleado que trabaja en un departamento. Puede que algunos empleados no estén asignados a ningún departamento.
- JOBS contiene los tipos de trabajos que puede tener cada empleado.
- JOB_HISTORY contiene el historial del trabajo de los empleados. Si un empleado cambia de departamento dentro de un mismo trabajo o cambia de trabajo dentro de un mismo departamento, se insertará una nueva fila en esta tabla con la información del antiguo trabajo del empleado.

Agenda

Primer día:

- I. Introducción
1. Introducción a PL/SQL
2. Declaración de Variables PL/SQL
3. Escritura de Sentencias Ejecutables
4. Interacción con el Servidor de Bases de Datos Oracle:
Sentencias SQL en Programas PL/SQL
5. Escritura de las Estructuras de Control

Segundo día:

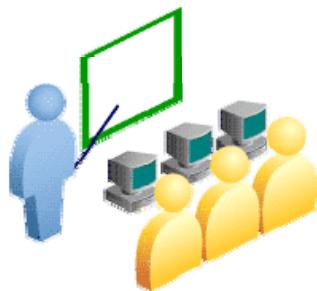
6. Trabajar con Tipos de Dato Compuestos
7. Uso de Cursores Explícitos
8. Manejo de Excepciones
9. Introducción a los Procedimientos y Funciones Almacenados

ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Información de las Cuentas de Clase

- El identificador de cuenta `HR` clonado está ya configurado.
- Su identificador de cuenta es `ora41`.
- La contraseña coincide con su identificador de cuenta.
- Cada máquina dispone de su propio entorno completo y está asignada a la misma cuenta.
- El instructor tiene un identificador distinto.



ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Apéndices Utilizados en Este Curso

- Apéndice A: Prácticas y Soluciones
- Apéndice B: Descripciones de las Tablas y Datos
- Apéndice C: Uso de SQL Developer
- Apéndice D: Uso de SQL*Plus
- Apéndice E: Uso de JDeveloper
- Apéndice F: Cursores REF
- Apéndice AP: Prácticas Adicionales y Soluciones

ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Entornos de Desarrollo de PL/SQL

La configuración de este curso proporciona las siguientes herramientas para desarrollar código PL/SQL:

- Oracle SQL Developer (se utiliza en este curso)
- Oracle SQL*Plus
- Oracle JDeveloper IDE



Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Entornos de Desarrollo de PL/SQL

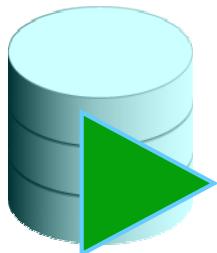
Oracle proporciona varias herramientas que se pueden utilizar para escribir código PL/SQL. Algunas de las herramientas de desarrollo disponibles en este curso son:

- **Oracle SQL Developer:** herramienta gráfica
- **Oracle SQL*Plus:** aplicación de ventanas o línea de comandos
- **Oracle JDeveloper:** entorno de desarrollo de integración (IDE) basado en ventanas

Nota: los ejemplos de código y pantallas que se presentan en las notas del curso se han generado a partir de la salida del entorno SQL Developer.

¿Qué Es Oracle SQL Developer?

- Oracle SQL Developer es una herramienta gráfica gratuita que mejora la productividad y simplifica las tareas de desarrollo de la base de datos.
- Puede conectarse a cualquier esquema de base de datos de destino de Oracle mediante la autenticación estándar de Oracle Database.
- En este curso se utiliza SQL Developer.
- El Apéndice C contiene detalles sobre el uso de SQL Developer.



SQL Developer

ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

¿Qué Es Oracle SQL Developer?

Oracle SQL Developer es una herramienta gráfica gratuita diseñada para mejorar la productividad y simplificar el desarrollo de las tareas diarias de la base de datos. Con sólo unos clics, puede crear y mantener fácilmente los procedimientos almacenados, probar sentencias SQL y visualizar los planes del optimizador.

SQL Developer, la herramienta visual para el desarrollo de la base de datos, simplifica las siguientes tareas:

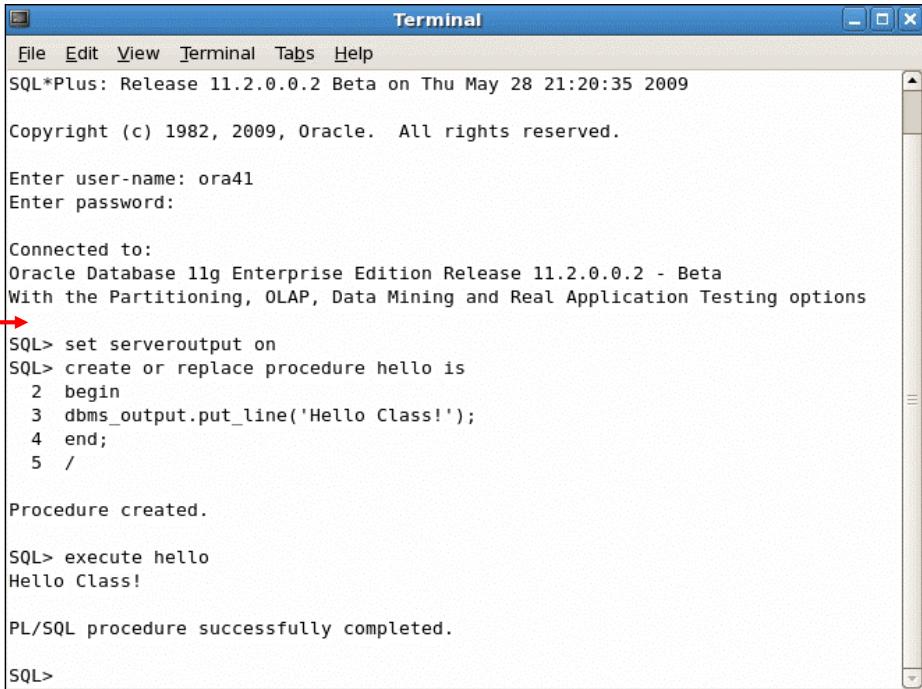
- Exploración y gestión de objetos de la base de datos
- Ejecución de sentencias y scripts SQL
- Edición y depuración de sentencias PL/SQL
- Creación de informes

Puede conectarse a cualquier esquema de base de datos de destino de Oracle mediante la autenticación estándar de la base de datos Oracle. Una vez conectado, puede realizar operaciones en los objetos de la base de datos.

Apéndice C

El Apéndice C de este curso proporciona una introducción al uso de la interfaz de SQL Developer. Consulte el apéndice para obtener información sobre la creación de una conexión de base de datos, así como la interacción con los datos mediante SQL y PL/SQL, etc.

Codificación de PL/SQL en SQL*Plus



The screenshot shows a terminal window titled "Terminal" running SQL*Plus. A red arrow points from the "Terminal" icon in the top-left corner to the window. The SQL*Plus session starts with the copyright notice and prompts for user name and password. It then connects to an Oracle Database 11g Enterprise Edition. The user creates a procedure named "hello" which prints "Hello Class!" using DBMS_OUTPUT.PUT_LINE. After creating the procedure, it is executed successfully, outputting "Hello Class!". The session ends with a message about the procedure being successfully completed.

```

Terminal
File Edit View Terminal Tabs Help
SQL*Plus: Release 11.2.0.0.2 Beta on Thu May 28 21:20:35 2009
Copyright (c) 1982, 2009, Oracle. All rights reserved.

Enter user-name: ora41
Enter password:

Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.0.2 - Beta
With the Partitioning, OLAP, Data Mining and Real Application Testing options

SQL> set serveroutput on
SQL> create or replace procedure hello is
  2 begin
  3   dbms_output.put_line('Hello Class!');
  4 end;
  5 /
Procedure created.

SQL> execute hello
Hello Class!

PL/SQL procedure successfully completed.

SQL>

```

ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Codificación de PL/SQL en SQL*Plus

Oracle SQL*Plus es una interfaz de línea de comandos que permite enviar sentencias SQL y bloques PL/SQL para su ejecución y recibir los resultados en una ventana de comandos o de aplicación.

SQL*Plus:

- Se incluye con la base de datos.
- Se instala en un sistema de cliente y servidor de bases de datos.
- Es accesible desde un ícono o la línea de comandos.

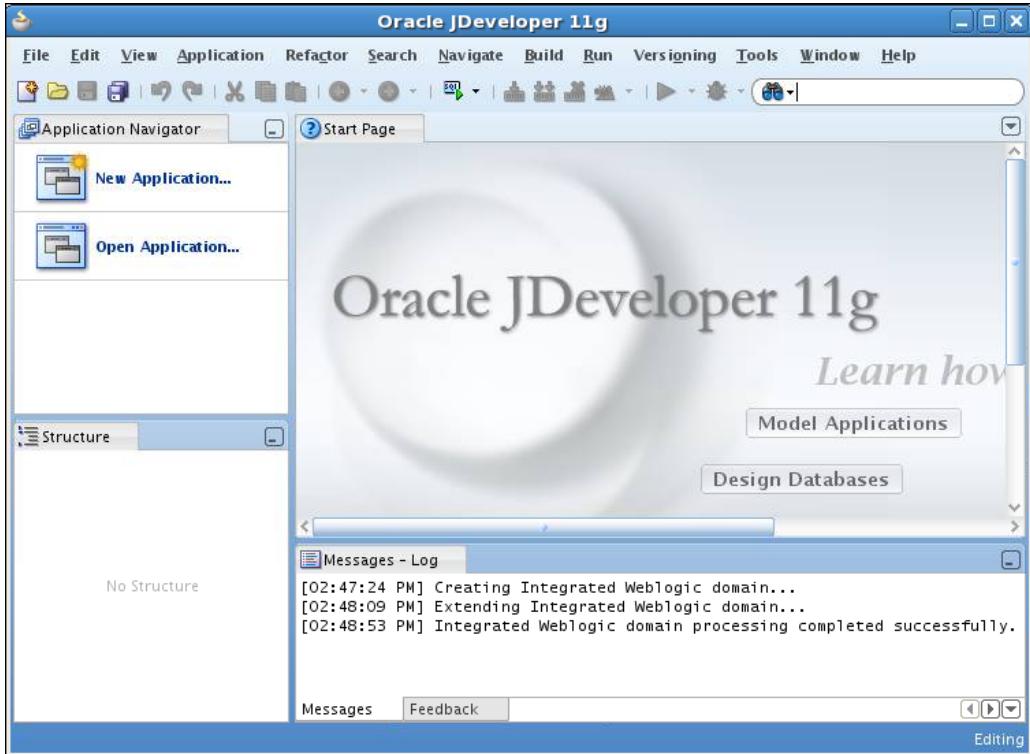
Al codificar subprogramas PL/SQL con SQL*Plus, no olvide lo siguiente:

- Los subprogramas se crean utilizando la sentencia CREATE SQL.
- Los subprogramas se ejecutan con un bloque PL/SQL anónimo o el comando EXECUTE.
- Si se utilizan los procedimientos del paquete DBMS_OUTPUT para imprimir texto en la pantalla, primero debe ejecutar el comando SET SERVEROUTPUT ON en la sesión.

Nota

- Para iniciar SQL*Plus en un entorno Linux, abra una ventana de terminal e introduzca el comando: sqlplus.
- Para obtener más información sobre el uso de SQL*Plus, consulte el Apéndice D.

Codificación de PL/SQL en Oracle JDeveloper



ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Codificación de PL/SQL en Oracle JDeveloper

Oracle JDeveloper permite a los desarrolladores crear, editar, probar y depurar código PL/SQL con una GUI sofisticada. Oracle JDeveloper forma parte de Oracle Developer Suite y está también disponible como producto independiente.

Al codificar PL/SQL en JDeveloper, tenga en cuenta lo siguiente:

- Primero debe crear una conexión a la base de datos para permitir que JDeveloper acceda a un propietario de esquema de base de datos para los subprogramas.
- A continuación, puede utilizar los menús contextuales de JDeveloper en la conexión a la base de datos para crear una nueva construcción de subprograma con el editor de códigos de JDeveloper incorporado.
- Se llama a un subprograma utilizando el comando Run del menú contextual del subprograma con nombre. El resultado aparece en la ventana JDeveloper Log Message, tal como se muestra en la parte inferior de la captura de pantalla.

Nota

- JDeveloper proporciona sintaxis con codificación de color en el editor de códigos de JDeveloper y es sensible a las sentencias y construcciones del lenguaje PL/SQL.
- Para obtener más información sobre cómo utilizar JDeveloper, consulte el Apéndice E.

Documentación sobre Oracle SQL y PL/SQL

- *Oracle Database New Features Guide*
- *Oracle Database PL/SQL Language Reference*
- *Oracle Database Reference*
- *Oracle Database SQL Language Reference*
- *Oracle Database Concepts*
- *Oracle Database PL/SQL Packages and Types Reference*
- *Oracle Database Advanced Application Developer's Guide 11g*
- *Oracle Database SQL Developer User's Guide Release 1.5*

ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Documentación sobre Oracle Database 11g

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

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

Resumen

En esta lección, debe haber aprendido lo siguiente:

- Describir los objetivos del curso
- Describir el esquema de base de datos HR que se utiliza en el curso
- Identificar los entornos de interfaz de usuario disponibles que se pueden utilizar en este curso
- Consultar los apéndices, la documentación y otros recursos disponibles

ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Visión General de la Práctica I: Introducción

En esta práctica se abordan los siguientes temas:

- Inicio de SQL Developer
- Creación de una conexión a la base de datos
- Exploración de tablas de esquema HR
- Definición de una preferencia de SQL Developer



Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Práctica I: Visión General

En esta práctica, se utiliza SQL Developer para ejecutar sentencias SQL para examinar los datos del esquema HR. También se crea un bloque anónimo simple.

Nota: todas las prácticas escritas utilizan SQL Developer como entorno de desarrollo. Si bien se recomienda utilizar SQL Developer, también puede utilizar los entornos SQL*Plus o JDeveloper disponibles en este curso.

1

Introducción a PL/SQL

ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Objetivos

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

- Explicar la necesidad de PL/SQL
- Explicar las ventajas de PL/SQL
- Identificar los distintos tipos de bloques PL/SQL
- Generar mensajes en PL/SQL



Objetivos

Esta lección presenta PL/SQL y las construcciones de programación PL/SQL. Aprenderá acerca de las ventajas de PL/SQL.

Agenda

- Descripción de las ventajas y la estructura de PL/SQL
- Examen de bloques PL/SQL
- Generación de mensajes de salida en PL/SQL

ORACLE

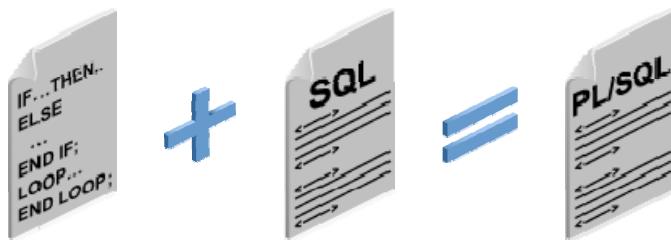
1 - 3

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Acerca de PL/SQL

PL/SQL:

- Significa extensión de lenguaje de procedimiento para SQL
- Es el lenguaje de acceso de datos estándar de Oracle Corporation para bases de datos relacionales
- Integra de forma ininterrumpida construcciones de procedimientos con SQL



ORACLE

1 - 4

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Acerca de PL/SQL

El lenguaje de consulta estructurado (SQL) es el lenguaje primario que se utiliza para acceder y modificar datos en bases de datos relacionales. Hay sólo algunos comandos SQL, por lo que puede aprenderlos y utilizarlos con facilidad.

Consideré el ejemplo:

```
SELECT first_name, department_id, salary FROM employees;
```

La anterior sentencia SQL es simple y sencilla. Sin embargo, si desea modificar cualquier dato recuperado de forma condicional, se encuentra con las limitaciones de SQL.

Consideré una sentencia con el problema ligeramente modificado: para cada empleado recuperado, comprueba el identificador de departamento y el salario. Dependiendo del rendimiento del departamento y también del salario de los empleados, puede que desee proporcionar diferentes incentivos a los empleados.

Observando el problema, es consciente de que tiene que ejecutar la sentencia SQL anterior, recopilar los datos y aplicar la lógica a los datos.

- Una solución consiste en escribir una sentencia SQL para cada departamento y ofrecer los incentivos a los empleados de dicho departamento. Recuerde que debe comprobar además el componente del salario antes de decidir la cantidad del incentivo. Esto resulta un poco complicado.
- Una solución más eficaz podría incluir sentencias condicionales. PL/SQL está diseñado para cumplir dichos requisitos. Proporciona una extensión de programación a SQL ya existente.

Acerca de PL/SQL

PL/SQL:

- Proporciona una estructura en bloque para unidades de código ejecutables. El mantenimiento del código se simplifica con una estructura muy bien definida.
- Proporciona construcciones de procedimientos como:
 - Variables, constantes y tipos de dato
 - Estructuras de control como sentencias condicionales y bucles
 - Unidades de programa reutilizables que se escriben una vez y se ejecutan muchas veces



1 - 5

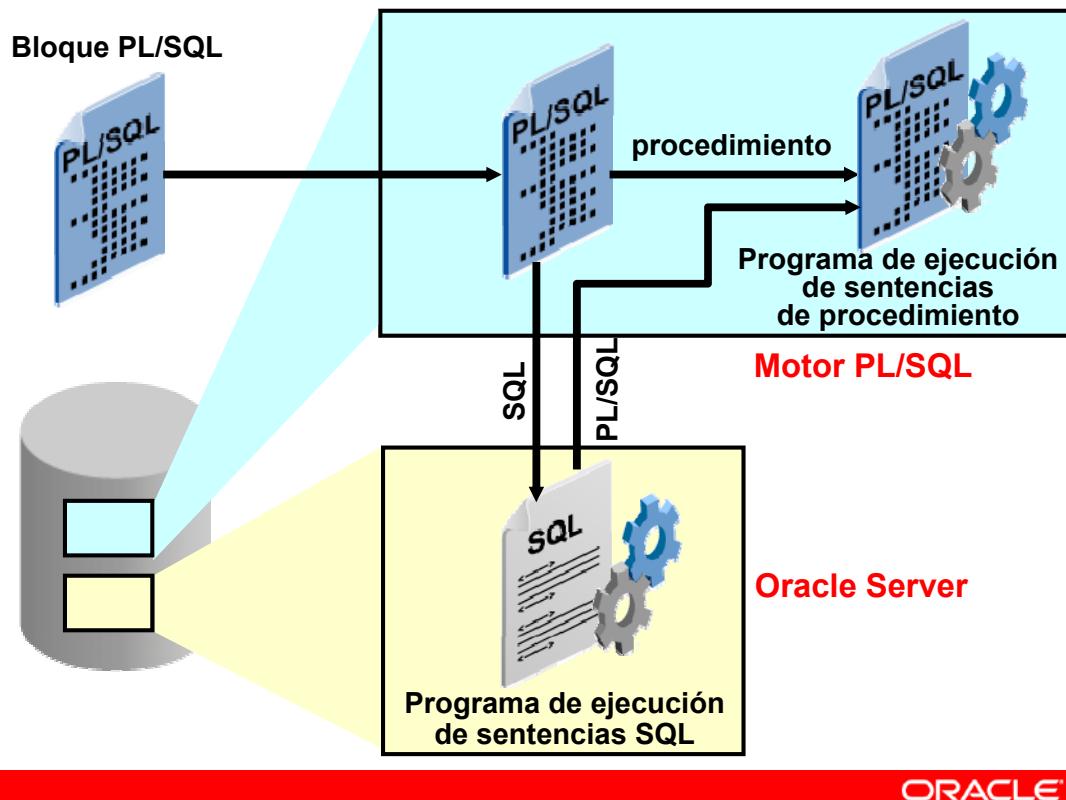
Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Acerca de PL/SQL (continuación)

PL/SQL define una estructura en bloque para escritura de código. El mantenimiento y la depuración del código se simplifican con dicha estructura porque se puede entender fácilmente el flujo y la ejecución de la unidad de programa.

PL/SQL ofrece modernas funciones de ingeniería de software, como encapsulado de datos, manejo de excepciones, ocultación de información y orientación de objetos. Proporciona lo último en programación a Oracle Server y al juego de herramientas de Oracle. PL/SQL ofrece todas las construcciones de procedimientos disponibles en cualquier lenguaje de tercera generación (3GL).

Arquitectura de Tiempo de Ejecución de PL/SQL



1 - 6

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Arquitectura de Tiempo de Ejecución de PL/SQL

El diagrama de la diapositiva muestra la ejecución de un bloque PL/SQL por parte de un motor PL/SQL. El motor PL/SQL reside en:

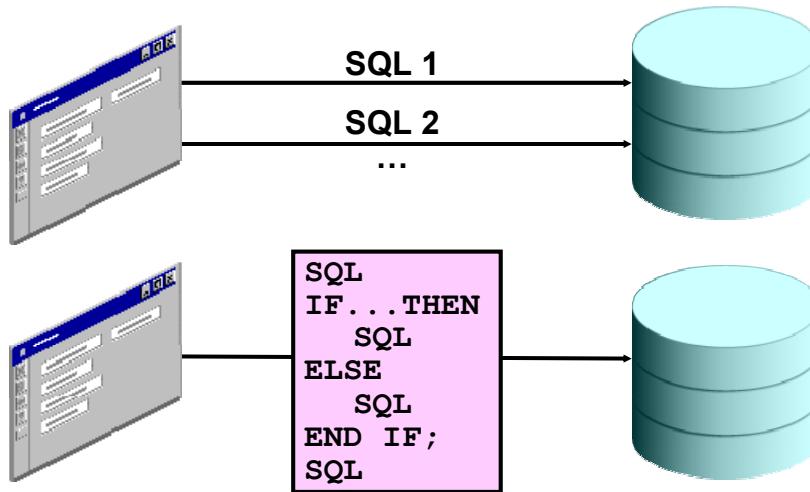
- La base de datos Oracle para ejecutar subprogramas almacenados
- El cliente Oracle Forms al ejecutar aplicaciones de cliente/servidor o en Oracle Application Server al utilizar Oracle Forms Services para ejecutar Forms en la Web

Independientemente del entorno de tiempo de ejecución de PL/SQL, la arquitectura básica se mantiene igual. Por lo tanto, todas las sentencias PL/SQL se procesan en el programa de ejecución de sentencias de procedimiento y todas las sentencias SQL se deben enviar al programa de ejecución de sentencias SQL para que Oracle Server las procese. El entorno SQL también puede llamar al entorno PL/SQL. Por ejemplo, se llama al entorno PL/SQL cuando se utiliza una función PL/SQL en una sentencia SELECT.

El motor PL/SQL es una máquina virtual que reside en memoria y procesa las instrucciones de código m de PL/SQL. Cuando el motor PL/SQL detecta una sentencia SQL, se produce un cambio de contexto para transferir la sentencia SQL a los procesos de Oracle Server. El motor PL/SQL espera a que termine la sentencia SQL y se devuelvan los resultados antes de continuar con el procesamiento de las siguientes sentencias del bloque PL/SQL. El motor PL/SQL de Oracle Forms se ejecuta en el cliente para la implantación del cliente o servidor y en el servidor de aplicaciones para la implantación de Forms Services. En cualquier caso, las sentencias SQL se envían normalmente por una red a Oracle Server para su procesamiento.

Ventajas de PL/SQL

- Integración de construcciones de procedimientos con SQL
- Rendimiento mejorado



ORACLE

Ventajas de PL/SQL

Integración de construcciones de procedimientos con SQL: La ventaja más importante de PL/SQL es la integración de construcciones de procedimientos con SQL. SQL es un lenguaje que no es de procedimientos. Al emitir un comando SQL, éste indica al servidor de bases de datos *qué* debe hacer. Sin embargo, no puede especificar *cómo* debe hacerlo. PL/SQL integra sentencias de control y sentencias condicionales con SQL, lo que proporciona un mayor control de las sentencias SQL y su ejecución. Ya ha visto un ejemplo anterior en esta lección, que muestra la necesidad de dicha integración.

Rendimiento mejorado: sin PL/SQL no podría combinar de manera lógica sentencias SQL como una sola unidad. Si ha diseñado una aplicación que contiene pantallas, debe contar con muchas pantallas distintas con campos en cada una de ellas. Cuando una pantalla envía datos, puede que tenga que ejecutar una serie de sentencias SQL. Las sentencias SQL se envían a la base de datos de una en una. Esto da como resultado numerosos recorridos en la red y una llamada a la base de datos para cada sentencia SQL, lo que aumenta el tráfico de red y se reduce el rendimiento (especialmente en un modelo cliente/servidor).

Con PL/SQL puede combinar todas estas sentencias SQL en una sola unidad de programa. La aplicación puede enviar todo el bloque a la base de datos en lugar de enviar las sentencias SQL de una en una. Esto reduce considerablemente el número de llamadas a la base de datos. Como muestra la diapositiva, si la aplicación tiene un gran flujo de SQL, puede utilizar bloques PL/SQL para agrupar sentencias SQL antes de enviarlas al servidor de bases de datos Oracle para su ejecución.

Ventajas de PL/SQL

- Desarrollo de programas basado en módulos
- Integración con herramientas de Oracle
- Portabilidad
- Manejo de excepciones



1 - 8

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Ventajas de PL/SQL (continuación)

Desarrollo de programas basado en módulos: la unidad básica de todos los programas PL/SQL es el bloque. Los bloques pueden estar en una secuencia o anidados en otros bloques. El desarrollo de programas basado en módulos tiene las siguientes ventajas:

- Puede agrupar de forma lógica sentencias relacionadas en los bloques.
- Puede anidar bloques en bloques más grandes para generar programas potentes.
- Puede dividir su aplicación en módulos más pequeños. Si va a diseñar una aplicación compleja, PL/SQL permite dividir la aplicación en módulos más pequeños, que se puedan gestionar y que estén relacionados de forma lógica.
- Puede mantener y depurar fácilmente el código.

En PL/SQL, los módulos se implantan con procedimientos, funciones y paquetes, que se describen en la lección titulada “Introducción a los Procedimientos y Funciones Almacenados”.

Integración con herramientas: el motor PL/SQL está integrado en herramientas de Oracle como Oracle Forms y Oracle Reports. Al utilizar estas herramientas, el motor PL/SQL disponible localmente procesa las sentencias de procedimientos y sólo se transfieren las sentencias SQL a la base de datos.

Ventajas de PL/SQL (continuación)

Portabilidad: los programas PL/SQL se pueden ejecutar en cualquier lugar donde se ejecute Oracle Server, independientemente del sistema operativo y la plataforma. No es necesario personalizarlos para cada nuevo entorno. Puede escribir paquetes de programas portátiles y crear bibliotecas que se puedan volver a utilizar en distintos entornos.

Manejo de excepciones: PL/SQL permite manejar excepciones de forma eficaz. Puede definir bloques independientes para gestionar excepciones. Aprenderá más sobre el manejo de excepciones en la lección titulada “Manejo de Excepciones”.

PL/SQL comparte el mismo sistema de tipo de dato que SQL (con algunas extensiones) y utiliza la misma sintaxis de expresiones.

Estructura de Bloque PL/SQL

- **DECLARE** (opcional)
 - Variables, cursores, excepciones definidas por el usuario
- **BEGIN** (obligatoria)
 - Sentencias SQL
 - Sentencias PL/SQL
- **EXCEPTION** (opcional)
 - Acciones que realizar cuando se producen excepciones
- **END;** (obligatoria)



ORACLE

1 - 10

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Estructura de Bloque PL/SQL

La diapositiva muestra un bloque PL/SQL básico. Un bloque PL/SQL se compone de cuatro secciones:

- **Declaraciones (opcional):** la sección de declaraciones empieza por la palabra clave **DECLARE** y termina donde empieza la sección ejecutable.
- **Inicio (obligatoria):** la sección ejecutable empieza por la palabra clave **BEGIN**. Esta sección debe tener al menos una sentencia. Sin embargo, la sección ejecutable de un bloque PL/SQL puede incluir cualquier número de bloques PL/SQL.
- **Manejo de excepciones (opcional):** la sección de excepciones está anidada en la sección ejecutable. Esta sección empieza por la palabra clave **EXCEPTION**.
- **Fin (obligatoria):** todos los bloques PL/SQL deben terminar en una sentencia **END**. Observe que **END** termina en punto y coma.

Estructura de Bloque PL/SQL (continuación)

En un bloque PL/SQL, las palabras clave DECLARE, BEGIN y EXCEPTION no terminan en punto y coma. No obstante, la palabra clave END, todas las sentencias SQL y las sentencias PL/SQL deben terminar en punto y coma.

Sección	Descripción	Inclusión
Declaraciones (DECLARE)	Contiene declaraciones de todas las variables, constantes, cursoras y excepciones definidas por el usuario a las que se hace referencia en las secciones ejecutable y de excepciones	Opcional
Ejecutable (BEGIN ... END)	Contiene sentencias SQL para recuperar datos de la base de datos y sentencias PL/SQL para manipular datos del bloque	Obligatoria
Excepciones (EXCEPTION)	Especifica las acciones que se realizan cuando surgen errores y condiciones anormales en la sección ejecutable	Opcional

Agenda

- Descripción de las ventajas y la estructura de PL/SQL
- Examen de bloques PL/SQL
- Generación de mensajes de salida en PL/SQL

ORACLE

1 - 12

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Tipos de Bloque

Procedimiento

```
PROCEDURE name
IS

BEGIN
    --statements

[EXCEPTION]

END ;
```

Función

```
FUNCTION name
RETURN datatype
IS

BEGIN
    --statements
    RETURN value;
[EXCEPTION]

END ;
```

Anónimo

```
[DECLARE]

BEGIN
    --statements

[EXCEPTION]

END ;
```

ORACLE

1 - 13

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Tipos de Bloque

Un programa PL/SQL comprende uno o más bloques. Estos bloques pueden ser totalmente independientes o estar anidados dentro de otro.

Hay tres tipos de bloques que forman un programa PL/SQL:

- Procedimientos
- Funciones
- Bloques anónimos

Procedimientos: los procedimientos son objetos con nombre que contienen sentencias SQL y/o PL/SQL.

Funciones: las funciones son objetos con nombre que contienen sentencias SQL y/o PL/SQL. A diferencia de los procedimientos, las funciones devuelven un valor del tipo de dato especificado.

Bloques anónimos

Los bloques anónimos son bloques sin nombre. Se declaran en línea en el punto de una aplicación donde se van a ejecutar y se compilan cada vez que la aplicación se ejecuta. Estos bloques no están almacenados en la base de datos. Se transfieren al motor PL/SQL para la ejecución en tiempo de ejecución. Los disparadores de componentes de Oracle Developer se componen de dichos bloques.

Si desea ejecutar el mismo bloque de nuevo, tiene que volver a escribir el bloque. No puede llamar al bloque que escribió anteriormente porque los bloques son anónimos y ya no existen una vez que se ejecutan.

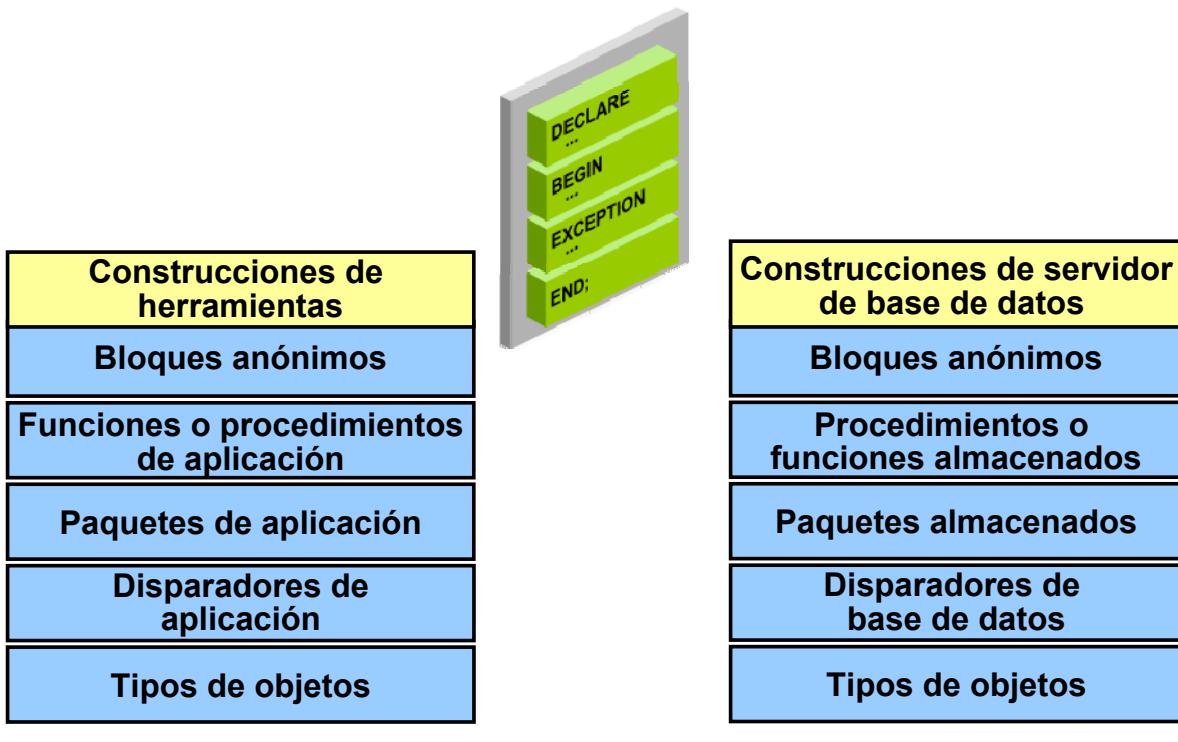
Tipos de Bloque (continuación)

Subprogramas

Los subprogramas son complementarios a los bloques anónimos. Se trata de bloques PL/SQL con nombre almacenados en la base de datos. Puesto que tienen nombre y están almacenados, puede llamarlos cuando lo deseé (dependiendo de la aplicación). Puede declararlos como procedimientos o funciones. Normalmente, se utiliza un procedimiento para realizar una acción y una función para calcular y devolver un valor.

Los subprogramas se pueden almacenar a nivel de servidor o aplicación. Con los componentes de Oracle Developer (Forms, Reports), puede declarar procedimientos y funciones como parte de la aplicación (pantalla o informe) y llamarlos desde otros procedimientos, funciones y disparadores de la misma aplicación cuando sea necesario.

Construcciones de Programa



ORACLE

Construcciones de Programa

La siguiente tabla describe diferentes construcciones de programa PL/SQL que utilizan el bloque PL/SQL básico. Las construcciones de programa están disponibles según el entorno en el que se ejecutan.

Construcción de programa	Descripción	Disponibilidad
Bloques anónimos	Bloques PL/SQL sin nombre que están embebidos en una aplicación o se emiten de forma interactiva	Todos los entornos PL/SQL
Funciones o procedimientos de aplicación	Bloques PL/SQL con nombre almacenados en una aplicación de Oracle Forms Developer o en una biblioteca compartida; pueden aceptar los parámetros y se pueden llamar por el nombre en repetidas ocasiones	Componentes de las herramientas de Oracle Developer (por ejemplo, Oracle Forms Developer y Oracle Reports)
Funciones o procedimientos almacenados	Bloques PL/SQL con nombre almacenados en el servidor de Oracle; pueden aceptar los parámetros y se pueden llamar por el nombre en repetidas ocasiones	Servidor de Oracle o herramientas de Oracle Developer
Paquetes (de aplicación o almacenados)	Módulos PL/SQL con nombre que agrupan procedimientos, funciones e identificadores relacionados	Servidor de Oracle y componentes de las herramientas de Oracle Developer (por ejemplo, Oracle Forms Developer)

Construcciones de Programa (continuación)

Construcción de programa	Descripción	Disponibilidad
Disparadores de base de datos	Bloques PL/SQL que se asocian a una tabla de base de datos y se arrancan automáticamente cuando se disparan en varios eventos	Servidor de Oracle o cualquier herramienta de Oracle que emita DML
Disparadores de aplicación	Bloques PL/SQL que se asocian a una tabla de base de datos o eventos de sistema. Se arrancan automáticamente cuando se disparan por un DML o un evento de sistema respectivamente	Componentes de las herramientas de Oracle Developer (por ejemplo, Oracle Forms Developer)
Tipos de objetos	Tipos de dato compuestos definidos por el usuario que encapsulan una estructura de datos junto con las funciones y procedimientos necesarios para manipular datos	Servidor de Oracle y herramientas de Oracle Developer

Examen de un Bloque Anónimo

Un bloque anónimo del espacio de trabajo de SQL Developer:



The screenshot shows the Oracle SQL Worksheet window. The title bar says "SQL Worksheet History". Below the title bar is a toolbar with several icons. The main area contains the following PL/SQL code:

```
DECLARE
    v_fname VARCHAR2(20);
BEGIN
    SELECT first_name INTO v_fname FROM employees
    WHERE employee_id=100;
END;
/
```

ORACLE

1 - 17

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Examen de un Bloque Anónimo

Para crear un bloque anónimo mediante SQL Developer, introduzca el bloque en el espacio de trabajo (como se muestra en la diapositiva).

Ejemplo

El bloque de ejemplo tiene la sección de declaraciones y la sección ejecutable. No tiene que prestar atención a la sintaxis de las sentencias del bloque; se explicará más adelante en este mismo curso.

El bloque anónimo obtiene `first_name` del empleado cuyo `employee_id` es 100 y lo almacena en una variable denominada `v_fname`.

Ejecución de un Bloque Anónimo

Haga clic en el botón Run Script para ejecutar el bloque anónimo:

The screenshot shows the Oracle SQL Worksheet interface. At the top, there's a toolbar with various icons. A yellow callout box points to the 'Run Script (o F5)' icon, which is highlighted with a red square. Below the toolbar, the main area contains the following PL/SQL code:

```
DECLARE
    v_fname VARCHAR2(20);
BEGIN
    SELECT first_name INTO v_fname FROM employees
    WHERE employee_id=100;
END;
/
```

At the bottom of the code area, there's a message: "anonymous block completed". The status bar at the bottom of the window shows "0.46330699 seconds".

ORACLE

Ejecución de un Bloque Anónimo

Para ejecutar un bloque anónimo, haga clic en el botón Run Script (o pulse F5).

Nota: aparece el mensaje “anonymous block completed” en la ventana Script Output después de ejecutarse el bloque.

Agenda

- Descripción de las ventajas y la estructura de PL/SQL
- Examen de bloques PL/SQL
- Generación de mensajes de salida en PL/SQL

ORACLE

1 - 19

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Activación de la Salida de un Bloque PL/SQL

1. Para activar la salida en SQL Developer, ejecute el siguiente comando antes de ejecutar el bloque PL/SQL:

```
SET SERVEROUTPUT ON
```

2. Utilice un paquete predefinido de Oracle y su procedimiento en el bloque anónimo:

- DBMS_OUTPUT.PUT_LINE

```
DBMS_OUTPUT.PUT_LINE('The First Name of the  
Employee is ' || v_fname);
```

Activación de la Salida de un Bloque PL/SQL

En el ejemplo mostrado en la diapositiva anterior, se ha almacenado un valor en la variable v_fname. Sin embargo, el valor no se ha impreso.

PL/SQL no tiene la funcionalidad de entrada o salida incorporada. Por tanto, necesita utilizar paquetes predefinidos de Oracle para la entrada y salida. Para generar una salida, debe llevar a cabo lo siguiente:

1. Ejecute el siguiente comando:

```
SET SERVEROUTPUT ON
```

Nota: para activar la salida en SQL*Plus, debe emitir de forma explícita el comando SET SERVEROUTPUT ON.

2. En el bloque PL/SQL, utilice el procedimiento PUT_LINE del paquete DBMS_OUTPUT para mostrar la salida. Transfiera el valor que se tiene que imprimir como argumento a este procedimiento (como se muestra en la diapositiva). A continuación, el procedimiento generará el argumento.

Visualización de la Salida de un Bloque PL/SQL

The screenshot shows the Oracle SQL Worksheet interface. In the top menu bar, 'SQL Worksheet' and 'History' are visible. Below the menu is a toolbar with various icons. The main workspace contains the following PL/SQL code:

```
SET SERVEROUTPUT ON
DECLARE
    v_fname VARCHAR(20);
BEGIN
    SELECT first_name
    INTO v_fname
    FROM employees
    WHERE employee_id = 100;
    DBMS_OUTPUT.PUT_LINE('The First Name of the Employee is ' || v_fname);
END;
/
```

A callout bubble points to the 'Run Script' icon in the toolbar with the text: "Pulse F5 para ejecutar el comando y el bloque PL/SQL." Below the code, the results section shows the output of the anonymous block:

anonymous block completed
The First Name of the Employee is Steven

ORACLE

Visualización de la Salida de un Bloque PL/SQL

Pulse F5 (o haga clic en el ícono Run Script) para ver la salida del bloque PL/SQL. Esta acción:

1. Ejecuta el comando SET SERVEROUTPUT ON
2. Ejecuta el bloque PL/SQL anónimo

La salida aparece en el separador Script Output.

Prueba

Un bloque PL/SQL *debe* constar de las tres secciones siguientes:

- Una sección de declaraciones, que empieza por la palabra clave DECLARE y termina donde empieza la sección ejecutable.
 - Una sección ejecutable, que empieza por la palabra clave BEGIN y termina en END.
 - Una sección de manejo de excepciones, que empieza por la palabra clave EXCEPTION y está anidada en la sección ejecutable.
- a.** Verdadero
b. Falso



Respuesta: b

Un bloque PL/SQL se compone de tres secciones:

- **Declaraciones (opcional):** la sección de declaraciones empieza por la palabra clave DECLARE y termina donde empieza la sección ejecutable.
- **Ejecutable (obligatoria):** la sección ejecutable empieza por la palabra clave BEGIN y termina en END. Esta sección debe tener en esencia al menos una sentencia. Observe que END termina en punto y coma. La sección ejecutable de un bloque PL/SQL puede, por su parte, incluir cualquier número de bloques PL/SQL.
- **Manejo de excepciones (opcional):** la sección de excepciones opcional está anidada en la sección ejecutable. Esta sección empieza por la palabra clave EXCEPTION.

Resumen

En esta lección, debe haber aprendido lo siguiente:

- Integrar sentencias SQL con construcciones de programa PL/SQL
- Describir las ventajas de PL/SQL
- Diferenciar entre los distintos tipos de bloque PL/SQL
- Generar mensajes en PL/SQL



Resumen

PL/SQL es un lenguaje que tiene funciones de programación que sirven como extensiones de SQL. SQL, que es un lenguaje que no es de procedimientos, se convierte en lenguaje de procedimientos con las construcciones de programación PL/SQL. Las aplicaciones PL/SQL se pueden ejecutar en cualquier plataforma o sistema operativo en el que se ejecute Oracle Server. En esta lección, ha aprendido a crear bloques PL/SQL básicos.

Práctica 1: Visión General

En esta práctica se abordan los siguientes temas:

- Identificación de los bloques PL/SQL que se ejecutan correctamente
- Creación y ejecución de un bloque PL/SQL simple



Práctica 1: Visión General

Esta práctica refuerza los conceptos básicos de PL/SQL que se abordan en esta lección.

- El ejercicio 1 se responde sobre papel y permite identificar los bloques PL/SQL que se ejecutan correctamente.
- El ejercicio 2 implica la creación y ejecución de un bloque PL/SQL simple.

2

Declaración de Variables PL/SQL

ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Objetivos

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

- Reconocer identificadores válidos y no válidos
- Mostrar los usos de las variables
- Declarar e inicializar variables
- Mostrar y describir los diversos tipos de dato
- Identificar las ventajas del uso del atributo %TYPE
- Declarar, utilizar e imprimir variables de enlace



Objetivos

Ya se han explicado los bloques PL/SQL básicos y sus secciones. En esta lección, aprenderá a distinguir los identificadores válidos de los que no son válidos. También aprenderá a declarar e inicializar variables en la sección de declaraciones de los bloques PL/SQL. En esta lección se describen, asimismo, los diversos tipos de dato. Además, conocerá el atributo %TYPE y sus ventajas.

Agenda

- Introducción a las variables
- Examen de los tipos de dato de las variables y el atributo %TYPE
- Examen de las variables de enlace

ORACLE

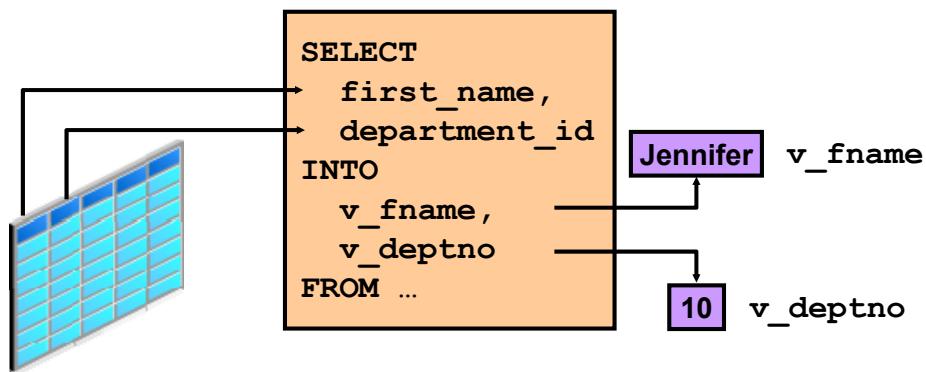
2 - 3

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Uso de Variables

Las variables sirven para lo siguiente:

- Almacenamiento temporal de datos
- Manipulación de valores almacenados
- Reutilización



ORACLE

2 - 4

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Uso de Variables

PL/SQL permite declarar variables y utilizarlas a continuación en sentencias SQL y de procedimiento.

Las variables se utilizan, principalmente, para almacenar datos y manipular valores almacenados. Observe la sentencia PL/SQL de la diapositiva. La sentencia recupera `first_name` y `department_id` de la tabla. Si es preciso manipular `first_name` o `department_id`, hay que almacenar el valor recuperado. Las variables facilitan el almacenamiento temporal del valor. El valor almacenado en estas variables se puede utilizar para procesar o manipular los datos. Las variables pueden almacenar cualquier objeto PL/SQL, por ejemplo, variables, tipos, cursos o subprogramas. La posibilidad de *reutilización* constituye otra ventaja de la declaración de variables. Después de declarar las variables, puede utilizarlas de forma repetida en una aplicación haciendo referencia a ellas varias veces en distintas sentencias.

Requisitos para Nombres de Variable

Un nombre de variable:

- Debe empezar por una letra
- Puede incluir letras y números
- Puede incluir caracteres especiales (como \$, _ y #)
- No debe contener más de 30 caracteres
- No debe incluir ninguna palabra reservada



ORACLE

2 - 5

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Requisitos para Nombres de Variable

En la diapositiva se enumeran las reglas de nomenclatura de las variables.

Manejo de Variables en PL/SQL

Las variables:

- Se declaran e inicializan (opcionalmente) en la sección de declaraciones
- Se utilizan y se les asignan nuevos valores en la sección ejecutable
- Se transfieren como parámetros a los subprogramas PL/SQL
- Se utilizan para contener la salida de los subprogramas PL/SQL

ORACLE

2 - 6

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Manejo de Variables en PL/SQL

Puede utilizar variables de las siguientes formas:

- **Declare e inicialice las variables en la sección de declaraciones:** puede declarar variables en la parte de declaraciones de cualquier bloque, subprograma o paquete PL/SQL. Las declaraciones asignan espacio de almacenamiento para los valores, especifican su tipo de dato y asignan un nombre a la ubicación de almacenamiento para que se pueda hacer referencia a ella. Las declaraciones también pueden asignar un valor inicial e imponer la restricción NOT NULL a la variable. No se permiten las referencias anticipadas. Se debe declarar una variable para poder hacer referencia a ella en otras sentencias, incluidas otras sentencias declarativas.
- **Utilícelas y asígneles nuevos valores en la sección ejecutable:** en la sección ejecutable, el valor existente de la variable se puede sustituir por un nuevo valor.
- **Transfiéralas como parámetros a los subprogramas PL/SQL:** los subprogramas aceptan el uso de parámetros. Se pueden transferir variables a modo de parámetros a los subprogramas.
- **Utilícelas para contener la salida de los subprogramas PL/SQL:** las variables se pueden utilizar para contener el valor que devuelven las funciones.

Declaración e Inicialización de Variables PL/SQL

Sintaxis:

```
identifier [CONSTANT] datatype [NOT NULL]
[ := | DEFAULT expr] ;
```

Ejemplos:

```
DECLARE
    v_hiredate      DATE ;
    v_deptno        NUMBER(2) NOT NULL := 10;
    v_location       VARCHAR2(13) := 'Atlanta';
    c_comm           CONSTANT NUMBER := 1400;
```

2 - 7

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Declaración e Inicialización de Variables PL/SQL

Debe declarar todos los identificadores PL/SQL de la sección de declaraciones antes de hacer referencia a ellos en el bloque PL/SQL. Puede asignar un valor inicial a la variable (tal y como se muestra en la diapositiva). No es necesario asignar un valor a una variable para declararla. Si hace referencia a otras variables de una declaración, asegúrese de declararlas por separado en una sentencia anterior.

En la sintaxis:

<i>identifier</i>	Es el nombre de la variable
CONSTANT	Limita la variable para que no se pueda cambiar el valor (Las constantes se deben inicializar)
<i>data type</i>	Es un tipo de dato escalar, compuesto, de referencia o LOB. (En este curso se abordan únicamente los tipos de dato escalar, compuesto y LOB.)
NOT NULL	Limita la variable para que contenga un valor (Las variables NOT NULL se deben inicializar)
<i>expr</i>	Es cualquier expresión PL/SQL que puede ser una expresión literal, otra variable o una expresión que incluya operadores y funciones.

Nota: además de las variables, también se pueden declarar cursos y excepciones en la sección de declaraciones. Dispone de más información sobre la declaración de cursos en la lección titulada “Uso de Cursos Explícitos” y sobre excepciones en la lección “Manejo de Excepciones”.

Ejecutar y analizar

Declaración e Inicialización de Variables PL/SQL

1

```
DECLARE
    v_myName VARCHAR2(20);
BEGIN
    DBMS_OUTPUT.PUT_LINE('My name is: ' || v_myName);
    v_myName := 'John';
    DBMS_OUTPUT.PUT_LINE('My name is: ' || v_myName);
END;
/
```

2

```
DECLARE
    v_myName VARCHAR2(20) := 'John';
BEGIN
    v_myName := 'Steven';
    DBMS_OUTPUT.PUT_LINE('My name is: ' || v_myName);
END;
/
```

ORACLE

2 - 8

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Explicar

Declaración e Inicialización de Variables PL/SQL (continuación)

Examine los dos bloques de código de la diapositiva.

- En el primer bloque, la variable `v_myName` se declara, pero no se inicializa. Se asigna el valor `John` a la variable en la sección ejecutable.
 - Los literales de cadena se deben incluir entre comillas simples. Si la cadena tiene comillas como en "Today's Date", la cadena sería 'Today''s Date'.
 - El operador de asignación es: “:=”.
 - Al procedimiento `PUT_LINE` se le llama transfiriendo la variable `v_myName`. El valor de la variable está concatenado a la cadena 'My name is:'.
 - La salida de este bloque anónimo es:



- En el segundo bloque, la variable `v_myName` se declara e inicializa en la sección de declaraciones. `v_myName` contiene el valor `John` después de la inicialización. Este valor se manipula en la sección ejecutable del bloque. La salida de este bloque anónimo es:

```
anonymous block completed
My name is: Steven
```

Delimitadores de Literales de Cadena

Analizar, Ejecutar y explicar

```

DECLARE
    v_event VARCHAR2(15);
BEGIN
    v_event := q'!Father's day!';
    DBMS_OUTPUT.PUT_LINE('3rd Sunday in June is :
    ''' || v_event );
    v_event := q'[Mother's day]';
    DBMS_OUTPUT.PUT_LINE('2nd Sunday in May is :
    ''' || v_event );
END;
/

```

Salida resultante

anonymous block completed
3rd Sunday in June is : Father's day
2nd Sunday in May is : Mother's day

ORACLE

2 - 9

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Delimitadores de Literales de Cadena

Si la cadena contiene un apóstrofe (idéntico a una comilla simple), debe duplicar la comilla, como en el siguiente ejemplo:

```
v_event VARCHAR2(15) := 'Father''s day';
```

La primera comilla actúa como carácter de escape. Esto complica la cadena, en especial si aparecen sentencias SQL como cadenas. Se puede especificar cualquier carácter que no esté presente en la cadena como delimitador. En la diapositiva se muestra cómo utilizar la notación q' para especificar el delimitador. El ejemplo utiliza ! y [como delimitadores. Considere el siguiente ejemplo:

```
v_event := q'!Father's day!';
```

Compárelo con el primer ejemplo de esta página. Debe empezar la cadena por q' si desea utilizar un delimitador. El carácter que sigue inmediatamente a la notación es el delimitador que se pretende utilizar. Introduzca la cadena después de especificar el delimitador, cierre el delimitador y cierre la notación con una sola comilla. El siguiente ejemplo muestra el uso de [como delimitador:

```
v_event := q'[Mother's day]';
```

Agenda

- Introducción a las variables
- Examen de los tipos de dato de las variables y el atributo %TYPE
- Examen de las variables de enlace

ORACLE

2 - 10

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Tipos de Variables

- Variables PL/SQL:
 - Escalar
 - Referencia
 - Objetos grandes (LOB)
 - Compuesta
- Variables no PL/SQL: Variables de enlace

ORACLE

Tipos de Variables

Todas las variables PL/SQL tienen un tipo de dato que especifica el formato de almacenamiento, las restricciones y un rango válido de valores. PL/SQL soporta varias categorías de tipos de dato, incluidas escalar, referencia, objeto grande (LOB) y compuesto.

- **Tipos de dato escalar:** los tipos de dato escalar contienen un único valor. Este valor depende del tipo de dato de la variable. Por ejemplo, la variable `v_myName` del ejemplo de la sección “Declaración e Inicialización de Variables PL/SQL” (de esta lección) es del tipo `VARCHAR2`. Por lo tanto, `v_myName` puede contener un valor de cadena. PL/SQL también soporta variables booleanas.
- **Tipos de dato de referencia:** los tipos de dato de referencia contienen determinados valores, llamados *punteros*, que apuntan a una ubicación de almacenamiento.
- **Tipos de dato LOB:** Los tipos de dato LOB contienen valores, llamados *localizadores*, que especifican la ubicación de objetos grandes (como imágenes gráficas) que están almacenados fuera de la tabla.
- **Tipos de dato compuestos:** los tipos de dato compuestos están disponibles si se utilizan las variables PL/SQL *collection* y *record*. Las recopilaciones y los registros de PL/SQL contienen elementos internos que se pueden tratar como variables individuales.

Las variables que no son PL/SQL incluyen variables de lenguaje del host declaradas en programas del precompilador, campos de pantalla de las aplicaciones de Forms y variables del host. Las variables del host se explican más adelante en esta misma lección.

Para obtener más información sobre LOB, consulte *PL/SQL User's Guide and Reference* (Guía del Usuario y Referencia PL/SQL).

Tipos de Variables

TRUE



15-ENE-09

Blancanieves
Hace mucho tiempo,
en un lejano país,
vivía una princesa llamada
Blancanieves. . .

Atlanta

256120.08

ORACLE

Tipos de Variables (continuación)

La diapositiva ilustra los siguientes tipos de dato:

- TRUE representa un valor booleano.
- 15-ENE-09 representa un dato DATE.
- La imagen representa un dato BLOB.
- El texto de la llamada puede representar un tipo de dato VARCHAR2 o CLOB.
- 256120.08 representa un tipo de dato NUMBER con precisión y escala.
- El rollo de película representa BFILE.
- El nombre de la ciudad, *Atlanta* representa un tipo de dato VARCHAR2.

Instrucciones para Declaración e Inicialización de Variables PL/SQL

- Siga reglas de nomenclatura consistentes.
- Utilice identificadores significativos para las variables.
- Inicialice las variables designadas como NOT NULL y CONSTANT.
- Inicialice las variables con el operador de asignación (:=) o la palabra clave DEFAULT:

```
v_myName VARCHAR2(20) := 'John' ;
```

```
v_myName VARCHAR2(20) DEFAULT 'John' ;
```

- Declare un identificador por línea para facilitar la lectura y el mantenimiento del código.



Instrucciones para Declaración e Inicialización de Variables PL/SQL

Estas son algunas instrucciones que se deben seguir al declarar variables PL/SQL.

- Siga reglas de nomenclatura consistentes, por ejemplo, puede utilizar name para representar una variable y c_name para representar una constante. Del mismo modo, para asignar un nombre a una variable, puede utilizar v_fname. La clave está en aplicar las reglas de nomenclatura de forma consistente para facilitar la identificación.
- Utilice identificadores significativos y adecuados para las variables. Por ejemplo, podría utilizar salary y sal_with_commission en lugar de salary1 y salary2.
- Si utiliza la restricción NOT NULL, debe asignar un valor al declarar la variable.
- En las declaraciones de constantes, debe anteponer la palabra clave CONSTANT al especificador de tipo. La siguiente declaración especifica una constante de tipo NUMBER y asigna el valor 50,000 a la constante. Las constantes se deben inicializar en su declaración; de lo contrario, aparece un error de compilación. Después de inicializar una constante, no se puede cambiar su valor.

```
sal CONSTANT NUMBER := 50000.00;
```

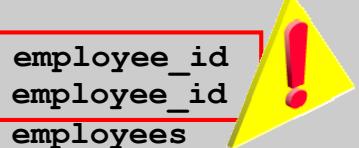
Instrucciones para Declaración de Variables PL/SQL

- Evite el uso de nombres de columna como identificadores.

```

DECLARE
    employee_id  NUMBER(6);
BEGIN
    SELECT
        INTO
        FROM
        WHERE
    END;
/

```



The code shows a PL/SQL block. Inside the SELECT statement, there are two occurrences of the column name 'employee_id' used as identifiers. This is highlighted with a red rectangular box around both instances. A yellow warning sign with an exclamation mark is positioned to the right of the boxed text.

- Utilice la restricción NOT NULL si la variable debe contener un valor.

ORACLE

Instrucciones para Declaración de Variables PL/SQL

- Inicialice las variables mediante una expresión que tenga el operador de asignación (:=) o con la palabra reservada DEFAULT. Si no se asigna ningún valor inicial, la nueva variable contiene NULL por defecto hasta que se asigna un valor. Para asignar o reasignar un valor a una variable, hay que escribir una sentencia de asignación PL/SQL. Sin embargo, es una práctica de programación aconsejable inicializar todas las variables.
- Dos objetos pueden tener el mismo nombre siempre que estén definidos en bloques diferentes. Si deben coexistir, se pueden cualificar con etiquetas para permitir su uso.
- Evite el uso de nombres de columna como identificadores. Si aparecen variables PL/SQL en sentencias SQL cuyo nombre es idéntico al de una columna, Oracle Server da por hecho que se hace referencia a la columna en cuestión. Aunque el código de ejemplo de la diapositiva funciona, el código escrito utilizando el mismo nombre para una tabla de base de datos y una variable no resulta fácil ni de leer ni de mantener.
- Imponga la restricción NOT NULL si la variable debe contener un valor. No puede asignar valores nulos a las variables definidas como NOT NULL. A continuación de la restricción NOT NULL debe aparecer una cláusula de inicialización.

```
pincode VARCHAR2(15) NOT NULL := 'Oxford';
```

Analizar y explicar

Reglas de Nomenclatura de Estructuras PL/SQL Utilizadas en este Curso

Estructura PL/SQL	Convención	Ejemplo
Variable	v_variable_name	v_rate
Constante	c_constant_name	c_rate
Parámetro de subprograma	p_parameter_name	p_id
Variable de enlace (host)	b_bind_name	b_salary
Cursor	cur_cursor_name	cur_emp
Registro	rec_record_name	rec_emp
Tipo	type_name_type	ename_table_type
Excepción	e_exception_name	e_products_invalid
Manejo de archivos	f_file_handle_name	f_file

ORACLE

2 - 15

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Reglas de Nomenclatura de Estructuras PL/SQL Utilizadas en este Curso

En la tabla de la diapositiva se muestran algunos ejemplos de las reglas de nomenclatura de las estructuras PL/SQL utilizadas en este curso.

Tipos de Dato Escalar

- Contienen un único valor
- Carecen de componentes internos

TRUE

15-ENE-09

A Dios rogando
y con el mazo dando.

256120.08

Atlanta

ORACLE

Tipos de Dato Escalar

PL/SQL proporciona diversos tipos de dato predefinidos. Así, es posible elegir entre enteros, coma flotante, caracteres, booleanos, fechas, recopilaciones y LOB. En esta lección se abordan los tipos básicos que se utilizan con frecuencia en los programas PL/SQL.

El tipo de dato escalar contiene un único valor y carece de componentes internos. Los tipos de dato escalar se pueden clasificar en cuatro categorías: número, carácter, fecha y booleano. Los tipos de dato de carácter y número disponen de subtipos que asocian un tipo base a una restricción. Por ejemplo, INTEGER y POSITIVE son subtipos del tipo base NUMBER.

Para obtener más información sobre los tipos de dato escalar (así como una lista completa), consulte *PL/SQL User's Guide and Reference* (Guía del Usuario y Referencia PL/SQL).

Tipos de Dato Escalar Base

- CHAR [(maximum_length)]
- VARCHAR2 (maximum_length)
- NUMBER [(precision, scale)]
- BINARY_INTEGER
- PLS_INTEGER
- BOOLEAN
- BINARY_FLOAT
- BINARY_DOUBLE

ORACLE

2 - 17

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Tipos de Dato Escalar Base

Tipo de Dato	Descripción
CHAR [(maximum_length)]	Tipo base para datos de carácter de longitud fija de hasta 32.767 bytes. Si no especifica ningún valor para maximum length, la longitud por defecto se define en 1.
VARCHAR2 (maximum_length)	Tipo base para datos de carácter de longitud variable de hasta 32.767 bytes. No hay ningún tamaño por defecto para las variables y las constantes VARCHAR2.
NUMBER [(precision, scale)]	Número que tiene precisión p y escala s . El rango de precisión p es de 1 a 38. El rango de escala s es de -84 a 127.
BINARY_INTEGER	Tipo base para enteros de entre -2,147,483,647 y 2,147,483,647.

Tipos de Dato Escalar Base (continuación)

Tipo de Dato	Descripción
PLS_INTEGER	Tipo base para enteros con signo de entre -2,147,483,647 y 2,147,483,647. Los valores de PLS_INTEGER necesitan menos almacenamiento y son más rápidos que los valores NUMBER. En Oracle Database 11g, los tipos de dato PLS_INTEGER y BINARY_INTEGER son idénticos. Las operaciones aritméticas con los valores PLS_INTEGER y BINARY_INTEGER son más rápidas que las realizadas con los valores NUMBER.
BOOLEAN	Tipo base que almacena uno de los tres posibles valores utilizados para los cálculos lógicos: TRUE, FALSE y NULL.
BINARY_FLOAT	Representa un número con coma flotante en formato IEEE 754. Necesita 5 bytes para almacenar el valor.
BINARY_DOUBLE	Representa un número con coma flotante en formato IEEE 754. Necesita 9 bytes para almacenar el valor.

Tipos de Dato Escalar Base

- DATE
- TIMESTAMP
- TIMESTAMP WITH TIME ZONE
- TIMESTAMP WITH LOCAL TIME ZONE
- INTERVAL YEAR TO MONTH
- INTERVAL DAY TO SECOND

ORACLE

Tipos de Dato Escalar Base (continuación)

Tipo de Dato	Descripción
DATE	Tipo base para fechas y horas. Los valores de DATE incluyen la hora del día en segundos desde la medianoche. El rango de fechas se sitúa entre el 4712 a.C. y el 9999 d.C.
TIMESTAMP	Tipo de dato que amplía el tipo de dato DATE, almacena el año, el mes, el día, la hora, el minuto, el segundo y la fracción de segundo. La sintaxis es TIMESTAMP [(precision)], donde el parámetro opcional precision especifica el número de dígitos presente en el campo de la fracción de segundo. Para especificar la precisión, se debe usar un entero del rango de 0 a 9. El valor por defecto es 6.
TIMESTAMP WITH TIME ZONE	Tipo de dato que amplía el tipo de dato TIMESTAMP; incluye el desplazamiento de zona horaria. El desplazamiento de zona horaria es la diferencia (en horas y minutos) entre la hora local y la Hora Universal Coordinada (UTC), antes denominada Hora Media de Greenwich. La sintaxis es TIMESTAMP [(precision)] WITH TIME ZONE, donde el parámetro opcional precision especifica el número de dígitos presente en el campo de la fracción de segundo. Para especificar la precisión, se debe usar un entero del rango de 0 a 9. El valor por defecto es 6.

Tipos de Dato Escalar Base (continuación)

Tipo de Dato	Descripción
TIMESTAMP WITH LOCAL TIME ZONE	<p>Tipo de dato que amplía el tipo de dato TIMESTAMP; incluye el desplazamiento de zona horaria. El desplazamiento de zona horaria es la diferencia (en horas y minutos) entre la hora local y la Hora Universal Coordinada (UTC), antes denominada Hora Media de Greenwich. La sintaxis es <code>TIMESTAMP [(precision)] WITH LOCAL TIME ZONE</code>, donde el parámetro opcional <code>precision</code> especifica el número de dígitos presente en el campo de la fracción de segundo. No se puede utilizar una variable o constante simbólica para especificar la precisión; se debe usar un literal entero del rango de 0 a 9. El valor por defecto es 6.</p> <p>Este tipo de dato difiere de <code>TIMESTAMP WITH TIME ZONE</code> en que, al insertar un valor en la columna de base de datos, el valor se normaliza en la zona horaria de la base de datos y en que el desplazamiento de zona horaria no se almacena en la columna. Cuando se recupera el valor, el servidor de Oracle lo devuelve en la zona horaria de la sesión local.</p>
INTERVAL YEAR TO MONTH	<p>Tipo de dato utilizado para almacenar y manipular intervalos de años y meses. La sintaxis es <code>INTERVAL YEAR [(precision)] TO MONTH</code>, donde <code>precision</code> especifica el número de dígitos del campo de años. No se puede utilizar una variable o constante simbólica para especificar la precisión; se debe usar un literal entero del rango de 0 a 4. El valor por defecto es 2.</p>
INTERVAL DAY TO SECOND	<p>Tipo de dato utilizado para almacenar y manipular intervalos de días, horas, minutos y segundos. La sintaxis es <code>INTERVAL DAY [(precision1)] TO SECOND [(precision2)]</code>, donde <code>precision1</code> y <code>precision2</code> especifican el número de dígitos de los campos de días y de segundos, respectivamente. En ninguno de los casos se puede utilizar una variable o constante simbólica para especificar la precisión; se debe usar un literal entero del rango de 0 a 9. Los valores por defecto son 2 y 6, respectivamente.</p>

Declaración de Variables Escalares

Ejemplos:

```
DECLARE
    v_emp_job          VARCHAR2(9);
    v_count_loop       BINARY_INTEGER := 0;
    v_dept_total_sal  NUMBER(9,2)  := 0;
    v_orderdate        DATE := SYSDATE + 7;
    c_tax_rate         CONSTANT NUMBER(3,2) := 8.25;
    v_valid            BOOLEAN NOT NULL := TRUE;
    ...
```

ORACLE

2 - 21

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Declaración de Variables Escalares

A continuación se definen los ejemplos de declaración de variables de la diapositiva:

- **v_emp_job:** variable para almacenar el cargo de un empleado
- **v_count_loop:** variable para contar las iteraciones de un bucle; se inicializa en 0
- **v_dept_total_sal:** variable para acumular el salario total de un departamento; se inicializa en 0
- **v_orderdate:** variable para almacenar la fecha de envío de un pedido; se inicializa en una semana desde el día actual
- **c_tax_rate:** variable constante para los impuestos que no cambia nunca en el bloque PL/SQL; se define en 8 . 25
- **v_valid:** indicador para señalar si un dato concreto es válido o no; se inicializa en TRUE

Atributo %TYPE

- Se utiliza para declarar variables según lo siguiente:
 - Una definición de columna de base de datos
 - Otra variable declarada
- Tiene antepuestos estos prefijos:
 - Nombre de tabla y columna de la base de datos
 - Nombre de la variable declarada

ORACLE

Atributo %TYPE

Las variables PL/SQL normalmente se declaran para contener y manipular los datos almacenados en las bases de datos. Cuando se declaran variables PL/SQL para contener valores de columnas, debe asegurarse de que el tipo de dato y la precisión de la variable oportuna son correctos. Si no lo son, se produce un error de PL/SQL durante la ejecución. Si se tienen que diseñar subprogramas grandes, puede resultar una tarea larga y proclive a los errores.

En lugar de codificar el tipo de dato y la precisión de las variables, se puede utilizar el atributo %TYPE para declararlas de acuerdo con otra variable declarada con anterioridad o con una columna de base de datos. El atributo %TYPE se utiliza con mayor frecuencia cuando el valor almacenado en la variable en cuestión se debe derivar de una tabla de la base de datos. Si se utiliza el atributo %TYPE para declarar una variable, es preciso anteponerle el nombre de la tabla y la columna de la base de datos. En caso de que se haga referencia a una variable antes declarada, se antepone el nombre de la variable antes declarada a la variable que se va a declarar.

Atributo %TYPE (continuación)

Ventajas del Atributo %TYPE

- Se pueden evitar errores provocados por tipos de dato no coincidentes o por una precisión incorrecta.
- Se puede evitar la codificación del tipo de dato de la variable.
- No hace falta cambiar la declaración de la variable en caso de que cambie la definición de la columna. Si ya se han declarado algunas variables para una tabla en particular prescindiendo del atributo %TYPE, el bloque PL/SQL puede devolver errores si se modifica la columna para la que se ha declarado la variable en cuestión. Sin embargo, cuando se utiliza el atributo %TYPE, PL/SQL determina el tipo de dato y el tamaño de la variable al compilar el bloque. Esto garantiza que la variable sea siempre compatible con la columna que se utiliza para rellenarla.

Declaración de Variables con el Atributo %TYPE

Sintaxis

```
identifier      table.column_name%TYPE;
```

Ejemplos

```
...
  v_emp_lname      employees.last_name%TYPE;
...
```

```
...
  v_balance        NUMBER(7,2);
  v_min_balance   v_balance%TYPE := 1000;
...
```

ORACLE

Declaración de Variables con el Atributo %TYPE

Declare variables para almacenar el apellido de un empleado. La variable `v_emp_lname` se define de forma que sea del mismo tipo de dato que la columna `v_last_name` de la tabla `employees`. El atributo `%TYPE` proporciona el tipo de dato de una columna de base de datos.

Declare variables para almacenar el saldo de una cuenta bancaria, así como el saldo mínimo, que es 1.000. La variable `v_min_balance` se define para que tenga el mismo tipo de dato que la variable `v_balance`. El atributo `%TYPE` proporciona el tipo de dato de una variable.

Las restricciones de columna de base de datos NOT NULL no se aplican a las variables que se declaran con `%TYPE`. Por lo tanto, si declara una variable con el atributo `%TYPE` que utiliza una columna de base de datos definida como NOT NULL, puede asignar el valor NULL a la variable.

Declaración de Variables Booleanas

- Sólo se puede asignar los valores TRUE, FALSE y NULL a las variables booleanas.
- Las expresiones condicionales utilizan operadores lógicos AND y OR, así como el operador unario NOT para comprobar los valores de las variables.
- Las variables siempre generan TRUE, FALSE o NULL.
- Se pueden utilizar expresiones aritméticas, de caracteres y de fechas para devolver un valor booleano.

ORACLE

2 - 25

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Declaración de Variables Booleanas

PL/SQL permite comparar variables tanto en sentencias SQL como en sentencias de procedimiento. Estas comparaciones, denominadas expresiones booleanas, constan de expresiones simples o complejas separadas por operadores relacionales. En las sentencias SQL, las expresiones booleanas sirven, por ejemplo, para especificar las filas de una tabla a las que afecta la sentencia. En las sentencias de procedimiento, las expresiones booleanas constituyen la base para el control condicional. NULL indica un valor que falta, que no es aplicable o que se desconoce.

Ejemplos

```
emp_sal1 := 50000;
emp_sal2 := 60000;
```

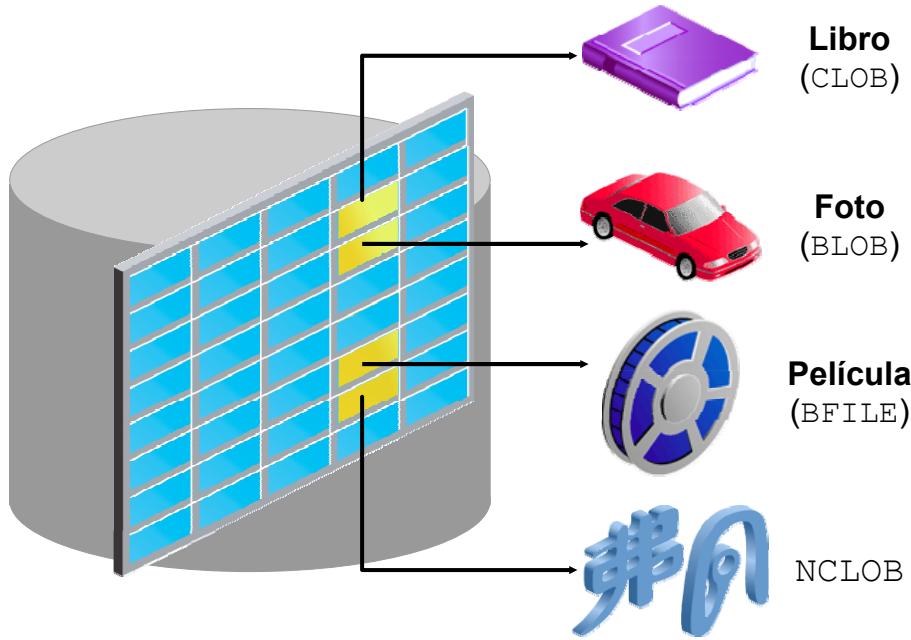
La siguiente expresión genera TRUE:

```
emp_sal1 < emp_sal2
```

Declare e inicialice una variable booleana:

```
DECLARE
    flag BOOLEAN := FALSE;
BEGIN
    flag := TRUE;
END;
/
```

Variables de Tipos de Datos LOB



ORACLE

Variables de Tipos de Datos LOB

Los objetos grandes (LOB) están concebidos para almacenar grandes cantidades de datos. Una columna de base de datos, por ejemplo, puede pertenecer a la categoría LOB. Con la categoría LOB de los tipos de dato (BLOB, CLOB, etc.), puede almacenar bloques de datos no estructurados (como texto, imágenes gráficas, videoclips u ondas sonoras) con un tamaño de hasta 128 TB dependiendo del tamaño de bloque de la base de datos. Los tipos de dato LOB facilitan el acceso de piecewise aleatorio y eficaz a los datos y pueden ser atributos de tipos de objeto.

- El tipo de dato de objeto grande de caracteres (CLOB) se utiliza para almacenar grandes bloques de datos de caracteres en las bases de datos.
- El tipo de dato de objeto grande binario (BLOB) se utiliza para almacenar grandes objetos binarios estructurados o no estructurados en las bases de datos. Al insertar o recuperar dichos datos de la base de datos, la base de datos no interpreta los datos. Son las aplicaciones externas que utilizan los datos las que deben interpretarlos.
- El tipo de dato de archivo binario (BFILE) se utiliza para almacenar archivos binarios de gran tamaño. A diferencia de otros LOB, BFILES se almacena fuera de la base de datos y no en la base de datos. Puede tratarse de archivos del sistema operativo. En la base de datos sólo se almacena un puntero en BFILE.
- El tipo de dato de objeto grande de caracteres del idioma nacional (NCLOB) se utiliza para almacenar grandes bloques de datos Unicode NCHAR de un solo byte o multibyte de ancho fijo en las bases de datos.

Tipos de Dato Compuestos: Registros y Recopilaciones

Registro PL/SQL:

TRUE	23-DEC-98	ATLANTA	
------	-----------	---------	---

Recopilaciones PL/SQL:

1	SMITH	1	5000
2	JONES	2	2345
3	NANCY	3	12
4	TIM	4	3456

↓ ↓ ↓ ↓
 PLS_INTEGER VARCHAR2 PLS_INTEGER NUMBER
 ↑ ↑ ↑ ↑

ORACLE

Tipos de Dato Compuestos: Registros y Recopilaciones

Como se ha mencionado anteriormente, el tipo de dato escalar contiene un único valor y carece de componentes internos. Los tipos de dato compuestos (denominados registros PL/SQL y recopilaciones PL/SQL) tienen componentes internos que se pueden considerar variables individuales.

- En un registro PL/SQL, los componentes internos pueden ser de diferentes tipos de dato, y se denominan campos. Puede acceder a cada campo con esta sintaxis: `record_name.field_name`. Una variable de registro puede contener una fila de tabla o algunas columnas de una fila de tabla. Cada campo de registro corresponde a una columna de tabla.
- En una recopilación PL/SQL, los componentes internos son siempre del mismo tipo de dato y se denominan elementos. Puede acceder a cada elemento con su script secundario único. Las listas y matrices son ejemplos clásicos de recopilaciones. Existen tres tipos de recopilaciones PL/SQL: matrices asociativas, tablas anidadas y tipos VARRAY.

Nota

- Los registros PL/SQL y las matrices asociativas se tratan en la lección titulada: “Trabajar con Tipos de Dato Compuestos”.
- Los tipos de dato NESTED TABLE y VARRAY se abordan en el curso titulado *Oracle Database 10g: PL/SQL Avanzado* u *Oracle Database 11g: PL/SQL Avanzado*.

Agenda

- Introducción a las variables
- Examen de los tipos de dato de las variables y el atributo %TYPE
- Examen de las variables de enlace

ORACLE

Variables de Enlace

Las variables de enlace:

- Se crean en el entorno
- Se llaman también variables del *host*
- Se crean con la palabra clave* VARIABLE
- Se utilizan en sentencias SQL y bloques PL/SQL
- Son accesibles incluso después de haber ejecutado los bloques PL/SQL
- Son objeto de referencia si se anteponen los dos puntos

Se puede generar la salida de los valores con el comando PRINT.

* Es necesaria si se utiliza SQL*Plus y SQL Developer

Variables de Enlace

Las variables de enlace son variables que se crean en el entorno de un host. Por este motivo, a veces se denominan variables del *host*.

Usos de las Variables de Enlace

Las variables de enlace se crean en el entorno y no en la sección de declaraciones de los bloques PL/SQL. Por lo tanto, las variables de enlace son accesibles incluso después de haber ejecutado los bloques. Cuando se crean variables de enlace, pueden utilizarlas y manipularlas varios subprogramas. Su uso en las sentencias SQL y los bloques PL/SQL es idéntico al de cualquier otra variable. Estas variables se pueden transferir como valores de tiempo de ejecución hacia subprogramas PL/SQL o desde ellos.

Nota: una variable de enlace es una variable de entorno, pero no es una variable global.

Creación de Variables de Enlace

Para crear una variable de enlace en SQL Developer, utilice el comando VARIABLE. Por ejemplo, una variable de tipo NUMBER y VARCHAR2 se declara del siguiente modo:

```
VARIABLE return_code NUMBER
VARIABLE return_msg  VARCHAR2(30)
```

Visualización de Valores en Variables de Enlace

Puede hacer referencia a la variable de enlace con SQL Developer y ver su valor con el comando PRINT.

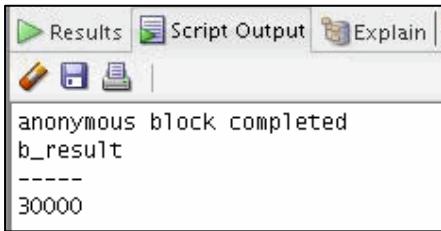
Variables de Enlace (continuación)

Ejemplo

Para hacer referencia a variables de enlace en los programas PL/SQL, hay que anteponer los dos puntos a la variable pertinente.

Por ejemplo, el siguiente bloque PL/SQL crea y utiliza la variable de enlace `b_result`. La salida resultante del comando `PRINT` se muestra debajo del código.

```
VARIABLE b_result NUMBER
BEGIN
    SELECT (SALARY*12) + NVL(COMMISSION_PCT,0) INTO :b_result
    FROM employees WHERE employee_id = 144;
END;
/
PRINT b_result
```



Nota: si se pretende crear una variable de enlace de tipo NUMBER, no se puede especificar la precisión y la escala, pero sí el tamaño de las cadenas de caracteres. El tipo NUMBER de Oracle se almacena del mismo modo sea cual sea la dimensión. Oracle Server utiliza el mismo número de bytes para almacenar 7, 70 o 0,0734. No resulta práctico calcular el tamaño de la representación numérica de Oracle a partir del formato del número, por lo que el código asigna siempre los bytes necesarios. Con las cadenas de caracteres, el usuario debe especificar el tamaño para poder asignar el número de bytes necesarios.

Referencia a Variables de Enlace

Ejemplo:

```
VARIABLE b_emp_salary NUMBER
BEGIN
    SELECT salary INTO :b_emp_salary
    FROM employees WHERE employee_id = 178;
END;
/
PRINT b_emp_salary
SELECT first_name, last_name
FROM employees
WHERE salary=:b_emp_salary;
```

Salida →

The screenshot shows the Oracle SQL Developer interface. On the left, there is a code editor window containing the provided PL/SQL code. On the right, there is a results window showing the output of the execution. The results window has tabs for 'Results', 'Script Output', 'Explain', and 'Autotrace'. The 'Results' tab is selected, displaying the following output:

FIRST_NAME	LAST_NAME
Oliver	Tuvault
Sarah	Sewall
Kimberely	Grant

Below the table, it says '3 rows selected'.

ORACLE

Referencia a Variables de Enlace

Como ya se ha explicado, después de crear una variable de enlace, puede hacer referencia a esa variable en otra sentencia SQL o programa PL/SQL.

En el ejemplo, se crea `b_emp_salary` como variable de enlace en el bloque PL/SQL. A continuación, se utiliza la sentencia `SELECT` de la siguiente forma.

Al ejecutar el bloque PL/SQL mostrado en la diapositiva, aparece la siguiente salida:

- El comando `PRINT` ejecuta:

```
b_emp_salary
-----
7000
```

- La salida de la sentencia SQL es la siguiente:

```
FIRST_NAME      LAST_NAME
-----          -----
Oliver          Tuvault
Sarah           Sewall
Kimberely       Grant
```

Nota: para mostrar todas las variables de enlace, utilice el comando `PRINT` sin variable.

Analizar, explicar y ejecutar

Uso de AUTOPRINT con Variables de Enlace

The screenshot shows the Oracle SQL Worksheet interface. In the main pane, there is PL/SQL code:

```
VARIABLE b_emp_salary NUMBER
SET AUTOPRINT ON
DECLARE
    v_empno NUMBER(6):=&empno;
BEGIN
    SELECT salary INTO :b_emp_salary
    FROM employees WHERE employee_id = v_empno;
END;
/
```

A red box highlights the line `SET AUTOPRINT ON`. A red arrow points from this line to the `Enter Substitution Variable` dialog box. This dialog box has a single input field labeled "EMPNO:" containing the value "178". Another red arrow points from this dialog to the "Script Output" window, which displays the results of the execution:

```
anonymous block completed
b_emp_salary
-----
7000
```

ORACLE

2 - 32

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Uso de AUTOPRINT con Variables de Enlace

El comando `SET AUTOPRINT ON` permite mostrar automáticamente las variables de enlace que se utilizan en los bloques PL/SQL correctos.

Ejemplo

En el código de ejemplo:

- Se crea una variable de enlace denominada `b_emp_salary` y se activa `AUTOPRINT`.
- Se declara una variable denominada `v_empno` y se utiliza una variable de sustitución para recibir entrada de usuario.
- Por último, se utilizan la variable de enlace y las variables temporales en la sección ejecutable del bloque PL/SQL.

Al introducir un número de empleado válido, en este caso 178, se imprime automáticamente la variable de enlace. La variable de enlace contiene el salario del número de empleado proporcionado por el usuario.

Contestar

Prueba

El atributo %TYPE:

- a. Se utiliza para declarar una variable según una definición de columna de base de datos
- b. Se utiliza para declarar una variable según la recopilación de columnas en una vista o tabla de base de datos
- c. Se utiliza para declarar una variable según la definición de otra variable declarada
- d. Se antepone con el nombre de columna y tabla de base de datos o el nombre de la variable declarada

ORACLE

2 - 33

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Respuesta: a, c, d

Atributo %TYPE

Las variables PL/SQL normalmente se declaran para contener y manipular los datos almacenados en las bases de datos. Cuando se declaran variables PL/SQL para contener valores de columnas, debe asegurarse de que el tipo de dato y la precisión de la variable oportuna son correctos. Si no lo son, se produce un error de PL/SQL durante la ejecución. Si se tienen que diseñar subprogramas grandes, puede resultar una tarea larga y proclive a los errores.

En lugar de codificar el tipo de dato y la precisión de las variables, se puede utilizar el atributo %TYPE para declararlas de acuerdo con otra variable declarada con anterioridad o con una columna de base de datos. El atributo %TYPE se utiliza con mayor frecuencia cuando el valor almacenado en la variable en cuestión se debe derivar de una tabla de la base de datos. Si se utiliza el atributo %TYPE para declarar una variable, es preciso anteponerle el nombre de la tabla y la columna de la base de datos. En caso de que se haga referencia a una variable antes declarada, se antepone el nombre de la variable antes declarada a la variable que se va a declarar. La ventaja de %TYPE es que no tendrá que cambiar la variable si se modifica la columna. Asimismo, si la variable se utiliza en un cálculo, no tiene que preocuparse de su precisión.

Atributo %ROWTYPE

El atributo %ROWTYPE se utiliza para declarar un registro que puede contener una fila completa de una tabla o vista. Dispone de más información sobre este atributo en la lección titulada “Trabajar con Tipos de Dato Compuestos”.

Resumen

En esta lección, debe haber aprendido lo siguiente:

- Reconocer identificadores válidos y no válidos
- Declarar variables en la sección de declaraciones de los bloques PL/SQL
- Inicializar variables y utilizarlas en la sección ejecutable
- Diferenciar los tipos de dato escalar y compuesto
- Utilizar el atributo %TYPE
- Utilizar variables de enlace



Resumen

Los bloques PL/SQL anónimos son unidades básicas sin nombre de los programas PL/SQL. Constan de un juego de sentencias SQL o PL/SQL para realizar una función lógica. La parte de declaraciones es la primera parte de los bloques PL/SQL y se utiliza para declarar objetos tales como variables, constantes, cursosres y definiciones de situaciones de error llamadas *excepciones*.

En esta lección, ha aprendido a declarar variables en la sección de declaraciones. Ha visto algunas instrucciones para la declaración de variables. Ha aprendido a inicializar variables al declararlas.

La parte ejecutable es la parte obligatoria de los bloques PL/SQL y contiene las sentencias SQL y PL/SQL para consultar y manipular los datos. Ha aprendido a inicializar las variables de la sección ejecutable, así como a utilizarlas y a manipular sus valores.

Crea de manera individual un bloque PL/SQL que cubra los puntos de la Práctica 2

Práctica 2: Visión General

En esta práctica se abordan los siguientes temas:

- Determinación de identificadores válidos
- Determinación de declaraciones de variables válidas
- Declaración de variables en bloques anónimos
- Uso del atributo %TYPE para declarar variables
- Declaración e impresión de variables de enlace
- Ejecución de bloques PL/SQL

ORACLE

2 - 35

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Práctica 2: Visión General

Los ejercicios 1, 2 y 3 se responden sobre papel.

Escritura de Sentencias Ejecutables



ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Objetivos

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

- Identificar unidades léxicas en los bloques PL/SQL
- Utilizar funciones SQL incorporadas en PL/SQL
- Describir cuándo tienen lugar conversiones implícitas y cuándo hay que manejar conversiones explícitas
- Escribir bloques anidados y cualificar variables con etiquetas
- Escribir código legible con sangrados adecuados
- Utilizar secuencias en expresiones PL/SQL



Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Objetivos

Ha aprendido a declarar variables y escribir sentencias ejecutables en los bloques PL/SQL. En esta lección, aprenderá cómo las unidades léxicas forman los bloques PL/SQL. Aprenderá a escribir bloques anidados. Además, conocerá el ámbito y la visibilidad de las variables de los bloques anidados y también aprenderá a cualificarlas con etiquetas.

Agenda

- Escritura de sentencias ejecutables en los bloques PL/SQL
- Escritura de bloques anidados
- Uso de operadores y desarrollo de código legible

ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Unidades Léxicas de los Bloques PL/SQL

Las unidades léxicas:

- Son los bloques integrantes de cualquier bloque PL/SQL
- Son secuencias de caracteres que incluyen letras, números, tabulaciones, espacios, retornos y símbolos
- Se clasifican como:
 - Identificadores: v_fname, c_percent
 - Delimitadores: ; , +, -
 - Literales: John, 428, True
 - Comentarios: --, /* */



Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Unidades Léxicas de los Bloques PL/SQL

Las unidades léxicas incluyen letras, números, caracteres especiales, tabulaciones, espacios, retornos y símbolos.

- **Identificadores:** los identificadores son los nombres otorgados a los objetos PL/SQL. Ya se ha explicado cómo distinguir los identificadores válidos y los no válidos. Recuerde que no se pueden utilizar palabras clave como identificadores.

Identificadores entre comillas:

- Haga que los identificadores sean sensibles a mayúsculas y minúsculas.
- Incluya determinados caracteres como, por ejemplo, espacios.
- Utilice palabras reservadas.

Ejemplos:

```
"begin date" DATE;
"end date"   DATE;
"exception thrown" BOOLEAN DEFAULT TRUE;
```

Todos los usos subsiguientes de estas variables deben llevar comillas dobles. Sin embargo, no se recomienda el uso de identificadores con comillas.

- **Delimitadores:** los delimitadores son símbolos que poseen un significado especial. Ya se ha explicado que el punto y coma (;) se utiliza para terminar las sentencias SQL y PL/SQL. Por lo tanto, ; es un ejemplo de delimitador.

Para obtener más información, consulte *PL/SQL User's Guide and Reference* (Guía del Usuario y Referencia PL/SQL).

Unidades Léxicas de los Bloques PL/SQL (continuación)

- **Delimitadores (continuación)**

Los delimitadores son símbolos simples o compuestos que poseen un significado especial en PL/SQL.

Símbolos Simples

Símbolo	Significado
+	Operador de suma
-	Operador de resta/negación
*	Operador de multiplicación
/	Operador de división
=	Operador de igualdad
@	Indicador de acceso remoto
;	Terminador de sentencia

Símbolos Compuestos

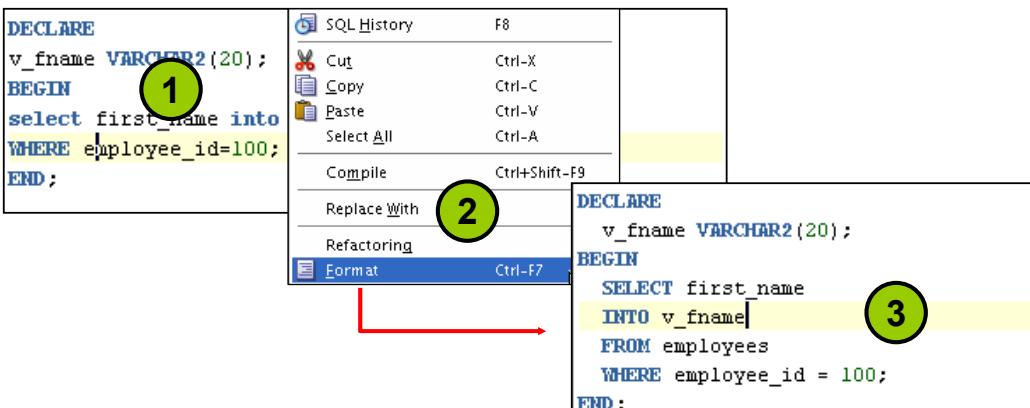
Símbolo	Significado
<>	Operador de no igualdad
!=	Operador de no igualdad
	Operador de concatenación
--	Indicador de comentario de una línea
/*	Delimitador de principio de comentario
*/	Delimitador de fin de comentario
:=	Operador de asignación

Nota: estas listas no están completas, sólo presentan un subjuego de delimitadores.

- **Literales:** cualquier valor asignado a una variable es un literal. Cualquier carácter, número, valor booleano o valor de fecha que no es un identificador es un literal. Los literales se clasifican del siguiente modo:
 - **Literales de caracteres:** todos los literales de cadena tienen el tipo de dato CHAR o VARCHAR2 y, por consiguiente, se denominan literales de caracteres (por ejemplo, John y 12C).
 - **Literales numéricos:** los literales numéricos representan enteros o valores reales (por ejemplo, 428 y 1.276).
 - **Literales booleanos:** los valores asignados a las variables booleanas son literales booleanos. TRUE, FALSE y NULL son literales booleanos o palabras clave.
- **Comentarios:** desde el punto de vista de la programación, es una práctica aconsejable explicar qué se intenta obtener con cada fragmento del código. Sin embargo, si se incluye la explicación en los bloques PL/SQL, el compilador es incapaz de interpretar estas instrucciones. Por lo tanto, tiene que existir alguna forma de indicar que no hace falta compilar dichas instrucciones. Los comentarios se utilizan principalmente con este fin. El compilador no interpreta ninguna instrucción que aparezca comentada.
 - Para comentar una sola línea, se utilizan dos guiones (--).
 - Para comentar varias líneas, se utilizan los delimitadores de inicio y fin /* y */.

Instrucciones y Sintaxis de Bloques PL/SQL

- Uso de Literales
 - Los literales de caracteres y fecha se deben incluir entre comillas simples.
 - Los números pueden ser valores simples o en notaciones científicas.
- Formato al Código: las sentencias pueden ocupar varias líneas.



ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Instrucciones y Sintaxis de Bloques PL/SQL

Uso de Literales

Un literal es cualquier valor explícito numérico, booleano, de cadena de caracteres o de fecha que no está representado por un identificador.

- Los literales de caracteres incluyen todos los caracteres imprimibles del juego de caracteres de PL/SQL: letras, números, espacios y símbolos especiales.
- Los literales numéricos pueden estar representados por un valor simple (por ejemplo, `-32.5`) o en una notación científica (por ejemplo, `2E5` significa $2 * 10^5 = 200.000$).

Formato de Código

En un bloque PL/SQL, una sentencia SQL puede ocupar varias líneas (como se muestra en el ejemplo 3 de la diapositiva).

Puede dar formato a una sentencia SQL que no lo tenga (como se muestra en el ejemplo 1 de la diapositiva) mediante el menú de acceso directo SQL Worksheet. Haga clic con el botón derecho en la opción SQL Worksheet activa y seleccione la opción Format en el menú de acceso directo que aparece (como se muestra en el ejemplo 2).

Nota: también puede utilizar la combinación de teclas de acceso directo Ctrl + F7 para dar formato al código.

Comentario del Código

- Anteponga dos guiones (--) a los comentarios de una sola línea.
- Incluya el comentario de un bloque entre los símbolos /* y */.

Ejemplo:

```
DECLARE
  ...
  v_annual_sal NUMBER (9,2);
BEGIN
  /* Compute the annual salary based on the
   monthly salary input from the user */
  v_annual_sal := monthly_sal * 12;
  --The following line displays the annual salary
  DBMS_OUTPUT.PUT_LINE(v_annual_sal);
END;
/
```

ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Comentario del Código

Debe comentar el código para documentar cada una de las fases y facilitar la depuración. En el código PL/SQL:

- Para comentar una sola línea, se utilizan dos guiones (--)
- Puede incluir un comentario entre los símbolos /* y */

Nota: para comentarios de varias líneas, puede anteponer dos guiones a cada línea de comentario o utilizar el formato de comentario de bloque.

Los comentarios son meramente informativos y no fuerzan ninguna condición ni ningún comportamiento en la lógica ni los datos. Una correcta colocación de los comentarios los convierte en una ayuda de incalculable valor para la legibilidad del código y su futuro mantenimiento.

Funciones SQL en PL/SQL

- Disponibles en sentencias de procedimiento:
 - Funciones de una sola fila
- No disponibles en sentencias de procedimiento:
 - DECODE
 - Funciones de grupo



Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Funciones SQL en PL/SQL

SQL proporciona varias funciones predefinidas para su uso en sentencias SQL. La mayoría de estas funciones (como las funciones de número y carácter de una fila, las funciones de conversión de tipo de dato y las funciones de registro de fecha y hora) son válidas en las expresiones PL/SQL.

Las siguientes funciones no están disponibles en sentencias de procedimiento:

- DECODE
- Funciones de grupo: AVG, MIN, MAX, COUNT, SUM, STDDEV y VARIANCE

Las funciones de grupo se aplican a grupos de filas de las tablas y, por lo tanto, sólo están disponibles en las sentencias SQL de los bloques PL/SQL. Las funciones aquí mencionadas son sólo un subjuego, no una lista completa.

Funciones SQL en PL/SQL: Ejemplos

Crea un bloque PLSQL para declarar la variable siguiente y mostrar el tamaño de la cadena en pantalla

- Obtenga la longitud de una cadena:

```
v_desc_size INTEGER(5);
v_prod_description VARCHAR2(70):='You can use this
product with your radios for higher frequency';

-- get the length of the string in prod_description
v_desc_size:= LENGTH(v_prod_description);
```

- Obtenga el número de meses que ha trabajado un empleado: **Realizar el ejercicio con cualquier empleado de la tabla EMPLOYEES**

```
v_tenure:= MONTHS_BETWEEN (CURRENT_DATE, v_hiredate);
```

ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Funciones SQL en PL/SQL: Ejemplos

Puede utilizar funciones SQL para manipular los datos. Estas funciones se agrupan en las siguientes categorías:

- Número
- Carácter
- Conversión
- Fecha
- Otras

Uso de Secuencias en Expresiones PL/SQL

Explica y crea una secuencia y realiza el ejercicio a partir de Oracle 11g

https://docs.oracle.com/cd/B12037_01/server.101/b10759/statements_6014.htm

A partir de 11g:

```
DECLARE
    v_new_id NUMBER;
BEGIN
    v_new_id := my_seq.NEXTVAL;
END;
/
```

Antes de 11g:

```
DECLARE
    v_new_id NUMBER;
BEGIN
    SELECT my_seq.NEXTVAL INTO v_new_id FROM Dual;
END;
/
```

ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Acceso a Valores de Secuencia

En Oracle Database 11g, puede utilizar las pseudocolumnas NEXTVAL y CURRVAL en cualquier contexto PL/SQL, donde pueda aparecer de forma válida una expresión del tipo de dato NUMBER. Aunque sigue siendo válido el estilo anterior de uso de una sentencia SELECT para consultar una secuencia, se recomienda que no se utilice.

Antes de Oracle Database 11g, se debía escribir una sentencia SQL para poder utilizar un valor de objeto de secuencia en una subrutina PL/SQL. Normalmente, se escribía una sentencia SELECT para hacer referencia a las pseudocolumnas de NEXTVAL y CURRVAL para obtener un número de secuencia. Este método creaba un problema de uso.

En Oracle Database 11g, se ha eliminado la limitación de escritura de sentencias SQL para recuperar valores de secuencia. Con la función de mejora de secuencia:

- Hay mayor posibilidad de uso de la secuencia
- El desarrollador tiene que escribir menos
- El código resultante es más claro

Conversión del Tipo de Dato

- Los datos se convierten a tipos de dato parecidos
- Existen dos tipos:
 - Conversión implícita
 - Conversión explícita
- Funciones:
 - TO_CHAR
 - TO_DATE
 - TO_NUMBER
 - TO_TIMESTAMP

ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Conversión del Tipo de Dato

En cualquier lenguaje de programación, la conversión de un tipo de dato a otro es una necesidad común. PL/SQL puede manejar estas conversiones con tipos de dato escalar. Las conversiones del tipo de dato pueden ser de dos tipos:

Conversión implícita: PL/SQL intenta convertir los tipos de dato de forma dinámica si aparecen mezclados en una sentencia. Considere el siguiente ejemplo:

```
DECLARE
    v_salary NUMBER(6):=6000;
    v_sal_hike VARCHAR2(5):='1000';
    v_total_salary v_salary%TYPE;
BEGIN
    v_total_salary:=v_salary + v_sal_hike;
END;
/
```

En este ejemplo, la variable `sal_hike` es del tipo VARCHAR2. Al calcular el salario total, PL/SQL convierte primero `sal_hike` a NUMBER y, a continuación, realiza la operación. El resultado es del tipo NUMBER.

Las conversiones implícitas se producen entre lo siguiente:

- Caracteres y números
- Caracteres y fechas

Conversión del Tipo de Dato (continuación)

Conversión explícita: para convertir valores de un tipo de dato a otro, utilice las funciones incorporadas. Por ejemplo, para convertir un valor CHAR en un valor DATE o NUMBER, utilice TO_DATE o TO_NUMBER, respectivamente.

Conversión del Tipo de Dato

1

```
-- implicit data type conversion  
v_date_of_joining DATE:= '02-Feb-2000';
```

2

```
-- error in data type conversion  
v_date_of_joining DATE:= 'February 02,2000';
```

3

```
-- explicit data type conversion  
v_date_of_joining DATE:= TO_DATE('February  
02,2000','Month DD, YYYY');
```

ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Conversión del Tipo de Dato (continuación)

Observe los tres ejemplos de conversión implícita y explícita del tipo de dato DATE de la diapositiva:

1. Ya que el literal de cadena asignado a date_of_joining tiene el formato por defecto, este ejemplo realiza una conversión implícita y asigna la fecha especificada a date_of_joining.
2. PL/SQL devuelve un error porque la fecha que se asigna no está en el formato por defecto.
3. La función TO_DATE se utiliza para convertir de forma explícita la fecha aportada en un formato concreto y asignarla a la variable date_of_joining del tipo de dato DATE.

Agenda

- Escritura de sentencias ejecutables en los bloques PL/SQL
- Escritura de bloques anidados
- Uso de operadores y desarrollo de código legible

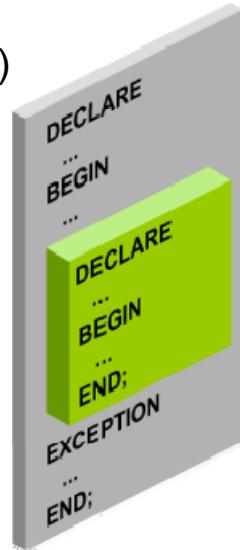
ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Bloques Anidados

Los bloques PL/SQL se pueden anidar.

- La sección ejecutable (`BEGIN ... END`) puede contener bloques anidados.
- La sección de excepciones puede contener bloques anidados.



ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Bloques Anidados

Al tratarse de un procedimiento, PL/SQL puede anidar sentencias. Se pueden anidar bloques siempre que se permita una sentencia ejecutable, de manera que el bloque anidado se convierte en una sentencia. Si en la sección ejecutable aparece código para muchas funcionalidades relacionadas lógicamente para soportar varias necesidades de negocio, dicha sección se puede dividir en bloques más pequeños. La sección de excepciones también puede contener bloques anidados.

Bloques Anidados: Ejemplo

```

DECLARE
  v_outer_variable VARCHAR2(20) := 'GLOBAL VARIABLE';
BEGIN
  DECLARE
    v_inner_variable VARCHAR2(20) := 'LOCAL VARIABLE';
  BEGIN
    DBMS_OUTPUT.PUT_LINE(v_inner_variable);
    DBMS_OUTPUT.PUT_LINE(v_outer_variable);
  END;
  DBMS_OUTPUT.PUT_LINE(v_outer_variable);
END;

```

anonymous block completed
 LOCAL VARIABLE
 GLOBAL VARIABLE
 GLOBAL VARIABLE

ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Bloques Anidados (continuación)

En el ejemplo de la diapositiva se muestra un bloque externo (principal) y un bloque anidado (secundario). La variable `v_outer_variable` se declara en el bloque externo y la variable `v_inner_variable` en el bloque interno.

`v_outer_variable` es local respecto al bloque externo, pero global respecto al bloque interno. Cuando se accede a esta variable en el bloque interno, PL/SQL busca primero una variable local en el bloque interno con ese nombre. No hay ninguna variable con el mismo nombre en el bloque interno, por lo tanto, PL/SQL busca la variable en el bloque externo. Por lo tanto, `v_outer_variable` se considera la variable global para todos los bloques delimitadores. A esta variable se puede acceder en el bloque interno, tal como se muestra en la diapositiva. Las variables declaradas en los bloques PL/SQL se consideran locales respecto al bloque pertinente y globales respecto a todos sus subbloques.

`v_inner_variable` es local respecto al bloque interno, pero no es global porque el bloque interno carece de bloques anidados. A esta variable sólo se puede acceder dentro del bloque interno. Si PL/SQL no encuentra la variable declarada localmente, realiza una búsqueda en sentido ascendente en la sección de declaraciones de los bloques principales. PL/SQL no efectúa la búsqueda en sentido descendente hacia los bloques secundarios.

Ámbito y Visibilidad de las Variables

```

DECLARE
    v_father_name VARCHAR2(20) :='Patrick';
    v_date_of_birth DATE:='20-Apr-1972';
BEGIN
    DECLARE
        v_child_name VARCHAR2(20) :='Mike';
        v_date_of_birth DATE:='12-Dec-2002';
    BEGIN
        DBMS_OUTPUT.PUT_LINE('Father''s Name: '||v_father_name);
        DBMS_OUTPUT.PUT_LINE('Date of Birth: '||v_date_of_birth);
        DBMS_OUTPUT.PUT_LINE('Child''s Name: '||v_child_name);
    END;
    DBMS_OUTPUT.PUT_LINE('Date of Birth: '||v_date_of_birth);
END;
/

```

ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Ámbito y Visibilidad de las Variables

La salida del bloque que se muestra en la diapositiva es la siguiente:

```

anonymous block completed
Father's Name: Patrick
Date of Birth: 12-DEC-02
Child's Name: Mike
Date of Birth: 20-APR-72

```

Observe la fecha de nacimiento que se imprime para el padre y el hijo. La salida no proporciona la información correcta, porque el *ámbito* y la *visibilidad* de las variables no se han aplicado correctamente.

- El *ámbito* de la variable es la parte del programa en la que se declara la variable y ésta se encuentra accesible.
- La *visibilidad* de la variable es la parte del programa en la que se puede acceder a la variable sin utilizar ningún cualificador.

Ámbito

- La variable `v_father_name` y la primera incidencia de la variable `v_date_of_birth` se declaran en el bloque externo. Estas variables tienen el *ámbito* del bloque en el que se declaran. Por lo tanto, el *ámbito* de dichas variables se limita al bloque externo.

Ámbito y Visibilidad de las Variables (continuación)

Ámbito (continuación)

- Las variables `v_child_name` y `v_date_of_birth` se declaran en el bloque interno o bloque anidado. Estas variables sólo son accesibles en el bloque anidado, no lo son en el bloque externo. Cuando una variable queda fuera del ámbito, PL/SQL libera la memoria utilizada para almacenarla y, por consecuencia, no es posible hacer referencia a dicha variable.

Visibilidad

- La variable `v_date_of_birth` declarada en el bloque externo tiene su ámbito incluso en el bloque interno. Sin embargo, esta variable no es visible en el bloque interno, porque éste dispone de una variable local con el mismo nombre.
 1. Examine el código de la sección ejecutable del bloque PL/SQL. Se pueden imprimir el nombre del padre, el nombre del hijo y la fecha de nacimiento. En este caso, sólo se puede imprimir la fecha de nacimiento del hijo, ya que la del padre no es visible.
 2. La fecha de nacimiento del padre es visible en el bloque externo y, por lo tanto, se puede imprimir.

Nota: no puede haber variables con idéntico nombre en un mismo bloque. Ahora bien, como se muestra en este ejemplo, existe la opción de declarar variables con el mismo nombre en dos bloques distintos (bloques anidados). Los dos elementos representados por los identificadores son distintos; los cambios aplicados en uno no afectan al otro.

Realizar

Uso de Cualificadores en Bloques Anidados

```
BEGIN <>outer>>
DECLARE
  v_father_name VARCHAR2(20) := 'Patrick';
  v_date_of_birth DATE := '20-Apr-1972';
BEGIN
  DECLARE
    v_child_name VARCHAR2(20) := 'Mike';
    v_date_of_birth DATE := '12-Dec-2002';
  BEGIN
    DBMS_OUTPUT.PUT_LINE('Father''s Name: ' || v_father_name);
    DBMS_OUTPUT.PUT_LINE('Date of Birth: '
                         || outer.v_date_of_birth);
    DBMS_OUTPUT.PUT_LINE('Child''s Name: ' || v_child_name);
    DBMS_OUTPUT.PUT_LINE('Date of Birth: ' || v_date_of_birth);
  END;
END;
END outer;
```

ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Uso de Cualificadores en Bloques Anidados

Un *cualificador* es una etiqueta asignada a un bloque. Sirve para acceder a las variables cuyo ámbito es válido, pero que no son visibles.

Ejemplo

En el código de ejemplo:

- El bloque externo tiene la etiqueta `outer`
- En el bloque interno, se utiliza el cualificador `outer` para acceder a la variable `v_date_of_birth` que se declara en el bloque externo. Por lo tanto, tanto la fecha de nacimiento del padre como la del hijo se pueden imprimir desde el bloque interno.
- La salida del código en la diapositiva muestra la información correcta:

```
anonymous block completed
Father's Name: Patrick
Date of Birth: 20-APR-72
Child's Name: Mike
Date of Birth: 12-DEC-02
```

Nota: el etiquetado no se limita al bloque externo, se puede aplicar a cualquier bloque.

Desafío: Determinación del Ámbito de las Variables

```

BEGIN <<outer>>
DECLARE
    v_sal      NUMBER(7,2) := 60000;
    v_comm     NUMBER(7,2) := v_sal * 0.20;
    v_message  VARCHAR2(255) := ' eligible for commission';
BEGIN
    DECLARE
        v_sal      NUMBER(7,2) := 50000;
        v_comm     NUMBER(7,2) := 0;
        v_total_comp NUMBER(7,2) := v_sal + v_comm;
    BEGIN
        1-> v_message := 'CLERK not'||v_message;
        outer.v_comm := v_sal * 0.30;
    END;
    2-> v_message := 'SALESMAN'||v_message;
END;
END outer;
/

```

ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Desafío: Determinación del Ámbito de las Variables

Evalúe el bloque PL/SQL de la diapositiva. Determine cada uno de los siguientes valores de acuerdo con las reglas de los ámbitos:

1. Valor de `v_message` en la posición 1
2. Valor de `v_total_comp` en la posición 2
3. Valor de `v_comm` en la posición 1
4. Valor de `outer.v_comm` en la posición 1
5. Valor de `v_comm` en la posición 2
6. Valor de `v_message` en la posición 2

Respuestas: Determinación del Ámbito de las Variables

Las respuestas a las preguntas de ámbito son las siguientes:

1. Valor de `v_message` en la posición 1: **CLERK no se puede elegir para comisión**
2. Valor de `v_total_comp` en la posición 2: **Error. v_total_comp no está visible porque se ha definido en el bloque interno.**
3. Valor de `v_comm` en la posición 1: **0**
4. Valor de `outer.v_comm` en la posición 1: **12000**
5. Valor de `v_comm` en la posición 2: **15000**
6. Valor de `v_message` en la posición 2: **SALESMANCLERK no se puede elegir para comisión**

Agenda

- Escritura de sentencias ejecutables en los bloques PL/SQL
- Escritura de bloques anidados
- Uso de operadores y desarrollo de código legible

ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Operadores de PL/SQL

- Lógicos
 - Aritméticos
 - Concatenación
 - Paréntesis para controlar el orden de las operaciones
- Operador exponencial (**)

Iguales que en SQL

ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Operadores de PL/SQL

Las operaciones que integran las expresiones se realizan en un orden concreto dependiendo de su prioridad. En la siguiente tabla se muestra el orden por defecto de las operaciones de la máxima prioridad a la mínima:

Operador	Operación
**	Exponencial
+, -	Identidad, negación
*, /	Multiplicación, división
+, -,	Adición, resta, concatenación
=, <, >, <=, >=, <>, !=, ~=, ^=, IS NULL, LIKE, BETWEEN, IN	Comparación
NOT	Negación lógica
AND	Conjunción
OR	Inclusión

Operadores de PL/SQL: Ejemplos

- Incremente el contador de un bucle.

```
loop_count := loop_count + 1;
```

- Defina el valor de un indicador booleano.

```
good_sal := sal BETWEEN 50000 AND 150000;
```

- Valide si el número de empleado contiene algún valor.

```
valid := (empno IS NOT NULL);
```



Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Operadores de PL/SQL (continuación)

Cuando se trabaja con valores nulos, puede evitar que se produzcan algunos errores comunes si se recuerdan las siguientes reglas:

- Las comparaciones donde aparecen valores nulos siempre generan NULL.
- Si se aplica el operador lógico NOT a un valor nulo, se genera NULL.
- En sentencias de control condicional, si la condición genera NULL, la secuencia de sentencias asociada no se ejecuta.

Instrucciones de Programación

Para facilitar el mantenimiento del código:

- Documente el código con comentarios
- Desarrolle una convención de mayúsculas y minúsculas para el código
- Desarrolle reglas de nomenclatura para los identificadores y otros objetos
- Mejore la legibilidad mediante sangrados



Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Instrucciones de Programación

Siga las instrucciones de programación que aparecen en la diapositiva para producir código limpio y reducir el mantenimiento al desarrollar bloques PL/SQL.

Convenciones de Código

En la siguiente tabla se proporcionan instrucciones para escribir código en mayúsculas o minúsculas, lo cual ayuda a distinguir las palabras clave de los objetos con nombre.

Categoría	Convención	Ejemplos
Sentencias SQL	Mayúsculas	SELECT, INSERT
Palabras clave PL/SQL	Mayúsculas	DECLARE, BEGIN, IF
Tipos de dato	Mayúsculas	VARCHAR2, BOOLEAN
Identificadores y parámetros	Minúsculas	v_sal, emp_cursor, g_sal, p_empno
Tablas de base de datos	Minúsculas, plural	employees, departments
Columnas de base de datos	Minúsculas, plural	employee_id, department_id

Sangrado del Código

Para mejorar la claridad, sangre cada nivel del código.

```
BEGIN
  IF x=0 THEN
    y:=1;
  END IF;
END;
/
```

```
DECLARE
  deptno      NUMBER(4);
  location_id NUMBER(4);
BEGIN
  SELECT department_id,
         location_id
  INTO   deptno,
         location_id
  FROM   departments
  WHERE  department_name
         = 'Sales';
...
END;
/
```

ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Sangrado del Código

Para mejorar la claridad y la legibilidad, sangre cada nivel del código. Para mostrar la estructura, se pueden dividir las líneas mediante retornos de carro y sangrarlas con espacios o tabulaciones. Compare la facilidad de lectura de las siguientes sentencias IF:

```
IF x>y THEN max:=x;ELSE max:=y;END IF;
```

```
IF x > y THEN
  max := x;
ELSE
  max := y;
END IF;
```

Prueba

Puede utilizar la mayoría de funciones SQL de una fila como las de número, carácter, conversión y fecha en expresiones PL/SQL.

- a. Verdadero
- b. Falso



Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Respuesta: a

Funciones SQL en PL/SQL

SQL proporciona varias funciones predefinidas para su uso en sentencias SQL. La mayoría de estas funciones (como las funciones de número y carácter de una fila, las funciones de conversión de tipo de dato y las funciones de registro de fecha y hora) son válidas en las expresiones PL/SQL.

Las siguientes funciones no están disponibles en sentencias de procedimiento:

- DECODE
 - Funciones de grupo: AVG, MIN, MAX, COUNT, SUM, STDDEV y VARIANCE
- Las funciones de grupo se aplican a grupos de filas de las tablas y, por lo tanto, sólo están disponibles en las sentencias SQL de los bloques PL/SQL. Las funciones aquí mencionadas son sólo un subjuego, no una lista completa.

Resumen

En esta lección, debe haber aprendido lo siguiente:

- Identificar unidades léxicas en los bloques PL/SQL
- Utilizar funciones SQL incorporadas en PL/SQL
- Escribir bloques anidados para desglosar las funcionalidades con relaciones lógicas
- Decidir cuándo se deben realizar conversiones explícitas
- Cualificar variables de bloques anidados
- Utilizar secuencias en expresiones PL/SQL



Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Resumen

Como PL/SQL es una extensión de SQL, las reglas generales de sintaxis que se aplican a SQL son también aplicables a PL/SQL.

En la parte ejecutable de los bloques se pueden definir cuantos bloques anidados se deseen. Los bloques definidos dentro de otro bloque se denominan subbloques. Sólo se pueden anidar bloques en la parte ejecutable de los bloques. Ya que la sección de excepciones también forma parte de la sección ejecutable, puede contener bloques anidados. Asegúrese de que el ámbito y la visibilidad de las variables son correctos cuando existen bloques anidados. Evite el uso de los mismos identificadores en los bloques principal y secundario.

Casi todas las funciones disponibles en SQL también son válidas en las expresiones PL/SQL. Las funciones de conversión sirven para convertir los valores de un tipo de dato a otro. Los operadores de comparación permiten comparar una expresión con otra. El resultado es siempre TRUE, FALSE o NULL. Por lo general, los operadores de comparación se utilizan en sentencias de control condicional y en la cláusula WHERE de las sentencias de manipulación de datos SQL. Los operadores relacionales facilitan la comparación arbitraria de expresiones complejas.

Práctica 3: Visión General

En esta práctica se abordan los siguientes temas:

- Revisión de las reglas de ámbito y anidación
- Escritura y prueba de bloques PL/SQL



Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Práctica 3: Visión General

Los ejercicios 1 y 2 se responden sobre papel.

4

Interacción con Oracle Database Server: Sentencias SQL en Programas PL/SQL

ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Objetivos

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

- Determinar qué sentencias SQL se pueden incluir directamente en el bloque ejecutable PL/SQL
- Manipular datos con sentencias DML en PL/SQL
- Utilizar sentencias de control de transacciones en PL/SQL
- Hacer uso de la cláusula `INTO` para contener los valores devueltos por las sentencias SQL
- Diferenciar cursores implícitos de cursores explícitos
- Utilizar atributos de cursor SQL



Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Objetivos

En esta lección, aprenderá a embeber las sentencias SQL estándar `SELECT`, `INSERT`, `UPDATE`, `DELETE` y `MERGE` en bloques PL/SQL. Aprenderá a incluir sentencias del lenguaje de definición de datos (DDL) y de control de transacciones en PL/SQL. Se explicará la necesidad del uso de cursores y aprenderá a diferenciar los dos tipos de cursores. En esta lección también se presentan los diversos atributos de cursor SQL que se pueden utilizar con cursores implícitos.

Agenda

- Recuperación de datos con PL/SQL
- Manipulación de datos con PL/SQL
- Introducción a los cursosres SQL



Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Sentencias SQL en PL/SQL

- Recupere filas de las bases de datos con el comando SELECT.
- Realice cambios en las filas de las bases de datos con comandos DML.
- Controle las transacciones con los comandos COMMIT, ROLLBACK o SAVEPOINT.



Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Sentencias SQL en PL/SQL

En los bloques PL/SQL, las sentencias SQL se utilizan para recuperar y modificar los datos de las tablas de las bases de datos. PL/SQL soporta los comandos de lenguaje de manipulación de datos (DML) y de control de transacciones. Los comandos DML se utilizan para modificar los datos de las tablas de las bases de datos. No obstante, recuerde los siguientes puntos al utilizar sentencias DML y comandos de control de transacciones en los bloques PL/SQL:

- La palabra clave END señala el final del bloque PL/SQL, no el final de una transacción. De igual modo que un bloque puede abarcar varias transacciones, una transacción también puede abarcar varios bloques.
- PL/SQL no soporta directamente las sentencias de lenguaje de definición de datos (DDL), por ejemplo, CREATE TABLE, ALTER TABLE o DROP TABLE. PL/SQL soporta enlaces anteriores, que no se pueden producir si las aplicaciones deben crear objetos de base de datos en tiempo de ejecución transfiriendo valores. Las sentencias DDL no se pueden ejecutar directamente, ya que son sentencias SQL dinámicas. Las sentencias SQL dinámicas se crean como cadenas de caracteres en tiempo de ejecución y pueden contener marcadores de posición para los parámetros. Por lo tanto, puede utilizar SQL dinámico para ejecutar las sentencias DDL en PL/SQL. Los detalles sobre la forma de trabajar con SQL dinámico se describen en el curso titulado *Oracle Database: Desarrollo de Unidades de Programa en PL/SQL*.
- PL/SQL no soporta directamente sentencias de lenguaje de control de datos (DCL) como GRANT o REVOKE. Puede utilizar SQL dinámico para ejecutarlas.

Sentencias SELECT en PL/SQL

Recupere datos de las bases de datos con sentencias SELECT.

Sintaxis:

```
SELECT select_list
INTO {variable_name[, variable_name]...  
| record_name}
FROM table
[WHERE condition];
```



Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Sentencias SELECT en PL/SQL

La sentencia SELECT permite recuperar datos de las bases de datos.

<i>select_list</i>	Lista de al menos una columna; puede incluir expresiones SQL, funciones de fila o funciones de grupo
<i>variable_name</i>	Variable escalar que contiene el valor recuperado
<i>record_name</i>	Registro PL/SQL que contiene los valores recuperados
<i>table</i>	Especifica el nombre de la tabla de la base de datos
<i>condition</i>	Se compone de nombres de columna, expresiones, constantes y operadores de comparación, incluidas variables y constantes PL/SQL

Instrucciones para la Recuperación de Datos en PL/SQL

- Termine todas las sentencias SQL con punto y coma (;).
- Todos los valores recuperados se deben almacenar en una variable mediante la cláusula INTO.
- La cláusula WHERE es opcional y se puede utilizar para especificar variables de entrada, constantes, literales y expresiones PL/SQL. Sin embargo, cuando se utiliza la cláusula INTO, sólo se recupera una fila y es obligatorio el uso de la cláusula WHERE.

Sentencias SELECT en PL/SQL (continuación)

- Especifique el mismo número de variables en la cláusula INTO que columnas de base de datos en la cláusula SELECT. Asegúrese de que se corresponden en posición y de que sus tipos de dato son compatibles.
- Utilice funciones de grupo como SUM en las sentencias SQL, ya que estas funciones se aplican a grupos de filas de la tabla pertinente.

Sentencias SELECT en PL/SQL

- La cláusula INTO es obligatoria.
- Las consultas deben devolver una única fila.

```
DECLARE
  v_fname VARCHAR2(25);
BEGIN
  SELECT first_name INTO v_fname
  FROM employees WHERE employee_id=200;
  DBMS_OUTPUT.PUT_LINE(' First Name is : '||v_fname);
END ;
/
```

anonymous block completed
First Name is : Jennifer

ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Sentencias SELECT en PL/SQL (continuación)

Cláusula INTO

La cláusula INTO es obligatoria y se produce entre las cláusulas SELECT y FROM. Se utiliza para especificar los nombres de las variables que contienen los valores que devuelve SQL de la cláusula SELECT. Debe especificar una variable para cada elemento seleccionado y el orden de las variables debe corresponder a los elementos seleccionados.

La cláusula INTO se utiliza para llenar variables PL/SQL o del host.

Las Consultas Deben Devolver una Única Fila

Las sentencias SELECT de un bloque PL/SQL pertenecen a la clasificación ANSI de SQL embebido, a la que se aplica la siguiente regla: las consultas deben devolver una única fila. Si la consulta devuelve más filas o no devuelve ninguna, se genera un error.

PL/SQL se ocupa de estos errores emitiendo excepciones estándar, comprendida en la sección de excepciones del bloque con las excepciones NO_DATA_FOUND y TOO_MANY_ROWS. Incluya una condición WHERE en la sentencia SQL para que la sentencia devuelva una sola fila. Aprenderá sobre el manejo de excepciones en la lección titulada “Manejo de Excepciones”.

Nota: en todos los casos en los que se utiliza DBMS_OUTPUT.PUT_LINE en los códigos de ejemplo, la sentencia SET SERVEROUTPUT ON precede al bloque.

Sentencias SELECT en PL/SQL (continuación)

Recuperación de Varias Filas de una Tabla y Operaciones con los Datos

Las sentencias SELECT que incluyen una cláusula INTO sólo pueden recuperar una única fila cada vez. Si necesita recuperar varias filas y realizar operaciones con los datos, debe utilizar cursosres explícitos. Más adelante en esta lección, se explicarán los cursosres y aprenderá sobre los cursosres explícitos en la sección “Uso de Cursosres Explícitos”.

Recuperación de Datos en PL/SQL: Ejemplo

Recupere hire_date y salary para el empleado especificado.

```

DECLARE
    v_emp_hiredate    employees.hire_date%TYPE;
    v_emp_salary       employees.salary%TYPE;
BEGIN
    SELECT    hire_date, salary
    INTO      v_emp_hiredate, v_emp_salary
    FROM     employees
    WHERE    employee_id = 100;
    DBMS_OUTPUT.PUT_LINE ('Hire date is :''' || v_emp_hiredate);
    DBMS_OUTPUT.PUT_LINE ('Salary is :''' || v_emp_salary);
END;
/

```

```

anonymous block completed
Hire date is : 17-JUN-87
Salary  is : 24000

```

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Recuperación de Datos en PL/SQL

En el ejemplo de la diapositiva, las variables v_emp_hiredate y v_emp_salary se declaran en la sección de declaraciones del bloque PL/SQL. En la sección ejecutable, los valores de las columnas hire_date y salary correspondientes al empleado con employee_id 100 se recuperan de la tabla employees. A continuación, se almacenan en las variables emp_hiredate y emp_salary, respectivamente. Observe cómo la cláusula INTO, junto con la sentencia SELECT, recupera los valores de columna de la base de datos y los almacena en las variables PL/SQL.

Nota: la sentencia SELECT recupera hire_date y, a continuación, salary. Las variables de la cláusula INTO deben estar, por tanto, en el mismo orden. Por ejemplo, si intercambia v_emp_hiredate con v_emp_salary en la sentencia de la diapositiva, la sentencia dará un error.

Recuperación de Datos en PL/SQL

Devuelva la suma de los salarios de todos los empleados del departamento especificado.

Ejemplo:

```
DECLARE
    v_sum_sal    NUMBER(10,2);
    v_deptno     NUMBER NOT NULL := 60;
BEGIN
    SELECT SUM(salary) -- group function
    INTO v_sum_sal FROM employees
    WHERE department_id = v_deptno;
    DBMS_OUTPUT.PUT_LINE ('The sum of salary is ' || v_sum_sal);
END;
```

```
anonymous block completed
The sum of salary is 28800
```

ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Recuperación de Datos en PL/SQL (continuación)

En el ejemplo de la diapositiva, las variables `v_sum_sal` y `v_deptno` se declaran en la sección de declaraciones del bloque PL/SQL. En la sección ejecutable, el salario total de los empleados del departamento con `department_id` 60 se calcula mediante la función de agregación SQL `SUM`. El salario total calculado se asigna a la variable `v_sum_sal`.

Nota: no se pueden utilizar funciones de grupo en la sintaxis PL/SQL. Se deben utilizar en las sentencias SQL incluidas en bloques PL/SQL, tal como se muestra en el ejemplo de la diapositiva.

Por ejemplo, *no puede* utilizar funciones de grupo con la siguiente sintaxis:

```
V_sum_sal := SUM(employees.salary);
```

Nomenclatura Ambigua

```

DECLARE
    hire_date      employees.hire_date%TYPE;
    sysdate        hire_date%TYPE;
    employee_id   employees.employee_id%TYPE := 176;
BEGIN
    SELECT      hire_date, sysdate
    INTO        hire_date, sysdate
    FROM       employees
    WHERE      employee_id = employee_id;
END;
/

```

```

Error report:
ORA-01422: exact fetch returns more than requested number of rows
ORA-06512: at line 6
01422. 00000 - "exact fetch returns more than requested number of rows"
*Cause:    The number specified in exact fetch is less than the rows returned.
*Action:   Rewrite the query or change number of rows requested

```

ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Nomenclatura Ambigua

En las sentencias SQL que pueden resultar ambiguas, los nombres de las columnas de base de datos tienen prioridad sobre los nombres de las variables locales.

El ejemplo que se muestra en la diapositiva se define de la siguiente forma: recupere la fecha de contratación y la fecha de hoy de la tabla `employees` para el valor `employee_id` 176. Este ejemplo produce una excepción no tratada de tiempo de ejecución porque, en la cláusula `WHERE`, los nombres de las variables PL/SQL son iguales que los nombres de las columnas de base de datos de la tabla `employees`.

La siguiente sentencia `DELETE` elimina todos los empleados de la tabla `employees` cuyo apellido es no nulo (no sólo “King”) porque Oracle Server presupone que ambas incidencias de `last_name` en la cláusula `WHERE` hacen referencia a la columna de base de datos:

```

DECLARE
    last_name VARCHAR2(25) := 'King';
BEGIN
    DELETE FROM employees WHERE last_name = last_name;
    .

```

Reglas de Nomenclatura

- Utilice una regla de nomenclatura para salvar la ambigüedad en la cláusula WHERE.
- Evite el uso de nombres de columna de base de datos como identificadores.
- Se pueden producir errores de sintaxis porque PL/SQL comprueba primero en la base de datos si hay alguna columna en la tabla.
- Los nombres de variables locales y parámetros formales tienen prioridad sobre los nombres de *tablas* de base de datos.
- Los nombres de *columnas* de tablas de base de datos tienen prioridad sobre los nombres de variables locales.

ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Reglas de Nomenclatura

Evite la ambigüedad de la cláusula WHERE siguiendo una regla de nomenclatura que distinga los nombres de columna de base de datos de los nombres de variables PL/SQL.

- Las columnas de base de datos y los identificadores deben tener nombres distintos.
- Se pueden producir errores de sintaxis porque PL/SQL comprueba primero en la base de datos si hay alguna columna en la tabla.

Nota: no hay ambigüedad posible en la cláusula SELECT, porque todos los identificadores que aparecen en ella deben ser nombres de columnas de base de datos SELECT. No hay ambigüedad posible en la cláusula INTO, porque los identificadores que aparecen en ella deben ser variables PL/SQL. Sólo hay posibilidad de confusión en la cláusula WHERE.

Agenda

- Recuperación de datos con PL/SQL
- Manipulación de datos con PL/SQL
- Introducción a los cursosres SQL

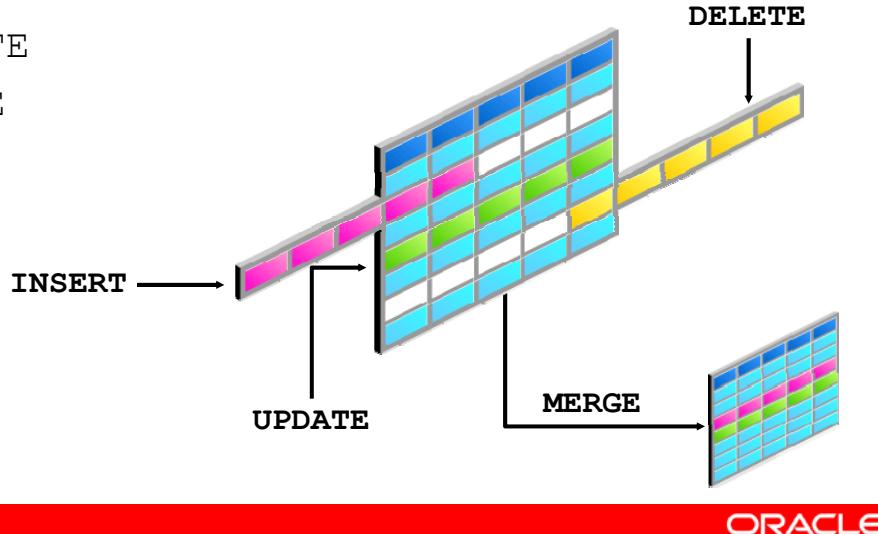


Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Uso de PL/SQL para Manipular Datos

Realice cambios en las tablas de las bases de datos con comandos DML:

- INSERT
- UPDATE
- DELETE
- MERGE



ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Uso de PL/SQL para Manipular Datos

Los datos de las bases de datos se manipulan con comandos DML. Puede emitir comandos DML como INSERT, UPDATE, DELETE y MERGE sin ninguna restricción en PL/SQL. Los bloqueos de fila (y los bloqueos de tabla) se liberan incluyendo las sentencias COMMIT o ROLLBACK en el código PL/SQL.

- La sentencia INSERT agrega filas nuevas a la tabla.
- La sentencia UPDATE modifica filas existentes de la tabla.
- La sentencia DELETE elimina filas de la tabla.
- La sentencia MERGE selecciona filas de una tabla para actualizar o insertar en otra tabla. La decisión sobre si se actualiza o se inserta en la tabla de destino se basa en una condición de la cláusula ON.

Nota: MERGE es una sentencia determinista, es decir, no se puede actualizar la misma fila de la tabla de destino varias veces en la misma sentencia MERGE. Debe tener privilegios de objeto INSERT y UPDATE en la tabla de destino y el privilegio SELECT en la tabla de origen.

Inserción de Datos: Ejemplo

Agregue nueva información de empleado a la tabla EMPLOYEES.

```
BEGIN
  INSERT INTO employees
  (employee_id, first_name, last_name, email,
   hire_date, job_id, salary)
  VALUES(employees_seq.NEXTVAL, 'Ruth', 'Cores',
  'RCORES', CURRENT_DATE, 'AD_ASST', 4000);
END;
/
```



Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Inserción de Datos

En el ejemplo de la diapositiva, se utiliza una sentencia `INSERT` en un bloque PL/SQL para insertar un registro en la tabla `employees`. Al utilizar el comando `INSERT` en un bloque PL/SQL, puede:

- Utilizar funciones SQL, como `USER` y `CURRENT_DATE`
- Generar valores de clave primaria con secuencias de base de datos existentes
- Derivar valores en el bloque PL/SQL

Nota: los datos de la tabla `employees` tienen que permanecer sin cambios. Aunque la tabla `employees` no es de sólo lectura, no se puede insertar, actualizar ni suprimir en esta tabla para garantizar una salida consistente, como se muestra en el código de ejemplo `code_04_15_s.sql`.

Actualización de Datos: Ejemplo

Aumente el salario de todos los empleados que sean oficinistas en el departamento de stock.

```
DECLARE
    sal_increase      employees.salary%TYPE := 800;
BEGIN
    UPDATE      employees
    SET          salary = salary + sal_increase
    WHERE        job_id = 'ST_CLERK';
END;
/
```

anonymous block completed	
FIRST_NAME	SALARY
Julia	4000
Irene	3500
James	3200
Steven	3000

...	
Curtis	3900
Randall	3400
Peter	3300

20 rows selected

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Actualización de Datos

Puede que exista ambigüedad en la cláusula SET de la sentencia UPDATE ya que, aunque el identificador situado a la izquierda del operador de asignación es siempre una columna de base de datos, el identificador de la derecha puede ser tanto una columna de base de datos como una variable PL/SQL. Recuerde que si los nombres de columna y los nombres de identificador son idénticos en la cláusula WHERE, Oracle Server buscará primero el nombre en la base de datos.

Recuerde que la cláusula WHERE se utiliza para determinar qué filas se verán afectadas. Si no se modifica ninguna fila, no se produce ningún error (a diferencia que con la sentencia SELECT en PL/SQL).

Nota: las asignaciones de variables PL/SQL siempre utilizan :=, mientras que las asignaciones de columnas SQL siempre utilizan =.

Supresión de Datos: Ejemplo

Suprima filas que pertenecen al departamento 10 de la tabla employees.

```
DECLARE
    deptno    employees.department_id%TYPE := 10;
BEGIN
    DELETE FROM employees
    WHERE department_id = deptno;
END ;
/
```



Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Supresión de Datos

La sentencia DELETE elimina las filas no deseadas de la tabla. Si no se utiliza la cláusula WHERE, se pueden eliminar todas las filas de la tabla oportuna, si no existe ninguna restricción de integridad.

Fusión de Filas

Inserte o actualice filas en la tabla `copy_emp` para que coincida con la tabla `employees`.

```
BEGIN
MERGE INTO copy_emp c
  USING employees e
    ON (e.employee_id = c.empno)
  WHEN MATCHED THEN
    UPDATE SET
      c.first_name      = e.first_name,
      c.last_name       = e.last_name,
      c.email           = e.email,
      . . .
  WHEN NOT MATCHED THEN
    INSERT VALUES(e.employee_id, e.first_name, e.last_name,
      . . ., e.department_id);
END;
/
```

ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Fusión de Filas

La sentencia MERGE inserta o actualiza filas en una tabla con datos de otra tabla. Cada fila se inserta o se actualiza en la tabla de destino de acuerdo con una condición de unión igualitaria.

En el ejemplo mostrado, se compara la columna `empno` de la tabla `copy_emp` con la columna `employee_id` de la tabla `employees`. Si se encuentra una coincidencia, se actualiza la fila para que coincida con la fila de la tabla `employees`. Si no se encuentra la fila, se inserta en la tabla `copy_emp`.

En la siguiente página se muestra el ejemplo completo de uso de MERGE en un bloque PL/SQL.

Fusión de Filas (continuación)

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

Agenda

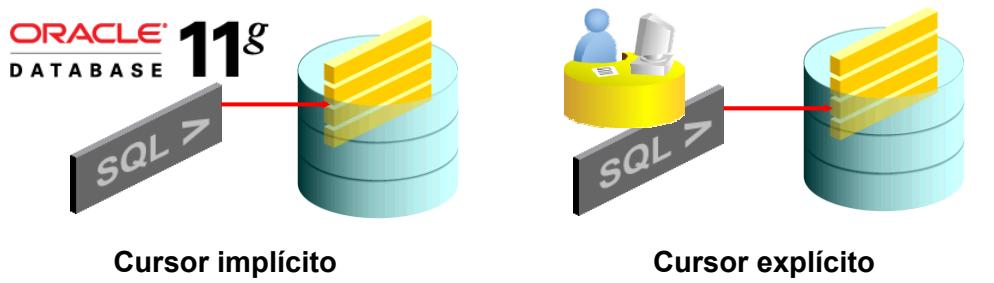
- Recuperación de datos con PL/SQL
- Manipulación de datos con PL/SQL
- Introducción a los cursosres SQL



Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Cursor SQL

- Un cursor es un puntero al área de memoria privada asignada por Oracle Server. Se utiliza para manejar el juego de resultados de una sentencia SELECT.
- Hay dos tipos de cursosres: implícitos y explícitos.
 - **Implícitos:** creados y gestionados de forma interna por Oracle Server para procesar sentencias SQL
 - **Explícitos:** declarados de forma explícita por el programador



Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Cursor SQL

Ya ha aprendido a incluir sentencias SQL que devuelvan una sola fila en los bloques PL/SQL. Los datos recuperados por las sentencias SQL se deben contener en variables mediante la cláusula INTO.

¿Dónde Procesa Oracle Server las Sentencias SQL?

Oracle Server asigna un área de memoria privada, llamada *área de contexto* para procesar las sentencias SQL. En esta área se analizan y procesan las sentencias SQL. Tanto la información necesaria para el procesamiento como la información recuperada tras éste se almacenan en esta área. No tiene ningún control sobre esta área porque Oracle Server la gestiona de forma interna.

Los cursosres son punteros que señalan el área de contexto. No obstante, estos cursosres son implícitos, y también los gestiona de forma automática Oracle Server. Cuando el bloque ejecutable emite una sentencia SQL, PL/SQL crea un cursor implícito.

Tipos de Cursosres

Hay dos tipos de cursosres:

- **Implícitos:** Oracle Server crea y gestiona los *cursosres implícitos*. No se puede acceder a ellos. Oracle Server crea estos cursosres cuando tiene que ejecutar una sentencia SQL.

Cursos SQL (continuación)

Tipos de Cursos (continuación)

- **Explícitos:** el programador puede recuperar varias filas de una tabla de base de datos, tener un puntero por cada fila recuperada y trabajar con las filas al mismo tiempo. En estos casos, existe la opción de declarar cursos de manera explícita para que satisfagan las necesidades de negocio. Los cursos declarados por los programadores se denominan *cursos explícitos*. Estos cursos se declaran en la sección de declaraciones de los bloques PL/SQL.

Atributos de Cursor SQL para Cursos Implícitos

Con los atributos de cursor SQL, puede probar el resultado de las sentencias SQL.

SQL%FOUND	Atributo booleano que se evalúa como TRUE si la sentencia SQL más reciente afecta, como mínimo, a una fila
SQL%NOTFOUND	Atributo booleano que se evalúa como TRUE si la sentencia SQL más reciente no afecta a ninguna fila
SQL%ROWCOUNT	Valor entero que representa el número de filas afectadas por la sentencia SQL más reciente.



Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Atributos de Cursor SQL para Cursos Implícitos

Los atributos de cursor SQL le permiten evaluar lo que ocurrió la última vez que se utilizó un cursor implícito. Se utilizan en sentencias PL/SQL, pero no en sentencias SQL.

Puede probar los atributos SQL%ROWCOUNT, SQL%FOUND y SQL%NOTFOUND en la sección ejecutable de un bloque para recopilar información después de ejecutarse el comando DML adecuado. PL/SQL no devuelve ningún error si la sentencia DML no afecta a ninguna de las filas de la tabla subyacente. Sin embargo, si una sentencia SELECT no recupera ninguna fila, PL/SQL sí devuelve una excepción.

Observe que el nombre de los atributos lleva el prefijo SQL. Estos atributos de cursor se utilizan con cursos implícitos que PL/SQL crea de forma automática y cuyos nombres desconoce. Por ello, se utiliza SQL en lugar del nombre de los cursos.

El atributo SQL%NOTFOUND es el contrario de SQL%FOUND. Este atributo puede servir como condición de salida de un bucle. Resulta útil en las sentencias UPDATE y DELETE cuando no se modifica ninguna fila porque no se devuelven excepciones en estos casos.

Aprenderá sobre los atributos de cursor explícito en la lección titulada “Uso de Cursos Explícitos”.

Atributos de Cursor SQL para Cursos Implícitos

Suprime filas que tienen el identificador de empleado especificado de la tabla employees. Imprima el número de filas suprimidas.

Ejemplo:

```
DECLARE
    v_rows_deleted VARCHAR2(30)
    v_empno employees.employee_id%TYPE := 176;
BEGIN
    DELETE FROM employees
    WHERE employee_id = v_empno;
    v_rows_deleted := (SQL%ROWCOUNT ||
                       ' row deleted.');
    DBMS_OUTPUT.PUT_LINE (v_rows_deleted);
END;
```

ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Atributos de Cursor SQL para Cursos Implícitos (continuación)

En el ejemplo de la diapositiva se suprime una fila con employee_id 176 de la tabla employees. El atributo SQL%ROWCOUNT permite imprimir el número de filas suprimidas.

Prueba

Si se utiliza la sentencia SELECT en PL/SQL, se necesita la cláusula INTO y las consultas pueden devolver una o varias filas.

- a. Verdadero
- b. Falso



Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Respuesta: b

Cláusula INTO

La cláusula INTO es obligatoria y se produce entre las cláusulas SELECT y FROM. Se utiliza para especificar los nombres de las variables que contienen los valores que devuelve SQL de la cláusula SELECT. Debe especificar una variable para cada elemento seleccionado y el orden de las variables debe corresponder a los elementos seleccionados.

La cláusula INTO se utiliza para llenar variables PL/SQL o del host.

Las Consultas Deben Devolver una Única Fila

Las sentencias SELECT de un bloque PL/SQL pertenecen a la clasificación ANSI de SQL embebido, a la que se aplica la siguiente regla: las consultas deben devolver una única fila. Si la consulta devuelve más filas o no devuelve ninguna, se genera un error.

PL/SQL se ocupa de estos errores emitiendo excepciones estándar, comprendida en la sección de excepciones del bloque con las excepciones NO_DATA_FOUND y TOO_MANY_ROWS. Incluya una condición WHERE en la sentencia SQL para que la sentencia devuelva una sola fila. Aprenderá sobre el manejo de excepciones posteriormente en el curso.

Resumen

En esta lección, debe haber aprendido lo siguiente:

- Embeber sentencias DML, sentencias de control de transacciones y sentencias DDL en PL/SQL
- Utilizar la cláusula `INTO`, que es obligatoria en todas las sentencias `SELECT` en PL/SQL
- Diferenciar cursos implícitos de cursos explícitos
- Utilizar atributos de cursor SQL para determinar la salida de las sentencias SQL



Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Resumen

Los comandos DML y las sentencias de control de transacciones se pueden utilizar en los programas PL/SQL sin ningún tipo de restricción. Los comandos DDL, en cambio, no se pueden utilizar directamente.

Las sentencias `SELECT` de los bloques PL/SQL sólo pueden devolver una fila. Es obligatorio utilizar la cláusula `INTO` para contener los valores recuperados por las sentencias `SELECT`.

Los cursos son punteros que señalan el área de memoria. Hay dos tipos de cursos. Los cursos implícitos los crea y gestiona de forma interna Oracle Server para ejecutar las sentencias SQL. Con estos cursos se pueden utilizar atributos de cursor SQL para determinar la salida de las sentencias SQL. Los cursos explícitos los declaran los programadores.

Práctica 4: Visión General

En esta práctica se abordan los siguientes temas:

- Selección de datos de tablas
- Inserción de datos en tablas
- Actualización de datos de tablas
- Supresión de registros de tablas



Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Escritura de las Estructuras de Control



ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Objetivos

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

- Identificar los usos y tipos de las estructuras de control
- Crear una sentencia IF
- Utilizar sentencias y expresiones CASE
- Crear e identificar sentencias de bucle
- Utilizar las instrucciones mientras se usan las estructuras de control condicionales



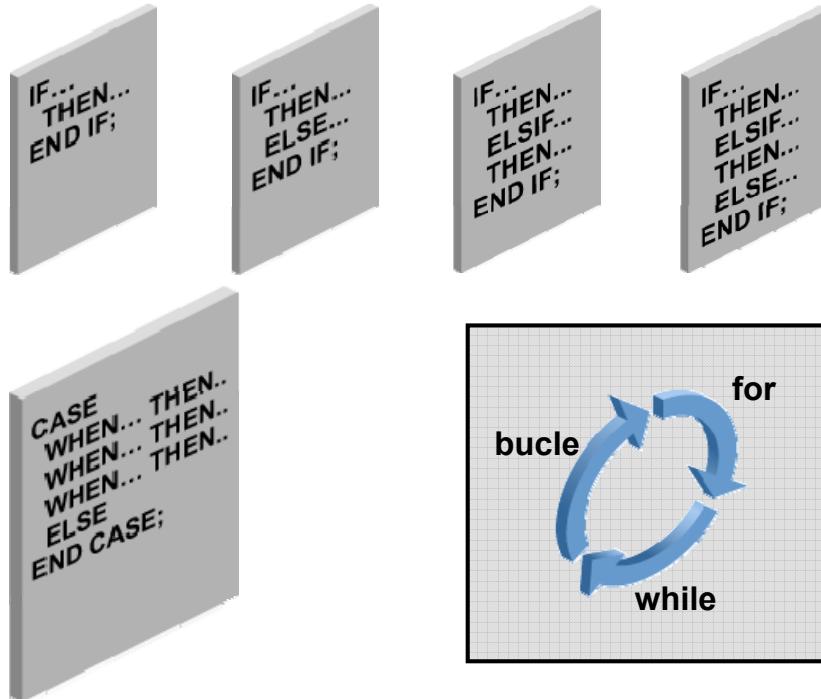
Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Objetivos

Ha aprendido a escribir bloques PL/SQL con secciones ejecutables y de declaraciones. Asimismo, ha aprendido a incluir expresiones y sentencias SQL en el bloque ejecutable.

En esta lección, aprenderá a utilizar estructuras de control, como sentencias IF, expresiones CASE y estructuras LOOP en un bloque PL/SQL.

Control del Flujo de Ejecución



ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Control del Flujo de Ejecución

Puede cambiar el flujo lógico de sentencias en el bloque PL/SQL con varias estructuras de control. Esta lección trata cuatro tipos de estructuras de control PL/SQL: construcciones condicionales con la sentencia IF, expresiones CASE, estructuras de control LOOP y la sentencia CONTINUE.

Agenda

- Uso de sentencias IF
- Uso de sentencias y expresiones CASE
- Creación e identificación de sentencias de bucle

ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Sentencia IF

Sintaxis:

```
IF condition THEN
    statements;
[ELSIF condition THEN
    statements;]
[ELSE
    statements;]
END IF;
```

ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Sentencia IF

La estructura de la sentencia PL/SQL IF es similar a la estructura de sentencias IF en otros lenguajes de procedimiento. Permite a PL/SQL realizar acciones de forma selectiva según las condiciones.

En la sintaxis:

- | | |
|-------------------|--|
| <i>condition</i> | Es una expresión o variable booleana que devuelve TRUE, FALSE o NULL. |
| THEN | Introduce una cláusula que asocia la expresión booleana a la secuencia de las sentencias que la siguen. |
| <i>statements</i> | Puede ser una o más sentencias PL/SQL o SQL. (Pueden incluir más sentencias IF con varias sentencias IF, ELSE y ELSIF anidadas.) Las sentencias de la cláusula THEN sólo se ejecutan si la condición de la cláusula IF asociada se evalúa como TRUE. |

Sentencia IF (continuación)

En la sintaxis:

- ELSIF Es una palabra clave que introduce una expresión booleana. (Si la primera condición genera FALSE o NULL, la palabra clave ELSIF introducirá condiciones adicionales).
- ELSE Introduce la cláusula por defecto que se ejecuta sólo si ninguno de los predicados anteriores (introducidos por IF y ELSIF) es TRUE. Las pruebas se ejecutan en secuencia para que un predicado posterior que debe ser verdadero sea vaciado con anterioridad por un predicado anterior que es verdadero.
- END IF Marca el final de una sentencia IF.

Nota: ELSIF y ELSE son opcionales en una sentencia IF. Puede tener cualquier número de palabras clave ELSIF, pero sólo una palabra clave ELSE en la sentencia IF. END IF marca el final de una sentencia IF y debe terminar en un punto y coma.

Sentencia IF Simple

REALIZAR

```
DECLARE
    v_myage  number:=31;
BEGIN
    IF v_myage  < 11
    THEN
        DBMS_OUTPUT.PUT_LINE(' I am a child ');
    END IF;
END ;
/
```

anonymous block completed

ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Sentencia IF Simple

Ejemplo de Sentencia IF Simple

La diapositiva muestra un ejemplo de una sentencia IF simple con la cláusula THEN.

- La variable v_myage se inicializa en 31.
- La condición de la sentencia IF devuelve FALSE porque v_myage no es menor que 11.
- Por lo tanto, el control no alcanzará nunca la cláusula THEN.

Adición de Expresiones Condicionales

Una sentencia IF puede tener varias expresiones condicionales relacionadas con operadores lógicos, como AND, OR y NOT.

Por ejemplo:

```
IF (myfirstname='Christopher' AND v_myage <11)
...
```

La condición utiliza el operador AND y, por lo tanto, se evalúa en TRUE sólo si ambas condiciones se evalúan como TRUE. No existe ninguna limitación en el número de expresiones condicionales. No obstante, estas sentencias se deben relacionar con operadores lógicos adecuados.

Sentencia IF THEN ELSE

REALIZAR

```
DECLARE
    v_myage  number:=31;
BEGIN
    IF v_myage  < 11
    THEN
        DBMS_OUTPUT.PUT_LINE(' I am a child ');
    ELSE
        DBMS_OUTPUT.PUT_LINE(' I am not a child ');
    END IF;
END ;
/
```

anonymous block completed
I am not a child

ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Sentencia IF THEN ELSE

Se ha agregado una cláusula ELSE al código en la diapositiva anterior. La condición no ha cambiado y, por lo tanto, se sigue evaluando en FALSE. Recuerde que las sentencias de la cláusula THEN sólo se ejecutan si la condición devuelve TRUE. En este caso, la condición devuelve FALSE y, por lo tanto, el control se mueve a la sentencia ELSE.

La salida del bloque se muestra debajo del código.

Cláusula IF ELSIF ELSE

REALIZAR

```

DECLARE
    v_myage number:=31;
BEGIN
    IF v_myage < 11 THEN
        DBMS_OUTPUT.PUT_LINE(' I am a child ');
    ELSIF v_myage < 20 THEN
        DBMS_OUTPUT.PUT_LINE(' I am young ');
    ELSIF v_myage < 30 THEN
        DBMS_OUTPUT.PUT_LINE(' I am in my twenties');
    ELSIF v_myage < 40 THEN
        DBMS_OUTPUT.PUT_LINE(' I am in my thirties');
    ELSE
        DBMS_OUTPUT.PUT_LINE(' I am always young ');
    END IF;
END;
/

```

anonymous block completed
I am in my thirties

ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Cláusula IF ELSIF ELSE

La cláusula IF puede contener varias cláusulas ELSIF y una cláusula ELSE. En el ejemplo se muestran las siguientes características de estas cláusulas:

- Las cláusulas ELSIF pueden tener condiciones, a diferencia de la cláusula ELSE.
- La condición de ELSIF debe estar seguida de la cláusula THEN, que se ejecuta si la condición de ELSIF devuelve TRUE.
- Cuando tiene varias cláusulas ELSIF, si la primera condición es FALSE o NULL, el control pasará a la siguiente cláusula ELSIF.
- Las condiciones se evalúan una a una desde el principio.
- Si todas las condiciones son FALSE o NULL, se ejecutarán las sentencias de la cláusula ELSE.
- La cláusula ELSE final es opcional.

En el ejemplo, la salida del bloque se muestra debajo del código.

Valor NULL en Sentencia IF

ANALIZA Y EXPLICA QUE OCURRE

```

DECLARE
    v_myage  number;
BEGIN
    IF v_myage  < 11 THEN
        DBMS_OUTPUT.PUT_LINE(' I am a child ');
    ELSE
        DBMS_OUTPUT.PUT_LINE(' I am not a child ');
    END IF;
END ;
/

```

anonymous block completed
I am not a child

ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Valor NULL en Sentencia IF

En el ejemplo de la diapositiva, la variable `v_myage` se declara, pero no se inicializa. La condición de la sentencia `IF` devuelve `NULL` en lugar de `TRUE` o `FALSE`. En este caso, el control pasa a la sentencia `ELSE`.

Instrucciones

- Puede realizar acciones de forma selectiva en función de las condiciones que se vayan a cumplir.
- Cuando escriba el código, recuerde la ortografía de las palabras clave:
 - `ELSIF` es una palabra.
 - `END IF` son dos palabras.
- Si la condición booleana de control es `TRUE`, se ejecutará la secuencia asociada de las sentencias. Si la condición es `FALSE` o `NULL`, dicha secuencia se pasará por alto. Se permite cualquier número de cláusulas `ELSIF`.
- Sangre las sentencias ejecutadas de forma condicional para su claridad.

Agenda

- Uso de sentencias IF
- Uso de sentencias y expresiones CASE
- Creación e identificación de sentencias de bucle

ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Expresiones CASE

- Una expresión CASE selecciona un resultado y lo devuelve.
- Para seleccionar el resultado, CASE utiliza expresiones. El valor devuelto por estas expresiones se utiliza para seleccionar una de varias alternativas.

```
CASE selector
  WHEN expression1 THEN result1
  WHEN expression2 THEN result2
  ...
  WHEN expressionN THEN resultN
  [ELSE resultN+1]
END ;
```



Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Expresiones CASE

Una expresión CASE devuelve un resultado basado en una o más alternativas. Para devolver el resultado, la expresión CASE utiliza un *selector*, que es una expresión cuyo valor se utiliza para devolver una de varias alternativas. El selector va seguido de una o más cláusulas WHEN, que se comprueban de forma secuencial. El valor del selector determina el resultado que se devuelve. Si el valor del selector es igual al valor de la expresión de una cláusula WHEN, se ejecutará dicha cláusula y se devolverá el resultado.

PL/SQL también proporciona una expresión CASE de búsqueda, que tiene el siguiente formato:

```
CASE
  WHEN search_condition1 THEN result1
  WHEN search_condition2 THEN result2
  ...
  WHEN search_conditionN THEN resultN
  [ELSE resultN+1]
END ;
```

Una expresión CASE de búsqueda no tiene selector. Además, las cláusulas WHEN de CASE contienen condiciones de búsqueda que generan un valor booleano y no incluyen expresiones que pueden generar valores de cualquier tipo.

Expresiones CASE: Ejemplo

Realizar

```
SET VERIFY OFF Omitir esta línea
DECLARE
    v_grade  CHAR(1) := UPPER('&grade');
    v_appraisal VARCHAR2(20);
BEGIN
    v_appraisal := CASE v_grade
        WHEN 'A' THEN 'Excellent'
        WHEN 'B' THEN 'Very Good'
        WHEN 'C' THEN 'Good'
        ELSE 'No such grade'
    END;
    DBMS_OUTPUT.PUT_LINE ('Grade: '|| v_grade || '
                           Appraisal ' || v_appraisal);
END;
/
```

ORACLE

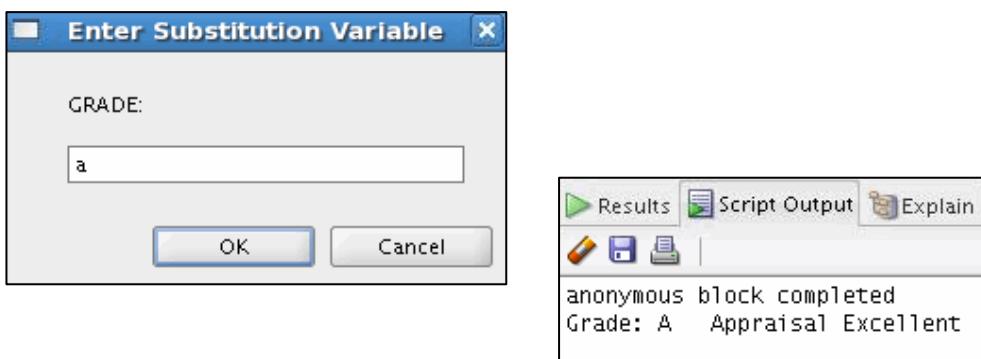
Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Expresiones CASE: Ejemplo

En el ejemplo de la diapositiva, la expresión CASE utiliza el valor de la variable v_grade como expresión. Este valor se acepta desde el usuario con una variable de sustitución. Según el valor que introduzca el usuario, la expresión CASE devolverá el valor de la variable v_appraisal en función del valor de v_grade.

Resultado

Al introducir a o A para v_grade, como se muestra en la ventana Substitution Variable, la salida del ejemplo es la siguiente:



Expresiones CASE de Búsqueda

```

DECLARE
    v_grade  CHAR(1) := UPPER('&grade');
    v_appraisal VARCHAR2(20);
BEGIN
    v_appraisal := CASE
        WHEN v_grade = 'A' THEN 'Excellent'
        WHEN v_grade IN ('B', 'C') THEN 'Good'
        ELSE 'No such grade'
    END;
    DBMS_OUTPUT.PUT_LINE ('Grade: ' || v_grade || '
                           Appraisal ' || v_appraisal);
END;
/

```

ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

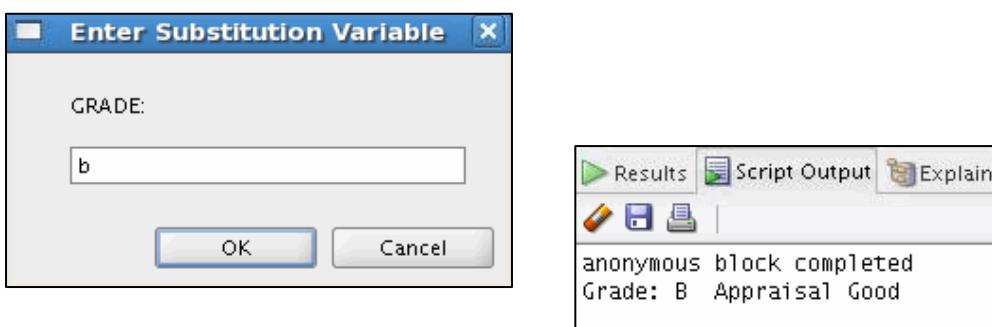
Expresiones CASE de Búsqueda

En el ejemplo anterior tiene una expresión de prueba única que es la variable `v_grade`. La cláusula `WHEN` comparaba un valor con esta expresión de prueba.

En las sentencias CASE de búsqueda, no hay una expresión de prueba. En su lugar, la cláusula `WHEN` contiene una expresión que da como resultado un valor booleano. Se vuelve a escribir el mismo ejemplo en esta diapositiva para mostrar las sentencias CASE de búsqueda.

Resultado

La salida del ejemplo es la siguiente si introduce b o B para `v_grade`:



Sentencia CASE

```

DECLARE
    v_deptid NUMBER;
    v_deptname VARCHAR2(20);
    v_emps NUMBER;
    v_mngid NUMBER:= 108;
BEGIN
    CASE v_mngid
        WHEN 108 THEN
            SELECT department_id, department_name
            INTO v_deptid, v_deptname FROM departments
            WHERE manager_id=108;
            SELECT count(*) INTO v_emps FROM employees
            WHERE department_id=v_deptid;
        WHEN 200 THEN
            ...
    END CASE;
    DBMS_OUTPUT.PUT_LINE ('You are working in the '|| v_deptname ||
    ' department. There are '||v_emps ||' employees in this
    department');
END;
/

```

ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Sentencia CASE

Recuerde el uso de la sentencia IF. Puede incluir *n* sentencias PL/SQL en la cláusula THEN y en la cláusula ELSE. Del mismo modo, puede incluir sentencias en la sentencia CASE, que es más legible comparada con varias sentencias IF y ELSIF.

Diferencias entre una Expresión CASE y una Sentencia CASE

Una expresión CASE evalúa la condición y devuelve un valor, mientras que una sentencia CASE evalúa la condición y realiza una acción. Una sentencia CASE puede ser un bloque PL/SQL completo.

- Las sentencias CASE terminan en END CASE;
- Las expresiones CASE terminan en END;

La salida del código de ejemplo de la diapositiva es la siguiente:

```

anonymous block completed
You are working in the Finance department. There are 6 employees in this department

```

Nota: mientras que es posible que una sentencia IF no realice ninguna acción (porque todas las condiciones sean falsas y la cláusula ELSE no sea obligatoria), una sentencia CASE debe ejecutar alguna sentencia PL/SQL.

Manejo de Valores Nulos

Cuando se trabaja con valores nulos, puede evitar que se produzcan algunos errores comunes si se recuerdan las siguientes reglas:

- Las comparaciones sencillas donde aparecen valores nulos siempre generan NULL.
- Si se aplica el operador lógico NOT a un valor nulo, se genera NULL.
- Si la condición genera NULL en sentencias de control condicional, la secuencia de sentencias asociada no se ejecuta.

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Manejo de Valores Nulos

Considere el siguiente ejemplo:

```
x := 5;
y := NULL;
...
IF x != y THEN -- yields NULL, not TRUE
    -- sequence_of_statements that are not executed
END IF;
```

Puede que espere que se ejecute la secuencia de sentencias porque x e y parecen desiguales. Sin embargo, los valores nulos son indeterminados. Se desconoce si x es igual o no a y. Por lo tanto, la condición IF genera NULL y se omite la secuencia de sentencias.

```
a := NULL;
b := NULL;
...
IF a = b THEN -- yields NULL, not TRUE
    -- sequence_of_statements that are not executed
END IF;
```

En el segundo ejemplo, puede que espere que se ejecute la secuencia de sentencias porque a y b parecen iguales. Pero, de nuevo, se desconoce la igualdad, por lo que la condición IF genera NULL y se omite la secuencia de sentencias.

Tablas Lógicas

Cree una condición booleana simple con un operador de comparación.

AND	<i>TRUE</i>	<i>FALSE</i>	<i>NULL</i>	OR	<i>TRUE</i>	<i>FALSE</i>	<i>NULL</i>	NOT	
<i>TRUE</i>	TRUE	FALSE	NULL	<i>TRUE</i>	TRUE	TRUE	TRUE	<i>TRUE</i>	FALSE
<i>FALSE</i>	FALSE	FALSE	FALSE	<i>FALSE</i>	TRUE	FALSE	NULL	<i>FALSE</i>	TRUE
<i>NULL</i>	NULL	FALSE	NULL	<i>NULL</i>	TRUE	NULL	NULL	<i>NULL</i>	NULL

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Tablas Lógicas

Puede crear una condición booleana simple combinando expresiones de números, caracteres y fechas con operadores de comparación.

Puede crear una condición booleana compleja combinando condiciones booleanas simples con los operadores lógicos AND, OR y NOT. Los operadores lógicos se utilizan para comprobar los valores de la variable booleana y devolver TRUE, FALSE o NULL. En las tablas lógicas de la diapositiva:

- FALSE tiene prioridad en una condición AND, mientras que TRUE tiene prioridad en una condición OR
- AND devuelve TRUE sólo si sus dos operandos son TRUE
- OR devuelve FALSE sólo si sus dos operandos son FALSE
- NULL AND TRUE siempre se evalúa en NULL porque no se sabe si el segundo operando se evalúa en TRUE

Nota: la negación de NULL (NOT NULL) origina un valor nulo porque estos valores son indeterminados.

¿Expresiones Booleanas o Expresiones Lógicas?

¿Cuál es el valor de flag en cada caso?

```
flag := reorder_flag AND available_flag;
```

REORDER_FLAG	AVAILABLE_FLAG	FLAG
TRUE	TRUE	? (1)
TRUE	FALSE	? (2)
NULL	TRUE	? (3)
NULL	FALSE	? (4)

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

¿Expresiones Booleanas o Expresiones Lógicas?

La tabla lógica AND puede ayudarle a evaluar las posibilidades de la condición booleana en la diapositiva.

Respuestas

1. TRUE
2. FALSE
3. NULL
4. FALSE

Agenda

- Uso de sentencias IF
- Uso de sentencias y expresiones CASE
- Creación e identificación de sentencias de bucle

ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Control Iterativo: Sentencias LOOP

- Los bucles repiten una sentencia (o una secuencia de sentencias) varias veces.
- Hay tres tipos de bucles:
 - Bucle básico
 - Bucle FOR
 - Bucle WHILE



ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Control Iterativo: Sentencias LOOP

PL/SQL proporciona varias utilidades para estructurar los bucles de forma que repitan una sentencia o una secuencia de sentencias varias veces. Los bucles se utilizan principalmente para ejecutar sentencias de forma repetida hasta alcanzar una condición de salida. Es obligatorio que haya una condición de salida en un bucle; de lo contrario, éste será infinito.

Las construcciones en bucle constituyen el tercer tipo de estructura de control. PL/SQL proporciona los siguientes tipos de bucles:

- Bucle básico que realiza acciones repetitivas sin condiciones generales
- Bucle FOR que realizan acciones iterativas basadas en un recuento
- Bucle WHILE que realizan acciones iterativas basadas en una condición

Nota: se puede utilizar una sentencia EXIT para terminar los bucles. Un bucle básico debe tener EXIT. El bucle FOR de cursor (que es otro tipo de bucle FOR) se aborda en la lección titulada “Uso de Cursos Explícitos”.

Bucles Básicos

Sintaxis:

```
LOOP
    statement1;
    . . .
    EXIT [WHEN condition];
END LOOP;
```



Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Bucles Básicos

La forma más sencilla de sentencia LOOP es el bucle básico, que delimita una secuencia de sentencias entre las palabras clave LOOP y END LOOP. Cada vez que el flujo de ejecución alcanza la sentencia END LOOP, el control se devuelve a la sentencia LOOP correspondiente situada por encima. Un bucle básico permite la ejecución de sus sentencias al menos una vez, incluso aunque la condición EXIT ya se haya cumplido al introducir el bucle. Sin la sentencia EXIT, el bucle sería infinito.

Sentencia EXIT

Puede utilizar la sentencia EXIT para terminar un bucle. El control pasa a la siguiente sentencia después de la sentencia END LOOP. Puede emitir EXIT como una acción en una sentencia IF o como una sentencia autónoma en el bucle. La sentencia EXIT debe estar dentro de un bucle. En el segundo caso, puede adjuntar una cláusula WHEN para permitir la terminación condicional del bucle. Cuando se detecte la sentencia EXIT, se evaluará la condición de la cláusula WHEN. Si la condición genera TRUE, el bucle termina y el control pasa a la siguiente sentencia después del bucle. Un bucle básico puede contener varias sentencias EXIT, pero se recomienda que sólo tenga un punto EXIT.

Bucle Básico: Ejemplo

```

DECLARE
    v_countryid      locations.country_id%TYPE := 'CA';
    v_loc_id          locations.location_id%TYPE;
    v_counter         NUMBER(2) := 1;
    v_new_city        locations.city%TYPE := 'Montreal';
BEGIN
    SELECT MAX(location_id) INTO v_loc_id FROM locations
    WHERE country_id = v_countryid;
    LOOP
        INSERT INTO locations(location_id, city, country_id)
        VALUES((v_loc_id + v_counter), v_new_city, v_countryid);
        v_counter := v_counter + 1;
        EXIT WHEN v_counter > 3;
    END LOOP;
END;
/

```



Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Bucle Básico: Ejemplo

El ejemplo de bucle básico que se muestra en la diapositiva se define de la siguiente forma: “Inserte tres nuevos identificadores de ubicaciones para el código de país CA y la ciudad Montreal”.

Nota

- Un bucle básico permite la ejecución de sus sentencias hasta que se cumpla la condición EXIT WHEN.
- Si la condición se coloca en el bucle para que no se compruebe hasta después de la ejecución de las sentencias de bucle, el bucle se ejecuta al menos una vez.
- No obstante, si la condición de salida se coloca en la parte superior del bucle, antes de cualquier otra sentencia ejecutable, y dicha condición es verdadera, el bucle se cerrará y las sentencias no se ejecutarán nunca.

Resultados

Para ver la salida, ejecute el código de ejemplo: code_05_22_s.sql.

Bucles WHILE

Sintaxis:

```
WHILE condition LOOP
    statement1;
    statement2;
    . . .
END LOOP;
```

Utilice el bucle WHILE para repetir sentencias mientras una condición sea TRUE.

ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Bucles WHILE

Puede utilizar el bucle WHILE para repetir una secuencia de sentencias hasta que la condición de control no siga siendo TRUE. La condición se evalúa al inicio de cada iteración. El bucle terminará cuando la condición sea FALSE o NULL. Si la condición es FALSE o NULL al inicio del bucle, no se realizarán más iteraciones. De esta forma, es posible que no se ejecute ninguna sentencia del bucle.

En la sintaxis:

<i>condition</i>	Es una expresión o variable booleana (TRUE, FALSE o NULL)
<i>statement</i>	Puede ser una o más sentencias PL/SQL o SQL

Si las variables relacionadas con las condiciones no cambian en el cuerpo del bucle, la condición seguirá siendo TRUE y el bucle no terminará.

Nota: si la condición genera NULL, el bucle no se usa y el control pasará a la siguiente sentencia.

Bucles WHILE: Ejemplo

```
DECLARE
    v_countryid    locations.country_id%TYPE := 'CA';
    v_loc_id        locations.location_id%TYPE;
    v_new_city      locations.city%TYPE := 'Montreal';
    v_counter       NUMBER := 1;
BEGIN
    SELECT MAX(location_id) INTO v_loc_id FROM locations
    WHERE country_id = v_countryid;
    WHILE v_counter <= 3 LOOP
        INSERT INTO locations(location_id, city, country_id)
        VALUES((v_loc_id + v_counter), v_new_city, v_countryid);
        v_counter := v_counter + 1;
    END LOOP;
END;
/
```



Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Bucles WHILE: Ejemplo

En el ejemplo de la diapositiva, se han agregado tres nuevos identificadores de ubicaciones para el código de país CA y la ciudad Montreal.

- Con cada iteración a través del bucle WHILE, se incrementa un contador (`v_counter`).
- Si el número de iteraciones es menor o igual que el número 3, se ejecutará el código del bucle y se insertará una fila en la tabla `locations`.
- Una vez que `v_counter` supere el número de ubicaciones nuevas para esta ciudad y país, la condición que controla el bucle se evaluará como FALSE y el bucle se terminará.

Resultados

Para ver la salida, ejecute el código de ejemplo: `code_05_24_s.sql`.

Bucles FOR

- Utilice un bucle FOR para abreviar la prueba del número de iteraciones.
- No declare el contador; ya está declarado implícitamente.

```
FOR counter IN [REVERSE]
    lower_bound..upper_bound LOOP
        statement1;
        statement2;
        . . .
    END LOOP;
```



Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Bucles FOR

Los bucles FOR tienen la misma estructura general que el bucle básico. Además, tienen una sentencia de control antes de la palabra clave LOOP para definir el número de iteraciones que realiza PL/SQL.

En la sintaxis:

<i>counter</i>	Es un entero declarado implícitamente cuyo valor aumenta o disminuye de forma automática (disminuye si se utiliza la palabra clave REVERSE) en 1 en cada iteración del bucle hasta alcanzar el límite superior o inferior.
REVERSE	Provoca la disminución del contador con cada iteración desde el límite superior al inferior. Nota: primero se sigue haciendo referencia al límite inferior.
<i>lower_bound</i>	Especifica el límite inferior para el rango de valores del contador.
<i>upper_bound</i>	Especifica el límite superior para el rango de valores del contador.

No declare el contador. Está declarado implícitamente como un entero.

Bucles FOR (continuación)

Nota: la secuencia de sentencias se ejecuta cada vez que se incrementa el contador, según determinen los dos límites. Los límites inferior y superior del rango del bucle pueden ser literales, variables o expresiones, pero se deben evaluar como enteros. Los límites se redondean a enteros, es decir, 11/3 y 8/5 son límites superiores o inferiores válidos. Los límites inferior y superior están dentro del rango del bucle. Si el límite inferior del rango del bucle se evalúa como un entero mayor que el límite superior, la secuencia de sentencias no se ejecutará.

Por ejemplo, la siguiente sentencia sólo se ejecutará una vez:

```
FOR i IN 3..3
LOOP
    statement1;
END LOOP;
```

Bucles FOR: Ejemplo

```
DECLARE
    v_countryid    locations.country_id%TYPE := 'CA';
    v_loc_id        locations.location_id%TYPE;
    v_new_city      locations.city%TYPE := 'Montreal';
BEGIN
    SELECT MAX(location_id) INTO v_loc_id
        FROM locations
        WHERE country_id = v_countryid;
    FOR i IN 1..3 LOOP
        INSERT INTO locations(location_id, city, country_id)
        VALUES((v_loc_id + i), v_new_city, v_countryid );
    END LOOP;
END ;
/
```



Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Bucles FOR: Ejemplo

Ya ha aprendido a insertar tres nuevas ubicaciones para el código de país CA y la ciudad Montreal con el bucle básico y el bucle WHILE. En el ejemplo de la diapositiva se muestra cómo lograr lo mismo con el bucle FOR.

Resultados

Para ver la salida, ejecute el código de ejemplo code_05_27_s.sql.

Reglas del Bucle FOR

- Haga referencia al contador sólo dentro del bucle; no está definido fuera del mismo.
- No haga referencia al contador como destino de una asignación.
- Ningún límite del bucle debe ser NULL.

ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Reglas del Bucle FOR

La diapositiva muestra las instrucciones que hay que seguir al escribir un bucle FOR.

Nota: no es necesario que los límites inferior y superior de una sentencia LOOP sean literales numéricos. Pueden ser expresiones que se convierten en valores numéricos.

Ejemplo:

```
DECLARE
    v_lower  NUMBER := 1;
    v_upper  NUMBER := 100;
BEGIN
    FOR i IN v_lower..v_upper LOOP
        ...
    END LOOP;
END;
/
```

Uso Sugerido para los Bucles

- Utilice el bucle básico cuando se deban ejecutar las sentencias que hay dentro del bucle al menos una vez.
- Utilice el bucle WHILE si se debe evaluar la condición al inicio de cada iteración.
- Utilice un bucle FOR si se conoce el número de iteraciones.



Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Uso Sugerido para los Bucles

Un bucle básico permite la ejecución de su sentencia al menos una vez, incluso si la condición ya se ha cumplido al introducir el bucle. Sin la sentencia EXIT, el bucle sería infinito.

Puede utilizar el bucle WHILE para repetir una secuencia de sentencias hasta que la condición de control no siga siendo TRUE. La condición se evalúa al inicio de cada iteración. El bucle terminará cuando la condición sea FALSE. Si la condición es FALSE al inicio del bucle, no se realizarán más iteraciones.

Los bucles FOR tienen una sentencia de control antes de la palabra clave LOOP para determinar el número de iteraciones que realiza PL/SQL. Utilice un bucle FOR si el número de iteraciones está predeterminado.

Etiquetas y Bucles Anidados

- Puede anidar bucles a varios niveles.
- Utilice etiquetas para distinguir entre bloques y bucles.
- Salga del bucle externo con la sentencia `EXIT` a la que hace referencia la etiqueta.



Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Etiquetas y Bucles Anidados

Puede anidar bucles `FOR`, `WHILE` y básicos unos dentro de otros. La terminación de un bucle anidado no finaliza el bucle delimitador a menos que se emita una excepción. Sin embargo, puede etiquetar bucles y salir del bucle externo con la sentencia `EXIT`.

Los nombres de las etiquetas siguen las mismas reglas que otros identificadores. La etiqueta se coloca antes de la sentencia, en la misma línea o en otra distinta. El espacio en blanco no tiene importancia en los análisis de PL/SQL, excepto dentro de los literales. Etiquete los bucles básicos colocando la etiqueta antes de la palabra `LOOP` en los delimitadores de etiquetas (`<<etiqueta>>`). En los bucles `FOR` y `WHILE`, coloque la etiqueta antes de `FOR` o `WHILE`.

Si el bucle está etiquetado, el nombre de la etiqueta se puede incluir (opcionalmente) después de la sentencia `END LOOP` para su claridad.

Etiquetas y Bucles Anidados: Ejemplo

```
...
BEGIN
  <<Outer_loop>>
  LOOP
    v_counter := v_counter+1;
  EXIT WHEN v_counter>10;
  <<Inner_loop>>
  LOOP
  ...
  EXIT Outer_loop WHEN total_done = 'YES';
  -- Leave both loops
  EXIT WHEN inner_done = 'YES';
  -- Leave inner loop only
  ...
END LOOP Inner_loop;
...
END LOOP Outer_loop;
END ;
/
```

ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Etiquetas y Bucles Anidados: Ejemplo

En el ejemplo de la diapositiva, hay dos bucles. El bucle externo se identifica con la etiqueta <<Outer_Loop>> y el interno se identifica con la etiqueta <<Inner_Loop>>.

Los identificadores se colocan antes de la palabra LOOP en los delimitadores de etiquetas (<<etiqueta>>). El bucle interno se anida dentro del bucle externo. Los nombres de las etiquetas se incluyen después de las sentencias END LOOP para su claridad.

Sentencia PL/SQL CONTINUE

- Definición
 - Agrega la funcionalidad para iniciar la siguiente iteración de bucle
 - Permite a los programadores transferir el control a la siguiente iteración de un bucle
 - Utiliza estructura y semántica paralelas a la sentencia EXIT
- Ventajas
 - Facilita el proceso de programación
 - Puede proporcionar un ligero aumento de rendimiento respecto a las soluciones de programación anteriores para simular la sentencia CONTINUE



ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Sentencia PL/SQL CONTINUE

La sentencia CONTINUE permite transferir el control de un bucle a una nueva iteración o salir del bucle. Muchos otros lenguajes de programación tienen esta funcionalidad. Con Oracle Database 11g, PL/SQL también la ofrece. Antes de Oracle Database 11g, se podía codificar una solución mediante variables booleanas y sentencias condicionales para simular la funcionalidad de programación CONTINUE. En algunos casos, las soluciones son menos eficientes.

La sentencia CONTINUE ofrece un método simplificado para controlar las iteraciones de bucle. Puede ser más eficiente que las anteriores soluciones de codificación.

El uso más común de la sentencia CONTINUE es para filtrar datos del cuerpo de un bucle antes de que se inicie el procesamiento principal.

Sentencia PL/SQL CONTINUE: Ejemplo 1

```

DECLARE
    v_total SIMPLE_INTEGER := 0;
BEGIN
    FOR i IN 1..10 LOOP
        1 v_total := v_total + i;
        dbms_output.put_line
            ('Total is: '|| v_total);
        CONTINUE WHEN i > 5;
        v_total := v_total + i;
        2 dbms_output.put_line
            ('Out of Loop Total is:
            '|| v_total);
    END LOOP;
END;
/

```

```

anonymous block completed
Total is: 1
Out of Loop Total is:
2
Total is: 4
Out of Loop Total is:
6
Total is: 9
Out of Loop Total is:
12
Total is: 16
Out of Loop Total is:
20
Total is: 25
Out of Loop Total is:
30
Total is: 36
Total is: 43
Total is: 51
Total is: 60
Total is: 70

```

ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Sentencia PL/SQL CONTINUE: Ejemplo 1

Se trata de un código de sólo 11g, ya que la sentencia Continue se ha introducido sólo en 11g.

En el ejemplo, hay dos asignaciones que utilizan la variable `v_total`:

1. La primera asignación se ejecuta por cada 10 iteraciones del bucle.
2. La segunda asignación se ejecuta para las cinco primeras iteraciones del bucle. La sentencia `CONTINUE` transfiere el control de un bucle a una nueva iteración, por lo tanto, en las últimas cinco iteraciones del bucle no se ejecuta la segunda asignación `TOTAL`.

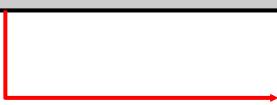
El resultado final de la variable `TOTAL` es 70.

Sentencia PL/SQL CONTINUE: Ejemplo 2

```

DECLARE
  v_total NUMBER := 0;
BEGIN
  <<BeforeTopLoop>>
  FOR i IN 1..10 LOOP
    v_total := v_total + 1;
    dbms_output.put_line
      ('Total is: ' || v_total);
    FOR j IN 1..10 LOOP
      CONTINUE BeforeTopLoop WHEN i + j > 5;
      v_total := v_total + 1;
    END LOOP;
  END LOOP;
END two_loop;

```



The screenshot shows the Oracle SQL Developer interface with the 'Script Output' tab selected. The output window displays the following text:

```

anonymous block completed
Total is: 1
Total is: 6
Total is: 10
Total is: 13
Total is: 15
Total is: 16
Total is: 17
Total is: 18
Total is: 19
Total is: 20

```

ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Sentencia PL/SQL CONTINUE: Ejemplo 2

Puede utilizar la sentencia CONTINUE para saltar a la siguiente iteración de un bucle externo. Se trata de un código de sólo 11g.

Para ello, se proporciona una etiqueta al bucle externo para identificar el lugar en el que se incluye la sentencia CONTINUE.

La sentencia CONTINUE del bucle más interior termina ese bucle si la condición WHEN es verdadera (como con la palabra clave EXIT). Cuando la sentencia CONTINUE termina el bucle más interno, el control se transfiere a la siguiente iteración del bucle más exterior, con la etiqueta BeforeTopLoop en este ejemplo.

Cuando termina este par de bucles, el valor de la variable TOTAL es 20.

También puede utilizar la sentencia CONTINUE en un bloque interior de código, que no contiene un bucle, ya que el bloque está anidado en el bucle exterior adecuado.

Restricciones

- La sentencia CONTINUE no puede aparecer fuera de un bucle, ya que generaría un error de compilación.
- No puede utilizar la sentencia CONTINUE para traspasar el límite de un procedimiento, función o método, ya que generaría un error de compilación.

Prueba

Hay tres tipos de bucles: básico, FOR y WHILE.

- a. Verdadero
- b. Falso



Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Respuesta: a

Tipos de Bucles

PL/SQL proporciona los siguientes tipos de bucles:

- Bucles básicos que realizan acciones repetitivas sin condiciones generales
- Bucles FOR que realizan acciones iterativas basadas en un recuento
- Bucles WHILE que realizan acciones iterativas basadas en una condición

Resumen

En esta lección debe haber aprendido a cambiar el flujo lógico de las sentencias con las siguientes estructuras de control:

- Condicional (sentencia IF)
- Expresiones y sentencias CASE
- Bucles:
 - Bucle básico
 - Bucle FOR
 - Bucle WHILE
- Sentencia EXIT
- Sentencia CONTINUE



Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Resumen

Un lenguaje se puede denominar lenguaje de programación sólo si proporciona estructuras de control para la implantación de la lógica de negocio. Estas estructuras de control se utilizan también para controlar el flujo del programa. PL/SQL es un lenguaje de programación que integra construcciones de programación con SQL.

Una construcción de control condicional comprueba la validez de una condición y realiza una acción en consecuencia. Utilice la construcción IF para realizar una ejecución condicional de sentencias.

Una construcción de control iterativo ejecuta una secuencia de sentencias de forma repetida, siempre que una condición concreta contenga TRUE. Utilice distintas construcciones en bucle para realizar operaciones iterativas.

Práctica 5: Visión General

En esta práctica se abordan los siguientes temas:

- Realización de acciones condicionales mediante sentencias IF
- Realización de pasos iterativos mediante estructuras LOOP



Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Práctica 5: Visión General

En esta práctica, creará bloques PL/SQL que incorporen bucles y estructuras de control condicionales. En los ejercicios se comprobará el conocimiento en escritura de varias sentencias IF y construcciones LOOP.



Trabajar con Tipos de Dato Compuestos

ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Objetivos

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

- Describir las recopilaciones y los registros PL/SQL
- Crear registros PL/SQL definidos por el usuario
- Crear un registro con el atributo %ROWTYPE
- Crear matrices asociativas
 - Tabla INDEX BY
 - Tabla de registros INDEX BY



Objetivos

Ya se ha iniciado en los tipos de dato compuestos. En esta lección, aprenderá más sobre ellos y sus usos.

Agenda

- Introducción a los tipos de dato compuestos
- Uso de registros PL/SQL
 - Manipulación de datos con registros PL/SQL
 - Ventajas del atributo %ROWTYPE
- Uso de recopilaciones PL/SQL
 - Examen de matrices asociativas
 - Introducción a las tablas anidadas
 - Introducción a VARRAY

ORACLE

Tipos de Dato Compuestos

- Pueden contener varios valores (a diferencia de los tipos escalares)
- Existen dos tipos:
 - Registros PL/SQL
 - Recopilaciones PL/SQL
 - Matriz asociativa (tabla INDEX BY)
 - Tabla anidada
 - VARRAY



6 - 4

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Tipos de Dato Compuestos

Ha aprendido que las variables del tipo de dato escalar sólo pueden contener un valor, mientras que una variable del tipo de dato compuesto puede contener varios valores del tipo de dato escalar o compuesto. Hay dos tipos de dato compuestos:

- **Registros PL/SQL:** los registros se utilizan para tratar datos relacionados, pero distintos como unidades lógicas. Un registro PL/SQL puede tener variables de diferentes tipos. Por ejemplo, puede definir un registro para incluir los detalles de empleado. Esto implica el almacenamiento del número de empleado como NUMBER, el nombre y apellido como VARCHAR2, etc. Al crear un registro para almacenar los detalles de empleado, se crea una unidad colectiva lógica. Esto facilita la manipulación y el acceso a los datos.
- **Recopilaciones PL/SQL:** las recopilaciones se utilizan para tratar datos como una sola unidad. Hay tres tipos de recopilaciones:
 - Matriz asociativa
 - Tabla anidada
 - VARRAY

¿Por Qué Utilizar Tipos de Dato Compuestos?

Tendrá todos los datos relacionados como una sola unidad. Podrá acceder a los datos y modificarlos fácilmente. Resultan más fáciles de gestionar, relacionar y transportar los datos si son compuestos. Se podría comparar al hecho de tener una sola maleta para PC portátil en lugar de tener distintas maletas para cada componente.

¿Registros o Recopilaciones PL/SQL?

- Utilice registros PL/SQL para almacenar valores de distintos tipos de dato, pero sólo una incidencia cada vez.
- Utilice recopilaciones PL/SQL cuando desee almacenar valores del mismo tipo de dato.

Registro PL/SQL:

TRUE	23-DEC-98	ATLANTA	
------	-----------	---------	---

Recopilación PL/SQL:

1	SMITH
2	JONES
3	BENNETT
4	KRAMER

↓ ↓
VARCHAR2 PLS_INTEGER

ORACLE

6 - 5

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

¿Registros o Recopilaciones PL/SQL?

Si los registros PL/SQL y las recopilaciones PL/SQL son tipos compuestos, ¿cómo selecciona el que debe utilizar?

- Utilice registros PL/SQL cuando desee almacenar valores de diferentes tipos de dato que están relacionados de forma lógica. Por ejemplo, puede crear un registro PL/SQL para incluir los detalles de empleado e indicar que todos los valores almacenados están relacionados porque proporcionan información acerca de un empleado determinado.
- Utilice recopilaciones PL/SQL cuando desee almacenar valores del mismo tipo de dato. Tenga en cuenta que este tipo de dato puede ser también del tipo compuesto (como registros). Puede definir una recopilación para incluir los nombres de todos los empleados. Puede que haya almacenado n nombres en la recopilación; sin embargo, el nombre 1 no está relacionado con el nombre 2. La única relación entre ellos es que son nombres de empleados. Estas recopilaciones son similares a las matrices de los lenguajes de programación, como C, C++ y Java.

Agenda

- Examen de los tipos de dato compuestos
- Uso de registros PL/SQL
 - Manipulación de datos con registros PL/SQL
 - Ventajas del atributo %ROWTYPE
- Uso de recopilaciones PL/SQL
 - Examen de matrices asociativas
 - Introducción a las tablas anidadas
 - Introducción a VARRAY

ORACLE

6 - 6

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Registros PL/SQL

- Deben contener uno o varios componentes (llamados **campos**) de cualquier tipo de dato de tabla escalar, RECORD o INDEX BY
- Son similares a las estructuras en la mayoría de los lenguajes de tercera generación (incluidos C y C++)
- Están definidos por el usuario y pueden ser el subjuego de una fila en una tabla
- Tratan una recopilación de campos como una unidad lógica
- Resultan convenientes para recuperar una fila de datos de una tabla para su procesamiento



6 - 7

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Registros PL/SQL

Un registro es un grupo de elementos de datos relacionados almacenados en campos, cada uno con su propio nombre y tipo de dato.

- Cada registro definido puede tener tantos campos como sean necesarios.
- Se pueden asignar valores iniciales a los registros y éstos se pueden definir como NOT NULL.
- Los campos sin valores iniciales se inicializan en NULL.
- Se puede utilizar la palabra clave DEFAULT, así como := en la inicialización de los campos.
- Puede definir los tipos RECORD y declarar los registros definidos por el usuario en la parte declarativa de cualquier bloque, subprograma o paquete.
- Puede declarar registros anidados y hacer referencia a ellos. Un registro puede ser el componente de otro registro.

Creación de un Registro PL/SQL

Sintaxis:

1

```
TYPE type_name IS RECORD
  (field_declaration[, field_declaration]...);
```

2

```
identifier type_name;
```

field_declaration:

```
field_name {field_type | variable%TYPE
  | table.column%TYPE | table%ROWTYPE}
  [[NOT NULL] {:= | DEFAULT} expr]
```

ORACLE

6 - 8

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Creación de un Registro PL/SQL

Los registros PL/SQL son tipos compuestos definidos por el usuario. Para utilizarlos, realice los siguientes pasos:

1. Defina el registro en la sección de declaraciones de un bloque PL/SQL. En la diapositiva se muestra la sintaxis para definir el registro.
2. Declare (e inicialice, opcionalmente) los componentes internos de este tipo de registro.

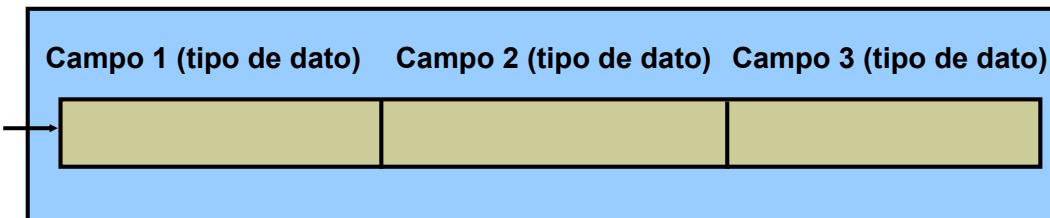
En la sintaxis:

<i>type_name</i>	Es el nombre del tipo RECORD. (Este identificador se utiliza para declarar registros).
<i>field_name</i>	Es el nombre de un campo dentro de un registro.
<i>field_type</i>	Es el tipo de dato del campo. (Representa cualquier tipo de dato PL/SQL excepto REF CURSOR. Puede utilizar los atributos %TYPE y %ROWTYPE).
<i>expr</i>	Es el <i>field_type</i> o un valor inicial.

La restricción NOT NULL evita la asignación de valores nulos a los campos especificados. Asegúrese de inicializar los campos NOT NULL.

Estructura de Registro PL/SQL

Declaraciones de campo:



Ejemplo:



ORACLE

6 - 9

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Estructura de Registro PL/SQL

A los campos de un registro se accede con el nombre de dicho registro. Para hacer referencia o inicializar un campo concreto, utilice la notación de puntos:

record_name.field_name

Por ejemplo, haga referencia al campo `job_id` en el registro `emp_record` de la siguiente forma:

`emp_record.job_id`

A continuación, puede asignar un valor al campo de registro:

`emp_record.job_id := 'ST_CLERK';`

En un bloque o subprograma, los registros definidos por el usuario se instancian al introducir el bloque o subprograma. Dejan de existir al salir del bloque o subprograma.

Atributo %ROWTYPE

- Declare una variable según la recopilación de columnas en una vista o tabla de base de datos.
- Anteponga %ROWTYPE con la vista o tabla de base de datos.
- Los campos del registro toman los nombres y los tipos de dato de las columnas de la tabla o vista.

Sintaxis:

```
DECLARE
  identifier  reference%ROWTYPE;
```

6 - 10

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Atributo %ROWTYPE

Ya sabe que %TYPE se utiliza para declarar una variable de un tipo de columna. La variable tendrá el mismo tamaño y tipo de dato que la columna de tabla. La ventaja de %TYPE es que no tendrá que cambiar la variable si se modifica la columna. Asimismo, si la variable es un número y se utiliza en un cálculo, no tiene que preocuparse de su precisión.

El atributo %ROWTYPE se utiliza para declarar un registro que puede contener una fila completa de una tabla o vista. Los campos del registro toman los nombres y los tipos de dato de las columnas de la tabla o vista. El registro puede almacenar también una fila completa de datos recuperados de un cursor o de una variable de cursor.

La diapositiva muestra la sintaxis para declarar un registro. En la sintaxis:

identifier Es el nombre seleccionado para todo el registro

reference Es el nombre de la tabla, vista, cursor o variable de cursor en el que se va a basar el registro. (Para que esta referencia sea válida, debe existir la tabla o la vista).

En el siguiente ejemplo, se declara un registro utilizando %ROWTYPE como especificador del tipo de dato:

```
DECLARE
  emp_record  employees%ROWTYPE;
  ...
```

Atributo %ROWTYPE (continuación)

El registro emp_record tendrá una estructura compuesta por los siguientes campos, representando cada uno de ellos una columna en la tabla employees.

Nota: esto no es el código, sino simplemente la estructura de la variable compuesta.

```
(employee_id      NUMBER(6),
first_name       VARCHAR2(20),
last_name        VARCHAR2(20),
email            VARCHAR2(20),
phone_number     VARCHAR2(20),
hire_date        DATE,
salary           NUMBER(8,2),
commission_pct   NUMBER(2,2),
manager_id       NUMBER(6),
department_id    NUMBER(4))
```

Para hacer referencia a un campo concreto, utilice la notación de puntos:

```
record_name.field_name
```

Por ejemplo, haga referencia al campo commission_pct en el registro emp_record de la siguiente forma:

```
emp_record.commission_pct
```

A continuación, puede asignar un valor al campo de registro:

```
emp_record.commission_pct := .35;
```

Asignación de Valores a Registros

Puede asignar una lista de valores comunes a un registro mediante la sentencia SELECT o FETCH. Asegúrese de que los nombres de columna aparecen en el mismo orden que los campos del registro. También puede asignar un registro a otro si los dos tienen los mismos tipos de dato correspondientes. Un registro del tipo employees%ROWTYPE y un tipo de registro definido por el usuario con campos análogos de la tabla employees tendrán el mismo tipo de dato. Por lo tanto, si un registro definido por el usuario contiene campos similares a los campos de un registro %ROWTYPE, puede asignar ese registro definido por el usuario al registro %ROWTYPE.

Creación de un Registro PL/SQL: Ejemplo

```

DECLARE
    TYPE t_rec IS RECORD
        (v_sal number(8),
         v_minsal number(8) default 1000,
         v_hire_date employees.hire_date%type,
         v_rec1 employees%rowtype);
    v_myrec t_rec;
BEGIN
    v_myrec.v_sal := v_myrec.v_minsal + 500;
    v_myrec.v_hire_date := sysdate;
    SELECT * INTO v_myrec.v_rec1
        FROM employees WHERE employee_id = 100;
    DBMS_OUTPUT.PUT_LINE(v_myrec.v_rec1.last_name ||' ' ||
        to_char(v_myrec.v_hire_date) ||' '|| to_char(v_myrec.v_sal));
END;

```

anonymous block completed
King 16-FEB-09 1500

ORACLE

6 - 12

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Creación de un Registro PL/SQL: Ejemplo

Las declaraciones de campos que se utilizan para definir un registro son como declaraciones de variables. Cada campo tiene un nombre único y un tipo de dato concreto. No existen tipos de dato predefinidos para registros PL/SQL, como ocurre con las variables escalares. Por lo tanto, debe crear primero el tipo de registro y, a continuación, declarar un identificador que utilice ese tipo.

En el ejemplo de la diapositiva, se crea un registro PL/SQL con el proceso de dos pasos necesarios:

1. Se define un tipo de registro (`t_rec`)
2. Se declara un registro (`v_myrec`) del tipo `t_rec`

Nota

- El registro contiene cuatro campos: `v_sal`, `v_minsal`, `v_hire_date` y `v_rec1`.
- `v_rec1` se define con el atributo `%ROWTYPE`, que es similar al atributo `%TYPE`. Con `%TYPE`, un campo hereda el tipo de dato de la columna especificada. Con `%ROWTYPE`, un campo hereda los nombres de columna y los tipos de dato de todas las columnas de la tabla de referencia.
- Se hace referencia a los campos del registro PL/SQL con la notación `<record>. <field>` o `<record>. <field>. <column>` para los campos definidos con el atributo `%ROWTYPE`.
- Puede agregar la restricción `NOT NULL` a cualquier declaración de campo para evitar la asignación de valores nulos a dicho campo. Recuerde que los campos declarados como `NOT NULL` se deben inicializar.

Ventajas del Uso del Atributo %ROWTYPE

- No es necesario conocer el número y los tipos de dato de las columnas subyacentes de base de datos; de hecho, puede cambiar en tiempo de ejecución.
- El atributo %ROWTYPE es útil si desea recuperar una fila con:
 - La sentencia SELECT *
 - Sentencias INSERT y UPDATE a nivel de fila



6 - 13

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Ventajas del Uso de %ROWTYPE

Las ventajas del uso del atributo %ROWTYPE se muestran en la diapositiva. Utilice el atributo %ROWTYPE cuando no esté seguro de la estructura de la tabla subyacente de base de datos.

La ventaja principal del uso de %ROWTYPE es que simplifica el mantenimiento. El uso de %ROWTYPE garantiza el cambio de forma dinámica de los tipos de dato de las variables que se declaran con este atributo cuando se modifique la tabla subyacente. Si una sentencia DDL cambia las columnas de una tabla, se invalidará la unidad de programa PL/SQL. Cuando el programa se recompile, reflejará automáticamente el nuevo formato de tabla.

El atributo %ROWTYPE resultará especialmente útil cuando desee recuperar una fila completa de una tabla. Cuando no exista este atributo, se verá obligado a declarar una variable para cada una de las columnas que recupere la sentencia SELECT.

Otro Ejemplo de Atributo %ROWTYPE

```

DECLARE
    v_employee_number number:= 124;
    v_emp_rec    employees%ROWTYPE;
BEGIN
    SELECT * INTO v_emp_rec FROM employees
    WHERE employee_id = v_employee_number;
    INSERT INTO retired_emps(empno, ename, job, mgr,
                           hiredate, leavedate, sal, comm, deptno)
    VALUES (v_emp_rec.employee_id, v_emp_rec.last_name,
            v_emp_rec.job_id, v_emp_rec.manager_id,
            v_emp_rec.hire_date, SYSDATE,
            v_emp_rec.salary, v_emp_rec.commission_pct,
            v_emp_rec.department_id);
END;
/

```

SELECT * FROM retired_emps;

	EMPNO	ENAME	JOB	MGR	HIREDATE	LEAVEDATE	SAL	COMM	DEPTNO
1	124	Mourgos	ST_MAN	100	16-NOV-99	16-JUN-09	5800	(null)	50

ORACLE

Otro Ejemplo de Atributo %ROWTYPE

En la diapositiva se muestra otro ejemplo del atributo %ROWTYPE. Si un empleado se jubila, se agregará información acerca del empleado a la tabla que contiene datos sobre los empleados jubilados. El usuario proporciona el número del empleado. El registro del empleado especificado por el usuario se recupera de la tabla employees y se almacena en la variable emp_rec, que se declara mediante el atributo %ROWTYPE.

La sentencia CREATE que crea la tabla retired_emps es:

```

CREATE TABLE retired_emps
(
    EMPNO      NUMBER(4), ENAME        VARCHAR2(10),
    JOB        VARCHAR2(9), MGR         NUMBER(4),
    HIREDATE   DATE, LEAVEDATE    DATE,
    SAL        NUMBER(7,2), COMM        NUMBER(7,2),
    DEPTNO     NUMBER(2)
)

```

Nota

- En la diapositiva se muestra el registro que se inserta en la tabla retired_emps.
- Para ver la salida que se muestra en la diapositiva, coloque el cursor en la sentencia SELECT al final del código de ejemplo en SQL Developer y pulse F9.
- El código de ejemplo completo está en code_6_14_n-s.sql.

Inserción de un Registro mediante %ROWTYPE

```

...
DECLARE
    v_employee_number number:= 124;
    v_emp_rec retired_emps%ROWTYPE;
BEGIN
    SELECT employee_id, last_name, job_id, manager_id,
    hire_date, hire_date, salary, commission_pct,
    department_id INTO v_emp_rec FROM employees
    WHERE employee_id = v_employee_number;
    INSERT INTO retired_emps VALUES v_emp_rec;
END ;
/
SELECT * FROM retired_emps;

```

	EMPNO	ENAME	JOB	MGR	HIREDATE	LEAVEDATE	SAL	COMM	DEPTNO
1	124	Mourgos	ST_MAN	100	16-NOV-99	16-NOV-99	5800	(null)	50

ORACLE

6 - 15

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Inserción de un Registro mediante %ROWTYPE

Compare la sentencia `INSERT` de la diapositiva anterior con la de esta diapositiva. El registro `emp_rec` es del tipo `retired_emps`. El número de campos del registro debe ser igual al número de nombres de los campos de la cláusula `INTO`. Puede utilizar este registro para insertar valores en una tabla. Esto hace que el código sea más legible.

Examine la sentencia `SELECT` de la diapositiva. Se ha seleccionado `hire_date` dos veces y se ha insertado el valor `hire_date` en el campo `leavedate` de `retired_emps`. Ningún empleado se jubilará en la fecha de contratación. El registro insertado se muestra en la diapositiva. (En la siguiente diapositiva verá cómo realizar esta actualización).

Nota: para ver la salida que se muestra en la diapositiva, coloque el cursor en la sentencia `SELECT` al final del código de ejemplo en SQL Developer y pulse F9.

Actualización de una Fila en una Tabla mediante un Registro

```

SET VERIFY OFF
DECLARE
    v_employee_number number:= 124;
    v_emp_rec retired_emps%ROWTYPE;
BEGIN
    SELECT * INTO v_emp_rec FROM retired_emps;
    v_emp_rec.leavedate:=CURRENT_DATE;
    UPDATE retired_emps SET ROW = v_emp_rec WHERE
        empno=v_employee_number;
END ;
/
SELECT * FROM retired_emps;

```

	EMPNO	ENAME	JOB	MGR	HIREDATE	LEAVEDATE	SAL	COMM	DEPTNO
1	124	Mourgos	ST_MAN	100	16-NOV-99	16-NOV-99	5800	(null)	50

ORACLE

Actualización de una Fila en una Tabla mediante un Registro

Ha aprendido a insertar una fila mediante un registro. Esta diapositiva muestra cómo actualizar una fila mediante un registro.

- La palabra clave ROW se utiliza para representar la fila completa.
- El código que se muestra en la diapositiva actualiza el valor de leavedate del empleado.
- El registro se actualiza como se muestra en la diapositiva.

Nota: para ver la salida que se muestra en la diapositiva, coloque el cursor en la sentencia SELECT al final del código de ejemplo en SQL Developer y pulse F9.

Agenda

- Examen de los tipos de dato compuestos
- Uso de registros PL/SQL
 - Manipulación de datos con registros PL/SQL
 - Ventajas del atributo %ROWTYPE
- Uso de recopilaciones PL/SQL
 - Examen de matrices asociativas
 - Introducción a las tablas anidadas
 - Introducción a VARRAY



6 - 17

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Agenda

Como se ha indicado anteriormente, las recopilaciones PL/SQL se utilizan cuando se desea almacenar valores del mismo tipo de dato. Este tipo de dato puede ser también del tipo compuesto (como registros).

Por lo tanto, las recopilaciones se utilizan para tratar datos como una sola unidad. Hay tres tipos de recopilaciones:

- Matriz asociativa
- Tabla anidada
- VARRAY

Nota: de estas tres recopilaciones, esta lección se centrará en la matriz asociativa. La tabla anidada y VARRAY se introducen sólo para comparaciones. Estas otras dos recopilaciones se describen con detalle en el curso titulado *Oracle Database 10g: PL/SQL Avanzado* u *Oracle Database 11g: PL/SQL Avanzado* (según la versión que utilice).

Matrices asociativas (tablas INDEX BY)

Una matriz asociativa es una recopilación PL/SQL con dos columnas:

- Clave primaria de tipo de dato entero o cadena
- Columna de tipo de dato escalar o de registro

Clave	Valores
1	JONES
2	HARDEY
3	MADURO
4	KRAMER

ORACLE

6 - 18

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

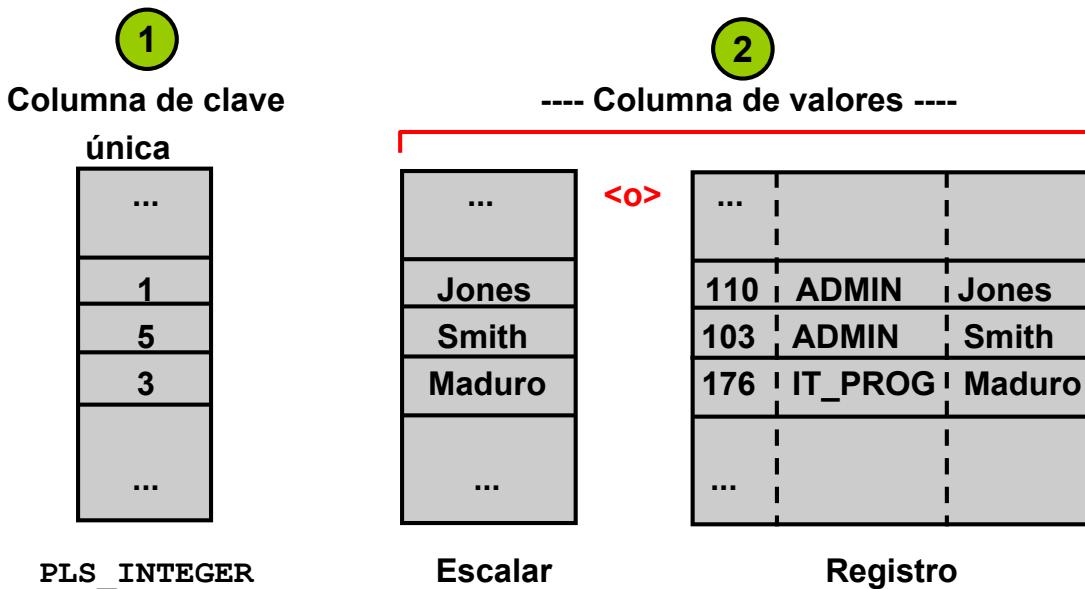
Matrices asociativas (tablas INDEX BY)

Una matriz asociativa es un tipo de recopilación PL/SQL. Es un tipo de dato compuesto y está definida por el usuario. Las matrices asociativas son juegos de pares clave-valor. Pueden almacenar datos utilizando un valor de clave primaria como índice, donde los valores clave no son necesariamente secuenciales. También se conocen como tablas *INDEX BY*.

Tienen sólo dos columnas y ninguna de ellas puede tener nombre:

- La primera columna, de tipo cadena o entero, actúa como clave primaria.
- La segunda columna, de tipo de dato escalar o registro, contiene valores.

Estructura de la Matriz Asociativa



ORACLE

Estructura de la Matriz Asociativa

Como ya se ha mencionado, las matrices asociativas tienen dos columnas. La segunda columna puede contener un valor por fila o varios valores.

Columna de Clave Única: el tipo de dato de la columna de clave puede ser:

- Numérico: `BINARY_INTEGER` o `PLS_INTEGER`. Estos dos tipos de dato numéricos necesitan menos almacenamiento que `NUMBER` y las operaciones aritméticas en estos tipos de dato son más rápidas que en `NUMBER`.
- `VARCHAR2` o uno de sus subtipos

Columna de “Valor”: la columna de valor puede ser un tipo de dato escalar o de registro. Una columna con tipo de dato escalar sólo puede contener un valor por fila, mientras que con el tipo de dato de registro puede contener varios valores por fila.

Otras Características

- Una matriz asociativa no se rellena en el momento de la declaración. No contiene claves ni valores ni se puede inicializar en su declaración.
- Se necesita una sentencia ejecutable explícita para llenar la matriz asociativa.
- Al igual que el tamaño de una tabla de base de datos, el tamaño de una matriz asociativa no tiene límite. Es decir, el número de filas puede aumentar dinámicamente, de forma que la matriz asociativa crecerá a medida que se agreguen nuevas filas. Tenga en cuenta que las claves no tienen que ser secuenciales y pueden ser positivas y negativas.

Pasos para Crear una Matriz Asociativa

Sintaxis:

```

1   TYPE type_name IS TABLE OF
      {column_type | variable%TYPE
      | table.column%TYPE} [NOT NULL]
      | table%ROWTYPE
      | INDEX BY PLS_INTEGER | BINARY_INTEGER
      | VARCHAR2(<size>);

2   identifier type_name;
  
```

Ejemplo:

```

...
TYPE ename_table_type IS TABLE OF
employees.last_name%TYPE
INDEX BY PLS_INTEGER;
...
ename_table ename_table_type;
  
```

ORACLE

Pasos para Crear una Matriz Asociativa

Se incluyen dos pasos en la creación de una matriz asociativa:

1. Declare un tipo de dato TABLE con la opción INDEX BY.
2. Declare una variable de dicho tipo de dato.

Sintaxis

type_name Nombre del tipo TABLE. (Este nombre se utiliza en declaraciones posteriores del identificador de la matriz.)

column_type Cualquier tipo de dato escalar o compuesto como VARCHAR2, DATE, NUMBER o %TYPE. (Puede utilizar el atributo %TYPE para proporcionar el tipo de dato de columna.)

identifier Es el nombre del identificador que representa una matriz asociativa completa.

Nota: la restricción NOT NULL evita que los valores nulos se asignen a la matriz asociativa.

Ejemplo

En el ejemplo, se ha declarado una matriz asociativa con el nombre de variable `ename_table` para almacenar los apellidos de los empleados.

Creación y Acceso a Matrices Asociativas

```

...
DECLARE
    TYPE ename_table_type IS TABLE OF
        employees.last_name%TYPE
        INDEX BY PLS_INTEGER;
    TYPE hiredate_table_type IS TABLE OF DATE
        INDEX BY PLS_INTEGER;
    ename_table      ename_table_type;
    hiredate_table   hiredate_table_type;
BEGIN
    ename_table(1)    := 'CAMERON';
    hiredate_table(8) := SYSDATE + 7;
    IF ename_table.EXISTS(1) THEN
        INSERT INTO ...
    ...
END;
/
...

```

anonymous block completed
 ENAME HIREDT

 CAMERON 23-JUN-09

 1 rows selected

6 - 21

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Creación y Acceso a Matrices Asociativas

En el ejemplo de la diapositiva se crean dos matrices asociativas, con los identificadores `ename_table` y `hiredate_table`.

Se utiliza la clave de cada matriz asociativa para acceder a un elemento de la matriz, con la siguiente sintaxis:

identifier(index)

En ambas matrices, el valor `index` pertenece al tipo `PLS_INTEGER`.

- Para hacer referencia a la primera fila de la matriz asociativa `ename_table`, especifique: `ename_table(1)`
- Para hacer referencia a la octava fila de la matriz asociativa `hiredate_table`, especifique: `hiredate_table(8)`

Nota

- El rango de magnitud de `PLS_INTEGER` es de -2.147.483.647 a 2.147.483.647. Por lo tanto, el valor de la clave primaria puede ser negativo. La indexación no tiene que empezar por 1.
- El método `exists(i)` devuelve TRUE si se devuelve una fila con índice `i`. Utilice el método `exists` para evitar que se produzca un error en relación con un elemento de tabla que no existe.
- El código de ejemplo completo está en `code_6_21_s.sql`.

Uso de los Métodos de Tablas INDEX BY

Los siguientes métodos facilitan el uso de las matrices asociativas:

- EXISTS
- COUNT
- FIRST
- LAST
- PRIOR
- NEXT
- DELETE



Uso de los Métodos de Tablas INDEX BY

El método de tabla INDEX BY es una función o procedimiento incorporado que opera en una tabla matriz asociativa y se llama mediante una notación de puntos.

Sintaxis: `table_name.method_name[(parameters)]`

Método	Descripción
EXISTS (<i>n</i>)	Devuelve TRUE si existe el <i>n</i> elemento en una matriz asociativa
COUNT	Devuelve el número de elementos que contiene actualmente una matriz asociativa
FIRST	<ul style="list-style-type: none"> • Devuelve el primer (menor) número de índice en una matriz asociativa • Devuelve NULL si la matriz asociativa está vacía
LAST	<ul style="list-style-type: none"> • Devuelve el último (mayor) número de índice en una matriz asociativa y • Devuelve NULL si la matriz asociativa está vacía
PRIOR (<i>n</i>)	Devuelve el número de índice que precede al índice <i>n</i> en una matriz asociativa
NEXT (<i>n</i>)	Devuelve el número de índice que supera al índice <i>n</i> en una matriz asociativa
DELETE	<ul style="list-style-type: none"> • DELETE elimina todos los elementos de una matriz asociativa • DELETE (<i>n</i>) elimina el <i>n</i> elemento de una matriz asociativa • DELETE (<i>m</i>, <i>n</i>) elimina todos los elementos del rango <i>m</i> ... <i>n</i> de una matriz asociativa

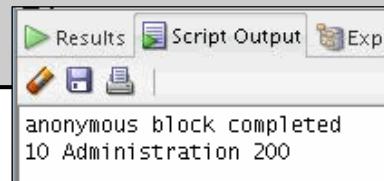
Opción de Tabla de Registros INDEX BY

Defina una matriz asociativa para contener una fila completa de una tabla.

```

DECLARE
    TYPE dept_table_type IS TABLE OF
        departments%ROWTYPE INDEX PLS_INTEGER;
    dept_table dept_table_type;
    -- Each element of dept_table is a record
Begin
    SELECT * INTO dept_table(1) FROM departments
    WHERE department_id = 10;
    DBMS_OUTPUT.PUT_LINE(dept_table(1).department_id || ||
                         dept_table(1).department_name || ||
                         dept_table(1).manager_id);
END;
/

```



ORACLE

6 - 23

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Opción de Tabla de Registros INDEX BY

Como se ha comentado anteriormente, una matriz asociativa declarada como tabla de tipo de dato escalar puede almacenar los detalles de una sola columna en una tabla de base de datos. Sin embargo, suele ser necesario almacenar todas las columnas recuperadas por una consulta. La opción de tabla de registros INDEX BY permite que una definición de matriz contenga información de todos los campos de una tabla de base de datos.

Creación y Referencia a una Tabla de Registros

Como se muestra en el ejemplo de matriz asociativa de la diapositiva, puede:

- Utilizar el atributo %ROWTYPE para declarar un registro que representa una fila en una tabla de base de datos.
- Hacer referencia a los campos de la matriz dept_table porque cada elemento de la matriz es un registro.

Las diferencias entre el atributo %ROWTYPE y el registro PL/SQL del tipo de dato compuesto son las siguientes:

- Los tipos de registro PL/SQL pueden estar definidos por el usuario, mientras que %ROWTYPE define el registro implícitamente.
- Los registros PL/SQL le permiten especificar los campos y sus tipos de dato al declararlos. Al utilizar %ROWTYPE, no se pueden especificar los campos. El atributo %ROWTYPE representa una fila de tabla con todos los campos basados en la definición de dicha tabla.
- Los registros definidos por el usuario son estáticos, pero los registros %ROWTYPE son dinámicos: se basan en una estructura de tabla. Si la estructura de la tabla cambia, la estructura del registro también recoge el cambio.

Opción de Tabla de Registros INDEX BY: Ejemplo 2

```

DECLARE
    TYPE emp_table_type IS TABLE OF
        employees%ROWTYPE INDEX BY PLS_INTEGER;
    my_emp_table emp_table_type;
    max_count      NUMBER(3) := 104;
BEGIN
    FOR i IN 100..max_count
    LOOP
        SELECT * INTO my_emp_table(i) FROM employees
        WHERE employee_id = i;
    END LOOP;
    FOR i IN my_emp_table.FIRST..my_emp_table.LAST
    LOOP
        DBMS_OUTPUT.PUT_LINE(my_emp_table(i).last_name);
    END LOOP;
END;
/

```

ORACLE

6 - 24

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Tabla de Registros INDEX BY: Ejemplo 2

En el ejemplo de la diapositiva se declara una matriz asociativa, con la opción de tabla de registros INDEX BY, para almacenar temporalmente los detalles de los empleados cuyos identificadores estén entre 100 y 104. El nombre de la variable de la matriz es `emp_table_type`.

Con un bucle, se recupera la información de los empleados de la tabla `EMPLOYEES` y se almacena en la matriz. Se utiliza otro bucle para imprimir los apellidos de la matriz. Observe el uso de los métodos `first` y `last` del ejemplo.

Nota: en la diapositiva se muestra una forma de trabajar con una matriz asociativa que utiliza el método de tabla de registros INDEX BY. Sin embargo, puede realizar lo mismo, pero de forma más eficiente mediante cursos. Los cursos se explican en la lección titulada “Uso de Cursos Explícitos”.

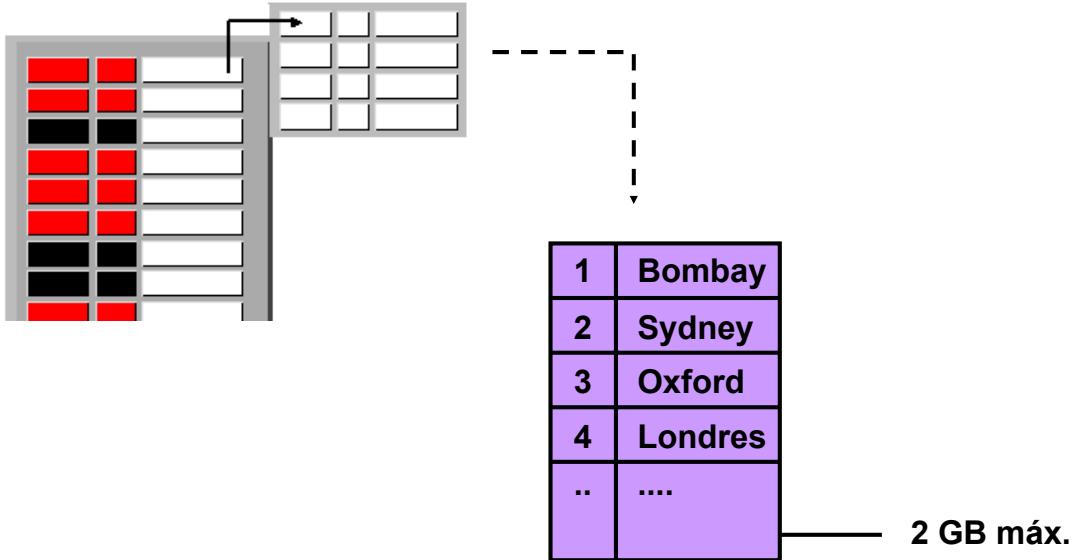
Los resultados del código de ejemplo son los siguientes:

```

Results Script Output Explain
anonymous block completed
King
Kochhar
De Haan
Hunold
Ernst

```

Tablas Anidadas



ORACLE

6 - 25

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Tablas Anidadas

La funcionalidad de las tablas anidadas es similar a la de las matrices asociativas; sin embargo, hay diferencias en la implantación de tablas anidadas.

- La tabla anidada es un tipo de dato válido en una tabla a nivel de esquema, pero no ocurre lo mismo con una matriz asociativa. Por lo tanto, a diferencia de las matrices asociativas, las tablas anidadas se pueden almacenar en la base de datos.
- El tamaño de una tabla anidada puede aumentar de forma dinámica, aunque su máximo es 2 GB.
- La “clave” no puede ser un valor negativo (a diferencia que con la matriz asociativa). Aunque la referencia se hace a la primera columna como clave, no hay ninguna clave en el caso de las tablas anidadas. Hay una columna con números.
- Los elementos se pueden suprimir desde cualquier lugar de una tabla anidada dejando una tabla dispersa con “claves” no secuenciales. Las filas de una tabla anidada no siguen un orden concreto.
- Al recuperar los valores de una tabla anidada, se proporcionan scripts secundarios consecutivos a las filas, empezando por 1.

Sintaxis

```
TYPE type_name IS TABLE OF
  {column_type | variable%TYPE
  | table.column%TYPE} [NOT NULL]
  | table.%ROWTYPE
```

Tablas Anidadas (continuación)

Ejemplo:

```
TYPE location_type IS TABLE OF locations.city%TYPE;
offices location_type;
```

Si no inicializa una tabla anidada, se inicializará de forma automática en NULL. Puede inicializar la tabla anidada `offices` mediante un constructor:

```
offices := location_type('Bombay', 'Tokyo', 'Singapore',
'London');
```

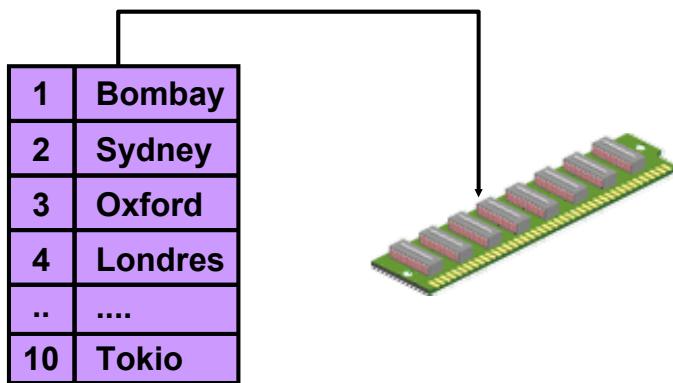
El código de ejemplo completo y la salida son los siguientes:

```
SET SERVEROUTPUT ON;

DECLARE
    TYPE location_type IS TABLE OF locations.city%TYPE;
    offices location_type;
    table_count NUMBER;
BEGIN
    offices := location_type('Bombay', 'Tokyo', 'Singapore',
        'London');
    FOR i in 1.. offices.count() LOOP
        DBMS_OUTPUT.PUT_LINE(offices(i));
    END LOOP;
END;
/
```

```
Results Script Output Explain
anonymous block completed
Bombay
Tokyo
Singapore
London
```

VARRAY



ORACLE

6 - 27

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

VARRAY

Las matrices de tamaño variable (VARRAY) son similares a las matrices asociativas, a excepción de que VARRAY tiene límite en cuanto a tamaño.

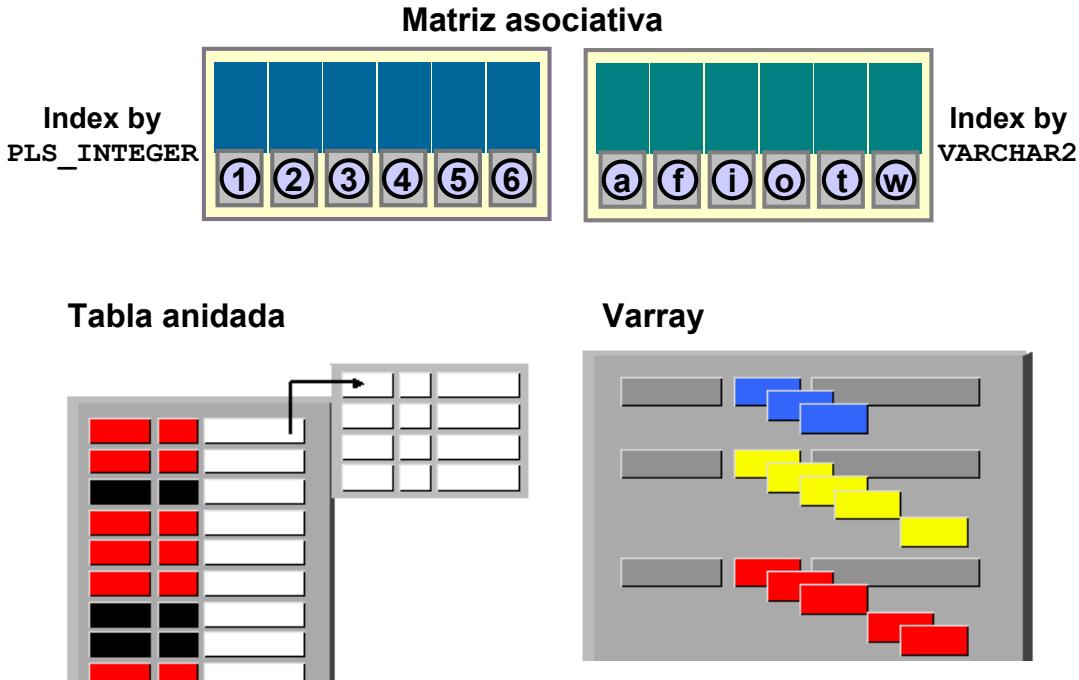
- VARRAY es válida en una tabla a nivel de esquema.
- Los elementos del tipo VARRAY se denominan VARRAY.
- VARRAY tiene un límite superior fijo. Debe especificar el límite superior cuando los declare. Esto es similar a las matrices del lenguaje C. El tamaño máximo de una VARRAY es de 2 GB, como en las tablas anidadas.
- La distinción entre tabla anidada y VARRAY reside en el modo de almacenamiento físico. Los elementos de VARRAY se almacenan en línea con los datos de la tabla, a menos que el tamaño de VARRAY sea mayor que 4 KB. Esto contrasta con las tablas anidadas, que siempre se almacenan fuera de línea.
- Puede crear un tipo VARRAY en la base de datos mediante SQL.

Ejemplo:

```
TYPE location_type IS VARRAY(3) OF locations.city%TYPE;
offices location_type;
```

El tamaño de VARRAY está restringido a 3. Puede inicializar VARRAY mediante constructores. Si intenta inicializar VARRAY con más de tres elementos, se mostrará el mensaje de error “Subscript outside of limit”.

Resumen de los Tipos de Recopilación



Resumen de los Tipos de Recopilación

Matrices Asociativas

Las matrices asociativas son juegos de pares clave-valor, en los que cada clave es única y se utiliza para localizar un valor correspondiente en la matriz. La clave se puede basar en un entero o en un carácter. El valor de la matriz puede ser un tipo de dato escalar (un único valor) o tipo de dato de registro (varios valores).

Ya que las matrices asociativas están destinadas a almacenar datos temporales, no puede utilizarlas con sentencias SQL como `INSERT` y `SELECT INTO`.

Tablas Anidadas

Las tablas anidadas contienen un juego de valores. Es decir, es una tabla dentro de otra tabla. Las tablas anidadas son ilimitadas, es decir, el tamaño de la tabla puede aumentar de forma dinámica. Las tablas anidadas están disponibles tanto en PL/SQL como en bases de datos. En PL/SQL, las tablas anidadas son como matrices de una dimensión cuyo tamaño puede aumentar dinámicamente.

Varrays

Las matrices de tamaño variable, o varrays, también son recopilaciones de elementos homogéneos que contienen un número fijo de elementos (aunque se puede cambiar el número de elementos en tiempo de ejecución). Utilizan números secuenciales como scripts secundarios. Se pueden definir tipos SQL equivalentes, permitiendo, por tanto, que las varrays se almacenen en las tablas de base de datos.

Prueba

Identifique las situaciones en las que se puede utilizar el atributo %ROWTYPE.

- a. Cuando no se está seguro de la estructura de la tabla de base de datos subyacente
- b. Cuando se quiere recuperar una fila entera de una tabla
- c. Cuando se quiere declarar una variable según una columna de base de datos o variable declarada anteriormente



Respuesta: a, b

Ventajas del Uso del Atributo %ROWTYPE

Utilice el atributo %ROWTYPE cuando no esté seguro de la estructura de la tabla subyacente de base de datos.

La ventaja principal del uso de %ROWTYPE es que simplifica el mantenimiento. El uso de %ROWTYPE garantiza el cambio de forma dinámica de los tipos de dato de las variables que se declaran con este atributo cuando se modifique la tabla subyacente. Si una sentencia DDL cambia las columnas de una tabla, se invalidará la unidad de programa PL/SQL. Cuando el programa se recompile, reflejará automáticamente el nuevo formato de tabla.

El atributo %ROWTYPE resultará especialmente útil cuando desee recuperar una fila completa de una tabla. Cuando no exista este atributo, se verá obligado a declarar una variable para cada una de las columnas que recupere la sentencia SELECT.

Resumen

En esta lección debe haber aprendido lo siguiente:

- Definir y hacer referencia a variables PL/SQL de tipos de dato compuestos
 - Registro PL/SQL
 - Matriz asociativa
 - Tabla INDEX BY
 - Tabla de registros INDEX BY
- Definir un registro PL/SQL mediante el atributo %ROWTYPE
- Comparar y contrastar los tres tipos de recopilación PL/SQL:
 - Matriz asociativa
 - Tabla anidada
 - VARRAY



Resumen

Un registro PL/SQL es una recopilación de campos individuales que representan una fila en una tabla. Mediante los registros, puede agrupar los datos en una estructura y, a continuación, manipularla como una entidad o unidad lógica. De esta forma, se reduce la codificación y se facilita el mantenimiento y la comprensión del código.

Al igual que los registros PL/SQL, una recopilación PL/SQL es otro tipo de dato compuesto. Las recopilaciones PL/SQL incluyen:

- Matrices asociativas (también se conocen como tablas INDEX BY). Son objetos del tipo TABLE y son similares a las tablas de base de datos, con una leve diferencia. Las llamadas tablas INDEX BY utilizan una clave primaria para proporcionarle acceso como el de la matriz a las filas. El tamaño de una matriz asociativa no tiene límite.
- Tablas anidadas. La clave de las tablas anidadas no puede contener un valor negativo, a diferencia de las tablas INDEX BY. La clave también debe estar en una secuencia.
- Matrices de tamaño variable (VARRAY). VARRAY es similar a las matrices asociativas, excepto en que VARRAY tiene un tamaño limitado.

Práctica 6: Visión General

En esta práctica se abordan los siguientes temas:

- Declaración de matrices asociativas
- Procesamiento de datos con matrices asociativas
- Declaración de un registro PL/SQL
- Procesamiento de datos mediante un registro PL/SQL



Práctica 6: Visión General

En esta práctica, defina, cree y utilice matrices asociativas y registros PL/SQL.

Uso de Cursos Explícitos

ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Objetivos

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

- Distinguir entre un cursor implícito y otro explícito
- Describir los motivos de uso de cursos explícitos
- Declarar y controlar cursos explícitos
- Utilizar bucles simples y bucles FOR de cursor para recuperar datos
- Declarar y utilizar cursos con parámetros
- Bloquear filas con la cláusula FOR UPDATE
- Hacer referencia a la fila actual con la cláusula WHERE CURRENT OF



Objetivos

Se han tratado los cursos implícitos que PL/SQL crea automáticamente al ejecutar una sentencia SQL SELECT o DML. En esta lección, aprenderá acerca de los cursos explícitos. Aprenderá a diferenciar entre cursos implícitos y explícitos. También aprenderá a declarar y controlar cursos simples y cursos con parámetros.

Agenda

- ¿Qué son los cursos explícitos?
- Uso de cursos explícitos
- Uso de cursos con parámetros
- Bloqueo de filas y referencia a la fila actual

ORACLE

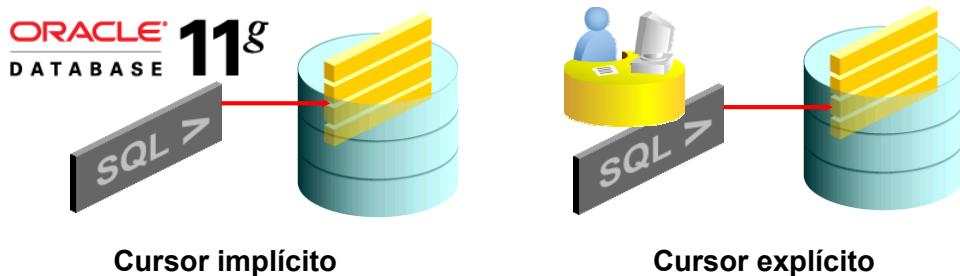
7 - 3

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Cursos

Todas las sentencias SQL ejecutadas por Oracle Server tienen un cursor individual asociado:

- Cursos implícitos: se declaran y se gestionan por PL/SQL para todas las sentencias DML y PL/SQL SELECT
- Cursos explícitos: se declaran y se gestionan por el programador



Cursos

Oracle Server utiliza áreas de trabajo (denominadas *áreas SQL privadas*) para ejecutar sentencias SQL y almacenar información del procesamiento. Puede utilizar cursos explícitos para asignar un nombre a un área SQL privada y acceder a la información almacenada.

Tipo de cursor	Descripción
Implícito	Los cursos implícitos se declaran implícitamente por PL/SQL para todas las sentencias SELECT DML y PL/SQL.
Explícito	Para las consultas que devuelven varias filas, los cursos explícitos se declaran y gestionan por el programador y se manipulan a través de sentencias específicas en las acciones ejecutables del bloque.

Oracle Server abre implícitamente un cursor para procesar cada sentencia SQL no asociada a un cursor declarado explícitamente. Mediante PL/SQL, puede hacer referencia a la mayoría de cursos implícitos recientes como el cursor SQL.

Operaciones de Cursosres Explícitos

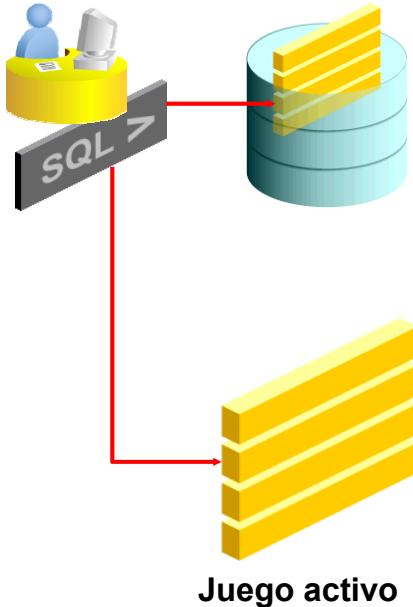


Tabla		
100	King	AD_PRES
101	Kochhar	AD_VP
102	De Haan	AD_VP
.	.	.
.	.	.
.	.	.
139	Seo	ST_CLERK
140	Patel	ST_CLERK
.	.	.

ORACLE

7 - 5

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Operaciones de Cursosres Explícitos

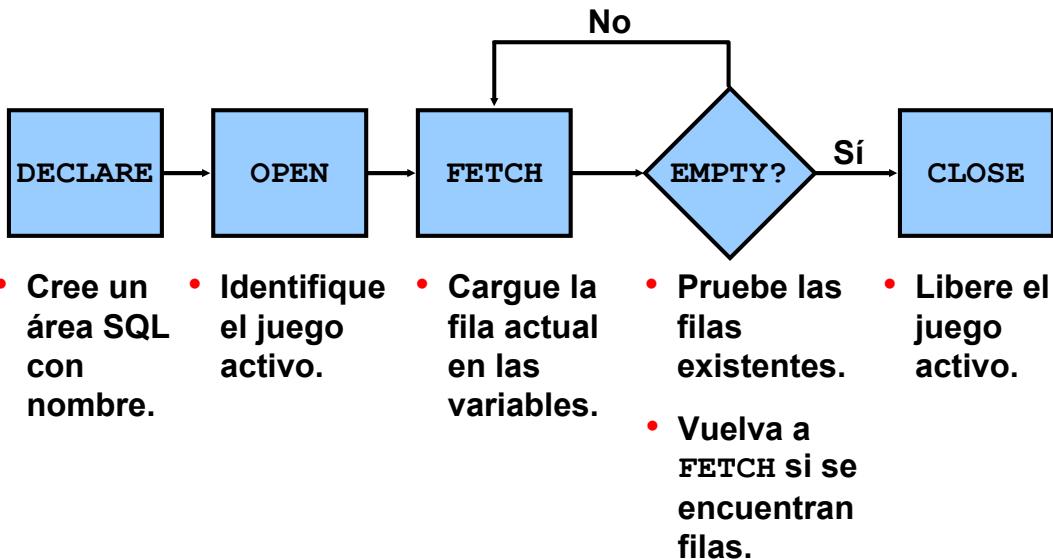
Declare cursosres explícitos en PL/SQL cuando tenga una sentencia SELECT que devuelva varias filas. Puede procesar todas las filas que devuelve la sentencia SELECT.

El juego de filas que devuelve una consulta de varias filas se denomina *juego activo*. Su tamaño es el número de filas que satisface los criterios de búsqueda. El diagrama de la diapositiva muestra cómo un cursor explícito “apunta” a la fila actual en el juego activo. Esto permite que el programa procese las filas de una en una.

Funciones del cursor explícito:

- Puede realizar un procesamiento fila por fila más allá de la primera fila devuelta por la consulta
- Realiza un seguimiento de la fila que se está procesando actualmente
- Permite al programador controlar manualmente los cursosres explícitos en el bloque PL/SQL

Control de Cursos Explícitos



ORACLE

7 - 6

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Control de Cursos Explícitos

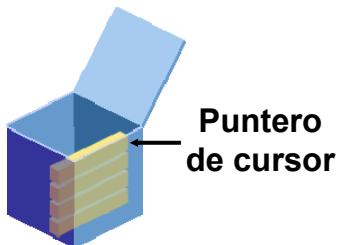
Ahora que tiene una comprensión conceptual de los cursos, revise los pasos para utilizarlos.

1. En la sección de declaraciones de un bloque PL/SQL, declare el cursor asignándole un nombre y definiendo la estructura de la consulta a la que se asocia.
2. Abra el cursor.
La sentencia OPEN ejecuta la consulta y enlaza cualquier variable a la que se haga referencia. Las filas identificadas por la consulta se denominan *juego activo* y pasan a estar disponibles para su recuperación.
3. Recupere los datos del cursor.
En el diagrama de flujo que se muestra en la diapositiva, después de cada recuperación realiza pruebas en el cursor en busca de una fila existente. Si no hay más filas que procesar, debe cerrar el cursor.
4. Cierre el cursor.
La sentencia CLOSE libera el juego de filas activo. Ya puede volver a abrir el cursor para establecer un nuevo juego activo.

Control de Cursos Explícitos

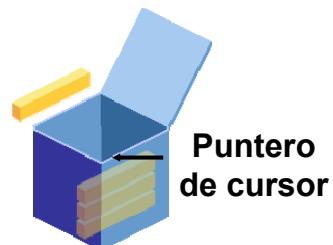
1

Abra el cursor.



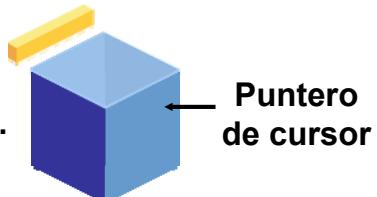
2

Recupere una fila.



3

Cierre el cursor.



ORACLE

Control de Cursos Explícitos (continuación)

Un programa PL/SQL abre un cursor, procesa las filas devueltas por una consulta y, a continuación, cierra el cursor. El cursor marca la posición actual en el juego activo.

1. La sentencia OPEN ejecuta la consulta asociada al cursor, identifica el juego activo y sitúa el cursor en la primera fila.
2. La sentencia FETCH recupera la fila actual y avanza el cursor a la siguiente fila hasta que no hay más filas o hasta que se satisface la condición especificada.
3. La sentencia CLOSE libera el cursor.

Agenda

- ¿Qué son los cursos explícitos?
- **Uso de cursos explícitos**
- Uso de cursos con parámetros
- Bloqueo de filas y referencia a la fila actual

ORACLE

7 - 8

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Declaración del Cursor

Sintaxis:

```
CURSOR cursor_name IS
    select_statement;
```

Ejemplos:

```
DECLARE
    CURSOR c_emp_cursor IS
        SELECT employee_id, last_name FROM employees
        WHERE department_id =30;
```

```
DECLARE
    v_locid NUMBER:= 1700;
    CURSOR c_dept_cursor IS
        SELECT * FROM departments
        WHERE location_id = v_locid;
    ...
```

ORACLE

7 - 9

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Declaración del Cursor

La sintaxis para declarar un cursor se muestra en la diapositiva. En la sintaxis:

cursor_name Es un identificador PL/SQL
select_statement Es una sentencia SELECT sin una cláusula INTO

El juego activo de un cursor está determinado por la sentencia SELECT de la declaración de cursores. Es obligatorio tener una cláusula INTO para una sentencia SELECT en PL/SQL. Sin embargo, tenga en cuenta que la sentencia SELECT de la declaración de cursores no puede tener una cláusula INTO. Esto se debe a que sólo define un cursor en la sección de declaraciones y no recupera ninguna fila en el cursor.

Nota

- No incluya la cláusula INTO en la declaración de cursores, ya que aparece más adelante en la sentencia FETCH.
- Si desea que las filas se procesen en una secuencia concreta, utilice la cláusula ORDER BY en la consulta.
- El cursor puede ser cualquier sentencia SELECT válida, incluidas uniones, subconsultas, etc.

Declaración del Cursor (continuación)

El cursor `c_emp_cursor` se declara para recuperar las columnas `employee_id` y `last_name` de los empleados que trabajan en el departamento con `department_id` 30.

El cursor `c_dept_cursor` se declara para recuperar todos los detalles del departamento con `location_id` 1700. Tenga en cuenta que se utiliza una variable mientras se declara el cursor.

Estas variables se consideran variables de enlace que deben ser visibles cuando declare el cursor.

Estas variables se examinan sólo una vez en el momento en que el cursor se abre. Ha aprendido que los cursosres explícitos se utilizan cuando ha recuperado y operado en varias filas en PL/SQL. Sin embargo, este ejemplo muestra que puede utilizar el cursor explícito aunque la sentencia SELECT devuelva sólo una fila.

Apertura del Cursor

```
DECLARE
  CURSOR c_emp_cursor IS
    SELECT employee_id, last_name FROM employees
    WHERE department_id =30;
  ...
BEGIN
  OPEN c_emp_cursor;
```



Apertura del Cursor

La sentencia OPEN ejecuta la consulta asociada al cursor, identifica el juego activo y sitúa el puntero del cursor en la primera fila. La sentencia OPEN se incluye en la sección ejecutable del bloque PL/SQL.

OPEN es una sentencia ejecutable que realiza las siguientes operaciones:

1. Asigna memoria dinámicamente para un área de contexto.
2. Analiza la sentencia SELECT
3. Enlaza las variables de entrada (establece los valores de las variables de entrada al obtener las direcciones de su memoria).
4. Identifica el juego activo (el juego de filas que cumplen los criterios de búsqueda). Las filas del juego activo no se recuperan en variables cuando se ejecuta la sentencia OPEN. Más bien, la sentencia FETCH recupera las filas del cursor a las variables.
5. Sitúa el puntero en la primera fila del juego activo.

Nota: si la consulta no devuelve ninguna fila cuando se abre el cursor, PL/SQL no emite ninguna excepción. Para conocer el número de filas devueltas con un cursor explícito, utilice el atributo <cursor_name>%ROWCOUNT.

Recuperación de Datos del Cursor

```

DECLARE
    CURSOR c_emp_cursor IS
        SELECT employee_id, last_name FROM employees
        WHERE department_id =30;
        v_empno employees.employee_id%TYPE;
        v_lname employees.last_name%TYPE;
BEGIN
    OPEN c_emp_cursor;
    FETCH c_emp_cursor INTO v_empno, v_lname;
    DBMS_OUTPUT.PUT_LINE( v_empno ||' '||v_lname);
END;
/

```

```

anonymous block completed
114  Raphaely

```

ORACLE

Recuperación de Datos del Cursor

La sentencia `FETCH` recupera las filas del cursor de una en una. Después de cada recuperación, el cursor avanza hasta la siguiente fila en el juego activo. Puede utilizar el atributo `%NOTFOUND` para determinar si se ha recuperado el juego activo completo.

Considere el ejemplo que se muestra en la diapositiva. Las dos variables, `empno` y `lname`, se declaran para contener los valores recuperados del cursor. Observe la sentencia `FETCH`.

Ha recuperado correctamente los valores del cursor a las variables. Sin embargo, hay seis empleados en el departamento 30, pero sólo se ha recuperado una fila. Para recuperar todas las filas, tiene que utilizar los bucles. En la siguiente diapositiva, verá cómo se utiliza un bucle para recuperar todas las filas.

La sentencia `FETCH` realiza las siguientes operaciones:

1. Lee los datos de la fila actual en las variables PL/SQL de salida.
2. Avanza el puntero hasta la siguiente fila del juego activo.

Recuperación de Datos del Cursor (continuación)

Puede incluir el mismo número de variables en la cláusula INTO de la sentencia FETCH que columnas hay en la sentencia SELECT; asegúrese de que los tipos de datos son compatibles. Haga coincidir cada variable para que correspondan con las columnas por la posición. Asimismo, puede definir un registro para el cursor y hacer referencia al registro en la cláusula FETCH INTO. Por último, pruebe si el cursor contiene las filas. Si una recuperación no adquiere ningún valor, no queda ninguna fila que procesar en el juego activo y no se registra ningún error.

Recuperación de Datos del Cursor

```

DECLARE
    CURSOR c_emp_cursor IS
        SELECT employee_id, last_name FROM employees
        WHERE department_id =30;
        v_empno employees.employee_id%TYPE;
        v_lname employees.last_name%TYPE;
BEGIN
    OPEN c_emp_cursor;
    LOOP
        FETCH c_emp_cursor INTO v_empno, v_lname;
        EXIT WHEN c_emp_cursor%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE( v_empno || ' ' || v_lname);
    END LOOP;
END ;
/

```

ORACLE

7 - 14

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Recuperación de Datos del Cursor (continuación)

Observe que se utiliza un LOOP simple para recuperar todas las filas. Además, el atributo de cursor %NOTFOUND se utiliza para probar la condición de salida. La salida del bloque PL/SQL es:

```

anonymous block completed
114  Raphaely
115  Khoo
116  Baida
117  Tobias
118  Himuro
119  Colmenares

```

Cierre del Cursor

```
...
LOOP
  FETCH c_emp_cursor INTO empno, lname;
  EXIT WHEN c_emp_cursor%NOTFOUND;
  DBMS_OUTPUT.PUT_LINE( v_empno || ' ' || v_lname);
END LOOP;
CLOSE c_emp_cursor;
END ;
/
```



Cierre del Cursor

La sentencia CLOSE desactiva el cursor, libera el área de contexto y anula la definición del juego activo. Cierre el cursor al terminar el procesamiento de la sentencia FETCH. Puede volver a abrir el cursor si es necesario. Un cursor se puede volver a abrir sólo si está cerrado. Si intenta recuperar datos de un cursor después de cerrarlo, se emitirá una excepción INVALID_CURSOR.

Nota: aunque se puede terminar el bloque PL/SQL sin cerrar los cursos, debe acostumbrarse a cerrar los cursos que declare explícitamente para liberar recursos.

Hay un límite máximo en el número de cursos abiertos por sesión, que determina el parámetro OPEN_CURSORS del archivo de parámetros de la base de datos. (OPEN_CURSORS = 50 por defecto).

Cursos y Registros

Procese las filas del juego activo recuperando los valores en un registro de PL/SQL.

```

DECLARE
    CURSOR c_emp_cursor IS
        SELECT employee_id, last_name FROM employees
        WHERE department_id =30;
        v_emp_record  c_emp_cursor%ROWTYPE;
BEGIN
    OPEN c_emp_cursor;
    LOOP
        FETCH c_emp_cursor INTO v_emp_record;
        EXIT WHEN c_emp_cursor%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE( v_emp_record.employee_id
                            ||' '||v_emp_record.last_name);
    END LOOP;
    CLOSE c_emp_cursor;
END;

```

7 - 16

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Cursos y Registros

Ya ha visto que puede definir los registros que tienen la estructura de columnas en una tabla. También puede definir un registro basado en la lista de columnas seleccionada en un cursor explícito. Esto es muy práctico para el procesamiento de las filas del juego activo, ya que puede simplemente recuperarlo en el registro. Por tanto, los valores de las filas se cargan directamente en los campos correspondientes del registro.

```

anonymous block completed
114 Raphaely
115 Khoo
116 Baida
117 Tobias
118 Himuro
119 Colmenares

```

Bucles FOR de Cursor

Sintaxis:

```
FOR record_name IN cursor_name LOOP
    statement1;
    statement2;
    . . .
END LOOP;
```

- El bucle FOR del cursor es un acceso directo para procesar cursores explícitos.
- Se produce una apertura, una recuperación, una salida y un cierre implícitos.
- El registro se declara implícitamente.

ORACLE

7 - 17

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Bucles FOR de Cursor

Ha aprendido a recuperar datos de cursores mediante bucles simples. Ahora aprenderá a utilizar un bucle FOR de cursor, que procesa las filas de un cursor explícito. Es un acceso directo puesto que se abre el cursor, se recupera una fila para cada iteración del bucle, el bucle sale cuando la última fila se procesa y el cursor se cierra automáticamente. El bucle se termina automáticamente al final de la iteración en la que se recupera la última fila.

En la sintaxis:

<i>record_name</i>	Es el nombre del registro declarado implícitamente
<i>cursor_name</i>	Es un identificador PL/SQL para el cursor declarado anteriormente

Instrucciones

- No declare el registro que controla el bucle; está declarado implícitamente.
- Pruebe los atributos de cursor durante el bucle si es necesario.
- Proporcione los parámetros para un cursor, si es necesario, entre paréntesis seguido del nombre del cursor en la sentencia FOR.

Bucles FOR de Cursor

```

DECLARE
    CURSOR c_emp_cursor IS
        SELECT employee_id, last_name FROM employees
        WHERE department_id =30;
BEGIN
    FOR emp_record IN c_emp_cursor
    LOOP
        DBMS_OUTPUT.PUT_LINE( emp_record.employee_id
        ||' '||emp_record.last_name);
    END LOOP;
END ;
/

```

```

anonymous block completed
114 Raphaely
115 Khoo
116 Baida
117 Tobias
118 Himuro
119 Colmenares

```

ORACLE

Bucles FOR de Cursor (continuación)

El ejemplo utilizado para demostrar el uso de un bucle simple para recuperar datos de los cursores se reescribe para utilizar el bucle FOR de cursor.

`emp_record` es el registro que se declara implícitamente. Puede acceder a los datos recuperados con este registro implícito (como se muestra en la diapositiva). Observe que no se declara ninguna variable que contenga los datos recuperados mediante la cláusula INTO. El código no tiene las sentencias OPEN y CLOSE para abrir y cerrar el cursor respectivamente.

Atributos de Cursor Explícito

Use atributos de cursor explícito para obtener información de estado sobre un cursor.

Atributo	Tipo	Descripción
%ISOPEN	Booleano	Se evalúa en TRUE si el cursor está abierto
%NOTFOUND	Booleano	Se evalúa en TRUE si la recuperación más reciente no devuelve una fila
%FOUND	Booleano	Se evalúa en TRUE si la recuperación más reciente devuelve una fila; complemento de %NOTFOUND
%ROWCOUNT	Número	Se evalúa en el número total de filas devueltas hasta el momento



Atributos de Cursor Explícito

Igual que ocurre con los cursores implícitos, existen cuatro atributos para obtener información sobre el estado de un cursor. Cuando se agregan al nombre de variable del cursor, estos atributos devuelven información útil sobre la ejecución de una sentencia de manipulación de cursor.

Nota: no puede hacer referencia a los atributos de cursor directamente en una sentencia SQL.

Atributo %ISOPEN

- Puede recuperar filas sólo cuando el cursor esté abierto.
- Utilice el atributo de cursor %ISOPEN antes de realizar una recuperación para probar si el cursor está abierto.

Ejemplo:

```
IF NOT c_emp_cursor%ISOPEN THEN
    OPEN c_emp_cursor;
END IF;
LOOP
    FETCH c_emp_cursor...
```

ORACLE

7 - 20

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Atributo %ISOPEN

- Puede recuperar filas sólo cuando el cursor esté abierto. Utilice el atributo de cursor %ISOPEN para determinar si el cursor está abierto.
- Recupere las filas en un bucle. Utilice los atributos de cursor para determinar cuándo salir del bucle.
- Utilice el atributo de cursor %ROWCOUNT para lo siguiente:
 - Procesar un número de filas exacto.
 - Recuperar las filas en un bucle y determinar cuándo salir del bucle.

Nota: %ISOPEN devuelve el estado del cursor: TRUE si está abierto y FALSE si no lo está.

%ROWCOUNT y %NOTFOUND: Ejemplo

```

DECLARE
    CURSOR c_emp_cursor IS SELECT employee_id,
        last_name FROM employees;
    v_emp_record  c_emp_cursor%ROWTYPE;
BEGIN
    OPEN c_emp_cursor;
    LOOP
        FETCH c_emp_cursor INTO v_emp_record;
        EXIT WHEN c_emp_cursor%ROWCOUNT > 10 OR
            c_emp_cursor%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE( v_emp_record.employee_id
            ||' '||v_emp_record.last_name);
    END LOOP;
    CLOSE c_emp_cursor;
END ; /

```

anonymous block completed

174 Abel
166 Ande
130 Atkinson
105 Austin
204 Baer
116 Baida
167 Banda
172 Bates
192 Bell
151 Bernstein

ORACLE

7 - 21

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

%ROWCOUNT y %NOTFOUND: Ejemplo

En el ejemplo de la diapositiva se recuperan los primeros 10 empleados uno por uno. Este ejemplo muestra cómo los atributos %ROWCOUNT y %NOTFOUND se pueden utilizar para condiciones de salida en un bucle.

Bucles FOR de Cursor mediante Subconsultas

No es necesario declarar el cursor.

```
BEGIN
  FOR emp_record IN (SELECT employee_id, last_name
    FROM employees WHERE department_id =30)
  LOOP
    DBMS_OUTPUT.PUT_LINE( emp_record.employee_id
      ||' '||emp_record.last_name);
  END LOOP;
END ;
/
```

```
anonymous block completed
114 Raphaely
115 Khoo
116 Baida
117 Tobias
118 Himuro
119 Colmenares
```

7 - 22

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Bucles FOR de Cursor mediante Subconsultas

Observe que no hay ninguna sección de declaraciones en este bloque PL/SQL. La diferencia entre los bucles FOR de cursor mediante subconsultas y el bucle FOR de cursor radica en la declaración de cursores. Si escribe bucles FOR de cursor mediante subconsultas, no tiene que declarar el cursor en la sección de declaraciones. Tiene que proporcionar la sentencia SELECT que determina el juego activo del propio bucle.

El ejemplo utilizado para ilustrar un bucle FOR de cursor se reescribe para ilustrar un bucle FOR de cursor mediante subconsultas.

Nota: no puede hacer referencia a atributos de cursor explícito si utiliza una subconsulta en un bucle FOR de cursor, ya que no puede asignar un nombre explícito al cursor.

Agenda

- ¿Qué son los cursos explícitos?
- Uso de cursos explícitos
- **Uso de cursos con parámetros**
- Bloqueo de filas y referencia a la fila actual

ORACLE

7 - 23

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Cursos con Parámetros

Sintaxis:

```
CURSOR cursor_name
  [ (parameter_name datatype, . . . ) ]
IS
  select_statement;
```

- Transfiera los valores de parámetro a un cursor cuando el cursor esté abierto y la consulta ejecutada.
- Abra un cursor explícito varias veces con un juego activo diferente cada vez.

```
OPEN cursor_name (parameter_value, . . . . .) ;
```

7 - 24

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Cursos con Parámetros

Puede transferir parámetros a un cursor. Esto significa que puede abrir y cerrar un cursor explícito varias veces en un bloque, devolviendo un juego activo diferente en cada ocasión. Para cada ejecución, el cursor anterior se cierra y se vuelve a abrir con un nuevo juego de parámetros.

Cada parámetro formal de la declaración de cursos debe tener un parámetro real correspondiente en la sentencia OPEN. Los tipos de dato de los parámetros son los mismos que los de las variables escalares, aunque no les proporciona ningún tamaño. Los nombres de parámetros son referencias en la expresión de consulta del cursor.

En la sintaxis:

<i>cursor_name</i>	Es un identificador PL/SQL para el cursor declarado
<i>parameter_name</i>	Es el nombre de un parámetro
<i>datatype</i>	Es el tipo de dato escalar del parámetro
<i>select_statement</i>	Es una sentencia SELECT sin una cláusula INTO

La notación de parámetros no ofrece mayor funcionalidad; simplemente le permite especificar los valores de entrada de forma sencilla y clara. Esto es especialmente útil cuando se hace referencia al mismo cursor en repetidas ocasiones.

Cursos con Parámetros

```

DECLARE
  CURSOR  c_emp_cursor (deptno NUMBER) IS
    SELECT employee_id, last_name
    FROM   employees
    WHERE  department_id = deptno;
    ...
BEGIN
  OPEN c_emp_cursor (10);
  ...
  CLOSE c_emp_cursor;
  OPEN c_emp_cursor (20);
  ...

```

```

anonymous block completed
200 Whalen
201 Hartstein
202 Fay

```

ORACLE

7 - 25

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Cursos con Parámetros (continuación)

Los tipos de dato de los parámetros son los mismos que los de las variables escalares, aunque no les proporciona ningún tamaño. Los nombres de parámetros son referencias en la consulta del cursor. En el siguiente ejemplo, se declara y se define un cursor con un parámetro:

```

DECLARE
  CURSOR c_emp_cursor(deptno NUMBER) IS SELECT ...

```

Las siguientes sentencias abren el cursor y devuelven diferentes juegos activos:

```

OPEN c_emp_cursor(10);
OPEN c_emp_cursor(20);

```

Puede transferir parámetros al cursor utilizado en un bucle FOR de cursor:

```

DECLARE
  CURSOR c_emp_cursor(p_deptno NUMBER, p_job VARCHAR2) IS
    SELECT ...
BEGIN
  FOR emp_record IN c_emp_cursor(10, 'Sales') LOOP ...

```

Agenda

- ¿Qué son los cursos explícitos?
- Uso de cursos explícitos
- Uso de cursos con parámetros
- Bloqueo de filas y referencia a la fila actual

ORACLE

7 - 26

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Cláusula FOR UPDATE

Sintaxis:

```
SELECT ...
FROM ...
FOR UPDATE [OF column_reference] [NOWAIT | WAIT n];
```

- Utilice el bloqueo explícito para denegar el acceso a otras sesiones durante una transacción.
- Bloquee las filas *antes* de la actualización o la supresión.

7 - 27

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Cláusula FOR UPDATE

Si hay varias sesiones para una única base de datos, existe la posibilidad de que las filas de una tabla concreta se hayan actualizado después de abrir el cursor. Verá los datos actualizados sólo cuando vuelva a abrir el cursor. Por tanto, es mejor tener bloqueos en las filas antes de actualizar o suprimir filas. Puede bloquear las filas con la cláusula FOR UPDATE de la consulta de cursor.

En la sintaxis:

<i>column_reference</i>	Es una columna de la tabla con la que se realiza la consulta (también se puede utilizar una lista de columnas).
NOWAIT	Devuelve un error de Oracle Server si las filas se bloquean en otra sesión.

La cláusula FOR UPDATE es la última de una sentencia SELECT, incluso después de ORDER BY (si existe). Al consultar varias tablas, puede utilizar la cláusula FOR UPDATE para limitar el bloqueo de filas a tablas concretas. FOR UPDATE OF *col_name (s)* bloquea las filas sólo en las tablas que contienen *col_name (s)*.

Cláusula FOR UPDATE (continuación)

La sentencia SELECT . . . FOR UPDATE identifica las filas que se actualizarán o suprimirán y, a continuación, bloquea cada fila del juego de resultados. Resulta útil cuando desea basar una actualización en los valores existentes de una fila. En ese caso, se debe asegurar de que la fila no se ha cambiado en otra sesión antes de la actualización.

La palabra clave opcional NOWAIT indica a Oracle Server que no espere si otro usuario ha bloqueado las filas solicitadas. Se devuelve inmediatamente el control al programa para que pueda realizar otro trabajo antes de intentar de nuevo adquirir el bloqueo. Si omite la palabra clave NOWAIT, Oracle Server espera hasta que las filas están disponibles.

Ejemplo:

```
DECLARE
    CURSOR c_emp_cursor IS
        SELECT employee_id, last_name, FROM employees
        WHERE department_id = 80 FOR UPDATE OF salary NOWAIT;
    ...

```

Si Oracle Server no puede adquirir los bloqueos de las filas que necesita en una operación SELECT FOR UPDATE, esperará de manera indefinida. Utilice NOWAIT para manejar estas situaciones. Si las filas se bloquean en otra sesión y ha especificado NOWAIT, al abrir el cursor se producirá un error. Puede intentar abrir el cursor más adelante. Puede utilizar WAIT en lugar de NOWAIT, especificar el número de segundos que esperar y determinar si las filas están desbloqueadas. Si las filas aún están bloqueadas después de n segundos, devolverá un error.

No es obligatorio que la cláusula FOR UPDATE OF haga referencia a una columna, pero se recomienda para una mejor legibilidad y mantenimiento.

Cláusula WHERE CURRENT OF

Sintaxis:

```
WHERE CURRENT OF cursor ;
```

- Utilice los cursores para actualizar o suprimir la fila actual.
- Incluya la cláusula FOR UPDATE en la consulta de cursor para bloquear primero las filas.
- Utilice la cláusula WHERE CURRENT OF para hacer referencia a la fila actual de un cursor explícito.

```
UPDATE employees  
SET salary = ...  
WHERE CURRENT OF c_emp_cursor;
```

Cláusula WHERE CURRENT OF

La cláusula WHERE CURRENT OF se utiliza junto con la cláusula FOR UPDATE para hacer referencia a la fila actual de un cursor explícito. La cláusula WHERE CURRENT OF se utiliza en la sentencia UPDATE o DELETE, mientras que la cláusula FOR UPDATE se especifica en la declaración de cursores. Puede utilizar la combinación para la actualización y supresión de la fila actual de la tabla de la base de datos correspondiente. Esto permite aplicar actualizaciones y supresiones a la fila que se consulta actualmente sin tener que hacer referencia explícitamente al identificador de fila. Debe incluir la cláusula FOR UPDATE en la consulta de cursor para que las filas se bloquen en OPEN.

En la sintaxis:

cursor Es el nombre de un cursor declarado (el cursor se debe haber declarado con la cláusula FOR UPDATE).

Prueba

Los cursos implícitos se declaran mediante PL/SQL de forma implícita para todas las sentencias DML y PL/SQL SELECT.

Oracle Server abre de forma implícita un cursor para procesar cada sentencia SQL que no está asociada a un cursor declarado de forma explícita.

- a. Verdadero
- b. Falso

ORACLE

7 - 30

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Respuesta: a

Resumen

En esta lección debe haber aprendido lo siguiente:

- Distinguir los tipos de cursor:
 - Los cursores implícitos se utilizan para todas las sentencias DML y las consultas de una sola fila.
 - Los cursores explícitos se utilizan para las consultas de cero, una o más filas.
- Crear y manejar cursores explícitos
- Utilizar bucles simples y bucles FOR de cursor para manejar varias filas en los cursores
- Evaluar el estado del cursor mediante los atributos de cursor
- Utilizar las cláusulas FOR UPDATE y WHERE CURRENT OF para actualizar o suprimir la fila actual recuperada



Resumen

Oracle Server utiliza áreas de trabajo para ejecutar sentencias SQL y almacenar la información de procesamiento. Puede utilizar una construcción PL/SQL denominada *cursor* para asignar un nombre a un área de trabajo y acceder a la información almacenada. Hay dos tipos de cursores: implícitos y explícitos. PL/SQL declara implícitamente un cursor para todas las sentencias de manipulación de datos SQL, incluidas las consultas que devuelven sólo una fila. Para las consultas que devuelven varias filas, debe declarar explícitamente un cursor para procesar las filas de forma individual.

Cada cursor explícito y variable de cursor tiene cuatro atributos: %FOUND, %ISOPEN, %NOTFOUND y %ROWCOUNT. Cuando se agregan al nombre de la variable del cursor, estos atributos devuelven información útil sobre la ejecución de una sentencia SQL. Puede utilizar los atributos de cursor en sentencias de procedimiento, pero no en sentencias SQL.

Utilice bucles simples o bucles FOR de cursor para trabajar en varias filas recuperadas por el cursor. Si utiliza bucles sencillos, tiene que abrir, recuperar y cerrar el cursor; sin embargo, los bucles FOR de cursor lo realizan implícitamente. Si actualiza o suprime filas, bloquee las filas mediante una cláusula FOR UPDATE. Esto garantiza que los datos que utiliza no se actualicen en otra sesión después de que abra el cursor. Utilice una cláusula WHERE CURRENT OF junto con la cláusula FOR UPDATE para hacer referencia a la fila actual recuperada por el cursor.

Práctica 7: Visión General

En esta práctica se abordan los siguientes temas:

- Declaración y uso de cursos explícitos para consultar las filas de una tabla
- Uso de un bucle FOR de cursor
- Aplicación de atributos de cursor para probar el estado del cursor
- Declaración y uso de cursos con parámetros
- Uso de las cláusulas FOR UPDATE y WHERE CURRENT OF



Práctica 7: Visión General

En esta práctica se aplican los conocimientos sobre cursos para procesar un número de filas de una tabla y llenar otra tabla con los resultados mediante un bucle FOR de cursor. También escribirá un cursor con parámetros.

Manejo de Excepciones

8

ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Objetivos

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

- Definir excepciones PL/SQL
- Reconocer excepciones no tratadas
- Enumerar y utilizar diferentes tipos de manejadores de excepciones PL/SQL
- Detectar errores inesperados
- Describir el efecto de la propagación de excepciones en bloques anidados
- Personalizar mensajes de excepción PL/SQL



Objetivos

Ha aprendido a escribir bloques PL/SQL con una sección de declaraciones y una sección ejecutable. Todos los códigos SQL y PL/SQL que se tienen que ejecutar están escritos en el bloque ejecutable.

Hasta ahora se ha asumido que el código funciona bien si presta atención a los errores de tiempo de compilación. Sin embargo, puede que el código provoque algunos errores inesperados en tiempo de ejecución. En esta lección, aprenderá a tratar estos errores en el bloque PL/SQL.

Agenda

- Descripción de las excepciones PL/SQL
- Detección de excepciones

ORACLE

8 - 3

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

¿Qué Es una Excepción?

```

DECLARE
    v_lname VARCHAR2(15);
BEGIN
    SELECT last_name INTO v_lname
    FROM employees
    WHERE first_name='John';
    DBMS_OUTPUT.PUT_LINE ('John''s last name is : ' ||v_lname);
END;

```

Results Script Output Explain Autotrace DBMS Output OWA Output

Error starting at line 3 in command:
 DECLARE
 v_lname VARCHAR2(15);
 BEGIN
 SELECT last_name INTO v_lname FROM employees WHERE
 first_name='John';
 DBMS_OUTPUT.PUT_LINE ('John''s last name is : ' ||v_lname);
 END;
 Error report:
 ORA-01422: exact fetch returns more than requested number of rows
 ORA-06512: at line 4
 01422. 00000 - "exact fetch returns more than requested number of rows"
 *Cause: The number specified in exact fetch is less than the rows returned.
 *Action: Rewrite the query or change number of rows requested

ORACLE

8 - 4

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

¿Qué Es una Excepción?

Considere el ejemplo que se muestra en la diapositiva. No hay errores de sintaxis en el código, lo que significa que puede ejecutar correctamente el bloque anónimo. La sentencia SELECT del bloque recupera el apellido de John.

Sin embargo, aparecerá el siguiente informe de error al ejecutar el código:

```

Error report:  

ORA-01422: exact fetch returns more than requested number of rows  

ORA-06512: at line 4  

01422. 00000 - "exact fetch returns more than requested number of rows"  

*Cause: The number specified in exact fetch is less than the rows returned.  

*Action: Rewrite the query or change number of rows requested

```

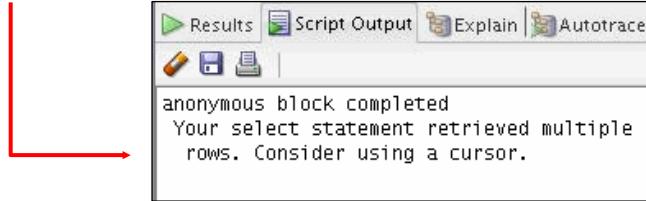
El código no funciona como se esperaba. Se esperaba que la sentencia SELECT recuperara sólo una fila; sin embargo, recupera varias filas. Estos errores que se producen en tiempo de ejecución se denominan *excepciones*. Cuando se produce una excepción, el bloque PL/SQL se termina. Puede manejar estas excepciones en el bloque PL/SQL.

Manejo de Excepciones: Ejemplo

```

DECLARE
    v_lname VARCHAR2(15);
BEGIN
    SELECT last_name INTO v_lname
    FROM employees
    WHERE first_name='John';
    DBMS_OUTPUT.PUT_LINE ('John''s last name is : ' ||v_lname);
EXCEPTION
    WHEN TOO_MANY_ROWS THEN
        DBMS_OUTPUT.PUT_LINE (' Your select statement retrieved
multiple rows. Consider using a cursor.');
END;
/

```



ORACLE

8 - 5

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Manejo de Excepciones: Ejemplo

Ha aprendido a escribir bloques PL/SQL con una sección de declaraciones (que empieza por la palabra clave DECLARE) y una sección ejecutable (que empieza y termina con las palabras clave BEGIN y END, respectivamente).

Para el manejo de la excepción, incluya otra sección opcional denominada *sección de excepciones*.

- Esta sección empieza por la palabra clave EXCEPTION.
- Si está presente, ésta debe ser la última sección en un bloque PL/SQL.

Ejemplo

En el ejemplo de la diapositiva, se vuelve a escribir el código de la diapositiva anterior para manejar la excepción producida. La salida del bloque también se muestra en la diapositiva.

Al agregar la sección EXCEPTION del código, el programa PL/SQL no termina repentinamente. Cuando se emite la excepción, el control pasa a la sección de excepciones y se ejecutan todas sus sentencias. El bloque PL/SQL termina de forma normal y correcta.

Descripción de Excepciones con PL/SQL

- Una excepción es un error PL/SQL que se emite durante la ejecución del programa.
- Una excepción se puede emitir:
 - Implícitamente por Oracle Server
 - Explícitamente por el programa
- Una excepción se puede manejar:
 - Detectándola con un manejador
 - Propagándola al entorno de llamada



8 - 6

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

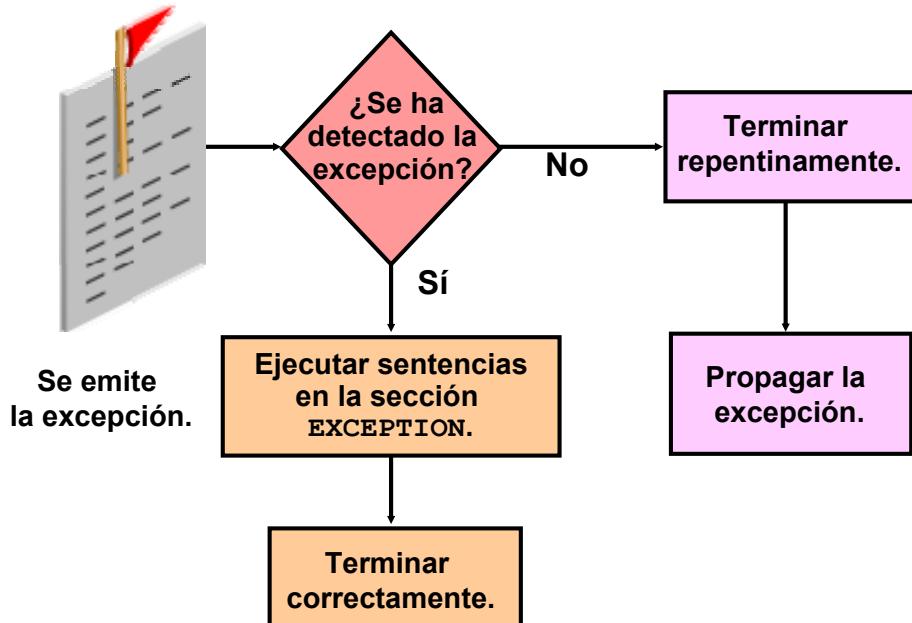
Descripción de Excepciones con PL/SQL

Una excepción es un error en PL/SQL que se emite durante la ejecución de un bloque. Un bloque siempre termina cuando PL/SQL emite una excepción, pero puede especificar un manejador de excepciones para realizar las acciones finales antes de que el bloque termine.

Dos Métodos para Emitir una Excepción

- Se produce un error de Oracle y la excepción relacionada se emite automáticamente. Por ejemplo, si se produce el error ORA-01403 cuando no se ha recuperado ninguna fila de la base de datos en una sentencia SELECT, PL/SQL emite la excepción NO_DATA_FOUND. Estos errores se convierten en excepciones predefinidas.
- Dependiendo de la funcionalidad de negocio que implante el programa, puede que tenga que emitir explícitamente una excepción. Genere una excepción explícitamente emitiendo la sentencia RAISE dentro del bloque. La excepción emitida puede ser definida por el usuario o predefinida. Hay también algunos errores de Oracle no predefinidos. Estos errores no son ningún error estándar de Oracle que no esté predefinido. Puede declarar excepciones explícitamente y asociarlas a los errores de Oracle no predefinidos.

Manejo de Excepciones



ORACLE

8 - 7

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Manejo de Excepciones

Detección de una Excepción

Incluya una sección EXCEPTION en el programa PL/SQL para detectar excepciones. Si la excepción se emite en la sección ejecutable del bloque, el procesamiento se diversifica en el manejador de excepciones correspondiente de la sección de excepciones del bloque. Si PL/SQL maneja de forma correcta la excepción, no se propaga al bloque delimitador o al entorno de llamada. El bloque PL/SQL termina de forma correcta.

Propagación de una Excepción

Si la excepción se emite en la sección ejecutable del bloque y no hay ningún manejador de excepciones correspondiente, el bloque PL/SQL termina con fallos y la excepción se propaga a un bloque delimitador o al entorno de llamada. El entorno de llamada puede ser cualquier aplicación (como SQL*Plus, que llama al programa PL/SQL).

Tipos de Excepción

- Predefinida de Oracle Server
 - No predefinida de Oracle Server
- } Se emite implícitamente
-
- Definida por el usuario
- Se emite explícitamente

8 - 8

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Tipos de Excepción

Hay tres tipos de excepciones.

Excepción	Descripción	Instrucciones para el manejo
Error predefinido del servidor de Oracle	Uno de los aproximadamente 20 errores que se producen con más frecuencia en código PL/SQL	No es necesario que declare estas excepciones. Están predefinidas por el servidor de Oracle y se producen implícitamente.
Error no predefinido del servidor de Oracle	Cualquier otro error estándar del servidor de Oracle	Debe declararlos en la sección de declaraciones; el servidor de Oracle lo produce implícitamente y puede capturar el error en el manejador de excepciones.
Error definido por el usuario	Condición que el desarrollador determina que es anormal	Debe declararlos en la sección de declaraciones y producirlos explícitamente.

Nota: algunas herramientas de la aplicación con PL/SQL del cliente (como Oracle Developer Forms) tienen sus propias excepciones.

Agenda

- Descripción de las excepciones PL/SQL
- Detección de excepciones

ORACLE

8 - 9

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Sintaxis para Detectar Excepciones

```

EXCEPTION
  WHEN exception1 [OR exception2 . . .] THEN
    statement1;
    statement2;
    . . .
  [WHEN exception3 [OR exception4 . . .] THEN
    statement1;
    statement2;
    . . .]
  [WHEN OTHERS THEN
    statement1;
    statement2;
    . . .]

```



Sintaxis para Detectar Excepciones

Puede detectar cualquier error incluido un manejador correspondiente dentro de la sección de manejo de excepciones del bloque PL/SQL. Cada manejador está formado por una cláusula WHEN, que especifica un nombre de excepción, seguido de una secuencia de sentencias que se ejecutan cuando se emite dicha excepción.

Puede incluir cualquier número de manejadores dentro de una sección EXCEPTION para manejar excepciones específicas. Sin embargo, no puede tener varios manejadores para una única excepción. La sintaxis de detección de excepciones incluye los siguientes elementos:

<i>exception</i>	Es el nombre estándar de una excepción predefinida o el nombre de una excepción definida por usuario declarada en la sección de declaraciones
<i>statement</i>	Es una o más sentencias PL/SQL o SQL
OTHERS	Es una cláusula de manejo de excepciones opcional que detecta cualquier excepción que no se ha manejado explícitamente

Sintaxis de Detección de Excepciones (continuación)

Manejador de Excepciones WHEN OTHERS

Como ya se ha indicado, la sección de manejo de excepciones detecta sólo aquellas excepciones que se especifican.

Para detectar las excepciones que no se especifican, utilice el manejador de excepciones OTHERS. Esta opción detecta cualquier excepción que aún no se ha manejado. Por este motivo, si se utiliza el manejador OTHERS, debe ser el último manejador de excepciones que se defina.

Por ejemplo:

```
WHEN NO_DATA_FOUND THEN  
    statement1;  
    ...  
WHEN TOO_MANY_ROWS THEN  
    statement1;  
    ...  
WHEN OTHERS THEN  
    statement1;
```

Ejemplo

Considere el ejemplo anterior. Si el programa emite la excepción NO_DATA_FOUND, se ejecutan las sentencias en el manejador correspondiente. Si se emite la excepción TOO_MANY_ROWS, se ejecutan las sentencias en el manejador correspondiente. Sin embargo, si se emite alguna otra excepción, se ejecutan las sentencias en el manejador de excepciones OTHERS.

El manejador OTHERS detecta todas las excepciones que aún no se han detectado. Algunas herramientas de Oracle tienen sus propias excepciones predefinidas que puede emitir para provocar eventos en la aplicación. El manejador OTHERS también detecta estas excepciones.

Instrucciones para la Detección de Excepciones

- La palabra clave EXCEPTION inicia la sección de manejo de excepciones.
- Se permiten varios manejadores de excepciones.
- Sólo se procesa un manejador antes de dejar el bloque.
- WHEN OTHERS es la última cláusula.

ORACLE

8 - 12

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Instrucciones para la Detección de Excepciones

- Inicie la sección de manejo de excepciones del bloque con la palabra clave EXCEPTION.
- Defina varios manejadores de excepciones, cada uno con su propio juego de acciones, para el bloque.
- Cuando se produce una excepción, PL/SQL procesa sólo un manejador antes de dejar el bloque.
- Ponga la cláusula OTHERS después de todas las cláusulas de manejo de excepciones.
- Sólo puede tener una cláusula OTHERS.
- Las excepciones no pueden aparecer en sentencias de asignación o sentencias SQL.

Detección de Errores Predefinidos de Oracle Server

- Referencia al nombre predefinido de la rutina de manejo de excepciones.
- Excepciones predefinidas de ejemplo:
 - NO_DATA_FOUND
 - TOO_MANY_ROWS
 - INVALID_CURSOR
 - ZERO_DIVIDE
 - DUP_VAL_ON_INDEX



8 - 13

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Detección de Errores Predefinidos de Oracle Server

Detecte un error predefinido de Oracle Server haciendo referencia a su nombre predefinido dentro de la rutina de manejo de excepciones correspondiente.

Para obtener una lista completa de excepciones predefinidas, consulte *PL/SQL User's Guide and Reference* (Guía del Usuario y Referencia PL/SQL).

Nota: PL/SQL declara excepciones predefinidas en el paquete STANDARD.

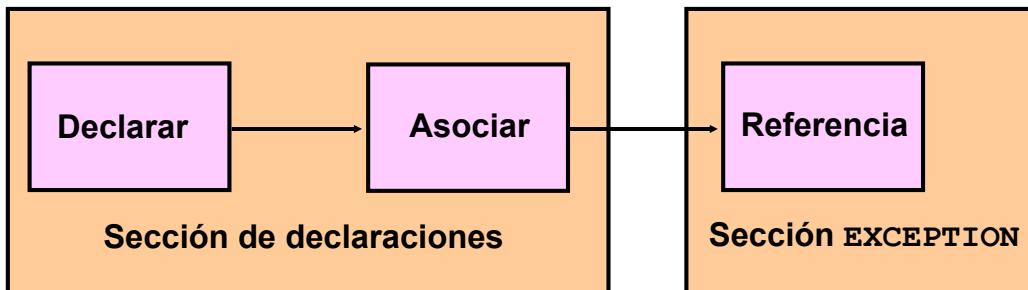
Excepciones Predefinidas

Nombre de Excepción	Número de error de servidor de Oracle	Descripción
ACCESS_INTO_NULL	ORA-06530	Se ha intentado asignar valores a los atributos de un objeto no inicializado.
CASE_NOT_FOUND	ORA-06592	Ninguna de las opciones de las cláusulas WHEN de una sentencia CASE está seleccionada y no hay ninguna cláusula ELSE.
COLLECTION_IS_NULL	ORA-06531	Se ha intentado aplicar métodos de recopilación distintos de EXISTS a una tabla o VARRAY anidada no inicializada.
CURSOR_ALREADY_OPEN	ORA-06511	Se ha intentado abrir un cursor ya abierto.
DUP_VAL_ON_INDEX	ORA-00001	Se ha intentado insertar un valor duplicado.
INVALID_CURSOR	ORA-01001	Se ha producido una operación de cursor no válida.
INVALID_NUMBER	ORA-01722	Fallo de conversión de cadena de caracteres a número.
LOGIN_DENIED	ORA-01017	Conexión a servidor de Oracle con un nombre de usuario o contraseña no válidos.
NO_DATA_FOUND	ORA-01403	SELECT de una sola fila no ha devuelto ningún dato.
NOT_LOGGED_ON	ORA-01012	El programa PL/SQL emite una llamada de base de datos sin conectarse al servidor de Oracle.
PROGRAM_ERROR	ORA-06501	PL/SQL tiene un problema interno.
ROWTYPE_MISMATCH	ORA-06504	La variable de cursor del host y la variable de cursor de PL/SQL de una asignación tienen tipos de retorno incompatibles.

Excepciones Predefinidas (continuación)

Nombre de Excepción	Número de error de servidor de Oracle	Descripción
STORAGE_ERROR	ORA-06500	PL/SQL se ha quedado sin memoria o está corrupta.
SUBSCRIPT_BEYOND_COUNT	ORA-06533	Se hace referencia a una tabla anidada o elemento VARRAY utilizando un número de índice mayor que el número de elementos en la recopilación.
SUBSCRIPT_OUTSIDE_LIMIT	ORA-06532	Se hace referencia a una tabla anidada o elemento VARRAY utilizando un número de índice que queda fuera del rango válido (-1 por ejemplo).
SYS_INVALID_ROWID	ORA-01410	La conversión de una cadena de caracteres en ROWID universal falla porque la cadena de caracteres no representa una ROWID válida.
TIMEOUT_ON_RESOURCE	ORA-00051	Se ha producido un timeout mientras el servidor de Oracle esperaba un recurso.
TOO_MANY_ROWS	ORA-01422	SELECT de una sola fila ha devuelto varias filas.
VALUE_ERROR	ORA-06502	Se ha producido un error aritmético, de conversión, de truncamiento o de restricción de tamaño.
ZERO_DIVIDE	ORA-01476	Se ha intentado dividir entre cero.

Detección de Errores No Predefinidos de Oracle Server



Asigne un nombre a la excepción.

Use PRAGMA EXCEPTION_INIT.

Maneje la excepción emitida.

ORACLE

Detección de Errores No Predefinidos de Oracle Server

Las excepciones no predefinidas son similares a las excepciones predefinidas; sin embargo, no se definen como excepciones PL/SQL en Oracle Server. Son errores estándar de Oracle. Puede crear excepciones con los errores estándar de Oracle mediante la función PRAGMA EXCEPTION_INIT. Estas excepciones se denominan excepciones no predefinidas.

Puede detectar un error no predefinido de Oracle Server declarándolo primero. Se emite la excepción declarada implícitamente. En PL/SQL, PRAGMA EXCEPTION_INIT indica al compilador que asocie el nombre de excepción a un número de error de Oracle. Esta acción permite hacer referencia a cualquier excepción interna mediante el nombre y escribir un manejador específico para la excepción.

Nota: PRAGMA (también denominado *pseudoinstrucciones*) es la palabra clave que indica que la sentencia es una directiva de compilador que no se procesa cuando se ejecuta el bloque PL/SQL. Más bien, indica al compilador PL/SQL que interprete todas las incidencias del nombre de excepción en el bloque como el número de error asociado de Oracle Server.

Detección de Errores No Predefinidos: Ejemplo

Para detectar el error de Oracle Server 01400 (“cannot insert NULL”):

```

DECLARE
    e_insert_excep EXCEPTION;
    PRAGMA EXCEPTION_INIT(e_insert_excep, -01400);
BEGIN
    INSERT INTO departments
    (department_id, department_name) VALUES (280, NULL);
EXCEPTION
    WHEN e_insert_excep THEN
        DBMS_OUTPUT.PUT_LINE('INSERT OPERATION FAILED');
        DBMS_OUTPUT.PUT_LINE(SQLERRM);
END;
/

```

8 - 17

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

ORACLE

Detección de Errores No Predefinidos: Ejemplo

En el ejemplo se muestran los tres pasos asociados a la detección de un error no predefinido:

1. Declare el nombre de la excepción en la sección de declaraciones, con la sintaxis:

```
exception      EXCEPTION;
```

En esta sintaxis, *exception* es el nombre de la excepción.

2. Asocie la excepción declarada al número de error de Oracle Server estándar con la función PRAGMA EXCEPTION_INIT. Utilice la siguiente sintaxis:

```
PRAGMA EXCEPTION_INIT(exception, error_number);
```

En la sintaxis, *exception* es la excepción declarada anteriormente y *error_number* es un número de error de Oracle Server estándar.

3. Haga referencia a la excepción declarada en la rutina de manejo de excepciones correspondiente.

Ejemplo

El ejemplo de la diapositiva intenta insertar el valor NULL para la columna *department_name* de la tabla *departments*. Sin embargo, la operación no se ha realizado correctamente porque *department_name* es una columna NOT NULL. Observe la siguiente línea del ejemplo:

```
DBMS_OUTPUT.PUT_LINE(SQLERRM);
```

La función SQLERRM se utiliza para recuperar el mensaje de error. Aprenderá más sobre SQLERRM en las siguientes diapositivas.

Funciones para la Detección de Excepciones

- **SQLCODE:** devuelve el valor numérico del código de error
- **SQLERRM:** devuelve el mensaje asociado al número de error

ORACLE

8 - 18

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Funciones para la Detección de Excepciones

Cuando se produce una excepción, puede identificar el código de error o el mensaje de error asociado utilizando dos funciones. Según los valores del código o el mensaje, puede decidir las acciones posteriores que realizar.

SQLCODE devuelve el número de error de Oracle para excepciones internas. **SQLERRM** devuelve el mensaje asociado al número de error.

Función	Descripción
SQLCODE	Devuelve el valor numérico del código de error (puede asignarlo a una variable NUMBER.)
SQLERRM	Devuelve los datos de caracteres que contienen el mensaje asociado al número de error

Valores de **SQLCODE**: Ejemplos

Valor SQLCODE	Descripción
0	No se ha encontrado ninguna excepción
1	Excepción definida por el usuario
+100	Excepción NO_DATA_FOUND
<i>número negativo</i>	Otro número de error del servidor de Oracle

Funciones para la Detección de Excepciones

```

DECLARE
    error_code      NUMBER;
    error_message   VARCHAR2 (255);
BEGIN
    ...
EXCEPTION
    ...
    WHEN OTHERS THEN
        ROLLBACK;
        error_code := SQLCODE ;
        error_message := SQLERRM ;
        INSERT INTO errors (e_user, e_date, error_code,
        error_message) VALUES (USER,SYSDATE,error_code,
        error_message);
    END;
/

```

ORACLE

8 - 19

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Funciones para la Detección de Excepciones (continuación)

Cuando se detecta una excepción en el manejador de excepciones WHEN OTHERS puede utilizar un juego de funciones genéricas para identificar estos errores. El ejemplo de la diapositiva ilustra los valores de SQLCODE y SQLERRM que se asignan a las variables y, a continuación, las variables que se utilizan en una sentencia SQL.

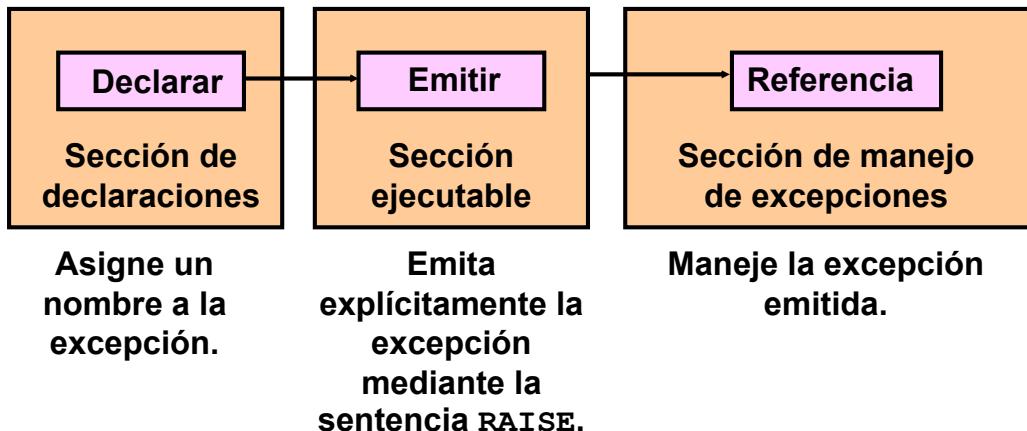
No puede utilizar SQLCODE o SQLERRM directamente en una sentencia SQL. En su lugar, debe asignar los valores a variables locales, a continuación utilice las variables en la sentencia SQL, como se muestra en el siguiente ejemplo:

```

DECLARE
    err_num NUMBER;
    err_msg VARCHAR2 (100);
BEGIN
    ...
EXCEPTION
    ...
    WHEN OTHERS THEN
        err_num := SQLCODE;
        err_msg := SUBSTR(SQLERRM, 1, 100);
        INSERT INTO errors VALUES (err_num, err_msg);
    END;
/

```

Detección de Excepciones Definidas por el Usuario



ORACLE

8 - 20

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Detección de Excepciones Definidas por el Usuario

PL/SQL permite definir las excepciones según los requisitos de la aplicación. Por ejemplo, puede que pida al usuario que introduzca un número de departamento.

Defina una excepción para tratar las condiciones de error en los datos de entrada. Compruebe si el número de departamento existe. Si no, puede que tenga que emitir la excepción definida por el usuario.

Las excepciones PL/SQL se deben:

- Declarar en la sección de declaraciones de los bloques PL/SQL
- Emitir explícitamente con sentencias RAISE
- Manejar en la sección EXCEPTION

Detección de Excepciones Definidas por el Usuario

```

DECLARE
    v_deptno NUMBER := 500;
    v_name VARCHAR2(20) := 'Testing';
    e_invalid_department EXCEPTION; ← 1
BEGIN
    UPDATE departments
    SET department_name = v_name
    WHERE department_id = v_deptno;
    IF SQL%NOTFOUND THEN
        RAISE e_invalid_department; ← 2
    END IF;
    COMMIT;
EXCEPTION
    WHEN e_invalid_department THEN
        DBMS_OUTPUT.PUT_LINE('No such department id.');
END;
/

```



ORACLE

8 - 21

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Detección de Excepciones Definidas por el Usuario (continuación)

Una excepción definida por el usuario se detecta al declararla o emitirla explícitamente.

1. Declare el nombre de la excepción definida por el usuario en la sección de declaraciones.

Sintaxis:

```
exception EXCEPTION;
```

En esta sintaxis, *exception* es el nombre de la excepción.

2. Utilice la sentencia RAISE para emitir la excepción explícitamente en la sección ejecutable.

Sintaxis:

```
RAISE exception;
```

En la sintaxis, *exception* es la excepción declarada anteriormente.

3. Haga referencia a la excepción declarada en la rutina de manejo de excepciones correspondiente.

Ejemplo

El bloque mostrado en la diapositiva actualiza *department_name* de un departamento. El usuario proporciona el número de departamento y el nuevo nombre. Si no existe el número de departamento indicado, no se actualiza ninguna fila en la tabla *departments*. Se emite una excepción y se imprime un mensaje para el usuario indicando que se ha introducido un número de departamento no válido.

Nota: utilice la sentencia RAISE sola en un manejador de excepciones para volver a emitir la misma excepción y propagarla en el entorno de llamada.

Propagación de Excepciones en un Subbloque

**Los subbloques
pueden manejar una
excepción o
transferirla al bloque
delimitador.**

```

DECLARE
    . . .
    e_no_rows      exception;
    e_integrity     exception;
    PRAGMA EXCEPTION_INIT (e_integrity, -2292);
BEGIN
    FOR c_record IN emp_cursor LOOP
        BEGIN
            SELECT . . .
            UPDATE . . .
            IF SQL%NOTFOUND THEN
                RAISE e_no_rows;
            END IF;
        END;
    END LOOP;
EXCEPTION
    WHEN e_integrity THEN . . .
    WHEN e_no_rows THEN . . .
END;
/

```

ORACLE

8 - 22

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Propagación de Excepciones en un Subbloque

Cuando un subbloque maneja una excepción, termina de forma normal. El control se reanuda en el bloque delimitador inmediatamente después de la sentencia END del subbloque.

Sin embargo, si un PL/SQL emite una excepción y el bloque actual no tiene un manejador para esa excepción, ésta se propaga a los sucesivos bloques delimitadores hasta que encuentra un manejador. Si ninguno de estos bloques maneja la excepción, se obtiene como resultado una excepción no tratada en el entorno del host.

Cuando la excepción se propaga a un bloque delimitador, se omiten las acciones ejecutables restantes de ese bloque.

Una ventaja de este comportamiento es que puede delimitar sentencias que necesitan su propio manejo de errores exclusivo en su propio bloque, mientras que dejan manejo de excepciones generales al bloque delimitador.

Observe en el ejemplo que las excepciones (no_rows y integrity) se declaran en el bloque externo. En el bloque interno, cuando se emite la excepción no_rows, PL/SQL busca la excepción que se maneja en el subbloque. Puesto que la excepción no se maneja en el subbloque, ésta se propaga al bloque externo, donde PL/SQL encuentra el manejador.

Procedimiento RAISE_APPLICATION_ERROR

Sintaxis:

```
raise_application_error (error_number,
                        message[, {TRUE | FALSE}]);
```

- Puede utilizar este procedimiento para emitir mensajes de error definidos por el usuario desde subprogramas almacenados.
- Puede informar de los errores de la aplicación y evitar la devolución de excepciones no tratadas.



8 - 23

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Procedimiento RAISE_APPLICATION_ERROR

Utilice el procedimiento RAISE_APPLICATION_ERROR para comunicar una excepción predefinida de forma interactiva al devolver un código de error no estándar y un mensaje de error. Con RAISE_APPLICATION_ERROR, puede informar de los errores de la aplicación y evitar la devolución de excepciones no tratadas.

En la sintaxis:

<i>error_number</i>	Es un número especificado por el usuario para la excepción entre -20.000 y -20.999
<i>message</i>	Es el mensaje especificado por el usuario para la excepción; es una cadena de caracteres de hasta 2.048 bytes
TRUE FALSE	Es un parámetro booleano opcional (si está definido en TRUE, el error se coloca en la pila de errores anteriores. Si está definido en FALSE, que es el valor por defecto, el error sustituye a todos los errores anteriores.)

Procedimiento RAISE_APPLICATION_ERROR

- Se utiliza en dos lugares diferentes:
 - Sección ejecutable
 - Sección de excepciones
- Devuelve condiciones de error al usuario de forma consistente con respecto a otros errores de Oracle Server



Procedimiento RAISE_APPLICATION_ERROR (continuación)

El procedimiento RAISE_APPLICATION_ERROR se puede utilizar en la sección ejecutable, en la sección de excepciones de un programa PL/SQL o en ambas. El error devuelto es consistente con la forma en la que Oracle Server produce un error predefinido, no predefinido o definido por el usuario. El número y el mensaje de error se muestra al usuario.

Procedimiento RAISE_APPLICATION_ERROR

Sección ejecutable:

```
BEGIN
...
DELETE FROM employees
  WHERE manager_id = v_mgr;
IF SQL%NOTFOUND THEN
  RAISE_APPLICATION_ERROR(-20202,
    'This is not a valid manager');
END IF;
...
```

Sección de excepciones:

```
...
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    RAISE_APPLICATION_ERROR (-20201,
      'Manager is not a valid employee.');
END;
```

ORACLE

8 - 25

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Procedimiento RAISE_APPLICATION_ERROR (continuación)

Esta diapositiva muestra que el procedimiento RAISE_APPLICATION_ERROR se puede utilizar en la sección ejecutable y de excepciones de un programa PL/SQL.

A continuación se presenta otro ejemplo de uso del procedimiento RAISE_APPLICATION_ERROR:

```
DECLARE
  e_name EXCEPTION;
BEGIN
...
DELETE FROM employees
  WHERE last_name = 'Higgins';
IF SQL%NOTFOUND THEN RAISE e_name;
END IF;
EXCEPTION
  WHEN e_name THEN
    RAISE_APPLICATION_ERROR (-20999, 'This is not a valid last
name'); ...
END;
/
```

Prueba

Puede detectar cualquier error incluyendo un manejador correspondiente dentro de la sección de manejo de excepciones del bloque PL/SQL.

- a. Verdadero
- b. Falso



Respuesta: a

Puede detectar cualquier error incluido un manejador correspondiente dentro de la sección de manejo de excepciones del bloque PL/SQL. Cada manejador está formado por una cláusula WHEN, que especifica un nombre de excepción, seguido de una secuencia de sentencias que se ejecutan cuando se emite dicha excepción. Puede incluir cualquier número de manejadores dentro de una sección EXCEPTION para manejar excepciones específicas. Sin embargo, no puede tener varios manejadores para una única excepción.

Resumen

En esta lección debe haber aprendido lo siguiente:

- Definir excepciones PL/SQL
- Agregar una sección EXCEPTION al bloque PL/SQL para tratar excepciones en tiempo de ejecución
- Manejar diferentes tipos de excepciones:
 - Excepciones predefinidas
 - Excepciones no predefinidas
 - Excepciones definidas por el usuario
- Propagar excepciones en bloques anidados y llamar a aplicaciones



Resumen

En esta lección, ha aprendido a tratar diferentes tipos de excepciones. En PL/SQL, una condición de advertencia o error en tiempo de ejecución se denomina excepción. Las excepciones predefinidas son condiciones de error que define Oracle Server. Las excepciones no predefinidas pueden ser cualquier error de Oracle Server estándar. Las excepciones definidas por el usuario son excepciones específicas de la aplicación. La función PRAGMA EXCEPTION_INIT se puede utilizar para asociar un nombre de excepción declarada a un error de Oracle Server.

Puede definir sus propias excepciones en la sección de declaraciones de cualquier bloque PL/SQL. Por ejemplo, puede definir una excepción denominada INSUFFICIENT_FUNDS para marcar cuentas bancarias al descubierto.

Cuando se produce un error, se emite una excepción. La ejecución normal se para y el control se transfiere a la sección de manejo de excepciones del bloque PL/SQL. Las excepciones internas se emiten de forma implícita (automáticamente) por el sistema de tiempo de ejecución; sin embargo, las excepciones definidas por el usuario se deben emitir explícitamente. Para manejar las excepciones generadas, escriba rutinas independientes denominadas manejadores de excepciones.

Práctica 8: Visión General

En esta práctica se abordan los siguientes temas:

- Creación y llamada de excepciones definidas por el usuario
- Manejo de excepciones con nombre de Oracle Server



Práctica 8: Visión General

En esta práctica, creará manejadores de excepciones para una excepción predefinida y una excepción de Oracle Server estándar.

Introducción a los Procedimientos y Funciones Almacenados

ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Objetivos

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

- Diferenciar entre los bloques anónimos y los subprogramas
- Crear un procedimiento simple y llamarlo desde un bloque anónimo
- Crear una función simple
- Crear una función simple que acepta un parámetro
- Diferenciar entre procedimientos y funciones



Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Objetivos

Se han explicado los bloques anónimos. Esta lección ofrece una introducción a los bloques con nombre, también denominados *subprogramas*. Los procedimientos y las funciones son subprogramas PL/SQL. En esta lección, aprenderá a diferenciar los bloques anónimos de los subprogramas.

Agenda

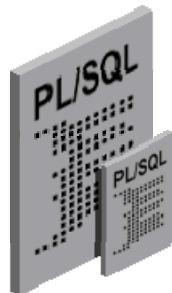
- Introducción a los procedimientos y las funciones
- Vista previa de procedimientos
- Vista previa de funciones

ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Procedimientos y Funciones

- Son bloques PL/SQL con nombre
- Se denominan subprogramas PL/SQL
- Tienen estructuras de bloques similares a los bloques anónimos:
 - Sección de declaraciones opcional (sin palabra clave DECLARE)
 - Sección ejecutable obligatoria
 - Sección opcional para manejar excepciones



ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Procedimientos y Funciones

Hasta este momento, los bloques anónimos eran los únicos ejemplos del código PL/SQL tratados en este curso. Como indica su nombre, los bloques *anónimos* son bloques PL/SQL ejecutables sin nombre. Puesto que no tienen nombre, no se pueden volver a utilizar ni almacenar para su uso posterior.

Los procedimientos y las funciones son bloques PL/SQL con nombre, también conocidos como *subprogramas*. Estos subprogramas se compilan y se almacenan en la base de datos. La estructura de bloques de los subprogramas es similar a la estructura de los bloques anónimos. Los subprogramas se pueden declarar no sólo a nivel de esquema sino también en cualquier otro bloque PL/SQL. Un subprograma contiene las siguientes secciones:

- **Sección de declaraciones:** los subprogramas pueden tener una sección de declaraciones opcional. Sin embargo, a diferencia de los bloques anónimos, la sección de declaraciones de los subprogramas no se inicia con la palabra clave DECLARE. La sección de declaraciones opcional sigue a la palabra clave IS o AS en la declaración de subprograma.
- **Sección ejecutable:** es la sección obligatoria del subprograma, que contiene la implantación de la lógica de negocio. Si observa el código de esta sección, puede determinar de forma sencilla la funcionalidad de negocio del subprograma. Esta sección empieza y termina con la palabra clave BEGIN y END respectivamente.
- **Sección de excepciones:** es una sección opcional que se incluye para manejar las excepciones.

Diferencias entre Bloques Anónimos y Subprogramas

Bloques Anónimos	Subprogramas
Bloques PL/SQL sin nombre	Bloques PL/SQL con nombre
Se compilan en cada ocasión	Se compilan sólo una vez
No se almacenan en la base de datos	Se almacenan en la base de datos
Otras aplicaciones no los pueden llamar	Tienen nombre y, por tanto, otras aplicaciones los pueden llamar
No devuelven valores	Si son funciones, deben devolver valores
No pueden utilizar parámetros	Pueden utilizar parámetros

ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Diferencias entre Bloques Anónimos y Subprogramas

La tabla de la diapositiva no sólo muestra las diferencias entre los bloques anónimos y los subprogramas, sino que también resalta las ventajas generales de los subprogramas.

Los bloques anónimos no son objetos de base de datos persistentes. Se compilan cada vez que se ejecutan. No se almacenan en la base de datos para volver a utilizarlos. Si desea volver a utilizarlos, debe volver a ejecutar el script que crea el bloque anónimo, que produce la recompilación y la ejecución.

Los procedimientos y las funciones se compilan y se almacenan en la base de datos en forma compilada. Sólo se vuelven a compilar cuando se modifican. Puesto que se almacenan en la base de datos, cualquier aplicación puede utilizar estos subprogramas basándose en los permisos necesarios. La aplicación que realiza la llamada puede transferir los parámetros a los procedimientos si el procedimiento está diseñado para aceptar parámetros. Asimismo, una aplicación que realiza la llamada puede recuperar un valor si llama a una función o a un procedimiento.

Agenda

- Introducción a los procedimientos y las funciones
- **Vista previa de procedimientos**
- Vista previa de funciones

ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Procedimiento: Sintaxis

```
CREATE [OR REPLACE] PROCEDURE procedure_name
  [(argument1 [mode1] datatype1,
    argument2 [mode2] datatype2,
    . . .)]
IS | AS
procedure_body;
```



Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Procedimiento: Sintaxis

La diapositiva muestra la sintaxis para crear procedimientos. En la sintaxis:

procedure_name Es el nombre del procedimiento que se va a crear

argument Es el nombre otorgado a los parámetros de procedimiento. Cada argumento está asociado a un modo y tipo de dato. Puede tener cualquier número de argumentos separados por comas.

mode Modo de argumento:
IN (valor por defecto)
OUT
IN OUT

datatype Es el tipo de dato del parámetro asociado. El tipo de dato de parámetros no puede tener tamaño explícito; en su lugar, utilice %TYPE.

Procedure_body Es el bloque PL/SQL que forma el código

La lista de argumentos es opcional en una declaración de procedimiento. Los procedimientos se explican en detalle en el curso *Oracle Database: Desarrollo de Unidades de Programa en PL/SQL*.

Creación de Procedimientos

```
...
CREATE TABLE dept AS SELECT * FROM departments;
CREATE PROCEDURE add_dept IS
    v_dept_id dept.department_id%TYPE;
    v_dept_name dept.department_name%TYPE;
BEGIN
    v_dept_id:=280;
    v_dept_name:='ST-Curriculum';
    INSERT INTO dept(department_id,department_name)
    VALUES(v_dept_id,v_dept_name);
    DBMS_OUTPUT.PUT_LINE(' Inserted '|| SQL%ROWCOUNT
    ||' row ');
END;
```



Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Creación de Procedimientos

En el código de ejemplo, el procedimiento `add_dept` inserta un departamento nuevo con el identificador de departamento 280 y el nombre de departamento ST-Curriculum.

Además, en el ejemplo se muestra lo siguiente:

- La sección de declaraciones de un procedimiento se inicia inmediatamente después de la declaración del procedimiento y no empieza por la palabra clave DECLARE.
- El procedimiento declara dos variables, `dept_id` y `dept_name`.
- El procedimiento utiliza el atributo de cursor implícito o el atributo SQL `SQL%ROWCOUNT` para verificar si la fila se insertó correctamente. Se debe devolver un valor 1 en este caso.

Nota: consulte más notas sobre el ejemplo en la página siguiente.

Procedimiento: Ejemplo

Nota

- Al crear cualquier objeto, las entradas se realizan en la tabla `user_objects`. Cuando el código de la diapositiva se ejecuta correctamente, puede comprobar los nuevos objetos en la tabla `user_objects` emitiendo el siguiente comando:

```
SELECT object_name,object_type FROM user_objects;
```

	OBJECT_NAME	OBJECT_TYPE
41	COPY_EMP	TABLE
42	DEPT	TABLE
43	GREET	PROCEDURE
44	ADD_DEPT	PROCEDURE
45	MY_SEQ	SEQUENCE

- El origen del procedimiento se almacena en la tabla `user_source`. Puede comprobar el origen del procedimiento emitiendo el siguiente comando:

```
SELECT * FROM user_source WHERE name='ADD_DEPT';
```

	NAME	TYPE	LINE	TEXT
1	ADD_DEPT	PROCEDURE	1	PROCEDURE add_dept IS
2	ADD_DEPT	PROCEDURE	2	v_dept_id dept.department_id%TYPE;
3	ADD_DEPT	PROCEDURE	3	v_dept_name dept.department_name%TYPE;
4	ADD_DEPT	PROCEDURE	4	BEGIN
5	ADD_DEPT	PROCEDURE	5	v_dept_id:=280;
6	ADD_DEPT	PROCEDURE	6	v_dept_name:='ST-Curriculum';
7	ADD_DEPT	PROCEDURE	7	INSERT INTO dept(department_id,department_name)
8	ADD_DEPT	PROCEDURE	8	VALUES(v_dept_id,v_dept_name);
9	ADD_DEPT	PROCEDURE	9	DBMS_OUTPUT.PUT_LINE(' Inserted' SQL%ROWCOUNT ' row');
10	ADD_DEPT	PROCEDURE	10	END;

Llamada de Procedimientos

```
...
BEGIN
  add_dept;
END ;
/
SELECT department_id, department_name FROM dept
WHERE department_id=280;
```

The screenshot shows the Oracle SQL Developer interface with the 'Results' tab selected. The output pane displays:

```
anonymous block completed
Inserted 1 row

DEPARTMENT_ID      DEPARTMENT_NAME
-----              -----
280                ST-Curriculum

1 rows selected
```

ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Llamada a Procedimientos

La diapositiva muestra cómo llamar a un procedimiento desde un bloque anónimo. Tiene que incluir la llamada al procedimiento en la sección ejecutable del bloque anónimo. Asimismo, puede llamar al procedimiento desde cualquier aplicación, como una aplicación de Forms o una aplicación Java. La sentencia SELECT del código comprueba si la fila se ha insertado correctamente.

También puede llamar a un procedimiento con la sentencia SQL CALL <procedure_name>.

Agenda

- Introducción a los procedimientos y las funciones
- Vista previa de procedimientos
- Vista previa de funciones

ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Función: Sintaxis

```
CREATE [OR REPLACE] FUNCTION function_name
  [(argument1 [mode1] datatype1,
    argument2 [mode2] datatype2,
    . . .)]
  RETURN datatype
  IS | AS
  function_body;
```



Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Función: Sintaxis

La diapositiva muestra la sintaxis para crear una función. En la sintaxis:

<i>function_name</i>	Es el nombre de la función que se va a crear
<i>argument</i>	Nombre proporcionado al parámetro de la función. (Cada argumento está asociado a un modo y tipo de dato. Puede tener cualquier número de argumentos separados por coma. Al llamar a la función transferirá el argumento.)
<i>mode</i>	Tipo del parámetro (sólo los parámetros IN se deben declarar.)
<i>datatype</i>	Es el tipo de dato del parámetro asociado
RETURN <i>datatype</i>	Es el tipo de dato del valor devuelto por la función
<i>function_body</i>	Es el bloque PL/SQL que forma el código de función

La lista de argumentos es opcional en una declaración de funciones. La diferencia entre un procedimiento y una función es que una función debe devolver un valor al programa de llamada. Por tanto, la sintaxis contiene *return_type* que especifica el tipo de dato del valor que devuelve la función. Puede que un procedimiento devuelva un valor a través de un parámetro OUT o IN OUT.

Creación de Funciones

```
CREATE FUNCTION check_sal RETURN Boolean IS
  v_dept_id employees.department_id%TYPE;
  v_empno   employees.employee_id%TYPE;
  v_sal      employees.salary%TYPE;
  v_avg_sal employees.salary%TYPE;
BEGIN
  v_empno:=205;
  SELECT salary,department_id INTO v_sal,v_dept_id FROM
employees
  WHERE employee_id= v_empno;
  SELECT avg(salary) INTO v_avg_sal FROM employees WHERE
department_id=v_dept_id;
  IF v_sal > v_avg_sal THEN
    RETURN TRUE;
  ELSE
    RETURN FALSE;
  END IF;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    RETURN NULL;
END;
```



Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Función: Ejemplo

La función `check_sal` se escribe para determinar si el salario de un empleado determinado es mayor o menor que el salario medio de todos los empleados que trabajan en el mismo departamento. La función devuelve TRUE si el salario del empleado es mayor que el salario medio de los empleados del departamento; de lo contrario, devuelve FALSE. La función devuelve NULL si se emite una excepción NO_DATA_FOUND.

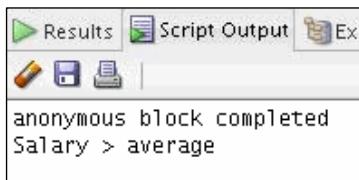
Observe que la función comprueba el empleado con el identificador de empleado 205. La función se codifica sólo para comprobar este identificador de empleado. Si desea comprobar otros empleados, tendrá que modificar la propia función. Puede resolver este problema declarando la función de modo que acepte un argumento. A continuación, puede transferir el identificador de empleado como parámetro.

Llamada de Funciones

```

BEGIN
  IF (check_sal IS NULL) THEN
    DBMS_OUTPUT.PUT_LINE('The function returned
      NULL due to exception');
  ELSIF (check_sal) THEN
    DBMS_OUTPUT.PUT_LINE('Salary > average');
  ELSE
    DBMS_OUTPUT.PUT_LINE('Salary < average');
  END IF;
END ;
/

```



ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Llamada a Funciones

Incluya la llamada a la función en la sección ejecutable del bloque anónimo.

La función se llama como parte de una sentencia. Recuerde que la función `check_sal` devuelve Boolean o NULL. Por tanto, la llamada a la función se incluye como la expresión condicional para el bloque IF.

Nota: puede utilizar el comando DESCRIBE para comprobar los argumentos y devolver el tipo de la función, como en el siguiente ejemplo:

```
DESCRIBE check_sal;
```

Transferencia de un Parámetro a la Función

```

DROP FUNCTION check_sal;
CREATE FUNCTION check_sal(p_empno employees.employee_id%TYPE)
RETURN Boolean IS
    v_dept_id employees.department_id%TYPE;
    v_sal      employees.salary%TYPE;
    v_avg_sal employees.salary%TYPE;
BEGIN
    SELECT salary,department_id INTO v_sal,v_dept_id FROM employees
        WHERE employee_id=p_empno;
    SELECT avg(salary) INTO v_avg_sal FROM employees
        WHERE department_id=v_dept_id;
    IF v_sal > v_avg_sal THEN
        RETURN TRUE;
    ELSE
        RETURN FALSE;
    END IF;
EXCEPTION
    ...

```

ORACLE

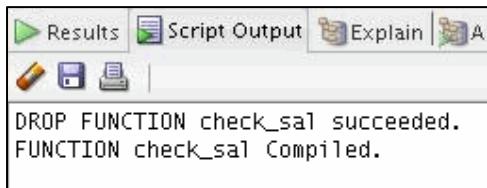
Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Transferencia de un Parámetro a la Función

Recuerde que la función se ha codificado para comprobar el salario del empleado con el identificador de empleado 205. El código que se muestra en la diapositiva elimina la restricción, ya que se vuelve a escribir para aceptar el número del empleado como parámetro. Ahora puede transferir números de diferentes empleados y comprobar sus salarios.

Los procedimientos se explican en detalle en el curso *Oracle Database: Desarrollo de Unidades de Programa en PL/SQL*.

La salida del código de ejemplo en la diapositiva es la siguiente:



The screenshot shows a database interface with several tabs at the top: 'Results' (highlighted in blue), 'Script Output', 'Explain', and 'A'. Below these are icons for 'Edit', 'Save', and 'Print'. The main area displays the following text:

```
DROP FUNCTION check_sal succeeded.
FUNCTION check_sal Compiled.
```

Llamada a Funciones con un Parámetro

```

BEGIN
  DBMS_OUTPUT.PUT_LINE('Checking for employee with id 205');
  IF (check_sal(205) IS NULL) THEN
    DBMS_OUTPUT.PUT_LINE('The function returned
      NULL due to exception');
  ELSIF (check_sal(205)) THEN
    DBMS_OUTPUT.PUT_LINE('Salary > average');
  ELSE
    DBMS_OUTPUT.PUT_LINE('Salary < average');
  END IF;
  DBMS_OUTPUT.PUT_LINE('Checking for employee with id 70');
  IF (check_sal(70) IS NULL) THEN
    DBMS_OUTPUT.PUT_LINE('The function returned
      NULL due to exception');
  ELSIF (check_sal(70)) THEN
    ...
  END IF;
END;
/

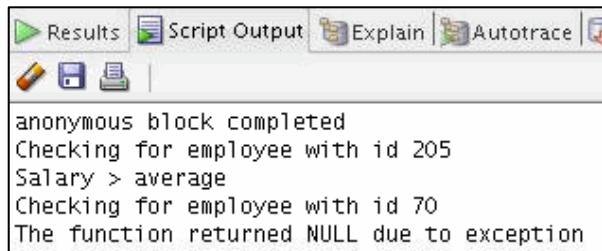
```

ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Llamada a Funciones con un Parámetro

El código de la diapositiva llama a la función dos veces al transferir los parámetros. La salida del código es la siguiente:



The screenshot shows the Oracle SQL developer interface with the 'Results' tab selected. The output window displays the following text:

```

anonymous block completed
Checking for employee with id 205
Salary > average
Checking for employee with id 70
The function returned NULL due to exception

```

Prueba

Subprogramas:

- a. Son bloques PL/SQL con nombre que se pueden llamar desde otras aplicaciones
- b. Se compilan sólo una vez
- c. Se almacenan en la base de datos
- d. No tienen que devolver valores si son funciones
- e. Pueden utilizar parámetros

ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Respuesta: a, b, c, e

Resumen

En esta lección debe haber aprendido lo siguiente:

- Crear un procedimiento simple
- Llamar al procedimiento desde un bloque anónimo
- Crear una función simple
- Crear una función simple que acepta parámetros
- Llamar a la función desde un bloque anónimo



Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Resumen

Puede utilizar bloques anónimos para diseñar cualquier funcionalidad en PL/SQL. Sin embargo, la restricción principal con los bloques anónimos es que no se almacenan y, por tanto, no se pueden volver a utilizar.

En lugar de crear bloques anónimos, puede crear subprogramas PL/SQL. Los procedimientos y las funciones se denominan subprogramas, que son bloques PL/SQL con nombre. Los subprogramas expresan lógica reutilizable mediante parámetros. La estructura de un procedimiento o una función es similar a la estructura de un bloque anónimo. Estos subprogramas se almacenan en la base de datos y, por tanto, son reutilizables.

Práctica 9: Visión General

En esta práctica se abordan los siguientes temas:

- Conversión de un bloque anónimo existente en un procedimiento
- Modificación del procedimiento para aceptar un parámetro
- Escritura de un bloque anónimo para llamar al procedimiento



Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Apéndice A

Prácticas y Soluciones

Tabla de Contenido

Prácticas y Soluciones de la Lección I.....	3
Práctica I-1: Acceso a los Recursos de SQL Developer	3
Práctica I-2: Introducción	3
Solución I-1: Acceso a los Recursos de SQL Developer.....	6
Solución I-2: Introducción	7
Prácticas y Soluciones de la Lección 1	14
Práctica 1: Introducción a PL/SQL	14
Solución 1: Introducción a PL/SQL.....	15
Prácticas y Soluciones de la Lección 2	16
Práctica 2: Declaración de Variables PL/SQL.....	16
Solución 2: Declaración de Variables PL/SQL	18
Prácticas y Soluciones de la Lección 3	21
Práctica 3: Escritura de Sentencias Ejecutables.....	21
Solución 3: Escritura de Sentencias Ejecutables	24
Prácticas y Soluciones de la Lección 4	28
Práctica 4: Interacción con Oracle Server.....	28
Solución 4: Interacción con Oracle Server	30
Prácticas y Soluciones de la Lección 5	33
Práctica 5: Escritura de las Estructuras de Control.....	33
Solución 5: Escritura de las Estructuras de Control.....	35
Prácticas y Soluciones de la Lección 6	38
Práctica 6: Trabajar con Tipos de Dato Compuestos.....	38
Solución 6: Trabajar con Tipos de Dato Compuestos	40
Prácticas y Soluciones de la Lección 7	45
Práctica 7-1: Uso de Cursores Explícitos.....	45
Práctica 7-2: Uso de Cursores Explícitos (Opcional)	48
Solución 7-1: Uso de Cursores Explícitos	49
Solución 7-2: Uso de Cursores Explícitos (Opcional)	54
Prácticas y Soluciones de la Lección 8	56
Práctica 8-1: Manejo de Excepciones Predefinidas	56
Práctica 8-2: Manejo de Excepciones de Oracle Server Estándar	58
Solución 8-1: Manejo de Excepciones Predefinidas.....	59
Solución 8-2: Manejo de Excepciones de Oracle Server Estándar	61
Prácticas y Soluciones de la Lección 9	62
Práctica 9: Creación y Uso de Procedimientos Almacenados	62
Solución 9: Creación y Uso de Procedimientos Almacenados	64

Prácticas y Soluciones de la Lección I

En estas prácticas, identificará los recursos de información para SQL Developer, ejecutará sentencias SQL con SQL Developer y examinará los datos del esquema de clase. En concreto:

- Iniciará SQL Developer
- Creará una conexión a la base de datos
- Examinará las tablas de esquema
- Definirá una preferencia de SQL Developer

Nota: todas las prácticas escritas utilizan SQL Developer como entorno de desarrollo. Si bien se recomienda utilizar SQL Developer, también puede utilizar los entornos SQL*Plus o JDeveloper disponibles en este curso.

Práctica I-1: Acceso a los Recursos de SQL Developer

En esta práctica, navegará a la página inicial de SQL Developer y examinará información útil sobre la herramienta.

- 1) Acceda a la página inicial de SQL Developer.
 - a) Acceda a la página inicial en línea de SQL Developer, disponible en:
http://www.oracle.com/technology/products/database/sql_developer/index.html
 - b) Marque la página para acceder más fácilmente a ella en el futuro.
- 2) Acceda al tutorial de SQL Developer disponible en línea en la dirección:
<http://st-curriculum.oracle.com/tutorial/SQLDeveloper/index.htm>. A continuación, revise las siguientes secciones y demostraciones asociadas:
 - a) Pasos Iniciales
 - b) Trabajar con Objetos de la Base de Datos
 - c) Acceso a los Datos

Práctica I-2: Introducción

- 1) Inicie SQL Developer.
- 2) Cree una conexión de base de datos con la siguiente información (**Indicación:** active la casilla de control Save Password.):
 - a) Connection Name: MyConnection
 - b) Username: ora41
 - c) Password: ora41
 - d) Hostname: localhost
 - e) Port: 1521
 - f) SID: orcl

Práctica I-2: Introducción (continuación)

- 3) Pruebe la nueva conexión. Si el estado es Success, conecte a la base de datos con esta nueva conexión.
 - a) En la ventana Database Connection, haga clic en el botón Test.
Nota: el estado de la conexión aparece en la esquina inferior izquierda de la ventana.
 - b) Si el estado es Success, haga clic en el botón Connect.
- 4) Examine la estructura de la tabla EMPLOYEES y muestre sus datos.
 - a) Amplíe la conexión MyConnection. Para ello, haga clic en el símbolo más situado junto a ella.
 - b) Amplíe el ícono Tables. Para ello, haga clic en el símbolo más situado junto a él.
 - c) Visualice la estructura de la tabla EMPLOYEES.
- 5) Utilice el separador EMPLOYEES para ver los datos de la tabla EMPLOYEES.
- 6) Utilice la hoja de trabajo de SQL para seleccionar los apellidos y los salarios de todos los empleados cuyo salario anual sea mayor de 10.000 dólares. Utilice tanto el ícono Execute Statement (F9) como el ícono Run Script (F5) para ejecutar la sentencia SELECT. Revise los resultados de ambos métodos de ejecución de las sentencias SELECT en los separadores adecuados.
Nota: dedique unos minutos a familiarizarse con los datos o a consultar el Apéndice B, donde se proporciona la descripción y los datos de todas las tablas del esquema HR que utilizará en este curso.
- 7) En el menú SQL Developer, seleccione Tools > Preferences. Aparecerá la ventana Preferences.
- 8) Seleccione Database > Worksheet Parameters. En el cuadro de texto “Select default path to look for scripts”, utilice el botón Browse para seleccionar la carpeta /home/oracle/labs/plsf. Esta carpeta contiene los scripts de código de ejemplo, los scripts de prácticas y los scripts de soluciones a las prácticas que se utilizarán en este curso. A continuación, en la ventana Preferences, haga clic en OK para guardar la configuración de Worksheet Parameter.
- 9) Familiarícese con la estructura de la carpeta /home/oracle/labs/plsf.
 - a) Seleccione File > Open. En la ventana Open se selecciona automáticamente la carpeta .../plsf como ubicación inicial. Esta carpeta contiene tres subcarpetas:
 - La carpeta /code_ex contiene los códigos de ejemplo utilizados en el material del curso. Cada script .sql está asociado a una página determinada de la lección.
 - La carpeta /labs contiene el código que se utiliza en determinadas prácticas de las lecciones. Se le pedirá que ejecute el script necesario en la práctica en cuestión.
 - La carpeta /soln contiene las soluciones de cada práctica. Cada script .sql está numerado con la referencia práctica_ejercicio asociada.

Práctica I-2: Introducción (continuación)

- b) También puede utilizar el separador Files para desplazarse por las carpetas y abrir los archivos de scripts.
- c) Con la ventana Open y el separador Files, desplácese por las carpetas y abra un archivo de script sin ejecutar el código.
- d) Cierre la hoja de trabajo de SQL.

Solución I-1: Acceso a los Recursos de SQL Developer

- 1) Acceda a la página inicial de SQL Developer.
 - a) Acceda a la página inicial en línea de SQL Developer, disponible en:
http://www.oracle.com/technology/products/database/sql_developer/index.html

La página inicial de SQL Developer se muestra de la siguiente forma:

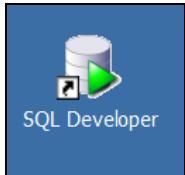
The screenshot shows the Oracle SQL Developer homepage within a Microsoft Internet Explorer browser window. The address bar contains the URL http://www.oracle.com/technology/products/database/sql_developer/index.html. The page itself is titled "Oracle SQL Developer" and features a sidebar with links for "PRODUCTS", "TECHNOLOGIES", and "COMMUNITY". The main content area includes a brief description of SQL Developer, download links for "FREE DOWNLOAD" and "Oracle SQL Developer 1.2", and sections for "Testimonials" and "Related Technologies".

- b) Marque la página para acceder más fácilmente a ella en el futuro.
- 2) Acceda al tutorial de SQL Developer disponible en línea en la dirección:
<http://st-curriculum.oracle.com/tutorial/SQLDeveloper/index.htm>. A continuación, revise las siguientes secciones y demostraciones asociadas:
 - a) Pasos Iniciales
 - b) Trabajar con Objetos de la Base de Datos
 - c) Acceso a los Datos

Solución I-2: Introducción

- 1) Inicie SQL Developer.

Haga clic en el ícono SQL Developer del escritorio.



- 2) Cree una conexión de base de datos con la siguiente información (**Indicación:** active la casilla de control Save Password.):
 - a) Connection Name: MyConnection
 - b) Username: ora41
 - c) Password: ora41
 - d) Hostname: localhost
 - e) Port: 1521
 - f) SID: orcl

Haga clic con el botón derecho en el nodo Connections de la página con separadores Connections y seleccione New Database Connection en el menú de acceso directo. Resultado: aparece la ventana New>Select Database Connection.

Utilice la información anterior para crear la nueva conexión de base de datos. Además, active la casilla de control Save Password. Por ejemplo:

 A screenshot of the 'New Database Connection' dialog box in SQL Developer. It shows the following configuration:

- Connection Name:** MyConnection
- Username:** ora41
- Password:** ora41 (displayed as *****)
- Save Password:**
- Access:** Oracle
- Role:** default
- OS Authentication:**
- Connection Type:** Basic
- Kerberos Authentication:**
- Proxy Connection:**
- Hostname:** localhost
- Port:** 1521
- SID:** orcl (selected radio button)
- Service name:** (disabled, grayed out)

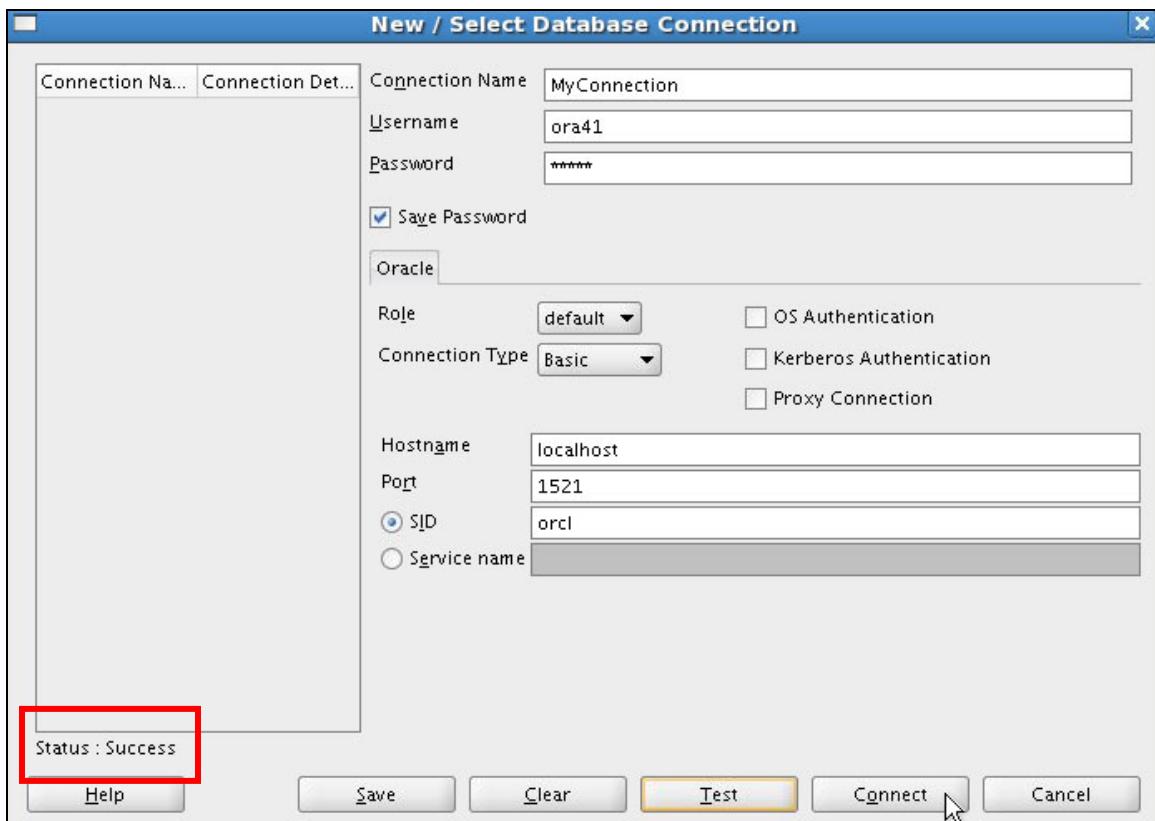
Solución I-2: Introducción (continuación)

- 3) Pruebe la nueva conexión. Si el estado es Success, conecte a la base de datos con esta nueva conexión.

- a) En la ventana Database Connection, haga clic en el botón Test.

Nota: el estado de la conexión aparece en la esquina inferior izquierda de la ventana.

- b) Si el estado es Success, haga clic en el botón Connect.



Nota: para mostrar las propiedades de una conexión existente, haga clic con el botón derecho en el nombre de la conexión en el separador Connections y seleccione Properties en el menú de acceso directo.

- 4) Examine la estructura de la tabla EMPLOYEES y muestre sus datos.
- Amplíe la conexión MyConnection. Para ello, haga clic en el símbolo más situado junto a ella.
 - Amplíe el ícono Tables. Para ello, haga clic en el símbolo más situado junto a él.
 - Visualice la estructura de la tabla EMPLOYEES.

Aumente el detalle de la tabla EMPLOYEES haciendo clic en el símbolo más situado junto a ella.

Haga clic en la tabla EMPLOYEES.

Resultado: en el separador Columns se muestran las columnas de la tabla EMPLOYEES, de la siguiente forma:

Solución I-2: Introducción (continuación)

The screenshot shows the Oracle SQL Developer interface. On the left, the Connections pane displays a connection named 'MyConnection' with tables like COUNTRIES, DEPARTMENTS, and EMPLOYEES selected. The main pane shows the 'EMPLOYEES' table with its columns: Employee ID, First Name, Last Name, Email, Phone Number, Hire Date, Job ID, Salary, Commission Pct, Manager ID, Department ID, Job History, Jobs, Locations, and Regions. The 'Data' tab is selected, showing the detailed structure of the table.

- 5) Utilice el separador EMPLOYEES para ver los datos de la tabla EMPLOYEES.

Para que aparezcan los datos de los empleados, haga clic en el separador Data.

Resultado: aparecen los datos de la tabla EMPLOYEES, como se muestra a continuación:

The screenshot shows the data for the EMPLOYEES table in the 'Data' view of Oracle SQL Developer. The table contains 13 rows of employee information, including their first name, last name, email, phone number, hire date, and job details.

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE
1	100	Steven	King	SKING	515.123.4567	17-JUN-
2	101	Neena	Kochhar	NKOCHHAR	515.123.4568	21-SEP-
3	102	Lex	De Haan	LDEHAAN	515.123.4569	13-JAN-
4	103	Alexander	Hunold	AHUNOLD	590.423.4567	03-JAN-
5	104	Bruce	Ernst	BERNST	590.423.4568	21-MAY-
6	105	David	Austin	DAUSTIN	590.423.4569	25-JUN-
7	106	Valli	Pataballa	VPATABAL	590.423.4560	05-FEB-
8	107	Diana	Lorentz	DLORENZ	590.423.5567	07-FEB-
9	108	Nancy	Greenberg	NGREENBE	515.124.4569	17-AUG-
10	109	Daniel	Faviet	DFAVIET	515.124.4169	16-AUG-
11	110	John	Chen	JCHEN	515.124.4269	28-SEP-
12	111	Ismael	Sciarra	ISCIARRA	515.124.4369	30-SEP-
13	112	Jose Manuel	Urman	JMURMAN	515.124.4469	07-MAR

Solución I-2: Introducción (continuación)

- 6) Utilice la hoja de trabajo de SQL para seleccionar los apellidos y los salarios de todos los empleados cuyo salario anual sea mayor de 10.000 dólares. Utilice tanto el icono Execute Statement (F9) como el icono Run Script (F5) para ejecutar la sentencia SELECT. Revise los resultados de ambos métodos de ejecución de las sentencias SELECT en los separadores adecuados.

Nota: dedique unos minutos a familiarizarse con los datos o a consultar el Apéndice B, donde se proporciona la descripción y los datos de todas las tablas del esquema HR que utilizará en este curso.

Para acceder a la hoja de trabajo de SQL, haga clic en el separador MyConnection.

Nota: este separador se abrió anteriormente, al aumentar el detalle de la conexión a la base de datos.

Introduzca la sentencia SELECT adecuada. Pulse F9 para ejecutar la consulta y F5 para ejecutarla con el método Run Script.

Por ejemplo, si pulsa F9, los resultados son similares a los siguientes:

The screenshot shows the Oracle SQL Developer interface. The top window is titled 'MyConnection' and contains the SQL statement:

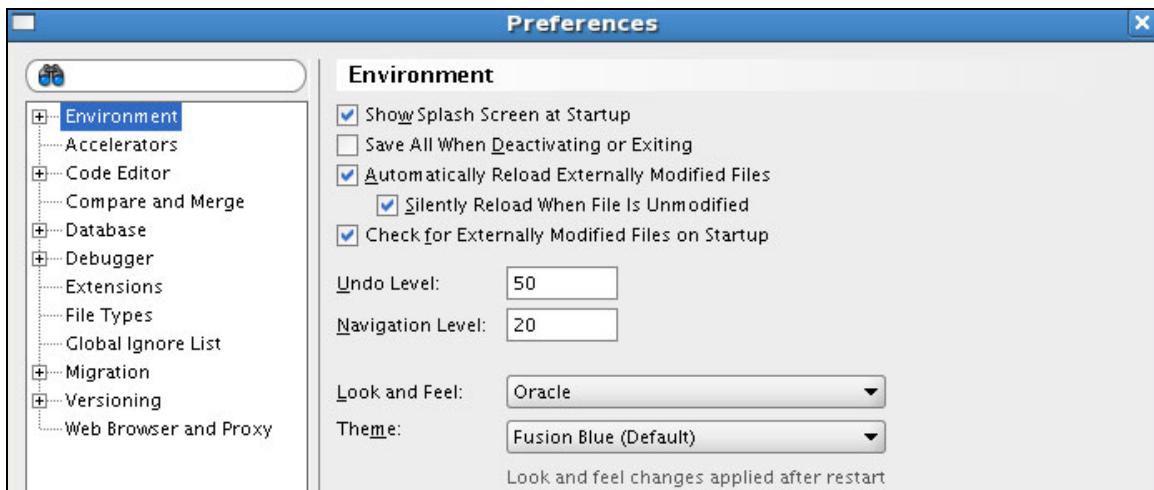
```
select last_name, salary
from employees
where salary > 10000;
```

The bottom window is titled 'Results' and displays the query results in a grid:

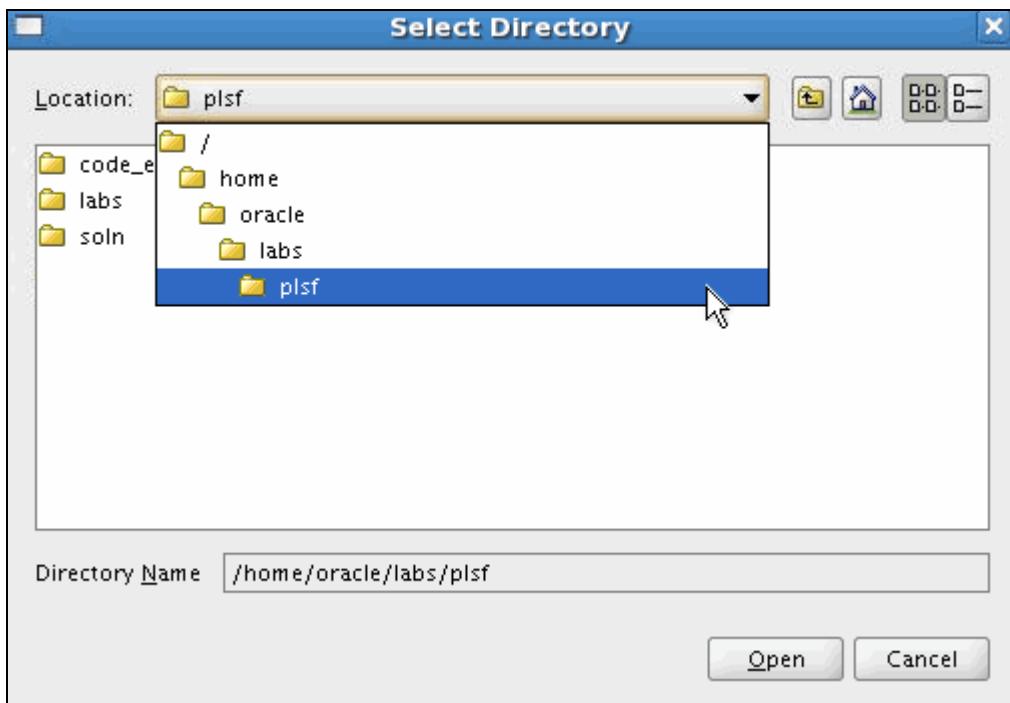
	LAST_NAME	SALARY
1	King	24000
2	Kochhar	17000
3	De Haan	17000
4	Greenberg	12000
5	Raphaely	11000
6	Russell	14000
7	Partners	13500

Solución I-2: Introducción (continuación)

- 7) En el menú SQL Developer, seleccione Tools > Preferences. Aparecerá la ventana Preferences.



- 8) Seleccione Database > Worksheet Parameters. En el cuadro de texto “Select default path to look for scripts”, utilice el botón Browse para seleccionar la carpeta /home/oracle/labs/plsf.

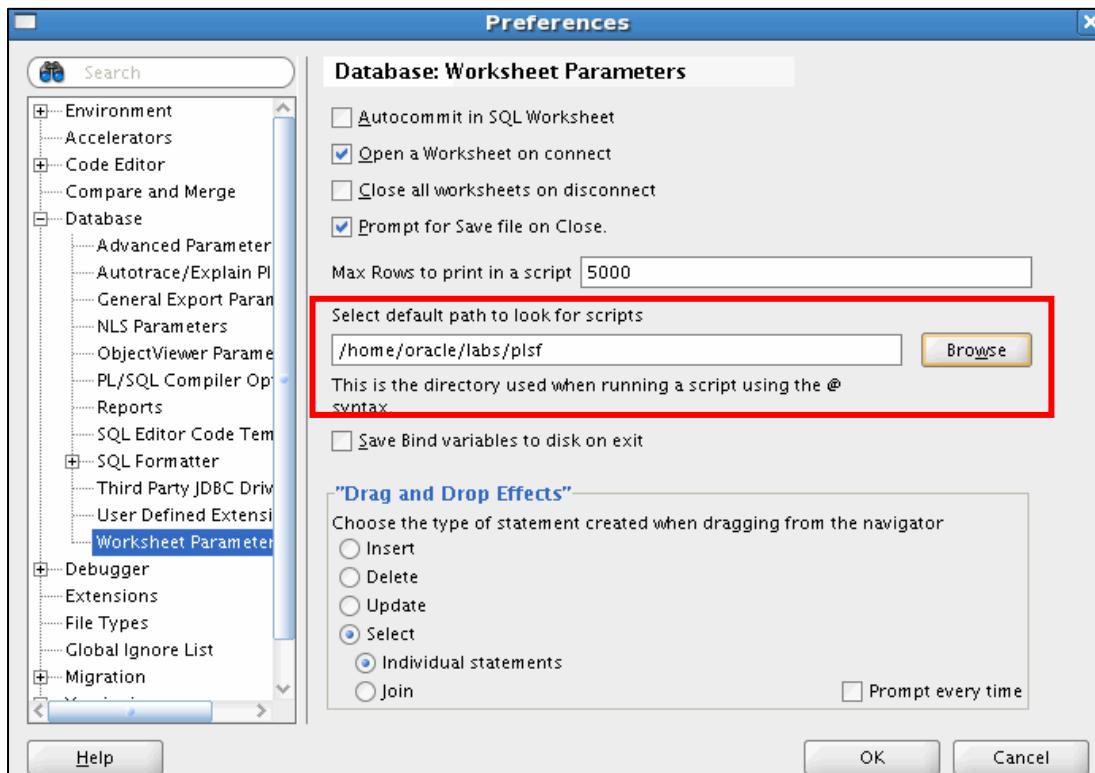


Esta carpeta contiene los scripts de código de ejemplo, los scripts de prácticas y los scripts de soluciones a las prácticas que se utilizarán en este curso.

Haga clic en Open para seleccionar la carpeta.

Solución I-2: Introducción (continuación)

A continuación, en la ventana Preferences, haga clic en OK para guardar la configuración de Worksheet Parameter.



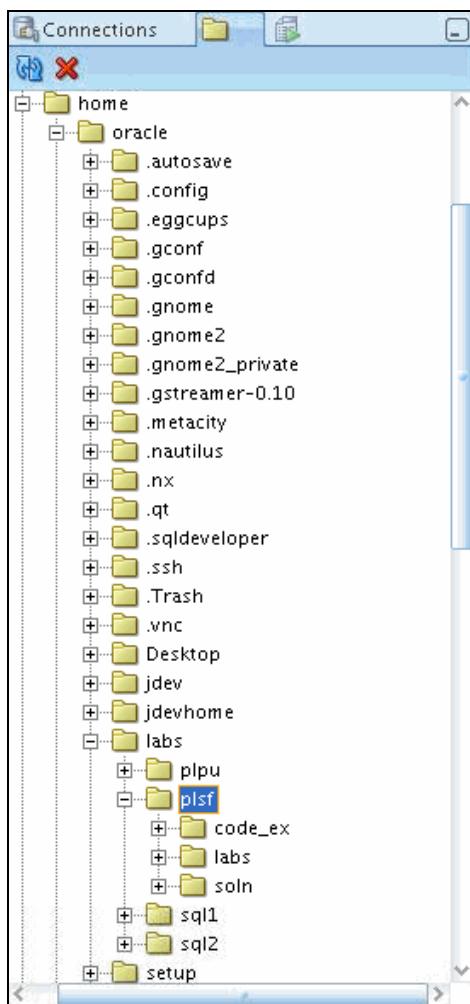
- 9) Familiarícese con la estructura de la carpeta /home/oracle/labs/plsf.
 - a) Seleccione File > Open. En la ventana Open se selecciona automáticamente la carpeta .../plsf como ubicación inicial. Esta carpeta contiene tres subcarpetas:



- La carpeta /code_ex contiene los códigos de ejemplo utilizados en el material del curso. Cada script .sql está asociado a una página determinada de la lección.
- La carpeta /labs contiene el código que se utiliza en determinadas prácticas de las lecciones. Se le pedirá que ejecute el script necesario en la práctica en cuestión.

Solución I-2: Introducción (continuación)

- La carpeta /soln contiene las soluciones de cada práctica. Cada script .sql está numerado con la referencia práctica_ejercicio asociada.
- b) También puede utilizar el separador Files para desplazarse por las carpetas y abrir los archivos de scripts.



- c) Con la ventana Open y el separador Files, desplácese por las carpetas y abra un archivo de script sin ejecutar el código.
- d) Cierre la hoja de trabajo de SQL.

Para cerrar los separadores de SQL Worksheet, haga clic en la X del separador, como se muestra aquí:



Prácticas y Soluciones de la Lección 1

La carpeta `/home/oracle/labs` es el directorio de trabajo en el que guardará los scripts que cree.

Las soluciones para todas las prácticas están en la carpeta `/home/oracle/labs/plsf/soln`.

Práctica 1: Introducción a PL/SQL

- 1) ¿Qué bloque PL/SQL de los siguientes se ejecuta correctamente?
 - a) BEGIN
END;
 - b) DECLARE
v_amount INTEGER(10);
END;
 - c) DECLARE
BEGIN
END;
 - d) DECLARE
v_amount INTEGER(10);
BEGIN
DBMS_OUTPUT.PUT_LINE(amount);
END;
- 2) Cree y ejecute un bloque anónimo simple cuya salida sea “Hello World”. Ejecute y guarde este script como `lab_01_02_soln.sql`.

Solución 1: Introducción a PL/SQL

1) ¿Qué bloque PL/SQL de los siguientes se ejecuta correctamente?

- a) BEGIN
END;
- b) DECLARE
v_amount INTEGER(10);
END;
- c) DECLARE
BEGIN
END;
- d) DECLARE
v_amount INTEGER(10);
BEGIN
DBMS_OUTPUT.PUT_LINE(amount);
END;

El bloque a no se ejecuta. No tiene ninguna sentencia ejecutable.

El bloque b no tiene la sección ejecutable obligatoria que empieza por la palabra clave BEGIN.

El bloque c tiene todas las partes necesarias, pero ninguna sentencia ejecutable.

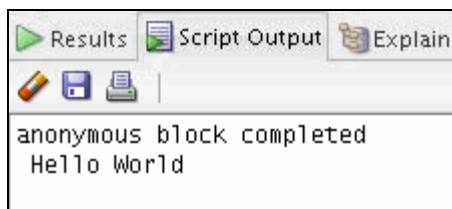
El bloque d se ejecuta correctamente.

2) Cree y ejecute un bloque anónimo simple cuya salida sea “Hello World”. Ejecute y guarde este script como lab_01_02_soln.sql.

Introduzca el siguiente código en el espacio de trabajo y pulse F5.

```
SET SERVEROUTPUT ON
BEGIN
DBMS_OUTPUT.PUT_LINE(' Hello World ');
END;
```

Debe aparecer la siguiente salida en el separador Script Output:



Haga clic en el botón Save. Seleccione la carpeta en la que desea guardar el archivo. Introduzca lab_01_02_soln.sql como nombre de archivo y haga clic en Save.

Prácticas y Soluciones de la Lección 2

Práctica 2: Declaración de Variables PL/SQL

En esta práctica, declarará variables PL/SQL.

1) Especifique identificadores válidos y no válidos:

- a) today
- b) last_name
- c) today's_date
- d) Number_of_days_in_February_this_year
- e) Isleap\$year
- f) #number
- g) NUMBER#
- h) number1to7

2) Identifique las declaraciones e inicializaciones de variables válidas y las no válidas:

- a) number_of_copies PLS_INTEGER;
- b) PRINTER_NAME constant VARCHAR2 (10);
- c) deliver_to VARCHAR2 (10):=Johnson;
- d) by_when DATE:= CURRENT_DATE+1;

3) Examine el siguiente bloque anónimo y seleccione la frase que sea verdadera de entre las siguientes.

```
DECLARE
    v_fname VARCHAR2(20);
    v_lname VARCHAR2(15) DEFAULT 'fernandez';
BEGIN
    DBMS_OUTPUT.PUT_LINE(v_fname || ' ' || v_lname);
END;
```

- a) El bloque se ejecuta correctamente y se imprime “fernandez”.
- b) El bloque produce un error porque se utiliza la variable fname sin inicializarse.
- c) El bloque se ejecuta correctamente y se imprime “null fernandez”.
- d) El bloque produce un error porque no se puede utilizar la palabra clave DEFAULT para inicializar una variable de tipo VARCHAR2.
- e) El bloque produce un error porque no se declara la variable v_fname.

Práctica 2: Declaración de Variables PL/SQL (continuación)

- 4) Modifique un bloque anónimo existente y guárdelo como nuevo script.
- Abra el script lab_01_02_soln.sql, creado en la Práctica 1.
 - En este bloque PL/SQL, declare las siguientes variables:
 - v_today del tipo DATE. Inicialice today con SYSDATE.
 - v_tomorrow del tipo today. Utilice el atributo %TYPE para declarar esta variable.
 - En la sección ejecutable:
 - Inicialice la variable v_tomorrow con una expresión, que calcule la fecha de mañana (agregue uno al valor de today)
 - Imprima el valor de v_today y tomorrow después de imprimir “Hello World”
 - Guarde el script como lab_02_04_soln.sql y ejecute.

La salida de ejemplo es como la siguiente (los valores de v_today y v_tomorrow serán diferentes para reflejar la fecha actual de hoy y la de mañana):

```
anonymous block completed
Hello World
TODAY IS : 05-JUN-09
TOMORROW IS : 06-JUN-09
```

- 5) Edite el script lab_02_04_soln.sql.
- Agregue código para crear dos variables de enlace: b_basic_percent y b_pf_percent. Ambas son del tipo NUMBER.
 - En la sección ejecutable del bloque PL/SQL, asigne los valores 45 y 12 a b_basic_percent y b_pf_percent, respectivamente.
 - Termine el bloque PL/SQL con “/” y muestre el valor de las variables de enlace con el comando PRINT.
 - Ejecute y guarde el script como lab_02_05_soln.sql. La salida de ejemplo es la siguiente:

```
anonymous block completed
b_basic_percent
--
45

b_pf_percent
--
12
```

Solución 2: Declaración de Variables PL/SQL

- 1) Especifique identificadores válidos y no válidos:

a) today	Válido
b) last_name	Válido
c) today's_date	No válido: no se permite el carácter “'”
d) Number_of_days_in_February_this_year	No válido: demasiado largo
e) Isleap\$year	Válido
f) #number	No válido: no puede empezar por “#”
g) NUMBER#	Válido
h) number1to7	Válido

- 2) Identifique las declaraciones e inicializaciones de variables válidas y las no válidas:

a) number_of_copies	PLS_INTEGER;	Válido
b) PRINTER_NAME	constant VARCHAR2(10);	No válido
c) deliver_to	VARCHAR2(10) :=Johnson;	No válido
d) by_when	DATE:= CURRENT_DATE+1;	Válido

La declaración b no es válida porque las variables constantes se deben inicializar durante la declaración.

La declaración c no es válida porque los literales de cadena deben aparecer entre comillas simples.

- 3) Examine el siguiente bloque anónimo y seleccione la frase que sea verdadera de entre las siguientes.

```
DECLARE
    v_fname VARCHAR2(20);
    v_lname VARCHAR2(15) DEFAULT 'fernandez';
BEGIN
    DBMS_OUTPUT.PUT_LINE(v_fname || ' ' || v_lname);
END;
```

- a) El bloque se ejecuta correctamente y se imprime “fernandez”.
 - b) El bloque produce un error porque se utiliza la variable fname sin inicializarse.
 - c) El bloque se ejecuta correctamente y se imprime “null fernandez”.
 - d) El bloque produce un error porque no se puede utilizar la palabra clave DEFAULT para inicializar una variable de tipo VARCHAR2.
 - e) El bloque produce un error porque no se declara la variable v_fname.
- a. **El bloque se ejecuta correctamente y se imprime “fernandez”.**

Solución 2: Declaración de Variables PL/SQL (continuación)

- 4) Modifique un bloque anónimo existente y guárdelo como nuevo script.

a) Abra el script lab_01_02_soln.sql, creado en la Práctica 1.

b) En el bloque PL/SQL, declare las siguientes variables:

1. Variable v_today del tipo DATE. Inicialice today con SYSDATE.

```
DECLARE
    v_today DATE:=SYSDATE;
```

2. Variable v_tomorrow del tipo today. Utilice el atributo %TYPE para declarar esta variable.

```
    v_tomorrow v_today%TYPE;
```

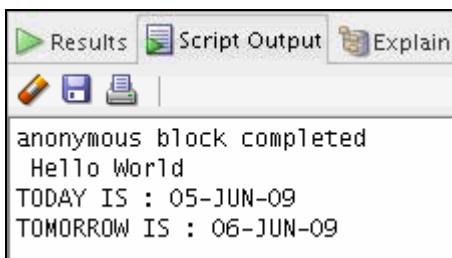
c) En la sección ejecutable:

1. Inicialice la variable v_tomorrow con una expresión, que calcule la fecha de mañana (agregue uno al valor de v_today)
2. Imprima el valor de v_today y v_tomorrow después de imprimir “Hello World”

```
BEGIN
    v_tomorrow:=v_today +1;
    DBMS_OUTPUT.PUT_LINE('Hello World ');
    DBMS_OUTPUT.PUT_LINE('TODAY IS : '|| v_today);
    DBMS_OUTPUT.PUT_LINE('TOMORROW IS : ' || v_tomorrow);
END;
```

- d) Guarde el script como lab_02_04_soln.sql y ejecute.

La salida de ejemplo es como la siguiente (los valores de v_today y v_tomorrow serán diferentes para reflejar la fecha actual de hoy y la de mañana):



Solución 2: Declaración de Variables PL/SQL (continuación)

5) Edite el script lab_02_04_soln.sql.

- a) Agregue código para crear dos variables de enlace: b_basic_percent y b_pf_percent. Ambas son del tipo NUMBER.

```
VARIABLE b_basic_percent NUMBER
VARIABLE b_pf_percent NUMBER
```

- b) En la sección ejecutable del bloque PL/SQL, asigne los valores 45 y 12 a b_basic_percent y b_pf_percent, respectivamente.

```
:b_basic_percent:=45;
:b_pf_percent:=12;
```

- c) Termine el bloque PL/SQL con “/” y muestre el valor de las variables de enlace con el comando PRINT.

```
/
PRINT b_basic_percent
PRINT b_pf_percent
```

O

```
PRINT
```

- d) Ejecute y guarde el script como lab_02_05_soln.sql. La salida de ejemplo es la siguiente:

```
anonymous block completed
b_basic_percent
--
45

b_pf_percent
--
12
```

Prácticas y Soluciones de la Lección 3

Práctica 3: Escritura de Sentencias Ejecutables

En esta práctica, examinará y escribirá sentencias ejecutables.

```
DECLARE
    v_weight      NUMBER(3) := 600;
    v_message     VARCHAR2(255) := 'Product 10012';
BEGIN
    DECLARE
        v_weight      NUMBER(3) := 1;
        v_message     VARCHAR2(255) := 'Product 11001';
        v_new_locn   VARCHAR2(50) := 'Europe';
    BEGIN
        v_weight := v_weight + 1;
        v_new_locn := 'Western ' || v_new_locn;
    1 →
        END;
        v_weight := v_weight + 1;
        v_message := v_message || ' is in stock';
        v_new_locn := 'Western ' || v_new_locn;
    2 →
    END;
/
```

- 1) Evalúe el bloque PL/SQL anterior y determine tanto el tipo de dato como el valor de cada una de las siguientes variables, según las reglas de los ámbitos.
 - a) Valor de `v_weight` en la posición 1:
 - b) Valor de `v_new_locn` en la posición 1:
 - c) Valor de `v_weight` en la posición 2:
 - d) Valor de `v_message` en la posición 2:
 - e) Valor de `v_new_locn` en la posición 2:

Práctica 3: Escritura de Sentencias Ejecutables (continuación)

```

DECLARE
    v_customer      VARCHAR2(50) := 'Womansport';
    v_credit_rating VARCHAR2(50) := 'EXCELLENT';
BEGIN
    DECLARE
        v_customer  NUMBER(7) := 201;
        v_name VARCHAR2(25) := 'Unisports';
    BEGIN
        v_credit_rating := 'GOOD';
        ...
    END;
    ...
END;

```

- 2) En el bloque PL/SQL anterior, determine el valor y el tipo de dato en cada uno de los siguientes casos.
 - a) Valor de `v_customer` en el bloque anidado:
 - b) Valor de `v_name` en el bloque anidado:
 - c) Valor de `v_credit_rating` en el bloque anidado:
 - d) Valor de `v_customer` en el bloque principal:
 - e) Valor de `v_name` en el bloque principal:
 - f) Valor de `v_credit_rating` en el bloque principal:
- 3) Utilice la misma sesión que utilizó para ejecutar las prácticas de la lección titulada “Declaración de Variables PL/SQL”. Si ha abierto una nueva sesión, ejecute `lab_02_05_soln.sql`. A continuación, edite `lab_02_05_soln.sql` de la siguiente forma:
 - a) Con la sintaxis de comentarios de una sola línea, comente las líneas que crean las variables de enlace y active SERVEROUTPUT.
 - b) Con los comentarios de varias líneas, comente en la sección ejecutable las líneas que asignan valores a las variables de enlace.
 - c) En la sección de declaraciones:
 1. Declare e inicialice dos variables temporales para sustituir las variables de enlace comentadas.
 2. Declare dos variables adicionales: `v_fname` del tipo VARCHAR2 y tamaño 15, y `v_emp_sal` del tipo NUMBER y tamaño 10.

Práctica 3: Escritura de Sentencias Ejecutables (continuación)

- d) Incluya la siguiente sentencia SQL en la sección ejecutable:

```
SELECT first_name, salary INTO v_fname, v_emp_sal  
  FROM employees WHERE employee_id=110;
```

- e) Cambie la línea que imprime “Hello World” para que imprima “Hello” y el nombre. A continuación, comente las líneas que muestran las fechas e imprima las variables de enlace.
- f) Calcule la aportación del empleado al fondo de previsión (PF). PF es el 12% del salario básico, y el salario básico es el 45% del salario. Utilice las variables locales para el cálculo. Intente utilizar sólo una expresión para calcular el valor de PF. Imprima el salario del empleado y su aportación a PF.
- g) Ejecute y guarde el script como lab_03_03_soln.sql. La salida de ejemplo es la siguiente:

```
anonymous block completed  
Hello John  
YOUR SALARY IS : 8200  
YOUR CONTRIBUTION TOWARDS PF:  
442.8
```

Solución 3: Escritura de Sentencias Ejecutables

En esta práctica, examinará y escribirá sentencias ejecutables.

```

DECLARE
    v_weight      NUMBER(3) := 600;
    v_message     VARCHAR2(255) := 'Product 10012';
BEGIN
    DECLARE
        v_weight      NUMBER(3) := 1;
        v_message     VARCHAR2(255) := 'Product 11001';
        v_new_locn   VARCHAR2(50) := 'Europe';
    BEGIN
        v_weight := v_weight + 1;
        v_new_locn := 'Western ' || v_new_locn;
    1  → END;
    v_weight := v_weight + 1;
    v_message := v_message || ' is in stock';
    v_new_locn := 'Western ' || v_new_locn;
    2  → END;
/

```

- 1) Evalúe el bloque PL/SQL anterior y determine tanto el tipo de dato como el valor de cada una de las siguientes variables, según las reglas de los ámbitos.
 - a) El valor de `v_weight` en la posición 1 es:
2
El tipo de dato es NUMBER.
 - b) El valor de `v_new_locn` en la posición 1 es:
Western Europe
El tipo de dato es VARCHAR2.
 - c) El valor de `v_weight` en la posición 2 es:
601
El tipo de dato es NUMBER.
 - d) El valor de `v_message` en la posición 2 es:
Product 10012 is in stock
El tipo de dato es VARCHAR2.
 - e) El valor de `v_new_locn` en la posición 2 es:
No válido porque v_new_locn no es visible fuera del subbloque

Solución 3: Escritura de Sentencias Ejecutables (continuación)

```

DECLARE
    v_customer      VARCHAR2(50) := 'Womansport';
    v_credit_rating VARCHAR2(50) := 'EXCELLENT';
BEGIN
    DECLARE
        v_customer  NUMBER(7) := 201;
        v_name      VARCHAR2(25) := 'Unisports';
    BEGIN
        v_credit_rating := 'GOOD';
        ...
    END;
    ...
END;

```

- 2) En el bloque PL/SQL anterior, determine el valor y el tipo de dato en cada uno de los siguientes casos.
- El valor de `v_customer` en el bloque anidado es:
201
El tipo de dato es NUMBER.
 - El valor de `v_name` en el bloque anidado es:
Unisports
El tipo de dato es VARCHAR2.
 - El valor de `v_credit_rating` en el bloque anidado es:
GOOD
El tipo de dato es VARCHAR2.
 - El valor de `v_customer` en el bloque principal es:
Womansport
El tipo de dato es VARCHAR2.
 - El valor de `v_name` en el bloque principal es:
Nulo. **name no está visible en el bloque principal y aparecerá un error.**
 - El valor de `v_credit_rating` en el bloque principal es:
EXCELLENT
El tipo de dato es VARCHAR2.
- 3) Utilice la misma sesión que utilizó para ejecutar las prácticas de la lección titulada “Declaración de Variables PL/SQL”. Si ha abierto una nueva sesión, ejecute `lab_02_05_soln.sql`. A continuación, edite `lab_02_05_soln.sql` de la siguiente forma:
- Con la sintaxis de comentarios de una sola línea, comente las líneas que crean las variables de enlace y active SERVEROUTPUT.

```

-- VARIABLE b_basic_percent NUMBER
-- VARIABLE b_pf_percent NUMBER
SET SERVEROUTPUT ON

```

Solución 3: Escritura de Sentencias Ejecutables (continuación)

- b) Con los comentarios de varias líneas, comente en la sección ejecutable las líneas que asignan valores a las variables de enlace.

```
/*:b_basic_percent:=45;
:b_pf_percent:=12;*/
```

- c) En la sección de declaraciones:

1. Declare e inicialice dos variables temporales para sustituir las variables de enlace comentadas.
2. Declare dos variables adicionales: v_fname del tipo VARCHAR2 y tamaño 15, y v_emp_sal del tipo NUMBER y tamaño 10.

```
DECLARE
    v_basic_percent NUMBER:=45;
    v_pf_percent NUMBER:=12;
    v_fname VARCHAR2(15);
    v_emp_sal NUMBER(10);
```

- d) Incluya la siguiente sentencia SQL en la sección ejecutable:

```
SELECT first_name, salary INTO v_fname, v_emp_sal
    FROM employees WHERE employee_id=110;
```

- e) Cambie la línea que imprime “Hello World” para que imprima “Hello” y el nombre. A continuación, comente las líneas que muestran las fechas e imprima las variables de enlace.

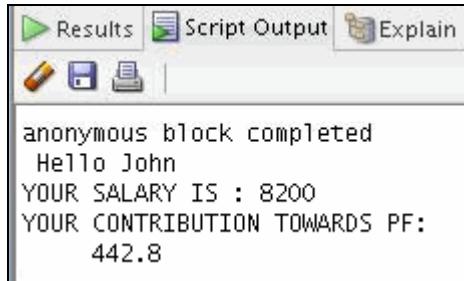
```
DBMS_OUTPUT.PUT_LINE('Hello '|| v_fname);
/* DBMS_OUTPUT.PUT_LINE('TODAY IS : '|| v_today);
DBMS_OUTPUT.PUT_LINE('TOMORROW IS : '|| v_tomorrow);*/
...
...
/
--PRINT b_basic_percent
--PRINT b_basic_percent
```

- f) Calcule la aportación del empleado al fondo de previsión (PF). PF es el 12% del salario básico, y el salario básico es el 45% del salario. Utilice las variables locales para el cálculo. Intente utilizar sólo una expresión para calcular el valor de PF. Imprima el salario del empleado y su aportación a PF.

```
DBMS_OUTPUT.PUT_LINE('YOUR SALARY IS : '||v_emp_sal);
DBMS_OUTPUT.PUT_LINE('YOUR CONTRIBUTION TOWARDS PF:
    '||v_emp_sal*v_basic_percent/100*v_pf_percent/100);
END;
```

Solución 3: Escritura de Sentencias Ejecutables (continuación)

- g) Ejecute y guarde el script como lab_03_03_soln.sql. La salida de ejemplo es la siguiente:



The screenshot shows the Oracle SQL Developer interface with the 'Results' tab selected. The output window displays the following text:
anonymous block completed
Hello John
YOUR SALARY IS : 8200
YOUR CONTRIBUTION TOWARDS PF:
442.8

Prácticas y Soluciones de la Lección 4

Práctica 4: Interacción con Oracle Server

En esta práctica, utilizará código PL/SQL para interactuar con Oracle Server.

- 1) Cree un bloque PL/SQL que seleccione el identificador de departamento superior en la tabla departments y lo almacene en la variable v_max_deptno. Muestre el identificador de departamento superior.
 - a) Declare una variable v_max_deptno de tipo NUMBER en la sección de declaraciones.
 - b) Inicie la sección ejecutable con la palabra clave BEGIN e incluya una sentencia SELECT para recuperar el valor máximo de department_id de la tabla departments.
 - c) Muestre v_max_deptno y termine el bloque ejecutable.
 - d) Ejecute y guarde el script como lab_04_01_soln.sql. La salida de ejemplo es la siguiente:

```
anonymous block completed  
The maximum department_id is : 270
```

- 2) Modifique el bloque PL/SQL creado en el paso 1 para insertar un nuevo departamento en la tabla departments.
 - a) Cargue el script lab_04_01_soln.sql. Declare dos variables: v_dept_name de tipo departments.department_name y v_dept_id de tipo NUMBER
Asigne 'Education' a v_dept_name en la sección de declaraciones.
 - b) Ya ha recuperado el número de departamento superior actual de la tabla departments. Agréguele 10 y asigne el resultado a v_dept_id.
 - c) Incluya una sentencia INSERT para insertar datos en las columnas department_name, department_id y location_id de la tabla departments.
Utilice valores en dept_name y dept_id para department_name y department_id, respectivamente, y utilice NULL para location_id.
 - d) Utilice el atributo SQL SQL%ROWCOUNT para mostrar el número de filas que se ven afectadas.
 - e) Ejecute una sentencia SELECT para comprobar si se ha insertado el nuevo departamento. Termine el bloque PL/SQL con "/" e incluya la sentencia SELECT en el script.

Práctica 4: Interacción con Oracle Server (continuación)

- f) Ejecute y guarde el script como lab_04_02_soln.sql. La salida de ejemplo es la siguiente:

```
anonymous block completed
The maximum department_id is : 270
SQL%ROWCOUNT gives 1

DEPARTMENT_ID      DEPARTMENT_NAME      MANAGER_ID      LOCATION_ID
-----              -----
280                Education           null            null

1 rows selected
```

- 3) En el paso 2, defina location_id en NULL. Cree un bloque PL/SQL que actualice location_id a 3000 para el nuevo departamento.

Nota: si ha terminado correctamente el paso 2, continúe con el paso 3a. De lo contrario, ejecute primero el script de solución /soln/sol_04_02.sql.

- Inicie el bloque ejecutable con la palabra clave BEGIN. Incluya la sentencia UPDATE para definir location_id en 3000 para el nuevo departamento (dept_id=280).
- Termine el bloque ejecutable con la palabra clave END. Termine el bloque PL/SQL con “/” e incluya una sentencia SELECT para mostrar el departamento que ha actualizado.
- Incluya una sentencia DELETE para suprimir el departamento agregado.
- Ejecute y guarde el script como lab_04_03_soln.sql. La salida de ejemplo es la siguiente:

```
anonymous block completed
DEPARTMENT_ID      DEPARTMENT_NAME      MANAGER_ID      LOCATION_ID
-----              -----
280                Education           null            3000

1 rows selected
1 rows deleted
```

Solución 4: Interacción con Oracle Server

En esta práctica, utilizará código PL/SQL para interactuar con Oracle Server.

- 1) Cree un bloque PL/SQL que seleccione el identificador de departamento superior en la tabla departments y lo almacene en la variable v_max_deptno. Muestre el identificador de departamento superior.
 - a) Declare una variable v_max_deptno de tipo NUMBER en la sección de declaraciones.

```
DECLARE
    v_max_deptno  NUMBER;
```

- b) Inicie la sección ejecutable con la palabra clave BEGIN e incluya una sentencia SELECT para recuperar el valor máximo de department_id de la tabla departments.

```
BEGIN
    SELECT MAX(department_id)  INTO v_max_deptno  FROM
        departments;
```

- c) Muestre v_max_deptno y termine el bloque ejecutable.

```
DBMS_OUTPUT.PUT_LINE('The maximum department_id is : ' ||
v_max_deptno);
END;
```

- d) Ejecute y guarde el script como lab_04_01_soln.sql. La salida de ejemplo es la siguiente:

```
anonymous block completed
The maximum department_id is : 270
```

- 2) Modifique el bloque PL/SQL creado en el paso 1 para insertar un nuevo departamento en la tabla departments.

- a) Cargue el script lab_04_01_soln.sql. Declare dos variables: v_dept_name de tipo departments.department_name y v_dept_id de tipo NUMBER
Asigne 'Education' a v_dept_name en la sección de declaraciones.

```
v_dept_name departments.department_name%TYPE := 'Education';
v_dept_id NUMBER;
```

Solución 4: Interacción con Oracle Server (continuación)

- b) Ya ha recuperado el número de departamento superior actual de la tabla departments. Agréguele 10 y asigne el resultado a v_dept_id.

```
v_dept_id := 10 + v_max_deptno;
```

- c) Incluya una sentencia INSERT para insertar datos en las columnas department_name, department_id y location_id de la tabla departments.

Utilice valores en dept_name y dept_id para department_name y department_id, respectivamente, y utilice NULL para location_id.

```
...
INSERT INTO departments (department_id, department_name,
location_id)
VALUES (v_dept_id, v_dept_name, NULL);
```

- d) Utilice el atributo SQL SQL%ROWCOUNT para mostrar el número de filas que se ven afectadas.

```
DBMS_OUTPUT.PUT_LINE (' SQL%ROWCOUNT gives ' || SQL%ROWCOUNT);
...
```

- e) Ejecute una sentencia SELECT para comprobar si se ha insertado el nuevo departamento. Termine el bloque PL/SQL con “/” e incluya la sentencia SELECT en el script.

```
...
/
SELECT * FROM departments WHERE department_id= 280;
```

- f) Ejecute y guarde el script como lab_04_02_soln.sql. La salida de ejemplo es la siguiente:

```
anonymous block completed
The maximum department_id is : 270
SQL%ROWCOUNT gives 1

DEPARTMENT_ID      DEPARTMENT_NAME      MANAGER_ID      LOCATION_ID
-----              -----
280                Education          null            null
1 rows selected
```

Solución 4: Interacción con Oracle Server (continuación)

- 3) En el paso 2, defina location_id en NULL. Cree un bloque PL/SQL que actualice location_id a 3000 para el nuevo departamento.

Nota: si ha terminado correctamente el paso 2, continúe con el paso 3a. De lo contrario, ejecute primero el script de solución /soln/sol_04_02.sql.

- a) Inicie el bloque ejecutable con la palabra clave BEGIN. Incluya la sentencia UPDATE para definir location_id en 3000 para el nuevo departamento (dept_id=280).

```
BEGIN
  UPDATE departments SET location_id=3000 WHERE
  department_id=280;
```

- b) Termine el bloque ejecutable con la palabra clave END. Termine el bloque PL/SQL con “/” e incluya una sentencia SELECT para mostrar el departamento que ha actualizado.

```
END;
/
SELECT * FROM departments WHERE department_id=280;
```

- c) Incluya una sentencia DELETE para suprimir el departamento agregado.

```
DELETE FROM departments WHERE department_id=280;
```

- d) Ejecute y guarde el script como lab_04_03_soln.sql. La salida de ejemplo es la siguiente:

anonymous block completed			
DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
280	Education		3000

1 rows selected

1 rows deleted

Prácticas y Soluciones de la Lección 5

Práctica 5: Escritura de las Estructuras de Control

En esta práctica, creará bloques PL/SQL que incorporen bucles y estructuras de control condicionales. En esta práctica se comprobará su conocimiento de varias sentencias IF y construcciones LOOP.

- 1) Ejecute el comando en el archivo `lab_05_01.sql` para crear la tabla `messages`. Escriba un bloque PL/SQL para insertar números en la tabla `messages`.
 - a) Inserte los números del 1 al 10, excluyendo el 6 y el 8.
 - b) Realice la confirmación antes del final del bloque.
 - c) Ejecute una sentencia SELECT para verificar que el bloque PL/SQL ha funcionado.

Resultado: debe ver la siguiente salida:

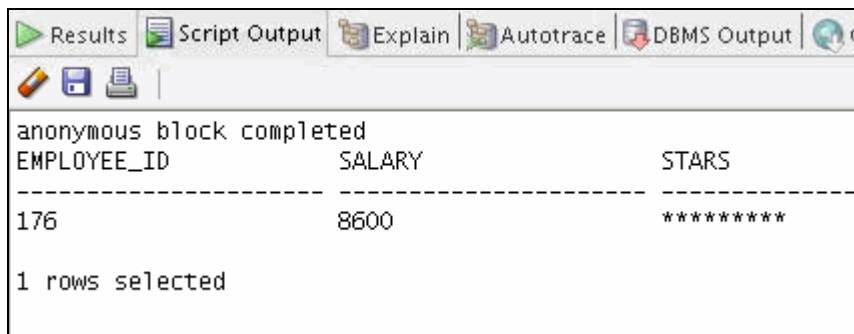
The screenshot shows the Oracle SQL Developer interface with the 'Results' tab selected. The output window displays the following text:
anonymous block completed
RESULTS

1
2
3
4
5
7
9
10
8 rows selected

- 2) Ejecute el script `lab_05_02.sql`. Este script crea una tabla `emp`, que es una réplica de la tabla `employees`. Modifica la tabla `emp` para agregar una nueva columna, `stars`, de tipo de dato VARCHAR2 y un tamaño de 50. Cree un bloque PL/SQL que inserte un asterisco en la columna `stars` por cada \$1000 del salario del empleado. Guarde el script como `lab_05_02_soln.sql`.
 - a) En la sección de declaraciones del bloque, declare una variable `v_emproto` de tipo `emp.employee_id` e inicialícela en 176. Declare una variable `v_asterisk` de tipo `emp.stars` e inicialícela en NULL. Cree una variable `v_sal` de tipo `emp.salary`.

Práctica 5: Escritura de las Estructuras de Control (continuación)

- b) En la sección ejecutable, escriba lógica para agregar un asterisco (*) a la cadena por cada 1.000 dólares del salario. Por ejemplo, si el empleado gana 8.000 dólares, la cadena de asteriscos debe incluir ocho asteriscos. Si el empleado gana 12.500 dólares, la cadena de asteriscos debe incluir 13 asteriscos.
- c) Actualice la columna stars para el empleado con la cadena de asteriscos. Realice la confirmación antes del final del bloque.
- d) Muestre la fila de la tabla emp para verificar que el bloque PL/SQL se ha ejecutado correctamente.
- e) Ejecute y guarde el script como lab_05_02_soln.sql. La salida es la siguiente:



The screenshot shows the Oracle SQL Developer interface with the 'Script Output' tab selected. The output window displays the results of an anonymous block execution. The output starts with 'anonymous block completed'. It then shows a table with three columns: EMPLOYEE_ID, SALARY, and STARS. A single row is displayed for employee ID 176, with a salary of 8600 and a corresponding star count of eight. The output concludes with '1 rows selected'.

EMPLOYEE_ID	SALARY	STARS
176	8600	*****

1 rows selected

Solución 5: Escritura de las Estructuras de Control

- 1) Ejecute el comando en el archivo lab_05_01.sql para crear la tabla messages.
Escriba un bloque PL/SQL para insertar números en la tabla messages.
 - a) Inserte los números del 1 al 10, excluyendo el 6 y el 8.
 - b) Realice la confirmación antes del final del bloque.

```
BEGIN
FOR i in 1..10 LOOP
  IF i = 6 or i = 8 THEN
    null;
  ELSE
    INSERT INTO messages(results)
    VALUES (i);
  END IF;
END LOOP;
COMMIT;
END;
/
```

- c) Ejecute una sentencia SELECT para verificar que el bloque PL/SQL ha funcionado.

```
SELECT * FROM messages;
```

Resultado: debe ver la siguiente salida:

The screenshot shows the Oracle SQL Developer interface with the 'Results' tab selected. The output window displays the following text:

```
anonymous block completed
RESULTS
-----
1
2
3
4
5
7
9
10
8 rows selected
```

Solución 5: Escritura de las Estructuras de Control (continuación)

- 2) Ejecute el script lab_05_02.sql. Este script crea una tabla emp, que es una réplica de la tabla employees. Modifica la tabla emp para agregar una nueva columna, stars, de tipo de dato VARCHAR2 y tamaño 50. Cree un bloque PL/SQL que inserte un asterisco en la columna stars por cada \$1000 del salario del empleado. Guarde el script como lab_05_02_soln.sql.
- a) En la sección de declaraciones del bloque PL/SQL, declare una variable v_empno de tipo emp.employee_id e inicialícela en 176. Declare una variable v_asterisk de tipo emp.stars e inicialícela en NULL. Cree una variable v_sal de tipo emp.salary.

```
DECLARE
    v_empno      emp.employee_id%TYPE := 176;
    v_asterisk   emp.stars%TYPE := NULL;
    v_sal        emp.salary%TYPE;
```

- b) En la sección ejecutable, escriba lógica para agregar un asterisco (*) a la cadena por cada 1.000 dólares del salario. Por ejemplo, si el empleado gana 8.000 dólares, la cadena de asteriscos debe incluir ocho asteriscos. Si el empleado gana 12.500 dólares, la cadena de asteriscos debe incluir 13 asteriscos.

```
BEGIN
    SELECT NVL(ROUND(salary/1000), 0) INTO v_sal
    FROM emp WHERE employee_id = v_empno;

    FOR i IN 1..v_sal
        LOOP
            v_asterisk := v_asterisk || '*';
    END LOOP;
```

- c) Actualice la columna stars para el empleado con la cadena de asteriscos. Realice la confirmación antes del final del bloque.

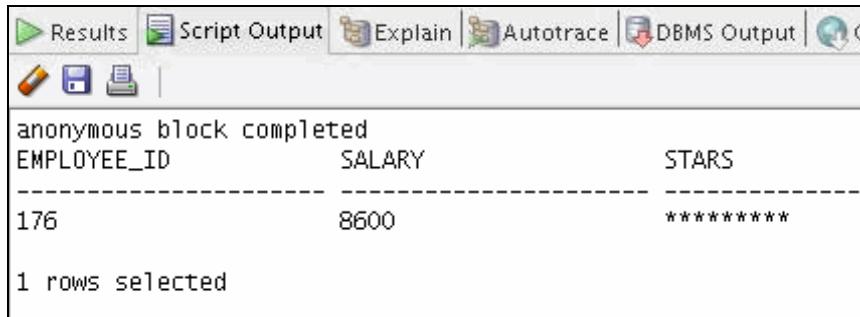
```
UPDATE emp SET stars = v_asterisk
WHERE employee_id = v_empno;
COMMIT;
END;
/
```

- d) Muestre la fila de la tabla emp para verificar que el bloque PL/SQL se ha ejecutado correctamente.

```
SELECT employee_id, salary, stars
FROM emp WHERE employee_id = 176;
```

Solución 5: Escritura de las Estructuras de Control (continuación)

- e) Ejecute y guarde el script como lab_05_02_soln.sql. La salida es la siguiente:



The screenshot shows the Oracle SQL Developer interface with the 'Results' tab selected. The output window displays the results of an anonymous block execution. The output is as follows:

```
anonymous block completed
EMPLOYEE_ID      SALARY      STARS
-----
176              8600        ****
1 rows selected
```

Prácticas y Soluciones de la Lección 6

Práctica 6: Trabajar con Tipos de Dato Compuestos

- 1) Escriba un bloque PL/SQL para imprimir información sobre un país determinado.
 - a) Declare un registro PL/SQL basado en la estructura de la tabla countries.
 - b) Declare una variable v_countryid. Asigne CA a v_countryid.
 - c) En la sección de declaraciones, utilice el atributo %ROWTYPE y declare la variable v_country_record del tipo countries.
 - d) En la sección ejecutable, obtenga toda la información de la tabla countries mediante v_countryid. Muestre la información seleccionada sobre el país. La salida de ejemplo es la siguiente:

```
anonymous block completed
Country Id: CA Country Name: Canada Region: 2
```

- e) Puede que desee ejecutar y probar el bloque PL/SQL para los países con los identificadores DE, UK y US.
- 2) Cree un bloque PL/SQL para recuperar los nombres de algunos departamentos de la tabla departments e imprimir el nombre de cada departamento en la pantalla, incorporando una matriz asociativa. Guarde el script como lab_06_02_soln.sql.
 - a) Declare una tabla INDEX BY dept_table_type del tipo departments.department_name. Declare una variable my_dept_table del tipo dept_table_type para almacenar temporalmente los nombres de los departamentos.
 - b) Declare dos variables: f_loop_count y v_deptno del tipo NUMBER. Asigne 10 a f_loop_count y 0 a v_deptno.
 - c) Con un bucle, recupere los nombres de 10 departamentos y almacene los nombres en la matriz asociativa. Empiece por department_id 10. Aumente v_deptno en 10 para cada iteración de bucle. La siguiente tabla muestra los valores department_id para los que se debe recuperar department_name.

DEPARTMENT_ID	DEPARTMENT_NAME
10	Administration
20	Marketing
30	Purchasing
40	Human Resources
50	Shipping
60	IT
70	Public Relations
80	Sales
90	Executive
100	Finance

Práctica 6: Trabajar con Tipos de Dato Compuestos (continuación)

- d) Si utiliza otro bucle, puede recuperar los nombres de los departamentos de la matriz asociativa y mostrarlos.
- e) Ejecute y guarde el script como lab_06_02_soln.sql. La salida es la siguiente:

```
anonymous block completed
Administration
Marketing
Purchasing
Human Resources
Shipping
IT
Public Relations
Sales
Executive
Finance
```

- 3) Modifique el bloque creado en la Práctica 2 para recuperar toda la información acerca de cada departamento de la tabla departments y mostrarla. Utilice una matriz asociativa con el método de tabla de registros INDEX BY.
 - a) Cargue el script lab_06_02_soln.sql.
 - b) Ha declarado que la matriz asociativa sea del tipo departments.department_name. Modifique la declaración de la matriz asociativa para almacenar temporalmente el número, el nombre y la ubicación de todos los departamentos. Utilice el atributo %ROWTYPE.
 - c) Modifique la sentencia SELECT para recuperar toda la información del departamento que está en la tabla departments y almacenarla en la matriz asociativa.
 - d) Si utiliza otro bucle, puede recuperar la información de los departamentos de la matriz asociativa y mostrarla.

La salida de ejemplo es la siguiente:

```
anonymous block completed
Department Number: 10 Department Name: Administration Manager Id: 200 Location Id: 1700
Department Number: 20 Department Name: Marketing Manager Id: 201 Location Id: 1800
Department Number: 30 Department Name: Purchasing Manager Id: 114 Location Id: 1700
Department Number: 40 Department Name: Human Resources Manager Id: 203 Location Id: 2400
Department Number: 50 Department Name: Shipping Manager Id: 121 Location Id: 1500
Department Number: 60 Department Name: IT Manager Id: 103 Location Id: 1400
Department Number: 70 Department Name: Public Relations Manager Id: 204 Location Id: 2700
Department Number: 80 Department Name: Sales Manager Id: 145 Location Id: 2500
Department Number: 90 Department Name: Executive Manager Id: 100 Location Id: 1700
Department Number: 100 Department Name: Finance Manager Id: 108 Location Id: 1700
```

Solución 6: Trabajar con Tipos de Dato Compuestos

1) Escriba un bloque PL/SQL para imprimir información sobre un país determinado.

- a) Declare un registro PL/SQL basado en la estructura de la tabla countries.
- b) Declare una variable v_countryid. Asigne CA a v_countryid.

```
SET SERVEROUTPUT ON
SET VERIFY OFF
DECLARE
    v_countryid varchar2(20) := 'CA';
```

- c) En la sección de declaraciones, utilice el atributo %ROWTYPE y declare la variable v_country_record del tipo countries.

```
v_country_record countries%ROWTYPE;
```

- d) En la sección ejecutable, obtenga toda la información de la tabla countries mediante v_countryid. Muestre la información seleccionada sobre el país. La salida de ejemplo es la siguiente:

```
BEGIN
    SELECT *
    INTO   v_country_record
    FROM   countries
    WHERE  country_id = UPPER(v_countryid);

    DBMS_OUTPUT.PUT_LINE ('Country Id: ' ||
                          v_country_record.country_id ||
                          ' Country Name: ' || v_country_record.country_name ||
                          ' Region: ' || v_country_record.region_id);

END;
```

```
anonymous block completed
Country Id: CA Country Name: Canada Region: 2
```

- e) Puede que desee ejecutar y probar el bloque PL/SQL para los países con los identificadores DE, UK y US.

Solución 6: Trabajar con Tipos de Dato Compuestos (continuación)

- 2) Cree un bloque PL/SQL para recuperar los nombres de algunos departamentos de la tabla departments e imprimir el nombre de cada departamento en la pantalla, incorporando una matriz asociativa. Guarde el script como lab_06_02_soln.sql.
- a) Declare una tabla INDEX BY dept_table_type del tipo departments.department_name. Declare una variable my_dept_table del tipo dept_table_type para almacenar temporalmente los nombres de los departamentos.

```
SET SERVEROUTPUT ON

DECLARE
    TYPE dept_table_type is table of
        departments.department_name%TYPE
    INDEX BY PLS_INTEGER;
    my_dept_table    dept_table_type;
```

- b) Declare dos variables: f_loop_count y v_deptno del tipo NUMBER. Asigne 10 a f_loop_count y 0 a v_deptno.

```
loop_count NUMBER (2):=10;
deptno      NUMBER (4):=0;
```

- c) Con un bucle, recupere los nombres de 10 departamentos y almacene los nombres en la matriz asociativa. Empiece por department_id 10. Aumente v_deptno en 10 para cada iteración del bucle. La siguiente tabla muestra department_id para los que se debe recuperar department_name y almacenar en la matriz asociativa.

DEPARTMENT_ID	DEPARTMENT_NAME
10	Administration
20	Marketing
30	Purchasing
40	Human Resources
50	Shipping
60	IT
70	Public Relations
80	Sales
90	Executive
100	Finance

Solución 6: Trabajar con Tipos de Dato Compuestos (continuación)

```
BEGIN
  FOR i IN 1..f_loop_count
  LOOP
    v_deptno:=v_deptno+10;
    SELECT department_name
    INTO my_dept_table(i)
    FROM departments
    WHERE department_id = v_deptno;
  END LOOP;
```

- d) Si utiliza otro bucle, puede recuperar los nombres de los departamentos de la matriz asociativa y mostrarlos.

```
FOR i IN 1..f_loop_count
LOOP
  DBMS_OUTPUT.PUT_LINE (my_dept_table(i));
END LOOP;
END;
```

- e) Ejecute y guarde el script como lab_06_02_soln.sql. La salida es la siguiente:

```
anonymous block completed
Administration
Marketing
Purchasing
Human Resources
Shipping
IT
Public Relations
Sales
Executive
Finance
```

- 3) Modifique el bloque creado en la Práctica 2 para recuperar toda la información acerca de cada departamento de la tabla departments y mostrarla. Utilice una matriz asociativa con el método de tabla de registros INDEX BY.
- Cargue el script lab_06_02_soln.sql.
 - Ha declarado que la matriz asociativa sea del tipo departments.department_name. Modifique la declaración de la matriz asociativa para almacenar temporalmente el número, el nombre y la ubicación de todos los departamentos. Utilice el atributo %ROWTYPE.

Solución 6: Trabajar con Tipos de Dato Compuestos (continuación)

```
SET SERVEROUTPUT ON

DECLARE
    TYPE dept_table_type is table of departments%ROWTYPE
    INDEX BY PLS_INTEGER;
    my_dept_table    dept_table_type;
    f_loop_count      NUMBER (2):=10;
    v_deptno         NUMBER (4):=0;
```

- c) Modifique la sentencia SELECT para recuperar toda la información del departamento que está en la tabla departments y almacenarla en la matriz asociativa.

```
BEGIN
    FOR i IN 1..f_loop_count
    LOOP
        v_deptno := v_deptno + 10;
        SELECT *
        INTO my_dept_table(i)
        FROM departments
        WHERE department_id = v_deptno;
    END LOOP;
```

- d) Si utiliza otro bucle, puede recuperar la información de los departamentos de la matriz asociativa y mostrarla.

```
FOR i IN 1..f_loop_count
LOOP
    DBMS_OUTPUT.PUT_LINE ('Department Number: ' ||
    my_dept_table(i).department_id
    || ' Department Name: ' ||
    my_dept_table(i).department_name
    || ' Manager Id: '|| my_dept_table(i).manager_id
    || ' Location Id: ' || my_dept_table(i).location_id);
END LOOP;
END;
```

Solución 6: Trabajar con Tipos de Dato Compuestos (continuación)

La salida de ejemplo es la siguiente:

```
anonymous block completed
Department Number: 10 Department Name: Administration Manager Id: 200 Location Id: 1700
Department Number: 20 Department Name: Marketing Manager Id: 201 Location Id: 1800
Department Number: 30 Department Name: Purchasing Manager Id: 114 Location Id: 1700
Department Number: 40 Department Name: Human Resources Manager Id: 203 Location Id: 2400
Department Number: 50 Department Name: Shipping Manager Id: 121 Location Id: 1500
Department Number: 60 Department Name: IT Manager Id: 103 Location Id: 1400
Department Number: 70 Department Name: Public Relations Manager Id: 204 Location Id: 2700
Department Number: 80 Department Name: Sales Manager Id: 145 Location Id: 2500
Department Number: 90 Department Name: Executive Manager Id: 100 Location Id: 1700
Department Number: 100 Department Name: Finance Manager Id: 108 Location Id: 1700
```

Prácticas y Soluciones de la Lección 7

Práctica 7-1: Uso de Cursos Explícitos

En esta lección, realizará dos ejercicios:

- Primero, utilizará un cursor explícito para procesar un número de filas de una tabla y llenar otra tabla con los resultados mediante un bucle FOR de cursor.
- Despues, escribirá un bloque PL/SQL que procese la información con dos cursos, incluido uno que utilice un parámetro.

- 1) Cree un bloque PL/SQL que realice las siguientes acciones:
 - a) En la sección de declaraciones, declare e inicialice una variable llamada `v_deptno` de tipo NUMBER. Asigne un valor de identificador de departamento válido (consulte los valores en la tabla del paso d).
 - b) Declare un cursor llamado `c_emp_cursor`, que recupere `last_name`, `salary` y `manager_id` de los empleados que trabajan en el departamento especificado en `v_deptno`.
 - c) En la sección ejecutable, utilice el bucle FOR de cursor para realizar operaciones en los datos recuperados. Si el salario del empleado es menor que 5.000 y si el identificador de superior es 101 o 124, aparece el mensaje “<<`last_name`>> Due for a raise”. De lo contrario, aparece el mensaje “<<`last_name`>> Not Due for a raise”.
 - d) Pruebe el bloque PL/SQL para los siguientes casos:

Department ID	Message
10	Whalen Due for a raise
20	Hartstein Not Due for a raise Fay Not Due for a raise
50	Weiss Not Due for a raise Fripp Not Due for a raise Kaufling Not Due for a raise Vollman Not Due for a raise. OConnell Due for a raise Grant Due for a raise
80	Russell Not Due for a raise Partners Not Due for a raise Errazuriz Not Due for a raise Cambrault Not Due for a raise . . . Livingston Not Due for a raise Johnson Not Due for a raise

Práctica 7-1: Uso de Cursos Explícitos (continuación)

- 2) A continuación, escriba un bloque PL/SQL que declare y utilice dos cursos: uno sin parámetro y otro con parámetro. El primer cursor recupera el número de departamento y el nombre del departamento de la tabla `departments` para todos los departamentos cuyos números de identificador sean menores que 100. El segundo cursor recibe el número de departamento como parámetro y recupera los detalles de los empleados que trabajan en ese departamento y cuyo `employee_id` sea menor que 120.
- Declare un cursor `c_dept_cursor` para recuperar `department_id` y `department_name` para los departamentos con `department_id` menor que 100. Ordene por `department_id`.
 - Declare otro cursor `c_emp_cursor` que tome el número de departamento como parámetro y recupere los siguientes datos de la tabla `employees`: `last_name`, `job_id`, `hire_date` y `salary` de los empleados que trabajan en ese departamento, con `employee_id` menor que 120.
 - Declare las variables que contienen los valores recuperados de cada cursor. Utilice el atributo `%TYPE` mientras declara las variables.
 - Abra `c_dept_cursor`, utilice un bucle simple y recupere los valores en las variables declaradas. Muestre el número y el nombre de departamento. Utilice el atributo de cursor adecuado para salir del bucle.
 - Abra `c_emp_cursor` transfiriendo el número de departamento actual como parámetro. Inicie otro bucle y recupere los valores de `emp_cursor` en las variables e imprima todos los detalles recuperados de la tabla `employees`.

Nota

- Compruebe si `c_emp_cursor` ya está abierto antes de abrirlo.
 - Utilice el atributo de cursor adecuado para la condición de salida.
 - Cuando termine el bucle, imprima una línea después de haber mostrado los detalles de cada departamento y cierre `c_emp_cursor`.
- Termine el primer bucle y cierre `c_dept_cursor`. A continuación, termine la sección ejecutable.
 - Ejecute el script. La salida de ejemplo es la siguiente:

Práctica 7-1: Uso de Cursosres Explícitos (continuación)

```
anonymous block completed
Department Number : 10 Department Name : Administration
-----
Department Number : 20 Department Name : Marketing
-----
Department Number : 30 Department Name : Purchasing
Raphaely    PU_MAN    07-DEC-94    11000
Khoo        PU_CLERK   18-MAY-95    3100
Baida       PU_CLERK   24-DEC-97    2900
Tobias      PU_CLERK   24-JUL-97    2800
Himuro      PU_CLERK   15-NOV-98    2600
Colmenares  PU_CLERK   10-AUG-99    2500
-----
Department Number : 40 Department Name : Human Resources
-----
Department Number : 50 Department Name : Shipping
-----
Department Number : 60 Department Name : IT
Hunold      IT_PROG    03-JAN-90    9000
Ernst       IT_PROG    21-MAY-91    6000
Austin      IT_PROG    25-JUN-97    4800
Pataballa   IT_PROG    05-FEB-98    4800
Lorentz     IT_PROG    07-FEB-99    4200
-----
Department Number : 70 Department Name : Public Relations
-----
Department Number : 80 Department Name : Sales
-----
Department Number : 90 Department Name : Executive
King        AD_PRES    17-JUN-87    24000
Kochhar     AD_VP      21-SEP-89    17000
De Haan     AD_VP      13-JAN-93    17000
```

Práctica 7-2: Uso de Cursos Explícitos (Opcional)

Si tiene tiempo, realice la siguiente práctica opcional. Creará un bloque PL/SQL que utiliza un cursor explícito para determinar los n salarios más altos de los empleados.

- 1) Ejecute el script `lab_07-2.sql` para crear la tabla `top_salaries` para almacenar los salarios de los empleados.
 - 2) En la sección de declaraciones, declare la variable `v_num` del tipo NUMBER que contenga el número n , que representa los n salarios más altos de la tabla `employees`. Por ejemplo, para ver los cinco salarios principales, introduzca 5. Declare otra variable `sal` del tipo `employees.salary`. Declare un cursor, `c_emp_cursor`, que recupere los salarios de los empleados en orden descendente. Recuerde que los salarios no deben estar duplicados.
 - 3) En la sección ejecutable, abra el bucle, recupere los n salarios principales e insértelos en la tabla `top_salaries`. Puede utilizar un bucle simple para realizar operaciones con los datos. Además, utilice los atributos `%ROWCOUNT` y `%FOUND` para la condición de salida.
- Nota:** asegúrese de agregar una condición de salida para evitar un bucle infinito.
- 4) Después de insertarlos en la tabla `top_salaries`, muestre las filas con una sentencia SELECT. La salida que se muestra representa los cinco salarios más altos de la tabla `employees`.

SALARY

24000
17000
17000
14000
13500

- 5) Pruebe distintos casos especiales, como `v_num = 0` o donde `v_num` es mayor que el número de empleados de la tabla `employees`. Vacíe la tabla `top_salaries` después de cada prueba.

Solución 7-1: Uso de Cursos Explícitos

En esta lección, realizará dos ejercicios:

- Primero, utilizará un cursor explícito para procesar un número de filas de una tabla y llenar otra tabla con los resultados mediante un bucle FOR de cursor.
- Después, escribirá un bloque PL/SQL que procese la información con dos cursos, incluido uno que utilice un parámetro.

1) Cree un bloque PL/SQL que realice las siguientes acciones:

- a) En la sección de declaraciones, declare e inicialice una variable llamada v_deptno de tipo NUMBER. Asigne un valor de identificador de departamento válido (consulte los valores en la tabla del paso d).

```
DECLARE
  v_deptno NUMBER := 10;
```

- b) Declare un cursor llamado c_emp_cursor, que recupere last_name, salary y manager_id de los empleados que trabajan en el departamento especificado en v_deptno.

```
CURSOR c_emp_cursor IS
  SELECT last_name, salary, manager_id
  FROM employees
  WHERE department_id = v_deptno;
```

- c) En la sección ejecutable, utilice el bucle FOR de cursor para realizar operaciones en los datos recuperados. Si el salario del empleado es menor que 5.000 y si el identificador de superior es 101 o 124, aparece el mensaje “<<last_name>> Due for a raise”. De lo contrario, aparece el mensaje “<<last_name>> Not Due for a raise”.

```
BEGIN
  FOR emp_record IN c_emp_cursor
  LOOP
    IF emp_record.salary < 5000 AND (emp_record.manager_id=101
    OR emp_record.manager_id=124) THEN
      DBMS_OUTPUT.PUT_LINE (emp_record.last_name || ' Due for
a raise');
    ELSE
      DBMS_OUTPUT.PUT_LINE (emp_record.last_name || ' Not Due
for a raise');
    END IF;
  END LOOP;
END;
```

Solución 7-1: Uso de Cursos Explícitos (continuación)

- d) Pruebe el bloque PL/SQL para los siguientes casos:

Department ID	Message
10	Whalen Due for a raise
20	Hartstein Not Due for a raise Fay Not Due for a raise
50	Weiss Not Due for a raise Fripp Not Due for a raise Kaufling Not Due for a raise Vollman Not Due for a raise. OConnell Due for a raise Grant Due for a raise
80	Russell Not Due for a raise Partners Not Due for a raise Errazuriz Not Due for a raise Cambrault Not Due for a raise . . . Livingston Not Due for a raise Johnson Not Due for a raise

- 2) A continuación, escriba un bloque PL/SQL que declare y utilice dos cursos: uno sin parámetro y otro con parámetro. El primer cursor recupera el número de departamento y el nombre del departamento de la tabla departments para todos los departamentos cuyos números de identificador sean menores que 100. El segundo cursor recibe el número de departamento como parámetro y recupera los detalles de los empleados que trabajan en ese departamento y cuyo employee_id sea menor que 120.
- a) Declare un cursor `c_dept_cursor` para recuperar `department_id` y `department_name` para los departamentos con `department_id` menor que 100. Ordene por `department_id`.

```
DECLARE
  CURSOR c_dept_cursor IS
    SELECT department_id, department_name
    FROM departments
    WHERE department_id < 100
    ORDER BY department_id;
```

Solución 7-1: Uso de Cursos Explícitos (continuación)

- b) Declare otro cursor `c_emp_cursor` que tome el número de departamento como parámetro y recupere los siguientes datos de la tabla `employees`: `last_name`, `job_id`, `hire_date` y `salary` de los empleados que trabajan en ese departamento, con `employee_id` menor que 120.

```
CURSOR c_emp_cursor(v_deptno NUMBER) IS
    SELECT last_name, job_id, hire_date, salary
    FROM employees
    WHERE department_id = v_deptno
    AND employee_id < 120;
```

- c) Declare las variables que contienen los valores recuperados de cada cursor. Utilice el atributo `%TYPE` mientras declara las variables.

```
v_current_deptno departments.department_id%TYPE;
v_current_dname departments.department_name%TYPE;
v_ename employees.last_name%TYPE;
v_job employees.job_id%TYPE;
v_hiredate employees.hire_date%TYPE;
v_sal employees.salary%TYPE;
```

- d) Abra `c_dept_cursor`, utilice un bucle simple y recupere los valores en las variables declaradas. Muestre el número y el nombre de departamento. Utilice el atributo de cursor adecuado para salir del bucle.

```
BEGIN
    OPEN c_dept_cursor;
    LOOP
        FETCH c_dept_cursor INTO v_current_deptno,
        v_current_dname;
        EXIT WHEN c_dept_cursor%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE ('Department Number : ' ||
        v_current_deptno || ' Department Name : ' ||
        v_current_dname);
```

Solución 7-1: Uso de Cursos Explícitos (continuación)

- e) Abra c_emp_cursor transfiriendo el número de departamento actual como parámetro. Inicie otro bucle y recupere los valores de emp_cursor en las variables e imprima todos los detalles recuperados de la tabla employees.

Nota

- Compruebe si c_emp_cursor ya está abierto antes de abrirlo.
- Utilice el atributo de cursor adecuado para la condición de salida.
- Cuando termine el bucle, imprima una línea después de haber mostrado los detalles de cada departamento y cierre c_emp_cursor.

```

IF c_emp_cursor%ISOPEN THEN
    CLOSE c_emp_cursor;
END IF;
OPEN c_emp_cursor (v_current_deptno);
LOOP
    FETCH c_emp_cursor INTO v_ename,v_job,v_hiredate,v_sal;
    EXIT WHEN c_emp_cursor%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE (v_ename || ' ' || v_job
                           || ' ' || v_hiredate || ' ' ||
                           v_sal);
END LOOP;
DBMS_OUTPUT.PUT_LINE ('-----');
CLOSE c_emp_cursor;

```

- f) Termine el primer bucle y cierre c_dept_cursor. A continuación, termine la sección ejecutable.

```

    END LOOP;
    CLOSE c_dept_cursor;
END;

```

- g) Ejecute el script. La salida de ejemplo es la siguiente:

Solución 7-1: Uso de Cursos Explícitos (continuación)

```
anonymous block completed
Department Number : 10 Department Name : Administration
-----
Department Number : 20 Department Name : Marketing
-----
Department Number : 30 Department Name : Purchasing
Raphaely    PU_MAN    07-DEC-94    11000
Khoo        PU_CLERK   18-MAY-95    3100
Baida       PU_CLERK   24-DEC-97    2900
Tobias      PU_CLERK   24-JUL-97    2800
Himuro      PU_CLERK   15-NOV-98    2600
Colmenares  PU_CLERK   10-AUG-99    2500
-----
Department Number : 40 Department Name : Human Resources
-----
Department Number : 50 Department Name : Shipping
-----
Department Number : 60 Department Name : IT
Hunold      IT_PROG    03-JAN-90    9000
Ernst       IT_PROG    21-MAY-91    6000
Austin      IT_PROG    25-JUN-97    4800
Pataballa   IT_PROG    05-FEB-98    4800
Lorentz     IT_PROG    07-FEB-99    4200
-----
Department Number : 70 Department Name : Public Relations
-----
Department Number : 80 Department Name : Sales
-----
Department Number : 90 Department Name : Executive
King        AD_PRES    17-JUN-87    24000
Kochhar     AD_VP      21-SEP-89    17000
De Haan     AD_VP      13-JAN-93    17000
```

Solución 7-2: Uso de Cursos Explícitos (Opcional)

Si tiene tiempo, realice el siguiente ejercicio opcional. Creará un bloque PL/SQL que utiliza un cursor explícito para determinar los n salarios más altos de los empleados.

- 1) Ejecute el script lab_07-2.sql para crear una nueva tabla top_salaries para almacenar los salarios de los empleados.
- 2) En la sección de declaraciones, declare la variable v_num del tipo NUMBER que contenga el número n que representa los n salarios más altos de la tabla employees. Por ejemplo, para ver los cinco salarios principales, introduzca 5. Declare otra variable v_sal del tipo employees.salary. Declare un cursor, c_emp_cursor, que recupere los salarios de los empleados en orden descendente. Recuerde que los salarios no deben estar duplicados.

```
DECLARE
    v_num          NUMBER(3) := 5;
    v_sal          employees.salary%TYPE;
    CURSOR         c_emp_cursor IS
        SELECT      salary
        FROM        employees
        ORDER BY    salary DESC;
```

- 3) En la sección ejecutable, abra el bucle, recupere los n salarios principales e insértelos en la tabla top_salaries. Puede utilizar un bucle simple para realizar operaciones con los datos. Además, utilice los atributos %ROWCOUNT y %FOUND para la condición de salida.

Nota: asegúrese de agregar una condición de salida para evitar un bucle infinito.

```
BEGIN
    OPEN c_emp_cursor;
    FETCH c_emp_cursor INTO v_sal;
    WHILE c_emp_cursor%ROWCOUNT <= v_num AND c_emp_cursor%FOUND
LOOP
    INSERT INTO top_salaries (salary)
    VALUES (v_sal);
    FETCH c_emp_cursor INTO v_sal;
END LOOP;
CLOSE c_emp_cursor;
END;
```

Solución 7-2: Uso de Cursos Explícitos (Opcional) (continuación)

- 4) Después de insertarlos en la tabla `top_salaries`, muestre las filas con una sentencia SELECT. La salida que se muestra representa los cinco salarios más altos de la tabla `employees`.

```
/  
SELECT * FROM top_salaries;
```

La salida de ejemplo es la siguiente:

SALARY

24000
17000
17000
14000
13500

- 5) Pruebe distintos casos especiales, como `v_num = 0` o donde `v_num` es mayor que el número de empleados de la tabla `employees`. Vacíe la tabla `top_salaries` después de cada prueba.

Prácticas y Soluciones de la Lección 8

Práctica 8-1: Manejo de Excepciones Predefinidas

En esta práctica, escribirá un bloque PL/SQL que aplique una excepción predefinida para procesar un único registro a la vez. El bloque PL/SQL seleccionará el nombre del empleado con un valor de salario determinado.

- 1) Ejecute el comando en el archivo `lab_05_01.sql` para volver a crear la tabla `messages`.
- 2) En la sección de declaraciones, declare dos variables: `v_ename` del tipo `employees.last_name` y `v_emp_sal` del tipo `employees.salary`. Inicialice la última en 6000.
- 3) En la sección ejecutable, recupere los apellidos de los empleados cuyos salarios son iguales al valor de `v_emp_sal`. Si el salario introducido devuelve sólo una fila, inserte en la tabla `messages` el nombre y el importe del salario del empleado.
Nota: no utilice cursores explícitos.
- 4) Si el salario introducido no devuelve ninguna fila, maneje la excepción con un manejador de excepciones adecuado e inserte en la tabla `messages` el mensaje “No employee with a salary of <salary>”.
- 5) Si el salario introducido devuelve varias filas, maneje la excepción con un manejador de excepciones adecuado e inserte en la tabla `messages` el mensaje “More than one employee with a salary of <salary>”.
- 6) Maneje cualquier otra excepción con un manejador de excepciones adecuado e inserte en la tabla `messages` el mensaje “Some other error occurred”.
- 7) Muestre las filas de la tabla `messages` para comprobar si el bloque PL/SQL se ha ejecutado correctamente. La salida es la siguiente:

RESULTS

More than one employee with a salary of 6000
1 rows selected

- 8) Cambie el valor inicializado de `v_emp_sal` a 2000 y vuelva a ejecutar. La salida es la siguiente:

Práctica 8-1: Manejo de Excepciones Predefinidas (continuación)

RESULTS

More than one employee with a salary of 6000
No employee with a salary of 2000

2 rows selected

Práctica 8-2: Manejo de Excepciones de Oracle Server Estándar

En esta práctica, escribirá un bloque PL/SQL que declare una excepción para el error de Oracle Server ORA-02292 (integrity constraint violated - child record found). El bloque comprobará la excepción y mostrará el mensaje de error.

- 1) En la sección de declaraciones, declare una excepción `e_childrecord_exists`. Asocie la excepción declarada al error de Oracle Server estándar –02292.
- 2) En la sección ejecutable, muestre “Deleting department 40....” Incluya una sentencia `DELETE` para suprimir el departamento con `department_id` 40.
- 3) Incluya una sección de excepciones para manejar la excepción `e_childrecord_exists` y muestre el mensaje adecuado.

La salida de ejemplo es la siguiente:

```
anonymous block completed
Deleting department 40.....
Cannot delete this department. There are employees in this department (child records exist.)
```

Solución 8-1: Manejo de Excepciones Predefinidas

En esta práctica, escribirá un bloque PL/SQL que aplique una excepción predefinida para procesar un único registro a la vez. El bloque PL/SQL seleccionará el nombre del empleado con un valor de salario determinado.

- 1) Ejecute el comando en el archivo `lab_05_01.sql` para volver a crear la tabla `messages`.
- 2) En la sección de declaraciones, declare dos variables: `v_ename` del tipo `employees.last_name` y `v_emp_sal` del tipo `employees.salary`. Inicialice la última en 6000.

```
DECLARE
    v_ename      employees.last_name%TYPE;
    v_emp_sal   employees.salary%TYPE := 6000;
```

- 3) En la sección ejecutable, recupere los apellidos de los empleados cuyos salarios son iguales al valor de `v_emp_sal`. Si el salario introducido devuelve sólo una fila, inserte en la tabla `messages` el nombre y el importe del salario del empleado.
- Nota:** no utilice cursosres explícitos.

```
BEGIN
    SELECT last_name
    INTO      v_ename
    FROM      employees
    WHERE     salary = v_emp_sal;
    INSERT INTO messages (results)
    VALUES (v_ename || ' - ' || v_emp_sal);
```

- 4) Si el salario introducido no devuelve ninguna fila, maneje la excepción con un manejador de excepciones adecuado e inserte en la tabla `messages` el mensaje “No employee with a salary of <salary>”.

```
EXCEPTION
    WHEN no_data_found THEN
        INSERT INTO messages (results)
        VALUES ('No employee with a salary of ' ||
                TO_CHAR(v_emp_sal));
```

Solución 8-1: Manejo de Excepciones Predefinidas (continuación)

- 5) Si el salario introducido devuelve varias filas, maneje la excepción con un manejador de excepciones adecuado e inserte en la tabla messages el mensaje “More than one employee with a salary of <salary>”.

```
WHEN too_many_rows THEN
    INSERT INTO messages (results)
    VALUES ('More than one employee with a salary of ' ||
            TO_CHAR(v_emp_sal));
```

- 6) Maneje cualquier otra excepción con un manejador de excepciones adecuado e inserte en la tabla messages el mensaje “Some other error occurred”.

```
WHEN others THEN
    INSERT INTO messages (results)
    VALUES ('Some other error occurred.');
END;
```

- 7) Muestre las filas de la tabla messages para comprobar si el bloque PL/SQL se ha ejecutado correctamente.

```
/  
SELECT * FROM messages;
```

La salida es la siguiente:

RESULTS

More than one employee with a salary of 6000
1 rows selected

- 8) Cambie el valor inicializado de v_emp_sal a 2000 y vuelva a ejecutar. La salida es la siguiente:

RESULTS

More than one employee with a salary of 6000
No employee with a salary of 2000
2 rows selected

Solución 8-2: Manejo de Excepciones de Oracle Server Estándar

En esta práctica, escribirá un bloque PL/SQL que declare una excepción para el error de Oracle Server ORA-02292 (integrity constraint violated - child record found). El bloque comprobará la excepción y mostrará el mensaje de error.

- 1) En la sección de declaraciones, declare una excepción e_childrecord_exists. Asocie la excepción declarada al error de Oracle Server estándar -02292.

```
SET SERVEROUTPUT ON
DECLARE
    e_childrecord_exists EXCEPTION;
    PRAGMA EXCEPTION_INIT(e_childrecord_exists, -02292);
```

- 2) En la sección ejecutable, muestre “Deleting department 40....” Incluya una sentencia DELETE para suprimir el departamento con department_id 40.

```
BEGIN
    DBMS_OUTPUT.PUT_LINE(' Deleting department 40.....');
    delete from departments where department_id=40;
```

- 3) Incluya una sección de excepciones para manejar la excepción e_childrecord_exists y muestre el mensaje adecuado.

```
EXCEPTION
    WHEN e_childrecord_exists THEN
        DBMS_OUTPUT.PUT_LINE(' Cannot delete this department. There
        are employees in this department (child records exist.) ');
END;
```

La salida de ejemplo es la siguiente:

anonymous block completed	
Deleting department 40.....	
Cannot delete this department. There are employees in this department (child records exist.)	

Prácticas y Soluciones de la Lección 9

Práctica 9: Creación y Uso de Procedimientos Almacenados

En esta práctica, modificará scripts existentes para crear y utilizar procedimientos almacenados.

- 1) Cargue el script `sol_02_04.sql` de la carpeta `/home/oracle/plsf/soln/`.
 - a) Modifique el script para convertir el bloque anónimo en un procedimiento denominado `greet`. (**Indicación:** elimine también el comando `SET SERVEROUTPUT ON`).
 - b) Ejecute el script para crear el procedimiento. La salida resultante debe ser como la siguiente:

The screenshot shows the Oracle SQL Developer interface with the 'Script Output' tab selected. The output window displays the message 'PROCEDURE greet Compiled.'

- c) Guarde este script como `lab_09_01_soln.sql`.
- d) Haga clic en el botón Clear para limpiar el espacio de trabajo.
- e) Cree y ejecute un bloque anónimo para llamar al procedimiento `greet`. (**Indicación:** asegúrese de activar SERVEROUTPUT al principio del bloque).

La salida debe ser similar a la siguiente:

The screenshot shows the Oracle SQL Developer interface with the 'Script Output' tab selected. The output window displays the results of an anonymous block execution:
anonymous block completed
Hello World
TODAY IS : 10-JUL-09
TOMORROW IS : 11-JUL-09

- 2) Modifique el script `lab_09_01_soln.sql` de la siguiente forma:

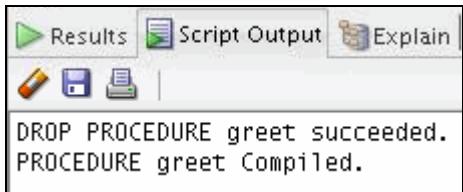
- a) Borre el procedimiento `greet` emitiendo el siguiente comando:

```
DROP PROCEDURE greet;
```

- b) Modifique el procedimiento para aceptar un argumento de tipo VARCHAR2. Llame al argumento `p_name`.
- c) Imprima Hello <name> (es decir, el contenido del argumento) en lugar de Hello World.

Práctica 9: Creación y Uso de Procedimientos Almacenados (continuación)

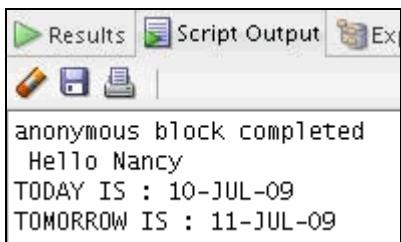
- d) Guarde el script como lab_09_02_soln.sql.
- e) Ejecute el script para crear el procedimiento. La salida resultante debe ser como la siguiente:



A screenshot of the Oracle SQL Developer interface. The title bar says "SQL Developer". Below it is a toolbar with icons for Run, Stop, Script, and Explain. The main area shows the following text:
DROP PROCEDURE greet succeeded.
PROCEDURE greet Compiled.

- f) Cree y ejecute un bloque anónimo para llamar al procedimiento greet con un valor de parámetro. El bloque también debe producir la salida.

La salida de ejemplo debe ser similar a la siguiente:



A screenshot of the Oracle SQL Developer interface. The title bar says "SQL Developer". Below it is a toolbar with icons for Run, Stop, Script, and Explain. The main area shows the following text:
anonymous block completed
Hello Nancy
TODAY IS : 10-JUL-09
TOMORROW IS : 11-JUL-09

Solución 9: Creación y Uso de Procedimientos Almacenados

En esta práctica, modificará scripts existentes para crear y utilizar procedimientos almacenados.

- 1) Cargue el script `sol_02_04.sql` de la carpeta `/home/oracle/plsf/soln/`.
 - a) Modifique el script para convertir el bloque anónimo en un procedimiento denominado `greet`. (**Indicación:** elimine también el comando `SET SERVEROUTPUT ON`).

```
CREATE PROCEDURE greet IS
    v_today DATE:=SYSDATE;
    V_tomorrow today%TYPE;
...

```

- b) Ejecute el script para crear el procedimiento. La salida resultante debe ser como la siguiente:

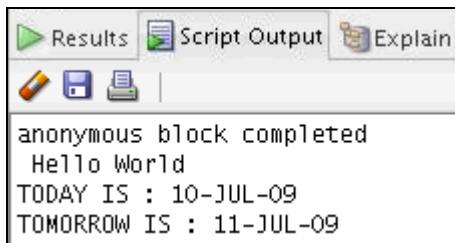


- c) Guarde este script como `lab_09_01_soln.sql`.
- d) Haga clic en el botón Clear para limpiar el espacio de trabajo.
- e) Cree y ejecute un bloque anónimo para llamar al procedimiento `greet`. (**Indicación:** asegúrese de activar SERVEROUTPUT al principio del bloque).

```
SET SERVEROUTPUT ON

BEGIN
    greet;
END;
```

La salida debe ser similar a la siguiente:



Solución 9: Creación y Uso de Procedimientos Almacenados (continuación)

- 2) Modifique el script lab_09_01_soln.sql de la siguiente forma:

- a) Borre el procedimiento greet emitiendo el siguiente comando:

```
DROP PROCEDURE greet;
```

- b) Modifique el procedimiento para aceptar un argumento de tipo VARCHAR2.
Llame al argumento p_name.

```
CREATE PROCEDURE greet(p_name VARCHAR2) IS
    v_today DATE:=SYSDATE;
    V_tomorrow today%TYPE;
```

- c) Imprima Hello <name> en lugar de Hello World.

```
BEGIN
    v_tomorrow:=v_today +1;
    DBMS_OUTPUT.PUT_LINE('Hello'|| p_name);
    ...

```

- d) Guarde el script como lab_09_02_soln.sql.
e) Ejecute el script para crear el procedimiento. La salida resultante debe ser como la siguiente:

```
DROP PROCEDURE greet succeeded.
PROCEDURE greet Compiled.
```

- f) Cree y ejecute un bloque anónimo para llamar al procedimiento greet con un valor de parámetro. El bloque también debe producir la salida.

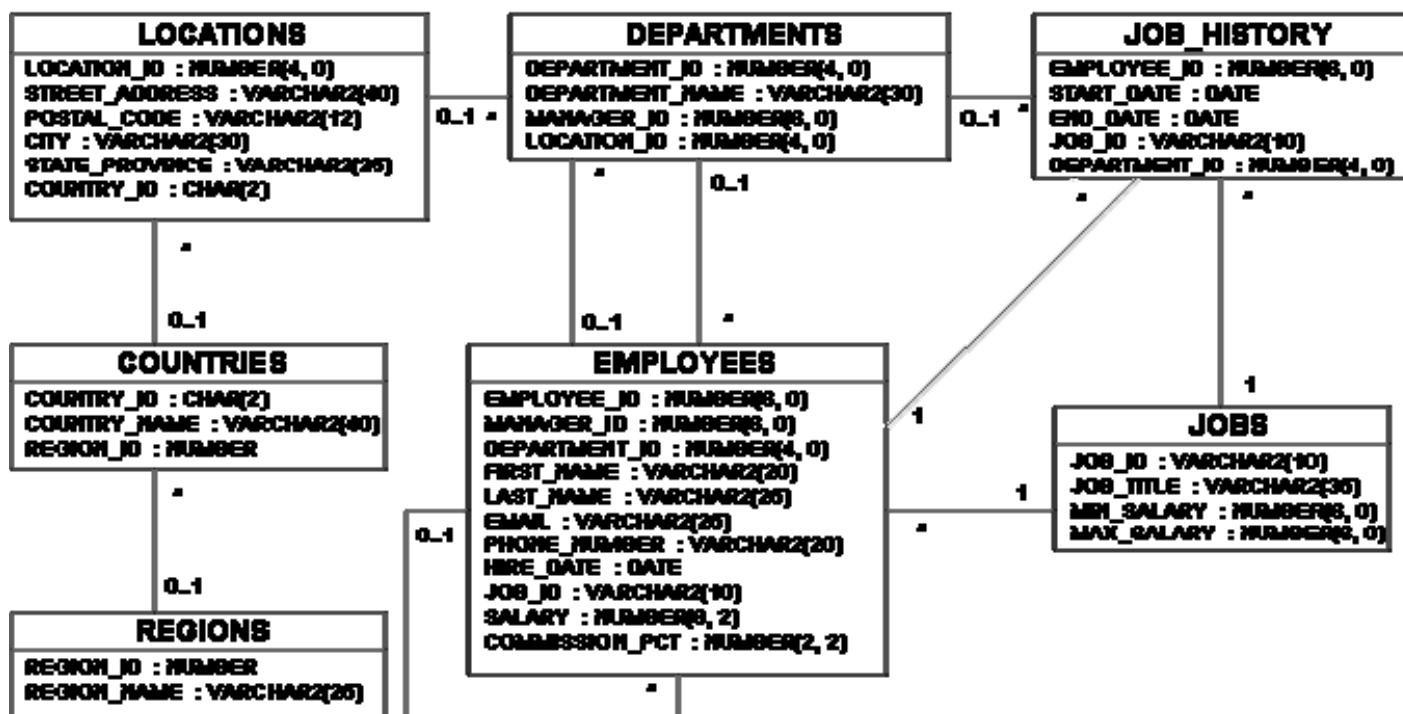
```
SET SERVEROUTPUT ON;
BEGIN
    greet('Nancy');
END;
```

La salida de ejemplo debe ser similar a la siguiente:

```
anonymous block completed
Hello Nancy
TODAY IS : 10-JUL-09
TOMORROW IS : 11-JUL-09
```

Apéndice B: Descripciones de las Tablas y Datos

DIAGRAMA DE ENTIDAD/RELACIÓN



Tablas del Esquema

```
SELECT * FROM tab;
```

TNAME	TABTYPE	CLUSTERID
COUNTRIES	TABLE	
DEPARTMENTS	TABLE	
EMPLOYEES	TABLE	
EMP_DETAILS_VIEW	VIEW	
JOBS	TABLE	
JOB_HISTORY	TABLE	
LOCATIONS	TABLE	
REGIONS	TABLE	

8 rows selected.

Tabla regions

DESCRIBE regions

Name	Null?	Type
REGION_ID	NOT NULL	NUMBER
REGION_NAME		VARCHAR2(25)

SELECT * FROM regions;

REGION_ID	REGION_NAME
1	Europe
2	Americas
3	Asia
4	Middle East and Africa

Tabla countries

DESCRIBE countries

Name	Null?	Type
COUNTRY_ID	NOT NULL	CHAR(2)
COUNTRY_NAME		VARCHAR2(40)
REGION_ID		NUMBER

SELECT * FROM countries;

CO	COUNTRY_NAME	REGION_ID
AR	Argentina	2
AU	Australia	3
BE	Belgium	1
BR	Brazil	2
CA	Canada	2
CH	Switzerland	1
CN	China	3
DE	Germany	1
DK	Denmark	1
EG	Egypt	4
FR	France	1
HK	HongKong	3
IL	Israel	4
IN	India	3
CO	COUNTRY_NAME	REGION_ID
IT	Italy	1
JP	Japan	3
KW	Kuwait	4
MX	Mexico	2
NG	Nigeria	4
NL	Netherlands	1
SG	Singapore	3
UK	United Kingdom	1
US	United States of America	2
ZM	Zambia	4
ZW	Zimbabwe	4

25 rows selected.

Tabla locations

DESCRIBE locations;

Name	Null?	Type
LOCATION_ID	NOT NULL	NUMBER(4)
STREET_ADDRESS		VARCHAR2(40)
POSTAL_CODE		VARCHAR2(12)
CITY	NOT NULL	VARCHAR2(30)
STATE_PROVINCE		VARCHAR2(25)
COUNTRY_ID		CHAR(2)

SELECT * FROM locations;

LOCATION_ID	STREET_ADDRESS	POSTAL_CODE	CITY	STATE_PROVINCE	CO
1000	1297 Via Cola di Rie	00989	Roma		IT
1100	93091 Calle della Testa	10934	Venice		IT
1200	2017 Shinjuku-ku	1689	Tokyo	Tokyo Prefecture	JP
1300	9450 Kamiya-cho	6823	Hiroshima		JP
1400	2014 Jabberwocky Rd	26192	Southlake	Texas	US
1500	2011 Interiors Blvd	99236	South San Francisco	California	US
1600	2007 Zagora St	50090	South Brunswick	New Jersey	US
1700	2004 Charade Rd	98199	Seattle	Washington	US
1800	147 Spadina Ave	M5V 2L7	Toronto	Ontario	CA
1900	6092 Boxwood St	YSW 9T2	Whitehorse	Yukon	CA
2000	40-5-12 Laogianggen	190518	Beijing		CN
2100	1298 Vileparle (E)	490231	Bombay	Maharashtra	IN
LOCATION_ID	STREET_ADDRESS	POSTAL_CODE	CITY	STATE_PROVINCE	CO
2400	8204 Arthur St		London		UK
2500	Magdalen Centre, The Oxford Science Park	OX9 9ZB	Oxford	Oxford	UK
2600	9702 Chester Road	09629850293	Stretford	Manchester	UK
2700	Schwanthalerstr. 7031	80925	Munich	Bavaria	DE
2800	Rua Frei Caneca 1360	01307-002	Sao Paulo	Sao Paulo	BR
2900	20 Rue des Corps-Saints	1730	Geneva	Geneve	CH
3000	Murtenstrasse 921	3095	Bern	BE	CH
3100	Pieter Breughelstraat 837	3029SK	Utrecht	Utrecht	NL
3200	Mariano Escobedo 9991	11932	Mexico City	Distrito Federal,	MX

23 rows selected.

Tabla departments

DESCRIBE departments

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

SELECT * FROM departments;

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
30	Purchasing	114	1700
40	Human Resources	203	2400
50	Shipping	121	1500
60	IT	103	1400
70	Public Relations	204	2700
80	Sales	145	2500
90	Executive	100	1700
100	Finance	108	1700
110	Accounting	205	1700
120	Treasury		1700
130	Corporate Tax		1700
140	Control And Credit		1700
DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
150	Shareholder Services		1700
160	Benefits		1700
170	Manufacturing		1700
180	Construction		1700
190	Contracting		1700
200	Operations		1700
210	IT Support		1700
220	NOC		1700
230	IT Helpdesk		1700
240	Government Sales		1700
250	Retail Sales		1700
260	Recruiting		1700
270	Payroll		1700

27 rows selected.

Tabla jobs

DESCRIBE jobs

Name	Null?	Type
JOB_ID	NOT NULL	VARCHAR2(10)
JOB_TITLE	NOT NULL	VARCHAR2(35)
MIN_SALARY		NUMBER(6)
MAX_SALARY		NUMBER(6)

SELECT * FROM jobs;

JOB_ID	JOB_TITLE	MIN_SALARY	MAX_SALARY
AD_PRES	President	20000	40000
AD_VP	Administration Vice President	15000	30000
AD_ASST	Administration Assistant	3000	6000
FI_MGR	Finance Manager	8200	16000
FI_ACCOUNT	Accountant	4200	9000
AC_MGR	Accounting Manager	8200	16000
AC_ACCOUNT	Public Accountant	4200	9000
SA_MAN	Sales Manager	10000	20000
SA_REP	Sales Representative	6000	12000
PU_MAN	Purchasing Manager	8000	15000
PU_CLERK	Purchasing Clerk	2500	5500
ST_MAN	Stock Manager	5500	8500
ST_CLERK	Stock Clerk	2000	5000
SH_CLERK	Shipping Clerk	2500	5500
JOB_ID	JOB_TITLE	MIN_SALARY	MAX_SALARY
IT_PROG	Programmer	4000	10000
MK_MAN	Marketing Manager	9000	15000
MK_REP	Marketing Representative	4000	9000
HR_REP	Human Resources Representative	4000	9000
PR_REP	Public Relations Representative	4500	10500

19 rows selected.

Tabla employees

```
DESCRIBE employees
```

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

Tabla employees (continuación)

Las cabeceras de las columnas commission_pct, manager_id y department_id están definidas en comm, mgrid y deptid, respectivamente, en la siguiente captura de pantalla para ajustar los valores de la tabla en la página.

```
SELECT * FROM employees;
```

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	comm	mgrid	deptid
EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	comm	mgrid	deptid
100	Steven	King	SKING	515.123.4567	17-JUN-87	AD_PRES	24000			90
101	Neena	Kochhar	NKOCHHAR	515.123.4568	21-SEP-89	AD_VP	17000		100	90
102	Lex	De Haan	LDEHAAN	515.123.4569	13-JAN-93	AD_VP	17000		100	90
103	Alexander	Hunold	AHUNOLD	590.423.4567	03-JAN-90	IT_PROG	9000		102	60
104	Bruce	Ernst	BERNST	590.423.4568	21-MAY-91	IT_PROG	6000		103	60
105	David	Austin	DAUSTIN	590.423.4569	25-JUN-97	IT_PROG	4800		103	60
106	Valli	Pataballa	VPATABAL	590.423.4560	06-FEB-98	IT_PROG	4800		103	60
107	Diana	Lorentz	DLORENTZ	590.423.5567	07-FEB-99	IT_PROG	4200		103	60
108	Nancy	Greenberg	NGREENBE	515.124.4569	17-AUG-94	FI_MGR	12000		101	100
109	Daniel	Faviet	DFAVIET	515.124.4169	16-AUG-94	FI_ACCOUNT	9000		108	100
110	John	Chen	JCHEN	515.124.4269	28-SEP-97	FI_ACCOUNT	8200		108	100
111	Ismael	Sciarrা	ISCIARRA	515.124.4369	30-SEP-97	FI_ACCOUNT	7700		108	100
112	Jose Manuel	Urman	JMURMAN	515.124.4469	07-MAR-98	FI_ACCOUNT	7800		108	100
113	Luis	Popp	LPOPP	515.124.4567	07-DEC-99	FI_ACCOUNT	6900		108	100
EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	comm	mgrid	deptid
114	Den	Raphaely	DRAPHEAL	515.127.4561	07-DEC-94	PU_MAN	11000		100	30
115	Alexander	Khoo	AKHOO	515.127.4562	18-MAY-95	PU_CLERK	3100		114	30
116	Shelli	Baida	SBAIDA	515.127.4563	24-DEC-97	PU_CLERK	2900		114	30
117	Sigal	Tobias	STOBIAS	515.127.4564	24-JUL-97	PU_CLERK	2800		114	30
118	Guy	Himuro	GHIMURO	515.127.4565	15-NOV-98	PU_CLERK	2600		114	30
119	Karen	Colmenares	KCOLMENA	515.127.4566	10-AUG-99	PU_CLERK	2500		114	30
120	Matthew	Weiss	MWEISS	650.123.1234	18-JUL-96	ST_MAN	8000		100	50
121	Adam	Fripp	AFRIPP	650.123.2234	10-APR-97	ST_MAN	8200		100	50
122	Payam	Kaufling	PKAUFLIN	650.123.3234	01-MAY-95	ST_MAN	7900		100	50
123	Shanta	Vollman	SVOLLMAN	650.123.4234	10-OCT-97	ST_MAN	6500		100	50
124	Kevin	Mourgos	KMOURGOS	650.123.5234	16-NOV-99	ST_MAN	5800		100	50
125	Julia	Nayer	JNAYER	650.124.1214	16-JUL-97	ST_CLERK	3200		120	50
126	Irene	Mikkilineni	IMIKKILI	650.124.1224	28-SEP-98	ST_CLERK	2700		120	50
127	James	Landry	JLANDRY	650.124.1334	14-JAN-99	ST_CLERK	2400		120	50

Tabla employees (continuación)

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	comm	mgrid	deptid
128	Steven	Markle	SMARKLE	650.124.1434	08-MAR-00	ST_CLERK	2200		120	50
129	Laura	Bissot	LBISSOT	650.124.5234	20-AUG-97	ST_CLERK	3300		121	50
130	Mozhe	Atkinson	MATKINSO	650.124.8234	30-OCT-97	ST_CLERK	2800		121	50
131	James	Marlow	JAMRLOW	650.124.7234	16-FEB-97	ST_CLERK	2500		121	50
132	TJ	Olson	TJOLSON	650.124.8234	10-APR-99	ST_CLERK	2100		121	50
133	Jason	Mallin	JMALLIN	650.127.1934	14-JUN-96	ST_CLERK	3300		122	50
134	Michael	Rogers	MROGERS	650.127.1834	26-AUG-98	ST_CLERK	2900		122	50
135	Ki	Gee	KGEE	650.127.1734	12-DEC-99	ST_CLERK	2400		122	50
136	Hazel	Philtanker	PHILHTAN	650.127.1634	06-FEB-00	ST_CLERK	2200		122	50
137	Renske	Ladwig	RLADWIG	650.121.1234	14-JUL-95	ST_CLERK	3600		123	50
138	Stephen	Stiles	SSTILES	650.121.2034	26-OCT-97	ST_CLERK	3200		123	50
139	John	Seo	JSEO	650.121.2019	12-FEB-98	ST_CLERK	2700		123	50
140	Joshua	Patel	JPATEL	650.121.1834	06-APR-98	ST_CLERK	2500		123	50
141	Trenna	Rajs	TRAJS	650.121.8009	17-OCT-95	ST_CLERK	3500		124	50
EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	comm	mgrid	deptid
142	Curtis	Davies	CDAMES	650.121.2994	29-JAN-97	ST_CLERK	3100		124	50
143	Randall	Matos	RMATOS	650.121.2874	15-MAR-98	ST_CLERK	2600		124	50
144	Peter	Vargas	PVARGAS	650.121.2004	09-JUL-98	ST_CLERK	2500		124	50
145	John	Russell	JRUSSEL	011.44.1344.429268	01-OCT-96	SA_MAN	14000	.4	100	80
146	Karen	Partners	KPARTNER	011.44.1344.467268	05-JAN-97	SA_MAN	13500	.3	100	80
147	Alberto	Errazuriz	AERRAZUR	011.44.1344.429278	10-MAR-97	SA_MAN	12000	.3	100	80
148	Gerald	Cambrault	GCAMBRAU	011.44.1344.619268	15-OCT-99	SA_MAN	11000	.3	100	80
149	Beni	Zlotkey	EZLOTKEY	011.44.1344.429018	29-JAN-00	SA_MAN	10500	.2	100	80
150	Peter	Tucker	PTUCKER	011.44.1344.129268	30-JAN-97	SA REP	10000	.3	145	80
151	David	Bernstein	DBERNSTE	011.44.1344.345268	24-MAR-97	SA REP	9500	.25	145	80
152	Peter	Hall	PHALL	011.44.1344.478968	20-AUG-97	SA REP	9000	.25	145	80
153	Christopher	Olsen	COLSEN	011.44.1344.498718	30-MAR-98	SA REP	8000	.2	145	80
154	Nanette	Cambrault	NCAMBRAU	011.44.1344.987668	09-DEC-98	SA REP	7500	.2	145	80
155	Oliver	Tuvault	OTUVVAULT	011.44.1344.486508	23-NOV-99	SA REP	7000	.15	145	80
EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	comm	mgrid	deptid
156	Janette	King	JKING	011.44.1345.429268	30-JAN-96	SA REP	10000	.35	146	80
157	Patrick	Sully	PSULLY	011.44.1345.929268	04-MAR-96	SA REP	9500	.35	146	80
158	Allan	McEwen	AMCEWEN	011.44.1345.829268	01-AUG-96	SA REP	9000	.35	146	80
159	Lindsey	Smith	LSMITH	011.44.1345.729268	10-MAR-97	SA REP	8000	.3	146	80
160	Louise	Doran	LDORAN	011.44.1345.629268	15-DEC-97	SA REP	7500	.3	146	80
161	Sarath	Sewall	SSEWALL	011.44.1345.529268	03-NOV-98	SA REP	7000	.25	146	80
162	Clara	Mishney	CMISHNEY	011.44.1346.129268	11-NOV-97	SA REP	10500	.25	147	80
163	Danielle	Greene	DGREENE	011.44.1346.229268	19-MAR-99	SA REP	9500	.15	147	80
164	Mattea	Marvins	MMARVINS	011.44.1346.329268	24-JAN-00	SA REP	7200	.1	147	80
165	David	Lee	DLEE	011.44.1346.529268	23-FEB-00	SA REP	6800	.1	147	80
166	Sundar	Ande	SANDE	011.44.1346.629268	24-MAR-00	SA REP	6400	.1	147	80
167	Amit	Banda	ABANDA	011.44.1346.729268	21-APR-00	SA REP	6200	.1	147	80
168	Lisa	Ozer	LOZER	011.44.1343.929268	11-MAR-97	SA REP	11500	.25	148	80
169	Harrison	Bloom	HBLOOM	011.44.1343.829268	23-MAR-98	SA REP	10000	.2	148	80

Tabla employees (continuación)

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	comm	mgrid	deptid
170	Tayler	Fox	TFOX	011.44.1343.729268	24-JAN-98	SA_REP	9600	.2	148	80
171	William	Smith	WSMITH	011.44.1343.629268	23-FEB-99	SA_REP	7400	.15	148	80
172	Elizabeth	Bates	EBATES	011.44.1343.529268	24-MAR-99	SA_REP	7300	.15	148	80
173	Sundita	Kumar	SKUMAR	011.44.1343.329268	21-APR-00	SA_REP	6100	.1	148	80
174	Elen	Abel	EABEL	011.44.1644.429267	11-MAY-96	SA_REP	11000	.3	149	80
175	Alyssa	Hutton	AHUTTON	011.44.1644.429266	19-MAR-97	SA_REP	8800	.25	149	80
176	Jonathon	Taylor	JTAYLOR	011.44.1644.429265	24-MAR-98	SA_REP	8600	.2	149	80
177	Jack	Livingston	JLIVINGSTON	011.44.1644.429264	23-APR-98	SA_REP	8400	.2	149	80
178	Kimberely	Grant	KGRANT	011.44.1644.429263	24-MAY-99	SA_REP	7000	.15	149	
179	Charles	Johnson	CJOHNSON	011.44.1644.429262	04-JAN-00	SA_REP	6200	.1	149	80
180	Winston	Taylor	WTAYLOR	650.507.9876	24-JAN-98	SH_CLERK	3200		120	50
181	Jean	Fleaur	JFLEAUR	650.507.9877	23-FEB-98	SH_CLERK	3100		120	50
182	Martha	Sullivan	MSULLIVA	650.507.9878	21-JUN-99	SH_CLERK	2500		120	50
183	Girard	Geoni	GGEONI	650.507.9879	03-FEB-00	SH_CLERK	2800		120	50
EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	comm	mgrid	deptid
184	Nandita	Sarchand	NSARCHAN	650.509.1876	27-JAN-96	SH_CLERK	4200		121	50
185	Alexis	Bull	ABULL	650.509.2876	20-FEB-97	SH_CLERK	4100		121	50
186	Julia	Dellinger	JDELLING	650.509.3876	24-JUN-98	SH_CLERK	3400		121	50
187	Anthony	Cabrio	ACABRIO	650.509.4876	07-FEB-99	SH_CLERK	3000		121	50
188	Kelly	Chung	KCHUNG	650.505.1876	14-JUN-97	SH_CLERK	3800		122	50
189	Jennifer	Dilly	JDILLY	650.505.2876	13-AUG-97	SH_CLERK	3600		122	50
190	Timothy	Gates	TGATES	650.505.3876	11-JUL-98	SH_CLERK	2900		122	50
191	Randall	Perkins	RPERKINS	650.505.4876	19-DEC-99	SH_CLERK	2500		122	50
192	Sarah	Bell	SBELL	650.501.1876	04-FEB-96	SH_CLERK	4000		123	50
193	Britney	Everett	BEVERETT	650.501.2876	03-MAR-97	SH_CLERK	3900		123	50
194	Samuel	McCain	SMCCAIN	650.501.3876	01-JUL-98	SH_CLERK	3200		123	50
195	Vance	Jones	VJONES	650.501.4876	17-MAR-99	SH_CLERK	2800		123	50
196	Alana	Walsh	AWALSH	650.507.9811	24-APR-98	SH_CLERK	3100		124	50
197	Kevin	Feehey	KFEENEY	650.507.9822	23-MAY-98	SH_CLERK	3000		124	50
EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	comm	mgrid	deptid
198	Donald	OConnell	DOCONNEL	650.507.9833	21-JUN-99	SH_CLERK	2600		124	50
199	Douglas	Grant	DGRANT	650.507.9844	13-JAN-00	SH_CLERK	2600		124	50
200	Jennifer	Whalen	JWHALEN	515.123.4444	17-SEP-87	AD_ASST	4400		101	10
201	Michael	Hartstein	MHARTSTE	515.123.5555	17-FEB-96	MK_MAN	13000		100	20
202	Pat	Fay	PFAY	603.123.6666	17-AUG-97	MK_REP	6000		201	20
203	Susan	Mavris	SMAVRIS	515.123.7777	07-JUN-94	HR_REP	6500		101	40
204	Hermann	Baer	HBAER	515.123.8888	07-JUN-94	PR_REP	10000		101	70
205	Shelley	Higgins	SHIGGINS	515.123.8080	07-JUN-94	AC_MGR	12000		101	110
206	William	Gietz	WGIETZ	515.123.8181	07-JUN-94	AC_ACCOUNT	8300		205	110

107 rows selected.

Tabla job_history

DESCRIBE job_history

Name	Null?	Type
EMPLOYEE_ID	NOT NULL	NUMBER(6)
START_DATE	NOT NULL	DATE
END_DATE	NOT NULL	DATE
JOB_ID	NOT NULL	VARCHAR2(10)
DEPARTMENT_ID		NUMBER(4)

SELECT * FROM job_history;

EMPLOYEE_ID	START_DATE	END_DATE	JOB_ID	deptid
102	13-JAN-93	24-JUL-98	IT_PROG	60
101	21-SEP-89	27-OCT-93	AC_ACCOUNT	110
101	28-OCT-93	15-MAR-97	AC_MGR	110
201	17-FEB-96	19-DEC-99	MK_REP	20
114	24-MAR-98	31-DEC-99	ST_CLERK	50
122	01-JAN-99	31-DEC-99	ST_CLERK	50
200	17-SEP-87	17-JUN-93	AD_ASST	90
176	24-MAR-98	31-DEC-98	SA_REP	80
176	01-JAN-99	31-DEC-99	SA_MAN	80
200	01-JUL-94	31-DEC-98	AC_ACCOUNT	90

10 rows selected.

C

Uso de SQL Developer

ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Objetivos

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

- Mostrar las funciones clave de Oracle SQL Developer
- Identificar las opciones de menú de Oracle SQL Developer
- Crear una conexión a la base de datos
- Gestionar objetos de bases de datos
- Utilizar la hoja de trabajo de SQL
- Guardar y ejecutar scripts SQL
- Crear y guardar informes



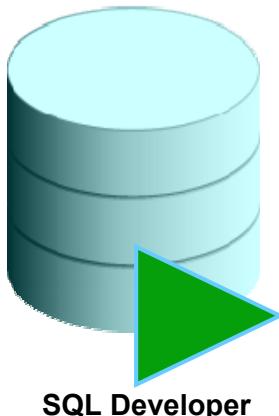
Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Objetivos

En este apéndice, se presentará la herramienta gráfica denominada SQL Developer. Aprenderá cómo utilizar SQL Developer para las tareas de desarrollo de la base de datos. Aprenderá cómo utilizar la hoja de trabajo de SQL para ejecutar sentencias y scripts SQL.

¿Qué Es Oracle SQL Developer?

- Oracle SQL Developer es una herramienta gráfica que mejora la productividad y simplifica las tareas de desarrollo de la base de datos.
- Puede conectarse a cualquier esquema de base de datos de destino de Oracle mediante la autenticación estándar de la base de datos Oracle.



ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

¿Qué Es Oracle SQL Developer?

Oracle SQL Developer es una herramienta gráfica gratuita diseñada para mejorar la productividad y simplificar el desarrollo de las tareas diarias de la base de datos. Con sólo unos clics, puede crear y depurar fácilmente los procedimientos almacenados, probar sentencias SQL y visualizar los planes del optimizador.

SQL Developer, la herramienta visual para el desarrollo de la base de datos, simplifica las siguientes tareas:

- Exploración y gestión de objetos de la base de datos
- Ejecución de sentencias y scripts SQL
- Edición y depuración de sentencias PL/SQL
- Creación de informes

Puede conectarse a cualquier esquema de base de datos de destino de Oracle mediante la autenticación estándar de la base de datos Oracle. Una vez conectado, puede realizar operaciones en los objetos de la base de datos.

La versión SQL Developer 1.2 está muy integrada con *Developer Migration Workbench*, que proporciona a los usuarios un único punto para examinar los datos y objetos de la base de datos de bases de datos de terceros, así como para migrar desde estas bases de datos a Oracle. También puede conectar a esquemas de bases de datos de terceros seleccionadas (no Oracle) como MySQL, Microsoft SQL Server y Microsoft Access, así como ver los metadatos y los datos de esas bases de datos.

Además, SQL Developer incluye soporte para Oracle Application Express 3.0.1 (Oracle APEX).

Especificaciones de SQL Developer

- Está incluido con Oracle Database 11g Versión 2
- Está desarrollado en Java
- Soporta plataformas Windows, Linux y Mac OS X
- Proporciona conectividad por defecto mediante el controlador JDBC Thin
- Se conecta a Oracle Database versión 9.2.0.1 y posteriores
- Se descarga de forma gratuita desde el siguiente enlace:
 - http://www.oracle.com/technology/products/database/sql_developer/index.html



Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

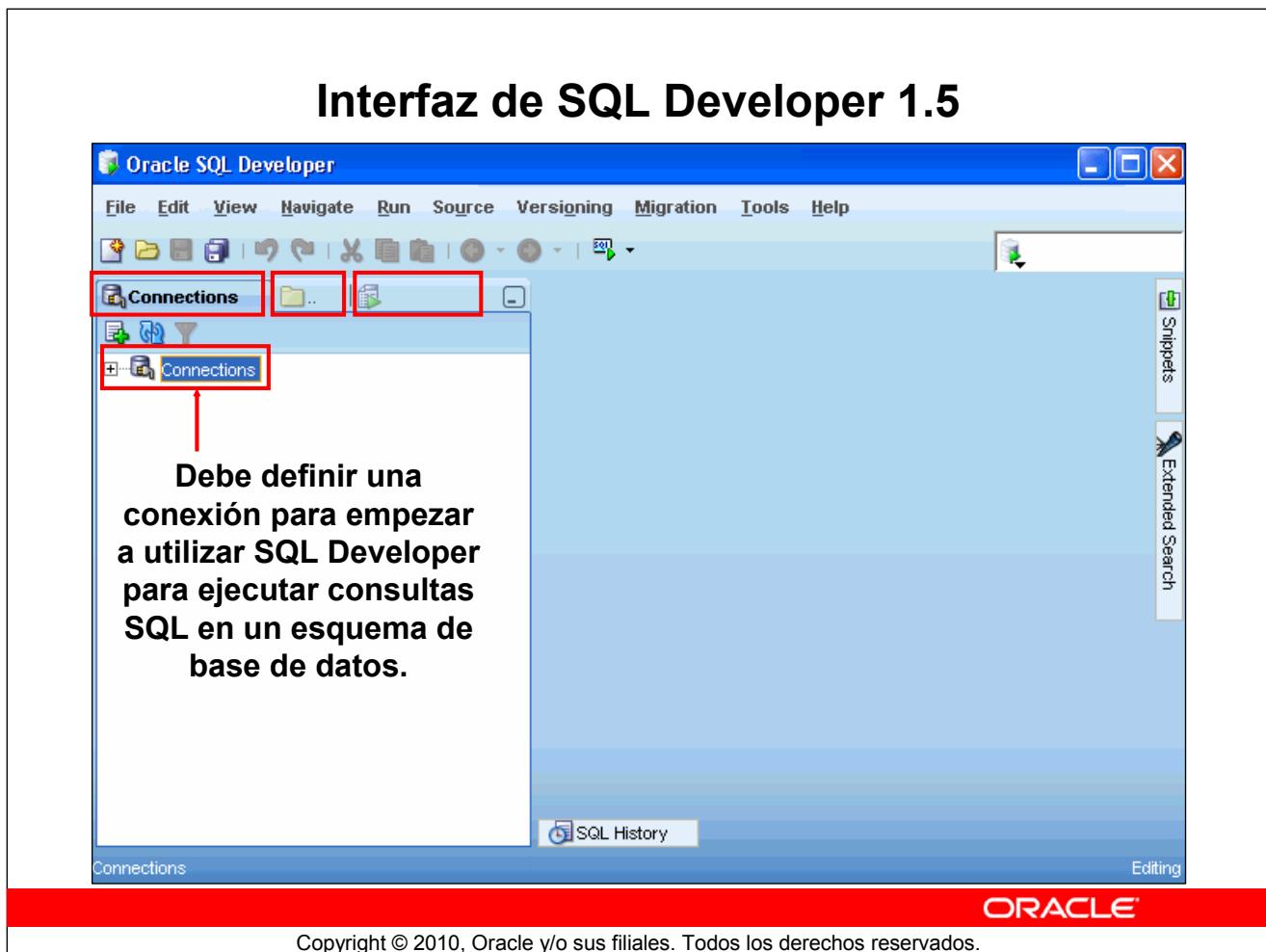
Especificaciones de SQL Developer

Oracle SQL Developer 1.5 se incluye con Oracle Database 11g Versión 2. SQL Developer está desarrollado en Java, aprovechando Oracle JDeveloper IDE (entorno de desarrollo de integración). Por tanto, se trata de una herramienta entre plataformas. La herramienta se ejecuta en plataformas con Windows, Linux y Mac OS X.

La conectividad por defecto a la base de datos se realiza a través del controlador Java Database Connectivity (JDBC) Thin, por lo que no es necesario el directorio raíz de Oracle. SQL Developer no necesita ningún programa de instalación. Sólo hay que descomprimir el archivo descargado. Con SQL Developer, los usuarios se pueden conectar a Oracle Database 9.2.0.1 y versiones posteriores y a todas las ediciones de Oracle Database, incluida Express Edition.

Nota

- Para las versiones de Oracle Database anteriores a Oracle Database 11g Versión 2, tendrá que descargar e instalar SQL Developer. SQL Developer 1.5 se descarga de forma gratuita desde el siguiente enlace:
 - http://www.oracle.com/technology/products/database/sql_developer/index.html
- Para obtener instrucciones sobre cómo instalar SQL Developer, puede visitar el siguiente enlace:
 - http://download.oracle.com/docs/cd/E12151_01/index.htm



Interfaz de SQL Developer 1.5

La interfaz de SQL Developer 1.5 contiene tres separadores principales de navegación, de izquierda a derecha:

- **Separador Connections:** si utiliza este separador, puede examinar los objetos y usuarios de la base de datos a los que tiene acceso.
- **Separador Files:** este separador, que se identifica mediante el icono de carpeta Files, le permite acceder a archivos desde la máquina local sin tener que utilizar el menú File > Open.
- **Separador Reports:** este separador, que se identifica mediante el icono Reports, le permite ejecutar informes predefinidos o crear y agregar sus propios informes.

Navegación General y Uso

SQL Developer utiliza la parte izquierda de la navegación para buscar y seleccionar objetos y la parte derecha para mostrar información sobre los objetos seleccionados. También puede personalizar muchos aspectos de la apariencia y del comportamiento de SQL Developer mediante la definición de preferencias.

Nota: debe definir al menos una conexión para poder conectar a un esquema de base de datos y emitir consultas SQL o ejecutar procedimientos y funciones.

Interfaz de SQL Developer 1.5 (continuación)

Menús

Los siguientes menús contienen entradas estándar, además de entradas de funciones específicas de SQL Developer:

- **View:** contiene opciones que afectan a lo que se muestra en la interfaz de SQL Developer.
- **Navigate:** contiene opciones para acceder a los paneles y para la ejecución de subprogramas.
- **Run:** contiene las opciones Run File y Execution Profile que son relevantes en la selección de una función o un procedimiento, así como en las opciones de depuración
- **Source:** contiene opciones para utilizarlas al editar funciones y procedimientos
- **Versioning:** proporciona soporte integrado para las siguientes versiones y sistemas de control de origen: sistema de versiones concurrentes (CVS) y subversión
- **Migration:** contiene opciones relacionadas con la migración de bases de datos de terceros a Oracle
- **Tools:** llama a las herramientas de SQL Developer como SQL*Plus, Preferences y la hoja de trabajo de SQL.

Nota: el menú Run también contiene opciones relevantes cuando se selecciona una función o procedimiento para su depuración. Se trata de las mismas opciones que se encuentran en el menú Debug de la versión 1.2.

Creación de una Conexión a la Base Datos

- Debe tener al menos una conexión a la base de datos para utilizar SQL Developer.
- Puede crear y probar conexiones para:
 - Varias bases de datos
 - Varios esquemas
- SQL Developer importa automáticamente las conexiones definidas en el archivo `tnsnames.ora` al sistema.
- Puede exportar conexiones a un archivo de lenguaje extensible de marcas (XML).
- Cada conexión a la base de datos adicional creada se muestra en la jerarquía de Connections Navigator.



Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Creación de una Conexión a la Base Datos

Una conexión es un objeto de SQL Developer que especifica la información necesaria para conectarse a una base de datos concreta como usuario específico de dicha base de datos. Para utilizar SQL Developer, debe tener al menos una conexión a la base de datos que puede ser existente, creada o importada.

Puede crear y probar conexiones para varias bases de datos y esquemas.

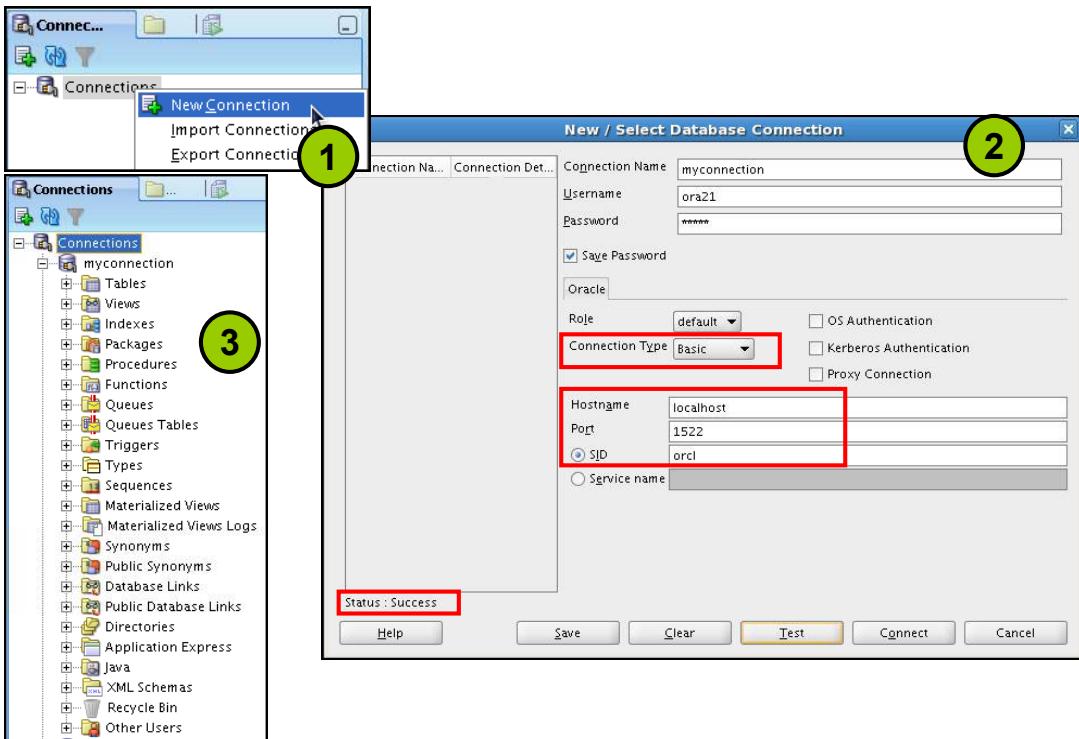
Por defecto, el archivo `tnsnames.ora` se encuentra en el directorio `$ORACLE_HOME/network/admin`, pero también puede estar en el directorio especificado por el valor de registro o la variable de entorno `TNS_ADMIN`. Al iniciar SQL Developer, cuando se muestra el cuadro de diálogo Database Connections, SQL Developer importa automáticamente las conexiones definidas en el archivo `tnsnames.ora` al sistema.

Nota: en los sistemas Windows, si existe el archivo `tnsnames.ora`, pero SQL Developer no está utilizando sus conexiones, defina `TNS_ADMIN` como una variable de entorno del sistema.

Puede exportar conexiones a un archivo XML para volver a utilizarlas más tarde.

Puede crear conexiones adicionales para conectarse a la misma base de datos pero como usuarios diferentes o para conectarse a distintas bases de datos.

Creación de una Conexión a la Base Datos



ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Creación de una Conexión a la Base Datos (continuación)

Para crear una conexión a la base de datos, realice los siguientes pasos:

1. En la página con separadores Connections, haga clic con el botón derecho en **Connections** y seleccione **New Connection**.
2. En la ventana New/Select Database Connection, introduzca el nombre de la conexión. Introduzca el nombre de usuario y la contraseña del esquema al que desea conectarse.
 - a) En el cuadro desplegable Role, puede seleccionar *default* o SYSDBA (seleccione SYSDBA para el usuario sys o cualquier usuario con privilegios de administrador de la base de datos).
 - b) También puede seleccionar el tipo de conexión como:
 - **Basic:** en este tipo, introduzca el nombre de host y el SID de la base de datos a la que desea conectarse. El puerto ya está definido en 1521. También puede optar por introducir el nombre de servicio directamente si utiliza una conexión de base de datos remota.
 - **TNS:** puede seleccionar cualquiera de los alias de base de datos importados del archivo *tnsnames.ora*.
 - **LDAP:** puede consultar los servicios de base de datos en Oracle Internet Directory, componente de Oracle Identity Management.
 - **Advanced:** puede definir una dirección URL de Java Database Connectivity (JDBC) personalizada para conectarse a la base de datos.
 - c) Haga clic en **Test** para asegurarse de que la conexión se ha definido correctamente.
 - d) Haga clic en **Connect**.

Creación de una Conexión a la Base Datos (continuación)

Si activa la casilla de control Save Password, la contraseña se guarda en un archivo XML. De este modo, cuando cierre la conexión a SQL Developer y la vuelva a abrir, no se le pedirá la contraseña.

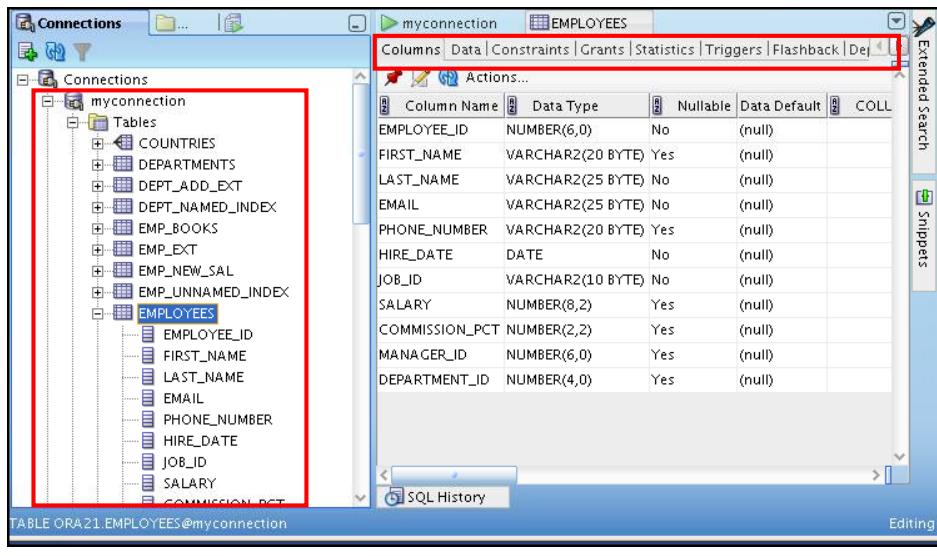
3. La conexión se agrega a Connections Navigator. Puede ampliar la conexión para ver los objetos de la base de datos y las definiciones de objetos, por ejemplo, dependencias, detalles, estadísticas, etc.

Nota: en la misma ventana New>Select Database Connection, puede definir las conexiones a orígenes de datos que no sean Oracle mediante los separadores Access, MySQL y SQL Server. Sin embargo, estas conexiones son conexiones de sólo lectura que le permiten examinar los objetos y datos de ese origen de datos.

Exploración de Objetos de Bases de Datos

Utilice Connections Navigator para:

- Examinar los objetos de un esquema de base de datos
- Revisar las definiciones de objetos de forma rápida



ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Exploración de Objetos de Bases de Datos

Una vez creada la conexión a la base de datos, puede utilizar Connections Navigator para examinar muchos objetos de un esquema de base de datos, entre los que se incluyen tablas, vistas, índices, paquetes, procedimientos, disparadores y tipos.

SQL Developer utiliza la parte izquierda de la navegación para buscar y seleccionar objetos y la parte derecha para mostrar la información sobre los objetos seleccionados. También puede personalizar muchos aspectos de la apariencia de SQL Developer mediante la definición de preferencias.

Puede ver la definición de los objetos desglosados en separadores de información que se transfieren al diccionario de datos. Por ejemplo, si selecciona una tabla en Navigator, se muestran los detalles sobre las columnas, restricciones, permisos, estadísticas, disparadores, etc. en una página con separadores fácil de leer.

Si desea ver la definición de la tabla EMPLOYEES como se muestra en la diapositiva, realice los siguientes pasos:

1. Amplíe el nodo Connections de Connections Navigator.
2. Amplíe Tables.
3. Haga clic en EMPLOYEES. Por defecto, se selecciona el separador Columns. Muestra la descripción de la columna de la tabla. Mediante el separador Data, puede ver los datos de la tabla y también introducir nuevas filas, datos de actualización y confirmar estos cambios en la base de datos.

Visualización de la Estructura de la Tabla

Utilice el comando DESCRIBE para mostrar la estructura de una tabla:

The screenshot shows the Oracle SQL Developer interface. At the top, there's a toolbar with various icons. Below it is a status bar showing "myconnection" and "1.69686902 seconds". The main area has tabs for "Results", "Script Output", "Explain", "Autotrace", "DBMS Output", and "OWA Output". The "Results" tab is selected. It displays the output of the DESCRIBE EMPLOYEES command. The output shows the columns of the EMPLOYEES table with their names, nullability, and data types. There are 11 rows selected. The table structure is as follows:

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

11 rows selected

ORACLE

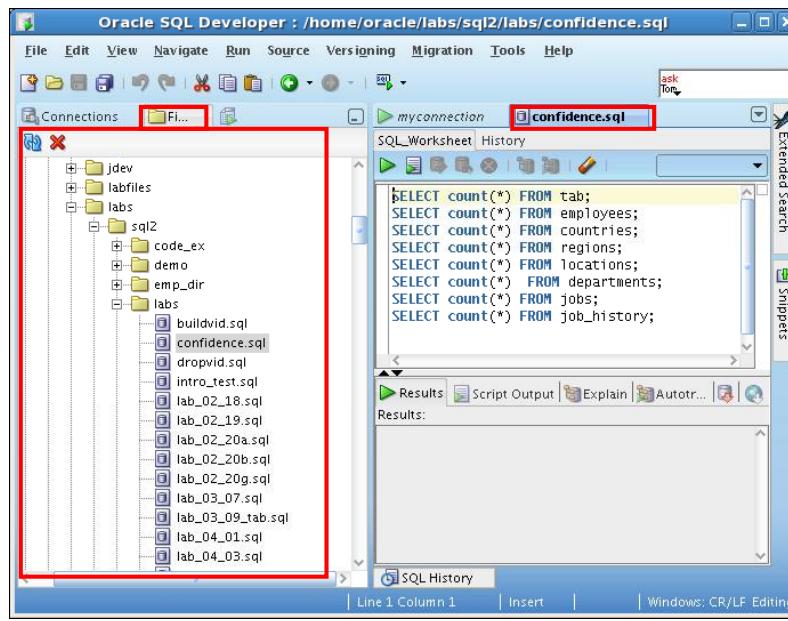
Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Visualización de la Estructura de la Tabla

En SQL Developer, también puede mostrar la estructura de una tabla mediante el comando DESCRIBE. El resultado del comando es una visualización de los nombres de columna y tipos de dato, así como una indicación de si una columna debe contener datos.

Exploración de Archivos

Utilice File Navigator para explorar el sistema de archivos y abrir los archivos del sistema.



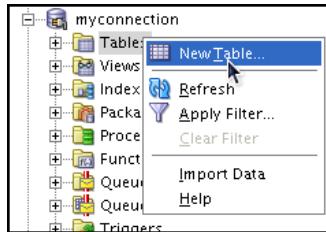
Exploración de Objetos de Bases de Datos

Puede utilizar File Navigator para examinar y abrir los archivos del sistema.

- Para ver File Navigator, haga clic en el separador Files o seleccione View > Files.
- Para ver el contenido de un archivo, haga clic dos veces en el nombre de un archivo para ver su contenido en el área SQL Worksheet.

Creación de un Objeto de Esquema

- SQL Developer soporta la creación de cualquier objeto de esquema mediante:
 - Ejecución de una sentencia SQL en la hoja de trabajo de SQL
 - Uso del menú contextual
- Edite los objetos mediante un cuadro de diálogo de edición o con uno de los muchos menús sensibles al contexto.
- Vea el lenguaje de definición de datos (DDL) para los ajustes, como la creación de un nuevo objeto o la edición de un objeto de esquema existente.



ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

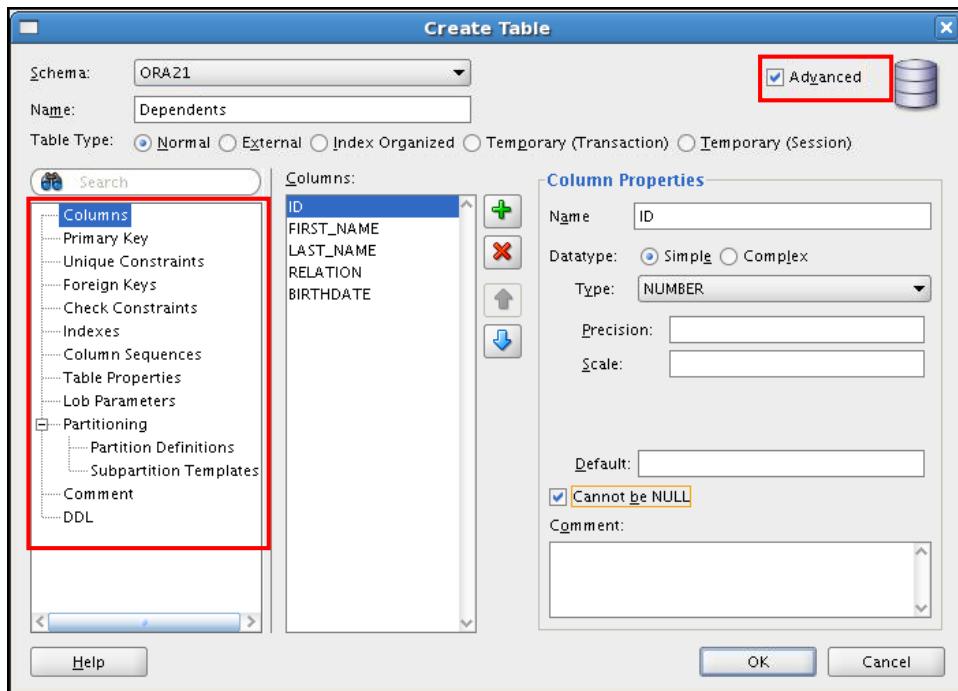
Creación de un Objeto de Esquema

SQL Developer soporta la creación de cualquier objeto de esquema mediante la ejecución de una sentencia SQL en la hoja de trabajo de SQL. Por otro lado, puede crear objetos mediante el menú contextual. Una vez creado, puede editar los objetos con un cuadro de diálogo de edición o con uno de los muchos menús sensibles al contexto.

Una vez que se han creado los nuevos objetos o se han editado los existentes, el DDL de estos ajustes estará disponible para la revisión. Una opción Export DDL está disponible si desea crear el DDL completo para uno o más objetos en el esquema.

La diapositiva muestra cómo crear una tabla mediante el menú contextual. Para abrir un cuadro de diálogo para la creación de una nueva tabla, haga clic con el botón derecho en Tables y seleccione New Table. Los cuadros de diálogo para la creación y edición de objetos de la base de datos tienen varios separadores, cada uno de ellos contiene un agrupamiento lógico de propiedades para dicho tipo de objeto.

Creación de una Nueva Tabla: Ejemplo



ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Creación de una Nueva Tabla: Ejemplo

En el cuadro de diálogo Create Table, si no activa la casilla de control Advanced, puede crear una tabla de forma rápida especificando columnas y algunas de las funciones más utilizadas.

Si activa la casilla de control Advanced, el cuadro de diálogo Create Table cambia a otro con varias opciones, en el que puede especificar un juego ampliado de funciones mientras crea la tabla.

En el ejemplo de la diapositiva se muestra cómo crear la tabla DEPENDENTS activando la casilla de control Advanced.

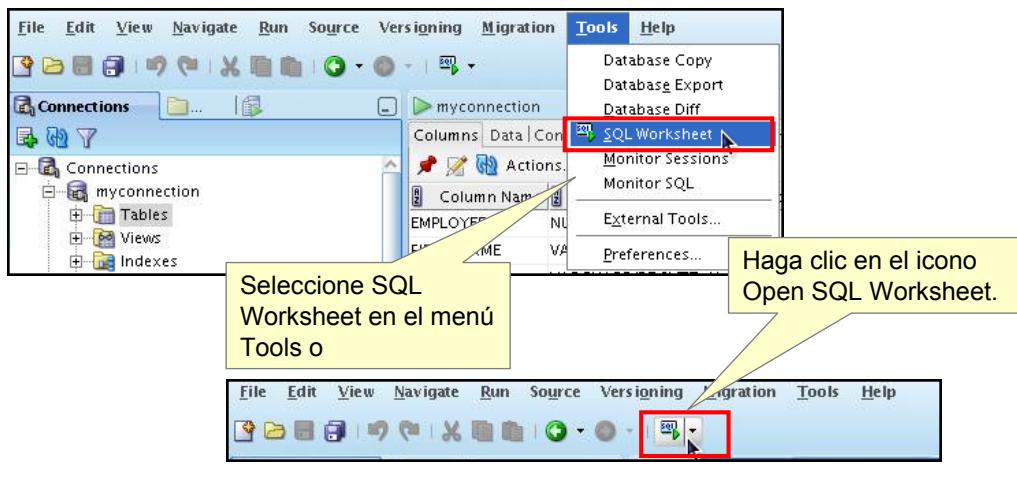
Para crear una nueva tabla, realice los siguientes pasos:

1. En Connections Navigator, haga clic con el botón derecho en Tables.
2. Seleccione New Table.
3. En el cuadro de diálogo Create Table, seleccione Advanced.
4. Especifique la información de columna.
5. Haga clic en OK.

Aunque no es necesario, también debe especificar una clave primaria con el separador Primary Key en el cuadro de diálogo. En ocasiones, puede que desee editar la tabla que ha creado. Para ello, haga clic con el botón derecho en la tabla en Connections Navigator y seleccione Edit.

Uso de la Hoja de Trabajo de SQL

- Utilice la hoja de trabajo de SQL para introducir y ejecutar sentencias SQL, PL/SQL y SQL *Plus.
- Especifique las acciones que se pueden procesar mediante la conexión a la base de datos asociada a la hoja de trabajo.



Uso de la Hoja de Trabajo de SQL

Al conectarse a una base de datos, automáticamente se abre una ventana de la hoja de trabajo de SQL para dicha conexión. Puede utilizar la hoja de trabajo de SQL para introducir y ejecutar sentencias SQL, PL/SQL y SQL *Plus. La hoja de trabajo de SQL soporta sentencias SQL*Plus hasta un determinado grado. Sin embargo, las sentencias SQL*Plus no soportadas por la hoja de trabajo de SQL se ignoran y no se transfieren a la base de datos.

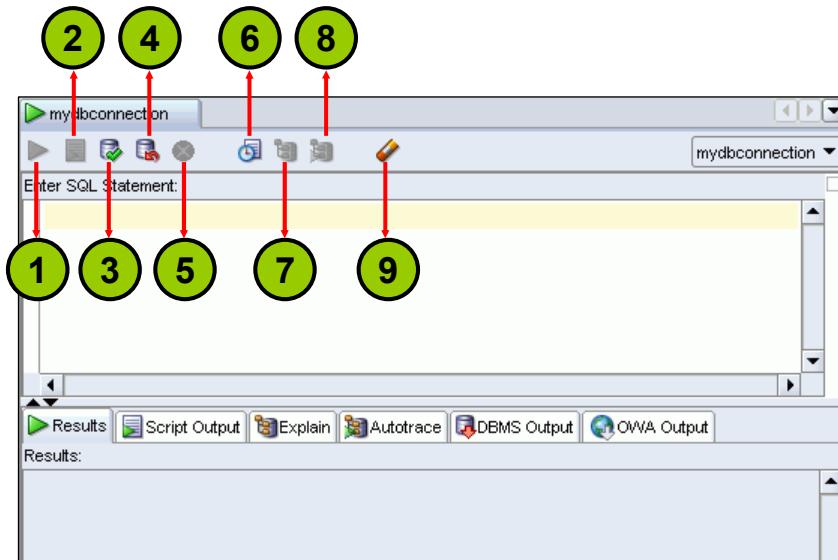
Puede especificar cualquier acción que se pueda procesar mediante la conexión a la base de datos asociada a la hoja de trabajo, como:

- Creación de una tabla
- Inserción de datos
- Creación y edición de un disparador
- Selección de datos de tablas
- Guardado de datos seleccionados en un archivo

Para mostrar una hoja de trabajo de SQL, utilice uno de estos métodos:

- Seleccione Tools > SQL Worksheet.
- Haga clic en el ícono Open SQL Worksheet.

Uso de la Hoja de Trabajo de SQL



ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

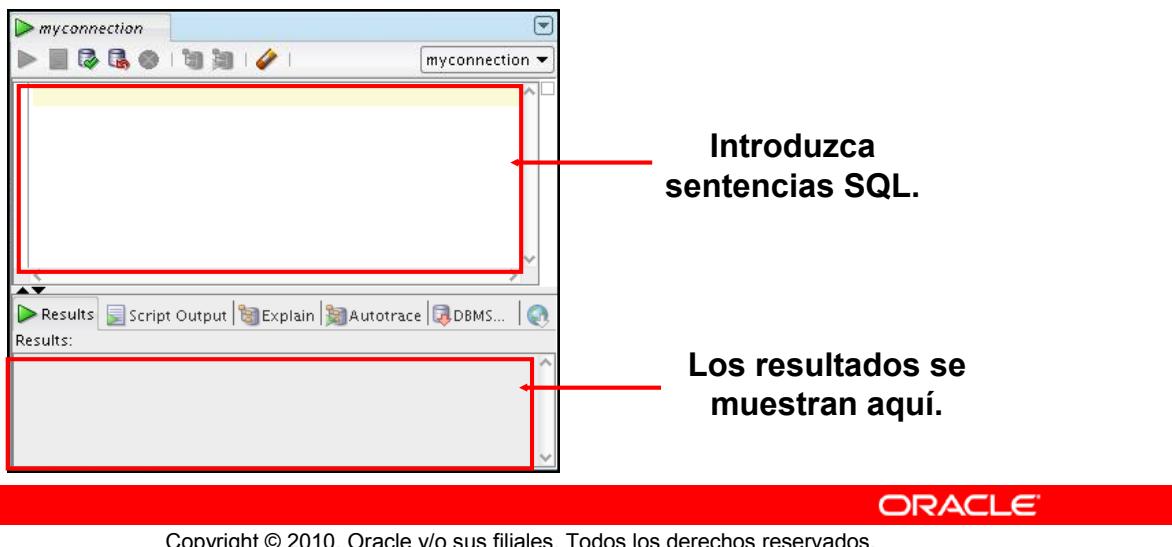
Uso de la Hoja de Trabajo de SQL (continuación)

Puede que desee utilizar teclas o iconos de acceso directo para realizar determinadas tareas como la ejecución de una sentencia SQL o de un script y la visualización del historial de sentencias SQL ejecutadas. Puede utilizar la barra de herramientas de SQL Worksheet que contiene iconos para realizar las siguientes tareas:

1. **Execute Statement:** ejecuta la sentencia donde se encuentra el cursor en la ventana Enter SQL Statement. Puede utilizar variables de enlace en las sentencias SQL pero no variables de sustitución.
2. **Run Script:** ejecuta todas las sentencias en la ventana Enter SQL Statement mediante Script Runner. Puede utilizar variables de sustitución en las sentencias SQL, pero no variables de enlace.
3. **Commit:** escribe cualquier cambio realizado en la base de datos y finaliza la transacción.
4. **Rollback:** desecha cualquier cambio realizado en la base de datos, sin escribirlos en la base de datos y finaliza la transacción.
5. **Cancel:** para la ejecución de cualquier sentencia que se esté ejecutando actualmente.
6. **SQL History:** muestra un cuadro de diálogo con información sobre las sentencias SQL ejecutadas.
7. **Execute Explain Plan:** genera el plan de ejecución, que puede ver haciendo clic en el separador Explain.
8. **Autotrace:** genera información de rastreo para la sentencia.
9. **Clear:** borra la sentencia o sentencias de la ventana Enter SQL Statement.

Uso de la Hoja de Trabajo de SQL

- Utilice la hoja de trabajo de SQL para introducir y ejecutar sentencias SQL, PL/SQL y SQL *Plus.
- Especifique las acciones que se pueden procesar mediante la conexión a la base de datos asociada a la hoja de trabajo.



Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Uso de la Hoja de Trabajo de SQL (continuación)

Al conectarse a una base de datos, automáticamente se abre una ventana de la hoja de trabajo de SQL para dicha conexión. Puede utilizar la hoja de trabajo de SQL para introducir y ejecutar sentencias SQL, PL/SQL y SQL *Plus. Todos los comandos SQL y PL/SQL están soportados, ya que se transfieren directamente desde la hoja de trabajo de SQL hasta Oracle Database. Pero la hoja de trabajo de SQL tiene que interpretar los comandos SQL*Plus utilizados en SQL Developer antes de transferirlos a la base de datos.

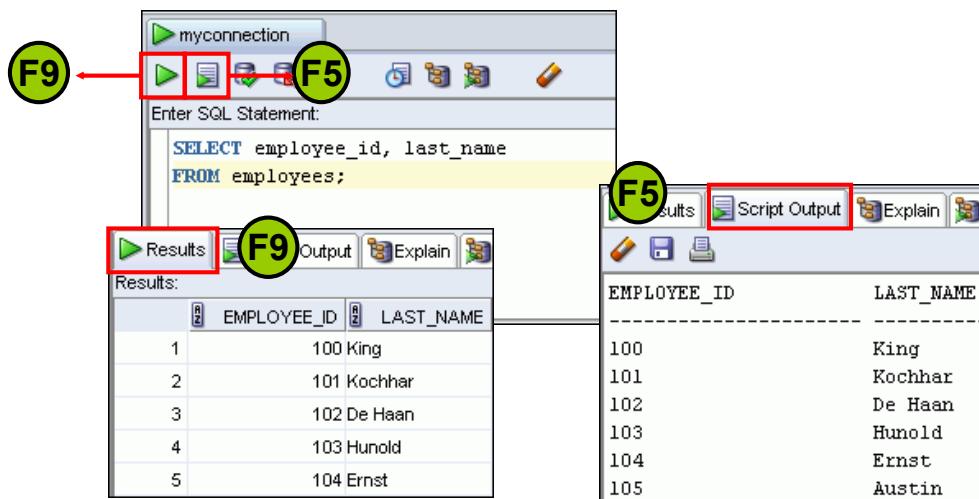
La hoja de trabajo de SQL soporta actualmente una serie de comandos de SQL*Plus. Sin embargo, los comandos no soportados por la hoja de trabajo de SQL se ignoran y no se envían a la base de datos Oracle. Mediante la hoja de trabajo de SQL, puede ejecutar sentencias SQL y algunos comandos de SQL*Plus.

Puede mostrar una hoja de trabajo de SQL mediante alguna de las siguientes dos opciones:

- Seleccione Tools > SQL Worksheet.
- Haga clic en el ícono Open SQL Worksheet.

Ejecución de Sentencias SQL

Utilice la ventana Enter SQL Statement para introducir una o varias sentencias SQL.



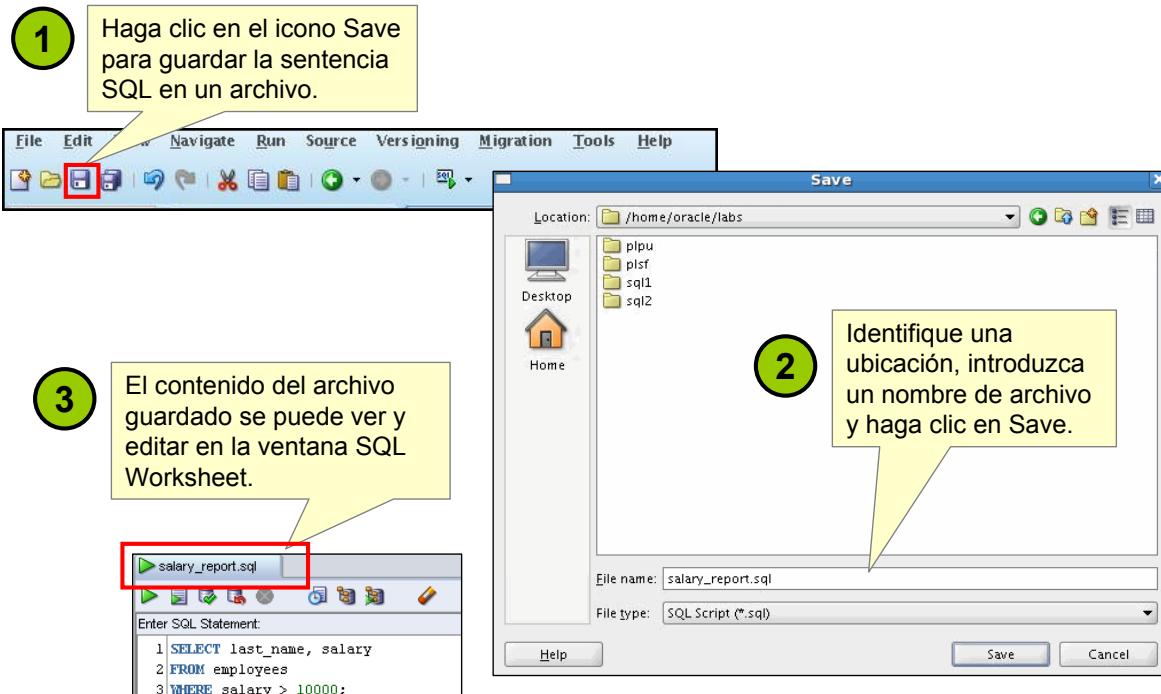
ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Ejecución de Sentencias SQL

En el ejemplo de la diapositiva se muestra la diferencia de la salida de la misma consulta cuando se utiliza la tecla F9 o Execute Statement, frente a la salida cuando se utiliza F5 o Run Script.

Guardado de Scripts SQL



ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Guardado de Scripts SQL

Puede guardar las sentencias SQL desde la hoja de trabajo de SQL en un archivo de texto. Para guardar el contenido de la ventana Enter SQL Statement, realice los siguientes pasos:

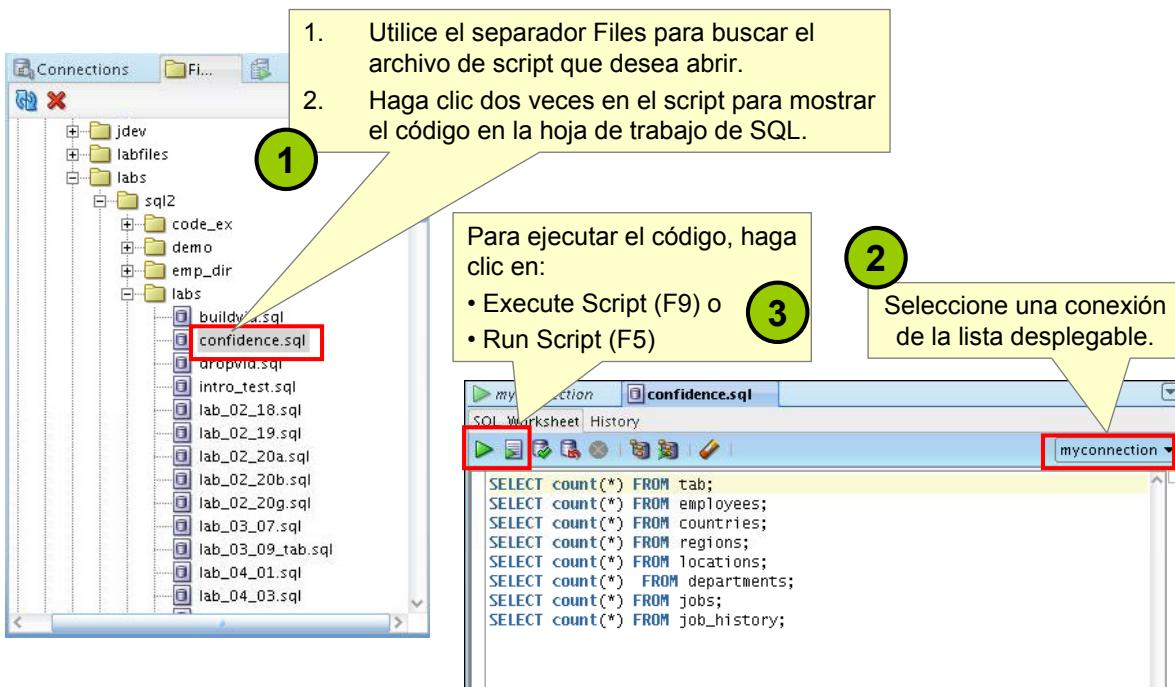
1. Haga clic en el ícono Save o utilice la opción de menú File > Save.
2. En el cuadro de diálogo Windows Save, introduzca un nombre de archivo y la ubicación donde desea guardar el archivo.
3. Haga clic en Save.

Una vez guardado el contenido en un archivo, la ventana Enter SQL Statement muestra una página con separadores con el contenido del archivo. Puede tener varios archivos abiertos a la vez. Cada archivo muestra una página con separadores.

Rutas de Acceso de los Scripts

Puede seleccionar una ruta de acceso por defecto para buscar los scripts y guardarlos. En Tools > Preferences > Database > Worksheet Parameters, introduzca un valor en el campo “Select default path to look for scripts”.

Ejecución de Archivos de Script Guardados: Método 1



ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Ejecución de Archivos de Script Guardados: Método 1

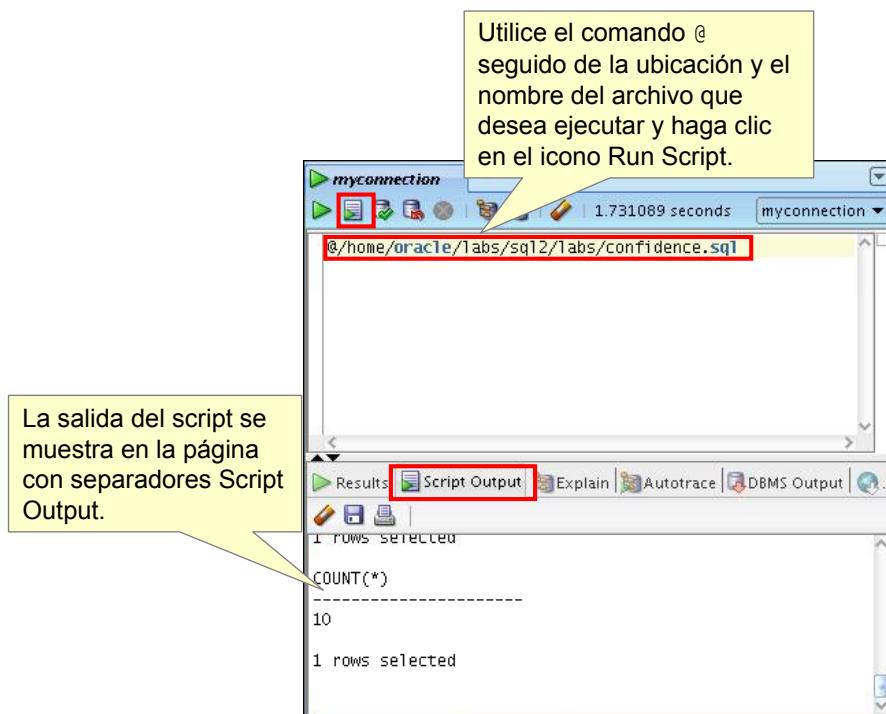
Para abrir un archivo de script y mostrar el código en el área SQL Worksheet, realice lo siguiente:

1. En el navegador de archivos, seleccione (o acceda al) archivo de script que desea abrir.
2. Haga clic dos veces para abrirlo. El código del archivo de script se muestra en el área SQL Worksheet.
3. Seleccione una conexión de la lista desplegable Connection.
4. Para ejecutar el código, haga clic en el ícono Run Script (F5) en la barra de herramientas de SQL Worksheet. Si no ha seleccionado una conexión de la lista desplegable Connection, aparecerá un cuadro de diálogo Connection. Seleccione la conexión que desea utilizar para ejecutar el script.

Como alternativa, también puede:

1. Seleccionar File > Open. Aparece el cuadro de diálogo Open.
2. En el cuadro de diálogo Open, seleccionar (o acceder al) archivo de script que desea abrir.
3. Hacer clic en Open. El código del archivo de script se muestra en el área SQL Worksheet.
4. Seleccione una conexión de la lista desplegable Connection.
5. Para ejecutar el código, haga clic en el ícono Run Script (F5) en la barra de herramientas de SQL Worksheet. Si no ha seleccionado una conexión de la lista desplegable Connection, aparecerá un cuadro de diálogo Connection. Seleccione la conexión que desea utilizar para ejecutar el script.

Ejecución de Archivos de Script Guardados: Método 2



ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

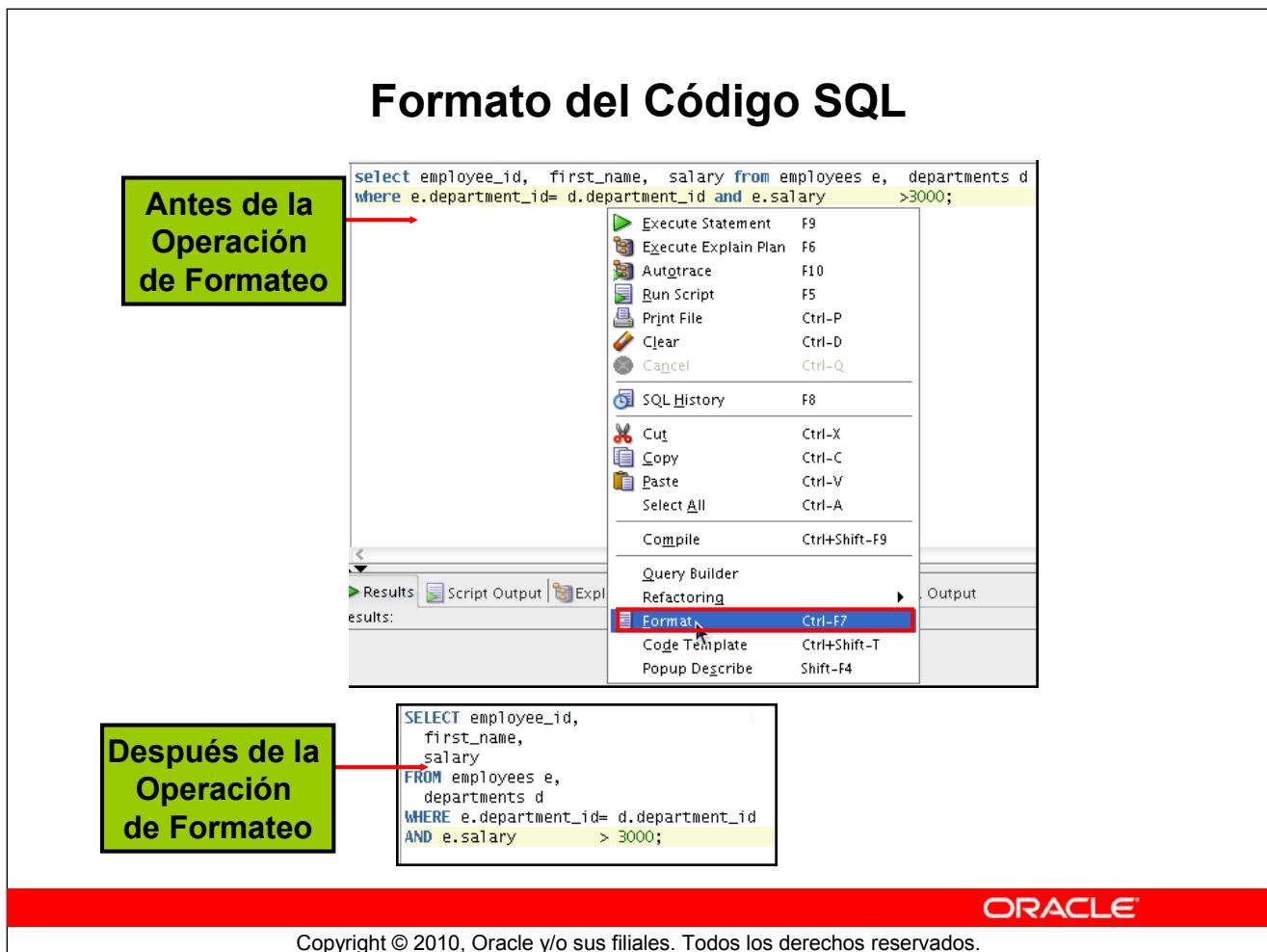
Ejecución de Archivos de Script Guardados: Método 2

Para ejecutar un script SQL guardado, realice lo siguiente:

1. Utilice el comando @, seguido de la ubicación y el nombre del archivo que desea ejecutar, en la ventana Enter SQL Statement.
2. Haga clic en el icono Run Script.

El resultado de la ejecución del archivo se muestra en la página con separadores Script Output.

También puede guardar la salida del script haciendo clic en el icono Save de la página con separadores Script Output. Aparece el cuadro de diálogo Windows File Save, donde puede identificar un nombre y una ubicación para el archivo.



Formato del Código SQL

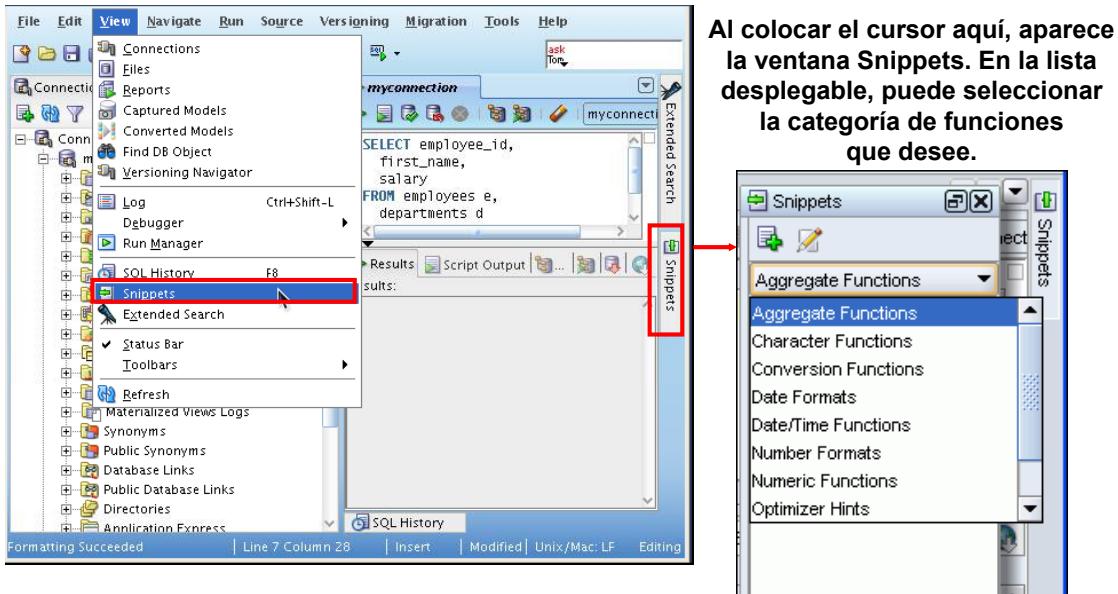
Puede que desee mejorar la apariencia del sangrado, espaciado, uso de mayúsculas y separación de líneas del código SQL. SQL Developer cuenta con una función para dar formato al código SQL.

Para dar formato al código SQL, haga clic con el botón derecho en el área de la sentencia y seleccione Format SQL.

En el ejemplo de la diapositiva, antes de dar formato, las palabras clave del código SQL no están en mayúscula y la sentencia no está sangrada correctamente. Después de dar formato, el código SQL mejora la apariencia con las palabras clave en mayúscula y la sentencia sangrada correctamente.

Uso de Fragmentos

Los fragmentos son partes de código, como sintaxis o ejemplos.



Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

ORACLE

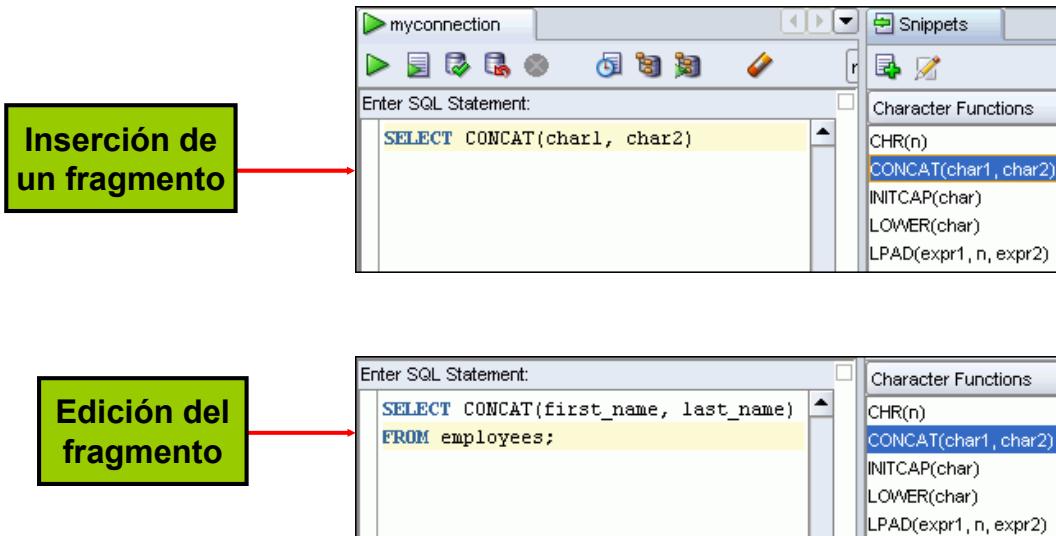
Uso de Fragmentos

Es posible que desee utilizar determinados fragmentos de código al utilizar la hoja de trabajo de SQL Worksheet o al crear o editar un procedimiento o función de PL/SQL. SQL Developer dispone de una función llamada Snippets, que son fragmentos de código, como funciones SQL, indicaciones del optimizador y otras técnicas de programación de PL/SQL. Puede arrastrar los fragmentos en la ventana Editor.

Para visualizar los fragmentos, seleccione View > Snippets.

Aparece la ventana Snippets a la derecha. Puede utilizar la lista desplegable para seleccionar un grupo. En el margen de la ventana derecha se encuentra un botón Snippets, para poder acceder a la ventana Snippets si se oculta.

Uso de Fragmentos: Ejemplo



ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Uso de Fragmentos: Ejemplo

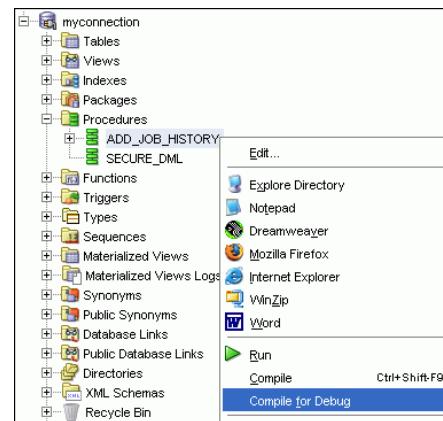
Para insertar un fragmento en el código en la hoja de trabajo de SQL Worksheet o en una función o procedimiento PL/SQL, arrastre el fragmento desde la ventana Snippets y suéltelo en el lugar deseado del código. A continuación, puede editar la sintaxis para que la función SQL sea válida en el contexto actual. Para ver una descripción breve de una función SQL en una ayuda de burbuja, coloque el cursor sobre el nombre de la función.

En el ejemplo de la diapositiva se muestra que `CONCAT (char1, char2)` se arrastra desde el grupo Character Functions a la ventana Snippets. A continuación, se edita la sintaxis de la función `CONCAT` y el resto de la sentencia se agrega como se describe a continuación:

```
SELECT CONCAT(first_name, last_name)
FROM employees;
```

Depuración de Procedimientos y Funciones

- Utilice SQL Developer para depurar funciones y procedimientos PL/SQL.
- Utilice la opción Compile for Debug para realizar una compilación PL/SQL para que se pueda depurar el procedimiento.
- Utilice las opciones del menú Debug para definir puntos de división y para ejecutar Step Into y Step Over.



ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Depuración de Procedimientos y Funciones

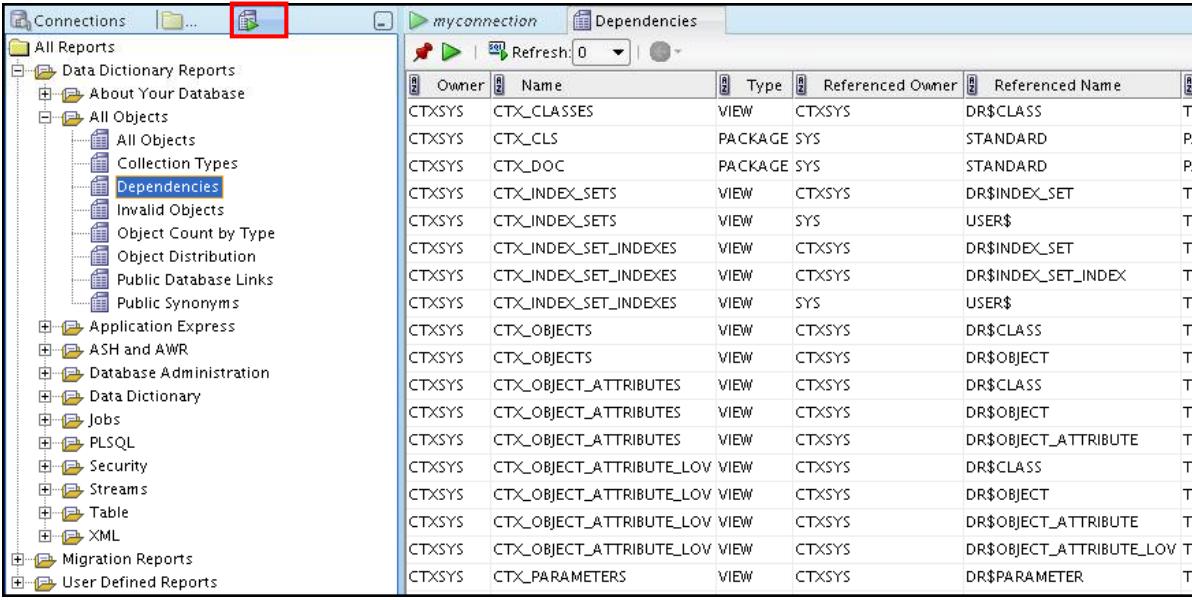
En SQL Developer, puede depurar procedimientos y funciones PL/SQL. Con las opciones del menú Debug, puede realizar las siguientes tareas de depuración:

- **Find Execution Point** permite ir al siguiente punto de ejecución.
- **Resume** permite continuar la ejecución.
- **Step Over** permite omitir el siguiente método e ir a la siguiente sentencia después del método.
- **Step Into** permite ir a la primera sentencia del siguiente método.
- **Step Out** permite salir del método actual e ir a la siguiente sentencia.
- **Step to End of Method** permite ir a la última sentencia del método actual.
- **Pause** permite detener la ejecución, pero no salir, con lo que se permite reanudar la ejecución.
- **Terminate** permite detener y salir de la ejecución. No se puede reanudar la ejecución desde este punto; en su lugar, para iniciar la ejecución o depuración desde el principio de la función o procedimiento, haga clic en el icono Run o Debug de la barra de herramientas con separadores Source.
- **Garbage Collection** permite eliminar objetos no válidos de la caché en favor de objetos a los que se acceda más frecuentemente y más válidos.

Estas opciones también están disponibles como iconos en la barra de herramientas de depuración.

Informes de Bases de Datos

SQL Developer proporciona un número predefinido de informes sobre la base de datos y sus objetos.



The screenshot shows the Oracle SQL Developer interface. On the left, there's a tree view under 'Connections' labeled 'myconnection'. A red box highlights the 'Reports' icon at the top of this tree. The tree includes categories like 'Data Dictionary Reports', 'All Objects' (with 'Dependencies' highlighted), and 'Migration Reports'. To the right is a table titled 'Dependencies' with the following data:

Owner	Name	Type	Referenced Owner	Referenced Name
CTXSYS	CTX_CLASSES	VIEW	CTXSYS	DR\$CLASS
CTXSYS	CTX_CLS	PACKAGE	SYS	STANDARD
CTXSYS	CTX_DOC	PACKAGE	SYS	STANDARD
CTXSYS	CTX_INDEX_SETS	VIEW	CTXSYS	DR\$INDEX_SET
CTXSYS	CTX_INDEX_SETS	VIEW	SYS	USER\$
CTXSYS	CTX_INDEX_SET_INDEXES	VIEW	CTXSYS	DR\$INDEX_SET
CTXSYS	CTX_INDEX_SET_INDEXES	VIEW	CTXSYS	DR\$INDEX_SET_INDEX
CTXSYS	CTX_INDEX_SET_INDEXES	VIEW	SYS	USER\$
CTXSYS	CTX_OBJECTS	VIEW	CTXSYS	DR\$CLASS
CTXSYS	CTX_OBJECTS	VIEW	CTXSYS	DR\$OBJECT
CTXSYS	CTX_OBJECT_ATTRIBUTES	VIEW	CTXSYS	DR\$CLASS
CTXSYS	CTX_OBJECT_ATTRIBUTES	VIEW	CTXSYS	DR\$OBJECT
CTXSYS	CTX_OBJECT_ATTRIBUTES	VIEW	CTXSYS	DR\$OBJECT_ATTRIBUTE
CTXSYS	CTX_OBJECT_ATTRIBUTE_LOV	VIEW	CTXSYS	DR\$CLASS
CTXSYS	CTX_OBJECT_ATTRIBUTE_LOV	VIEW	CTXSYS	DR\$OBJECT
CTXSYS	CTX_OBJECT_ATTRIBUTE_LOV	VIEW	CTXSYS	DR\$OBJECT_ATTRIBUTE
CTXSYS	CTX_OBJECT_ATTRIBUTE_LOV	VIEW	CTXSYS	DR\$OBJECT_ATTRIBUTE_LOV
CTXSYS	CTX_PARAMETERS	VIEW	CTXSYS	DR\$PARAMETER

ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Informes de Bases de Datos

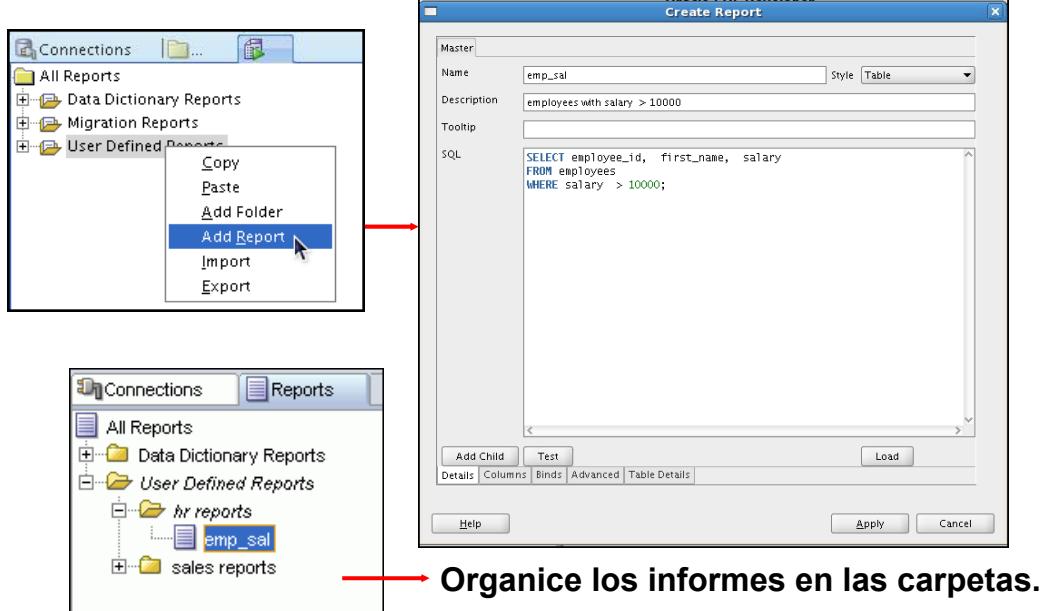
SQL Developer proporciona diferentes informes sobre la base de datos y sus objetos. Estos informes se pueden agrupar en las siguientes categorías:

- Informes About Your Database
- Informes Database Administration
- Informes Table
- Informes PL/SQL
- Informes Security
- Informes XML
- Informes Jobs
- Informes Streams
- Informes All Objects
- Informes Data Dictionary
- Informes User Defined

Para visualizar los informes, haga clic en el separador Reports situado a la izquierda de la ventana. Los informes individuales se muestran en los paneles con separadores situados a la derecha de la ventana. Para cada informe, puede seleccionar (mediante una lista desplegable) la conexión a la base de datos para la que desea mostrar el informe. Para los informes sobre objetos, sólo se muestran aquellos objetos que sean visibles para el usuario de la base de datos asociado a la conexión a la base de datos seleccionada y las filas ordenadas por propietario. También puede crear sus propios informes definidos por el usuario.

Creación de un Informe Definido por el Usuario

Cree y guarde informes definidos por el usuario para un uso repetido.



ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Creación de un Informe Definido por el Usuario

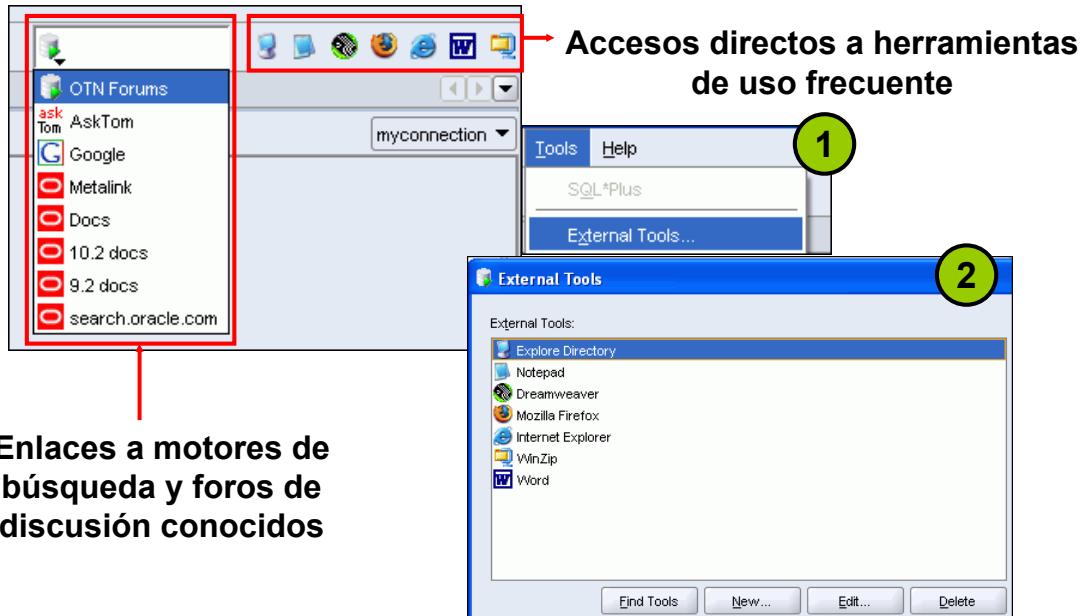
Los informes definidos por el usuario son los informes creados por los usuarios de SQL Developer. Para crear un informe definido por el usuario, realice los siguientes pasos:

1. Haga clic con el botón derecho en el nodo User Defined Reports en Reports y seleccione Add Report.
2. En el cuadro de diálogo Create Report, especifique el nombre del informe y la consulta SQL para recuperar la información para el informe. A continuación, haga clic en Apply.

En el ejemplo de la diapositiva, se especifica el nombre del informe como `emp_sal`. Se proporciona una descripción opcional que indica que el informe contiene los detalles de los empleados con `salary >= 10000`. La sentencia SQL completa para la recuperación de la información que se mostrará en el informe definido por el usuario se especifica en el cuadro SQL. También puede incluir una ayuda de burbuja opcional que se muestre al pasar el cursor brevemente sobre el nombre del informe en la pantalla del navegador Reports.

Puede organizar los informes definidos por el usuario en carpetas y puede crear una jerarquía de carpetas y subcarpetas. Para crear una carpeta para los informes definidos por el usuario, haga clic con el botón derecho en el nodo User Defined Reports o en cualquier nombre de carpeta de dicho nodo y seleccione Add Folder. La información sobre los informes definidos por el usuario, incluidas las carpetas de dichos informes, se almacena en un archivo denominado `UserReports.xml` en el directorio de información específica del usuario.

Motores de Búsqueda y Herramientas Externas



ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Motores de Búsqueda y Herramientas Externas

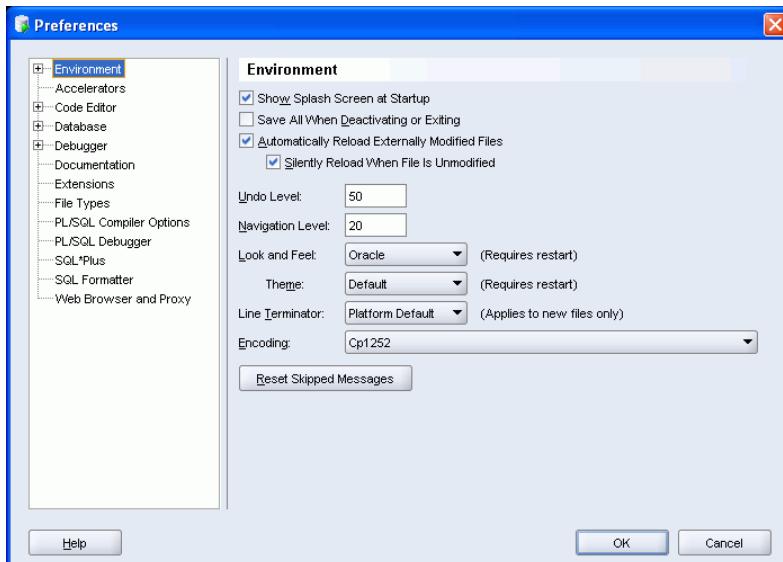
Para mejorar la productividad de los desarrolladores SQL, SQL Developer ha agregado enlaces rápidos a motores de búsqueda y foros de discusión conocidos, como AskTom, Google, etc. Asimismo, existen iconos de acceso directo a algunas de las herramientas de uso más frecuente, como Bloc de Notas, Microsoft Word y Dreamweaver.

Puede agregar herramientas externas a la lista existente o incluso suprimir los accesos directos a herramientas que no utilice frecuentemente. Para ello, realice los siguientes pasos:

1. En el menú Tools, seleccione External Tools.
2. En el cuadro de diálogo External Tools, seleccione New para agregar nuevas herramientas. Seleccione Delete para eliminar las herramientas de la lista.

Definición de Preferencias

- Personalice la interfaz y el entorno de SQL Developer.
- En el menú Tools, seleccione Preferences.



ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

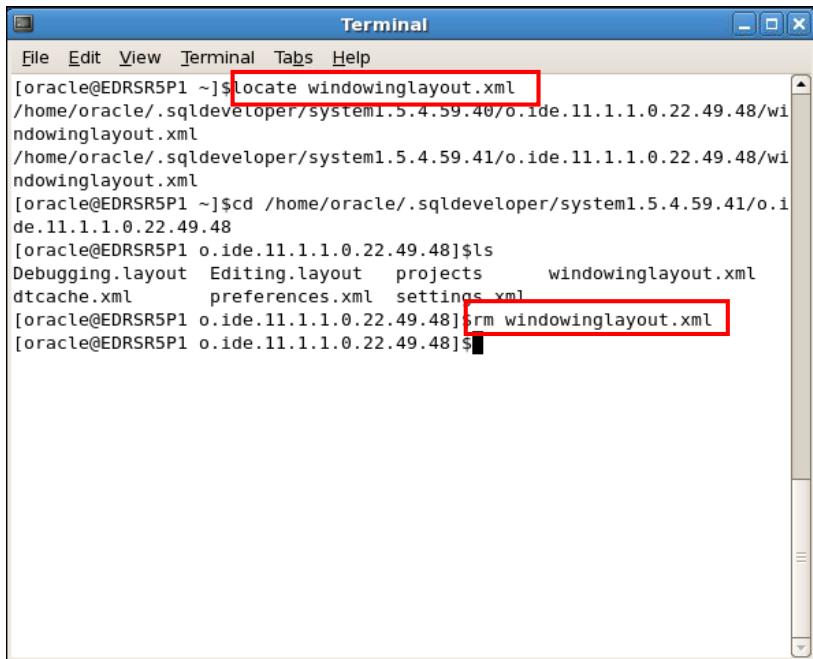
Definición de Preferencias

Puede personalizar muchos aspectos de la interfaz y del entorno de SQL Developer mediante la modificación de las preferencias de SQL Developer según sus necesidades. Para modificar las preferencias de SQL Developer, seleccione Tools y, a continuación, Preferences.

Las preferencias se agrupan en las siguientes categorías:

- Entorno
- Aceleradores (accesos directos de teclado)
- Editores de códigos
- Base de datos
- Depurador
- Documentación
- Extensiones
- Tipos de archivo
- Migración
- Compiladores PL/SQL
- Depurador PL/SQL, etc.

Restablecimiento del Diseño de SQL Developer



```
Terminal
File Edit View Terminal Tabs Help
[oracle@EDRSR5P1 ~]$ locate windowinglayout.xml
/home/oracle/.sqldeveloper/system1.5.4.59.40/o.ide.11.1.1.0.22.49.48/windowinglayout.xml
/home/oracle/.sqldeveloper/system1.5.4.59.41/o.ide.11.1.1.0.22.49.48/windowinglayout.xml
[oracle@EDRSR5P1 ~]$ cd /home/oracle/.sqldeveloper/system1.5.4.59.41/o.ide.11.1.1.0.22.49.48
[oracle@EDRSR5P1 o.ide.11.1.1.0.22.49.48]$ ls
Debugging.layout  Editing.layout  projects      windowinglayout.xml
dtcache.xml       preferences.xml   settings.xml
[oracle@EDRSR5P1 o.ide.11.1.1.0.22.49.48]$ rm windowinglayout.xml
[oracle@EDRSR5P1 o.ide.11.1.1.0.22.49.48]$
```



Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Restablecimiento del Diseño de SQL Developer

Mientras trabaja con SQL Developer, si desaparece Connections Navigator o si no puede acoplar la ventana Log en su lugar original, realice los pasos siguientes para corregir el problema:

1. Salga de SQL Developer.
2. Abra una ventana de terminal y utilice el comando locate para encontrar la ubicación de windowinglayout.xml.
3. Vaya al directorio que contenga windowinglayout.xml y suprímalo.
4. Reinicie SQL Developer.

Resumen

En este apéndice, debe haber aprendido cómo utilizar SQL Developer para realizar las siguientes acciones:

- Examinar, crear y editar objetos de bases de datos
- Ejecutar sentencias SQL y scripts en la hoja de trabajo de SQL
- Crear y guardar informes personalizados



Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Resumen

SQL Developer es una herramienta gráfica gratuita para simplificar las tareas de desarrollo de la base de datos. Con SQL Developer, puede examinar, crear y editar objetos de bases de datos. Puede utilizar la hoja de trabajo de SQL para ejecutar scripts y sentencias SQL. SQL Developer permite crear y guardar su propio juego especial de informes para un uso repetido.

Uso de SQL*Plus

ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Objetivos

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

- Conectarse a SQL*Plus
- Editar comandos SQL
- Formatear la salida con comandos SQL*Plus
- Interactuar con archivos de script

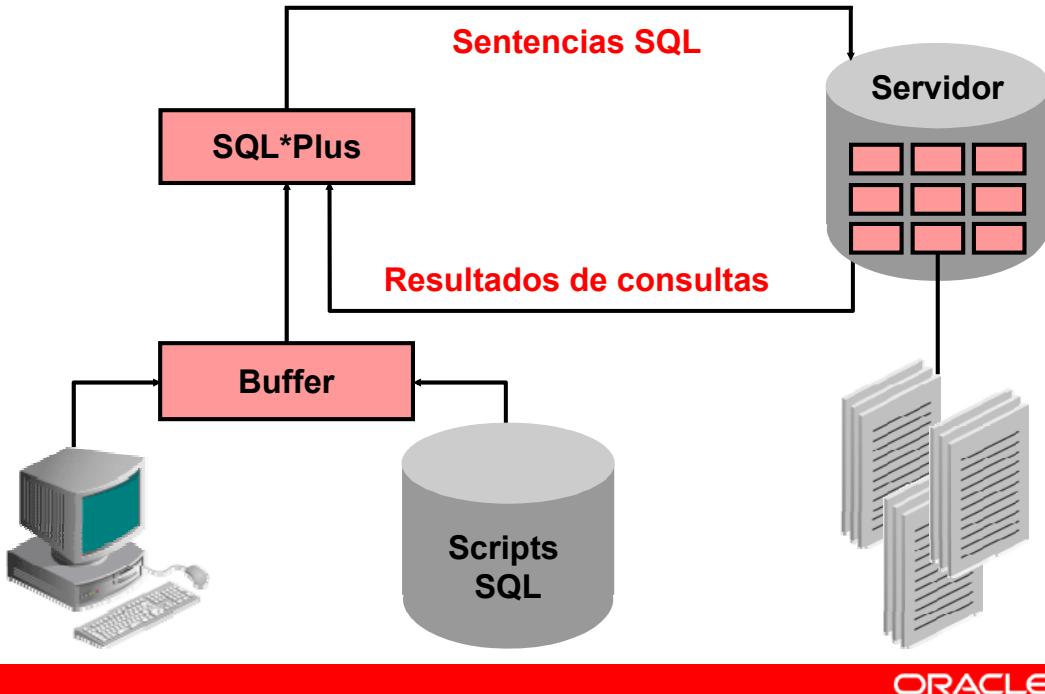


Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Objetivos

Puede que desee crear sentencias SELECT que se puedan utilizar repetidamente. En este apéndice también se aborda el uso de comandos SQL*Plus para ejecutar sentencias SQL. Aprenderá cómo formatear la salida mediante comandos SQL*Plus, editar comandos SQL y guardar scripts en SQL*Plus.

Interacción de SQL y SQL*Plus



Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

ORACLE

SQL y SQL*Plus

SQL es un lenguaje de comandos que se utiliza para la comunicación con Oracle Server desde cualquier herramienta o aplicación. Oracle SQL contiene muchas extensiones. Al introducir una sentencia SQL, ésta se almacena en una parte de la memoria denominada *buffer SQL* y permanece allí hasta que introduzca una nueva sentencia SQL. SQL*Plus es una herramienta de Oracle que reconoce y envía sentencias SQL en Oracle9i Server para su ejecución. Contiene su propio lenguaje de comandos.

Funciones de SQL

- Las pueden utilizar una gran variedad de usuarios, incluidos aquéllos con poca o ninguna experiencia de programación.
- Es un lenguaje que no es de procedimientos.
- Reduce la cantidad de tiempo necesario para crear y mantener sistemas.
- Es un lenguaje como el inglés.

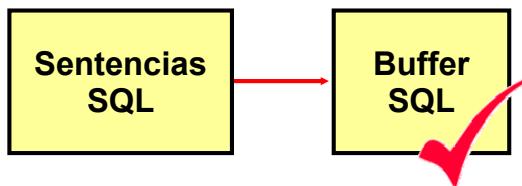
Funciones de SQL*Plus

- Acepta la entrada ad hoc de sentencias.
- Acepta la entrada de SQL de los archivos.
- Proporciona un editor de líneas para modificar sentencias SQL.
- Controla la configuración de entorno.
- Formatea resultados de consulta en informes básicos.
- Accede a bases de datos locales y remotas.

Sentencias SQL frente a Comandos SQL*Plus

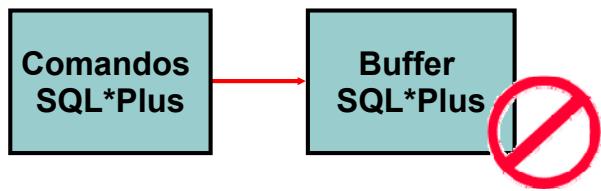
SQL

- Un lenguaje
- Estándar de ANSI
- Las palabras clave no se pueden abreviar.
- Las sentencias manipulan definiciones de tablas y datos en la base de datos.



SQL*Plus

- Un entorno
- Propiedad de Oracle
- Las palabras clave se pueden abreviar.
- Los comandos no permiten la manipulación de valores en la base de datos.



ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

SQL y SQL*Plus (continuación)

En la siguiente tabla se compara SQL y SQL*Plus:

SQL	SQL*Plus
Lenguaje para la comunicación con el servidor de Oracle para acceder a los datos.	Reconoce las sentencias SQL y las envía al servidor.
Se basa en SQL del estándar ANSI (American National Standards Institute).	Interfaz propiedad de Oracle para ejecutar sentencias SQL.
Manipula la definiciones de tablas y datos en la base de datos.	No permite la manipulación de valores en la base de datos.
Se introduce en el buffer SQL en una o más líneas.	Se introduce en una línea al mismo tiempo y no se almacena en el buffer SQL.
No tiene ningún carácter de continuación.	Utiliza un guión (-) como carácter de continuación si el comando es más largo que una línea.
No se puede abreviar.	Se puede abreviar.
Utiliza un carácter de terminación para ejecutar comandos inmediatamente.	No necesita caracteres de terminación; ejecuta los comandos inmediatamente.
Utiliza funciones para realizar algunas tareas de formato.	Utiliza comandos para formatear datos.

SQL*Plus: Visión General

- Conéctese a SQL*Plus.
- Describa la estructura de la tabla.
- Edite la sentencia SQL.
- Ejecute SQL desde SQL*Plus.
- Guarde sentencias SQL en archivos y agregue sentencias SQL a los archivos.
- Ejecute archivos guardados.
- Cargue comandos del archivo en el buffer para la edición.

ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

SQL*Plus

SQL*Plus es un entorno en el que puede:

- Ejecutar sentencias SQL para recuperar, modificar, agregar y eliminar datos de la base de datos.
- Formatear, realizar cálculos, almacenar e imprimir resultados de consulta como informes.
- Crear archivos de script para almacenar sentencias SQL para un uso repetido en el futuro.

Los comandos SQL*Plus se pueden dividir en las siguientes categorías principales:

Categoría	Objetivo
Entorno	Afectar al comportamiento general de sentencias SQL para la sesión.
Formato	Formatear resultados de consulta.
Manipulación de archivos	Guardar, cargar y ejecutar archivos de comandos.
Ejecución	Enviar sentencias SQL del buffer SQL al servidor de Oracle.
Edición	Modificar sentencias SQL en el buffer.
Interacción	Crear y transferir variables a la sentencia SQL, imprimir valores de variables y mensajes en la pantalla.
Otros	Conectarse a la base de datos, manipular el entorno SQL*Plus y mostrar definiciones de columnas.

Conexión a SQL*Plus

```

Terminal
File Edit View Terminal Tabs Help
[oracle@EDRSR5P1 ~]$sqlplus
SQL*Plus: Release 11.2.0.0.2 Beta on Tue May 26 19:59:06 2009
Copyright (c) 1982, 2009, Oracle. All rights reserved.
Enter user-name: ora21@orcl
Enter password:
Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.0.2 - Beta
With the Partitioning, OLAP, Data Mining and Real Application Testing options
SQL>

```

sqlplus [username[/password[@database]]]

```

Terminal
File Edit View Terminal Tabs Help
[oracle@EDRSR5P1 ~]$sqlplus ora21/ora21@orcl
SQL*Plus: Release 11.2.0.0.2 Beta on Tue May 26 19:58:06 2009
Copyright (c) 1982, 2009, Oracle. All rights reserved.

Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.0.2 - Beta
With the Partitioning, OLAP, Data Mining and Real Application Testing options
SQL>

```

ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Conexión a SQL*Plus

La forma de llamar a SQL*Plus dependerá del tipo de sistema operativo que esté ejecutando con Oracle Database.

Para conectarse desde un entorno Linux, realice los siguientes pasos:

1. Haga clic con el botón derecho en el escritorio de Linux y seleccione el terminal.
2. Introduzca el comando `sqlplus` que se muestra en la diapositiva.
3. Introduzca el nombre de usuario, la contraseña y el nombre de la base de datos.

En la sintaxis:

<code>username</code>	Es el nombre de usuario de la base de datos
<code>password</code>	Es la contraseña de la base de datos (la contraseña será visible si la introduce aquí).
<code>@database</code>	Es la cadena de conexión de la base de datos

Nota: para asegurarse de la integridad de la contraseña, no la introduzca en la petición de datos del sistema operativo. En su lugar, introduzca sólo el nombre de usuario. Introduzca la contraseña en la petición de datos de la contraseña.

Visualización de la Estructura de la Tabla

Utilice el comando SQL*Plus `DESCRIBE` para mostrar la estructura de una tabla:

```
DESC[RIBE] tablename
```

ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Visualización de la Estructura de la Tabla

En SQL*Plus, puede mostrar la estructura de una tabla mediante el comando `DESCRIBE`. El resultado del comando es una visualización de los nombres de columna y tipos de dato, así como una indicación de si una columna debe contener datos.

En la sintaxis:

`tablename` Es el nombre de cualquier tabla existente, vista o sinónimo al que puede acceder el usuario

Para describir la tabla `DEPARTMENTS`, utilice este comando:

```
SQL> DESCRIBE DEPARTMENTS
      Name          Null?    Type
----- 
DEPARTMENT_ID      NOT NULL NUMBER(4)
DEPARTMENT_NAME    NOT NULL VARCHAR2(30)
MANAGER_ID         NUMBER(6)
LOCATION_ID        NUMBER(4)
```

Visualización de la Estructura de la Tabla

```
DESCRIBE departments
```

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

ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Visualización de la Estructura de Tabla (continuación)

En el ejemplo de la diapositiva se muestra la información sobre la estructura de la tabla DEPARTMENTS. En el resultado:

Null? : especifica si una columna debe contener datos (NOT NULL indica que una columna debe contener datos.)

Type : muestra el tipo de dato de una columna.

Comandos de Edición SQL*Plus

- A [PPEND] *text*
- C [HANGE] / *old* / *new*
- C [HANGE] / *text* /
- CL [EAR] BUFF [ER]
- DEL
- DEL *n*
- DEL *m n*

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Comandos de Edición SQL*Plus

Los comandos SQL*Plus se introducen en una línea al mismo tiempo y no se almacenan en el buffer SQL.

Comando	Descripción
A [PPEND] <i>text</i>	Agrega texto al final de la línea actual.
C [HANGE] / <i>old</i> / <i>new</i>	Cambia el texto antiguo por el nuevo en la línea actual.
C [HANGE] / <i>text</i> /	Suprime el texto de la línea actual.
CL [EAR] BUFF [ER]	Suprime todas las líneas del buffer SQL.
DEL	Suprime la línea actual.
DEL <i>n</i>	Suprime la línea <i>n</i> .
DEL <i>m n</i>	Suprime de las líneas <i>m</i> hasta la <i>n</i> inclusive.

Instrucciones

- Si pulsa Intro antes de que haya terminado la ejecución de un comando, SQL*Plus le solicitará un número de línea.
- Termine el buffer SQL introduciendo uno de los caracteres de terminación (punto y coma o barra) o pulsando Intro dos veces. A continuación, aparecerá la petición de datos de SQL.

Comandos de Edición SQL*Plus

- `I [NPUT]`
- `I [NPUT] text`
- `L [IST]`
- `L [IST] n`
- `L [IST] m n`
- `R [UN]`
- `n`
- `n text`
- `0 text`

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Comandos de Edición SQL*Plus (continuación)

Comando	Descripción
<code>I [NPUT]</code>	Inserta un número indefinido de líneas.
<code>I [NPUT] text</code>	Inserta una línea que consta de <i>texto</i> .
<code>L [IST]</code>	Muestra todas las líneas en el buffer SQL.
<code>L [IST] n</code>	Muestra una línea (especificada por <i>n</i>).
<code>L [IST] m n</code>	Muestra un rango de líneas (<i>m</i> a <i>n</i> inclusive).
<code>R [UN]</code>	Muestra y ejecuta la sentencia SQL actual en el buffer.
<code>n</code>	Especifica la línea para crear la línea actual.
<code>n text</code>	Sustituye la línea <i>n</i> con <i>texto</i> .
<code>0 text</code>	Inserta una línea delante de la línea 1.

Nota: puede introducir sólo un comando SQL*Plus para cada petición de datos de SQL. Los comandos SQL*Plus no se almacenan en el buffer. Para que un comando SQL*Plus continúe en la siguiente línea, finalice la primera línea con un guión (-).

Uso de LIST, n y APPEND

```
LIST
 1  SELECT last_name
 2* FROM   employees
```

```
1
1* SELECT last_name
```

```
A , job_id
1* SELECT last_name, job_id
```

```
LIST
 1  SELECT last_name, job_id
 2* FROM   employees
```

ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Uso de LIST, n y APPEND

- Utilice el comando L [IST] para mostrar el contenido del buffer SQL. El asterisco (*) situado junto a la línea 2 en el buffer indica que la línea 2 es la línea actual. Cualquier edición que realice se aplica a la línea actual.
- Introduzca el número (n) de la línea que desea editar para cambiar el número de la línea actual. Se muestra la nueva línea actual.
- Utilice el comando A [PPEND] para agregar texto a la línea actual. Se muestra la línea recién editada. Verifique el nuevo contenido del buffer mediante el comando LIST.

Nota: muchos de los comandos SQL*Plus, incluidos LIST y APPEND, se pueden abreviar sólo con sus primeras letras. LIST se puede abreviar con L; APPEND se puede abreviar con A.

Uso del Comando CHANGE

```
LIST
```

```
1* SELECT * from employees
```

```
c/employees/departments
```

```
1* SELECT * from departments
```

```
LIST
```

```
1* SELECT * from departments
```

ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Uso del Comando CHANGE

- Utilice L [LIST] para mostrar el contenido del buffer.
- Utilice el comando C [CHANGE] para modificar el contenido de la línea actual del buffer SQL. En este caso, sustituya la tabla employees por la tabla departments. Se muestra la nueva línea actual.
- Utilice el comando L [LIST] para verificar el nuevo contenido del buffer.

Comandos de Archivos SQL*Plus

- `SAVE filename`
- `GET filename`
- `START filename`
- `@ filename`
- `EDIT filename`
- `SPOOL filename`
- `EXIT`

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Comandos de Archivos SQL*Plus

Las sentencias SQL se comunican con Oracle Server. Los comandos SQL*Plus controlan el entorno, formatean los resultados de la consulta y gestionan archivos. Puede utilizar los comandos descritos en la siguiente tabla:

Comando	Descripción
<code>SAV[E] filename [.ext]</code> [REP[LACE]APP[END]]	Guarda el contenido actual del buffer SQL en un archivo. Utilice APPEND para agregar a un archivo existente; utilice REPLACE para sobrescribir un archivo existente. La extensión por defecto es <code>.sql</code> .
<code>GET filename [.ext]</code>	Escribe el contenido de un archivo guardado anteriormente en el buffer SQL. La extensión por defecto del nombre de archivo es <code>.sql</code> .
<code>STA[RT] filename [.ext]</code>	Ejecuta el archivo de comandos guardado anteriormente.
<code>@ filename</code>	Ejecuta un archivo de comandos guardado anteriormente (igual que START).
<code>ED[IT]</code>	Llama al editor y guarda el contenido del buffer en un archivo denominado <code>afiedt.buf</code> .
<code>ED[IT] [filename [.ext]]</code>	Llama al editor para editar el contenido de un archivo guardado.
<code>SPO[OL] [filename [.ext]] OFF OUT</code>	Almacena los resultados de la consulta en un archivo. OFF cierra el archivo de spool. OUT cierra el archivo de spool y envía los resultados del archivo a la impresora.
<code>EXIT</code>	Sale de SQL*Plus.

Uso de los Comandos SAVE y START

```
LIST
```

```
1  SELECT last_name, manager_id, department_id  
2* FROM employees
```

```
SAVE my_query
```

```
Created file my_query
```

```
START my_query
```

```
LAST_NAME
```

```
MANAGER_ID DEPARTMENT_ID
```

```
-----  
King
```

```
90
```

```
Kochhar
```

```
100
```

```
90
```

```
...
```

```
107 rows selected.
```

ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Uso de los Comandos SAVE y START

SAVE

Utilice el comando SAVE para almacenar el contenido actual del buffer en un archivo. De esta forma, podrá almacenar los scripts usados con frecuencia para utilizarlos en el futuro.

START

Utilice el comando START para ejecutar un script en SQL*Plus. También puede utilizar el símbolo @ para ejecutar un script.

```
@my_query
```

Comando SERVEROUTPUT

- Utilice el comando `SET SERVEROUT [PUT]` para controlar si se debe mostrar la salida de los procedimientos almacenados o de los bloques PL/SQL en SQL*Plus.
- El límite de la longitud de línea `DBMS_OUTPUT` aumenta de 255 bytes a 32.767 bytes.
- El tamaño por defecto ahora es ilimitado.
- Los recursos no están preasignados cuando se define SERVEROUTPUT.
- Debido a que no hay ninguna penalización de rendimiento, utilice `UNLIMITED` a menos que desee conservar la memoria física.

```
SET SERVEROUT [PUT] {ON | OFF} [SIZE {n | UNLIMITED}]
[FOR [MAT] {WRA[PPED] | WOR[D_WRAPPED] | TRU[NATED]}]
```

ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Comando SERVEROUTPUT

La mayoría de los programas PL/SQL realizan la entrada y salida mediante sentencias SQL para almacenar datos en tablas de base de datos o para consultar dichas tablas. Las demás entradas o salidas PL/SQL se realizan mediante API que interactúan con otros programas. Por ejemplo, el paquete `DBMS_OUTPUT` tiene procedimientos como `PUT_LINE`. Para ver el resultado fuera de PL/SQL, se necesita otro programa, como SQL*Plus, para leer y mostrar los datos transferidos a `DBMS_OUTPUT`.

SQL*Plus no muestra los datos de `DBMS_OUTPUT` a no ser que antes emita el comando SQL*Plus `SET SERVEROUTPUT ON`, de la siguiente forma:

```
SET SERVEROUTPUT ON
```

Nota

- `SIZE` define el número de bytes de la salida almacenados en buffer en el servidor de Oracle Database. El valor por defecto es `UNLIMITED`. `n` no puede ser menor que 2.000 ni mayor que 1.000.000.
- Para obtener más información sobre SERVEROUTPUT, consulte *Oracle Database PL/SQL User's Guide and Reference 11g* (Guía del Usuario y Referencia PL/SQL de Oracle Database 11g).

Uso del Comando SQL*Plus SPOOL

```
SPO[OL] [file_name[.ext]] [CRE[ATE] | REP[LACE] | APP[END]] | OFF | OUT]
```

Opción	Descripción
file_name[.ext]	Envía la salida al nombre de archivo especificado
CRE[ATE]	Crea un nuevo archivo con el nombre especificado
REP[LACE]	Sustituye el contenido de un archivo existente. Si el archivo no existe, REPLACE crea el archivo.
APP[END]	Agrega el contenido del buffer al final del archivo especificado
OFF	Para el envío
OUT	Para el envío y manda el archivo a la impresora estándar (por defecto) de la computadora

ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Uso del Comando SQL*Plus SPOOL

El comando SPOOL almacena el resultado de la consulta en un archivo o bien envía el archivo a una impresora. El comando SPOOL se ha mejorado. Ahora puede agregar o sustituir un archivo existente, donde antes sólo se podía utilizar SPOOL para crear (y sustituir) un archivo. REPLACE es el valor por defecto.

Para enviar la salida generada por comandos en un script sin que la salida aparezca en pantalla, utilice SET TERMOUT OFF. SET TERMOUT OFF no afecta a la salida de comandos que se ejecutan de forma interactiva.

Debe utilizar comillas simples en los nombres de archivos que contengan espacio en blanco. Para crear un archivo HTML válido con comandos SPOOL APPEND, debe utilizar el comando PROMPT o uno similar para crear la cabecera y el pie de página HTML. El comando SPOOL APPEND no analiza etiquetas HTML. Defina SQLPLUSCOMPAT [IBILITY] en 9.2 o una versión anterior para desactivar los parámetros CREATE, APPEND y SAVE.

Uso del Comando AUTOTRACE

- Muestra un informe después de la ejecución correcta de sentencias SQL DML como SELECT, INSERT, UPDATE o DELETE.
- El informe puede ahora incluir estadísticas de ejecución y la ruta de acceso de ejecución de la consulta.

```
SET AUTOT[RACE] {ON | OFF | TRACE[ONLY]} [EXP[LAIN]]  
[STATISTICS]
```

```
SET AUTOTRACE ON  
-- The AUTOTRACE report includes both the optimizer  
-- execution path and the SQL statement execution  
-- statistics
```



Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Uso del Comando AUTOTRACE

EXPLAIN muestra la ruta de acceso de ejecución de la consulta realizando un EXPLAIN PLAN. STATISTICS muestra las estadísticas de la sentencia SQL. El formato del informe AUTOTRACE puede variar en función de la versión del servidor al que esté conectado y de la configuración del servidor. El paquete DBMS_XPLAN proporciona una forma fácil de mostrar la salida del comando EXPLAIN PLAN en varios formatos predefinidos.

Nota

- Para obtener más información sobre paquetes y subprogramas, consulte *Oracle Database PL/SQL Packages and Types Reference 11g Guide* (Referencia de Tipos y Paquetes PL/SQL de Oracle Database 11g).
- Para obtener más información sobre EXPLAIN PLAN, consulte *Oracle Database SQL Reference 11g* (Referencia SQL de Oracle Database 11g).
- Para obtener más información sobre los planes de ejecución y las estadísticas, consulte *Oracle Database Performance Tuning Guide 11g* (Guía de Ajuste de Rendimiento de Oracle Database 11g).

Resumen

En este apéndice, debe haber aprendido cómo utilizar SQL*Plus como un entorno para realizar las siguientes acciones:

- Ejecutar sentencias SQL
- Editar sentencias SQL
- Formatear la salida
- Interactuar con archivos de script



Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Resumen

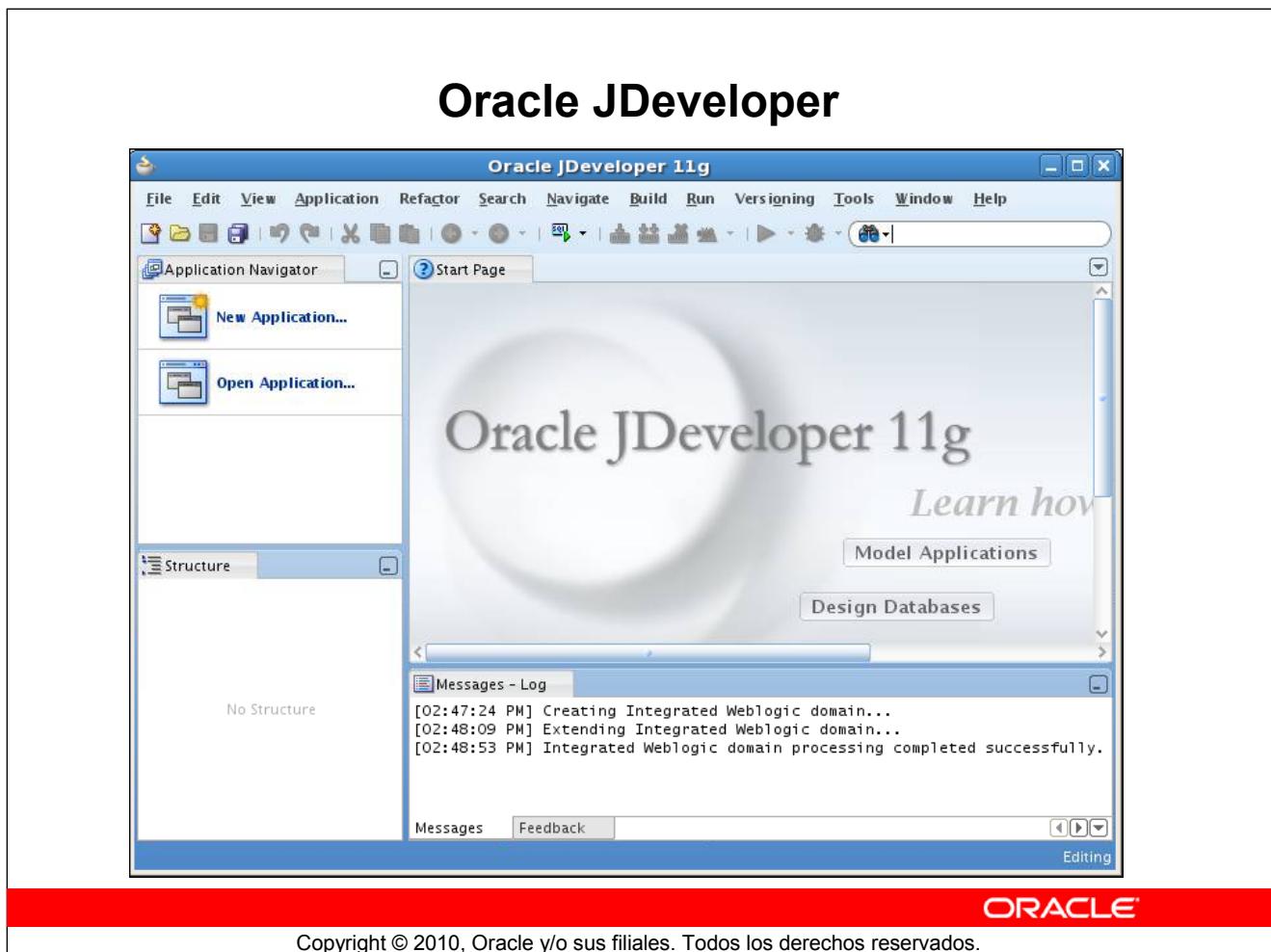
SQL*Plus es un entorno de ejecución que puede utilizar para enviar comandos SQL al servidor de la base datos para editar y guardar los comandos SQL. Puede ejecutar los comandos desde la petición de datos de SQL o desde un archivo de script.



Uso de JDeveloper

ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

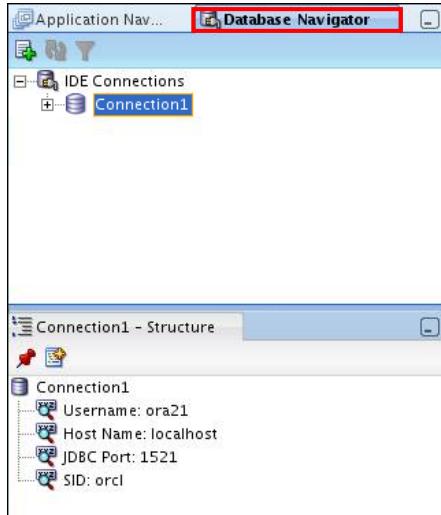


Oracle JDeveloper

Oracle JDeveloper es un entorno de desarrollo de integración (IDE) para desarrollar y desplegar aplicaciones Java y servicios web. Soporta cada etapa del ciclo de vida de desarrollo de software (SDLC), del modelado al despliegue. Tiene funciones que permiten utilizar los últimos estándares de la industria para Java, XML y SQL y desarrollar una aplicación.

Oracle JDeveloper 11g inicia un nuevo enfoque al desarrollo J2EE con funciones que permiten un desarrollo visual y declarativo. Este enfoque innovador hace que el desarrollo J2EE sea sencillo y eficaz.

Database Navigator

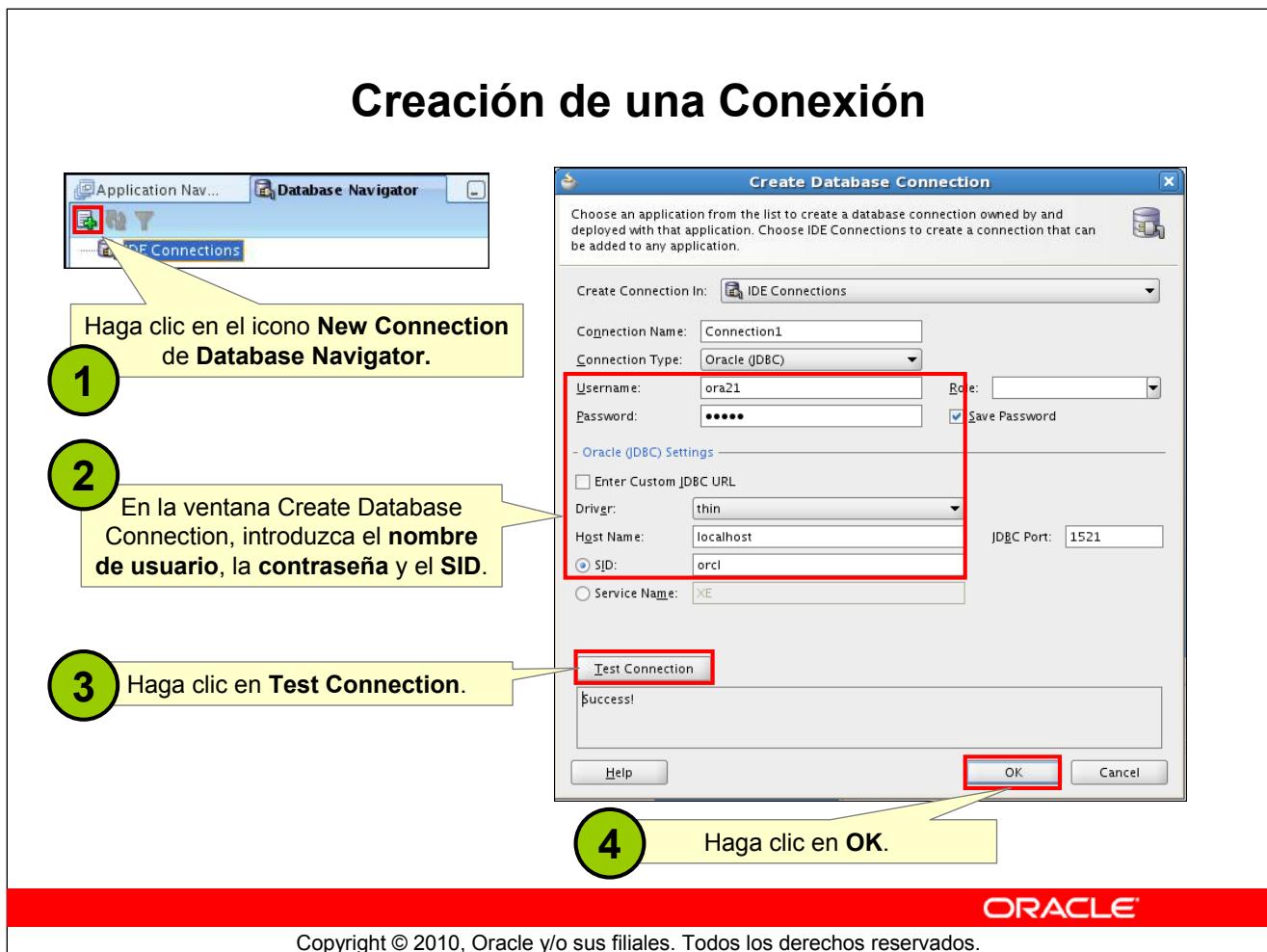


ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Database Navigator

Con Oracle JDeveloper, puede almacenar la información necesaria para conectar a la base de datos en un objeto denominado “conexión”. Una conexión se almacena como parte de la configuración de IDE y se puede exportar e importar para compartirlo fácilmente entre grupos de usuarios. Una conexión tiene diferentes fines, desde la exploración de la base de datos y la creación de aplicaciones hasta el despliegue.



Creación de una Conexión

Una conexión es un objeto que especifica la información necesaria para conectarse a una base de datos concreta como usuario específico de dicha base de datos. Puede crear y probar conexiones para varias bases de datos y esquemas.

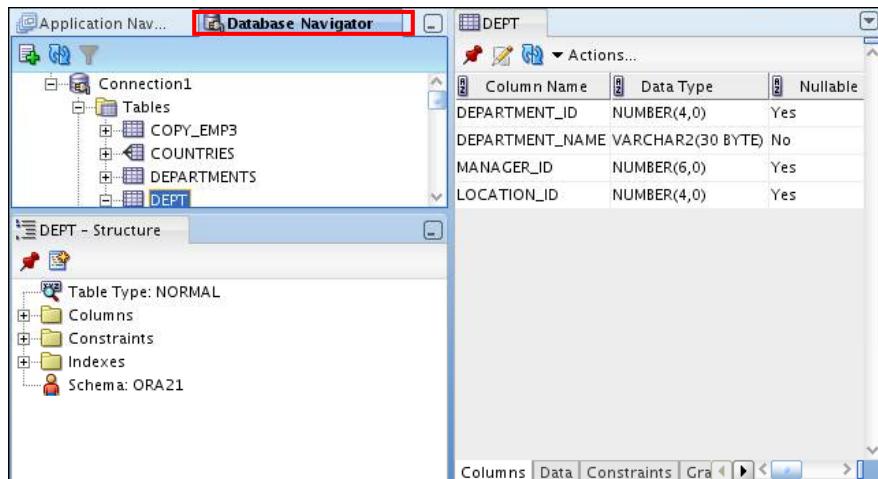
Para crear una conexión a la base de datos, realice los siguientes pasos:

1. Haga clic en el ícono New Connection de Database Navigator.
2. En la ventana Create Database Connection, introduzca el nombre de la conexión. Introduzca el nombre de usuario y la contraseña del esquema al que desea conectarse. Introduzca el SID de la base de datos a la que desea conectarse.
3. Haga clic en Test para asegurarse de que la conexión se ha definido correctamente.
4. Haga clic en OK.

Exploración de Objetos de Bases de Datos

Utilice Database Navigator para:

- Examinar los objetos de un esquema de base de datos
- Revisar las definiciones de objetos de forma rápida



ORACLE

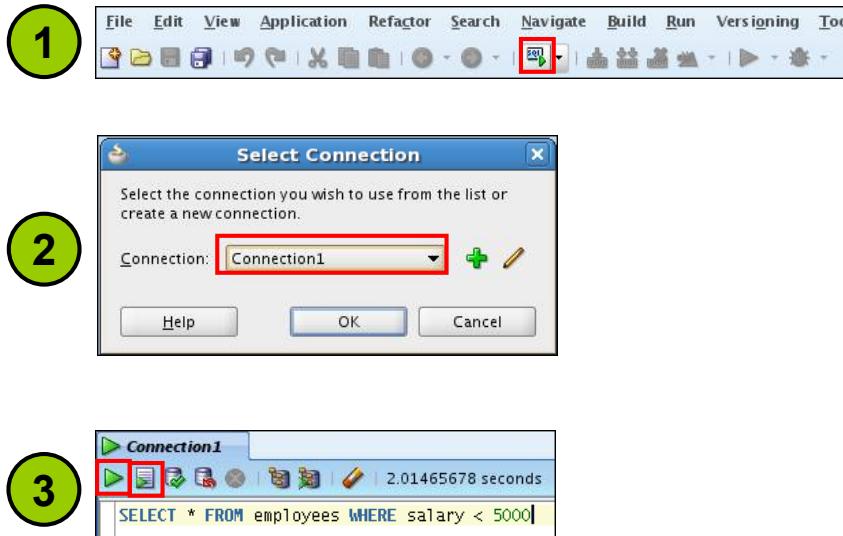
Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Exploración de Objetos de Bases de Datos

Una vez creada la conexión a la base de datos, puede utilizar Database Navigator para examinar muchos objetos de un esquema de base de datos, entre los que se incluyen tablas, vistas, índices, paquetes, procedimientos, disparadores y tipos.

Puede ver la definición de los objetos desglosados en separadores de información que se transfieren al diccionario de datos. Por ejemplo, si selecciona una tabla en Navigator, se muestran los detalles sobre las columnas, las restricciones, los permisos, las estadísticas, los disparadores, etc. en una página con separadores fácil de leer.

Ejecución de Sentencias SQL



ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Ejecución de Sentencias SQL

Para ejecutar una sentencia SQL, realice los siguientes pasos:

1. Haga clic en el ícono Open SQL Worksheet.
2. Seleccione la conexión.
3. Ejecute el comando SQL con una de las siguientes opciones:
 - El botón **Execute statement** o pulsando F9. La salida es la siguiente:

A screenshot of the SQL Worksheet showing the results of the query 'SELECT * FROM employees WHERE salary < 5000'. The results are displayed in a grid format:

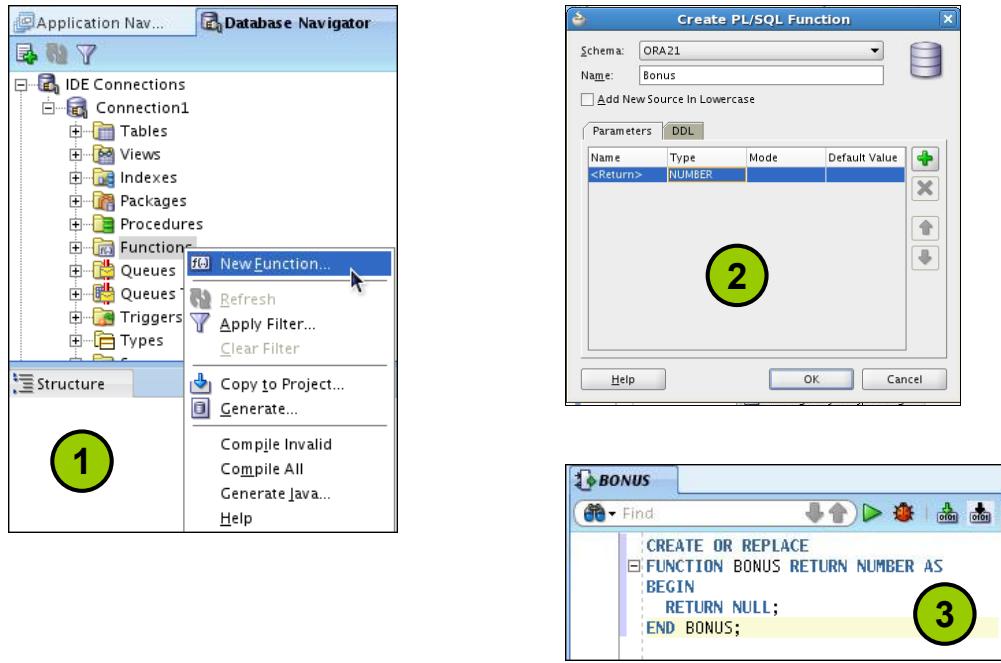
EMPLOYEE_ID	FIRST_NAME	LAST_NAME
100	Steven	King
101	Neena	Kochhar

- El botón **Run Script** o pulsando F5. La salida es la siguiente:

A screenshot of the SQL Worksheet showing the results of the query 'SELECT * FROM employees WHERE salary < 5000' using the 'Script Output' tab. The results are displayed in a grid format:

EMPLOYEE_ID	FIRST_NAME	LAST_NAME
100	Steven	King

Creación de Unidades de Programa



Esqueleto de la Función

ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Creación de Unidades de Programa

Para crear una unidad de programa PL/SQL, realice los siguientes pasos:

1. Seleccione View > Database Navigator. Seleccione y amplíe una conexión de base de datos. Haga clic con el botón derecho en una carpeta que corresponda al tipo de objeto (procedimientos, paquetes o funciones). Seleccione “New [Procedures|Packages|Functions]”.
2. Introduzca un nombre válido para la función, el paquete o el procedimiento y haga clic en OK.
3. Se crea una definición de esqueleto que se abre en la ventana Code Editor. A continuación, puede editar el subprograma para que se ajuste a sus necesidades.

Compilación

Messages - Log
Compiling...
Context: MakeSelectedCommand selection=Element containing file
/home/oracle/Middleware/jdk160_11/jre/bin/java -jar /
[5:14:32 PM] Successful compilation: 0 errors, 0 warnings.

Compilación con errores

Compiler - Log
Project: /home/oracle/jdeveloper/mywork/Application1/Project1/Project1
/home/oracle/jdeveloper/mywork/Application1/Project1/src/project1
Error(7,13): duplicate definition of variable a in constructor Hello()
Error(7,15): ; expected
Error(8,28): variable a might not have been initialized

Compilación sin errores

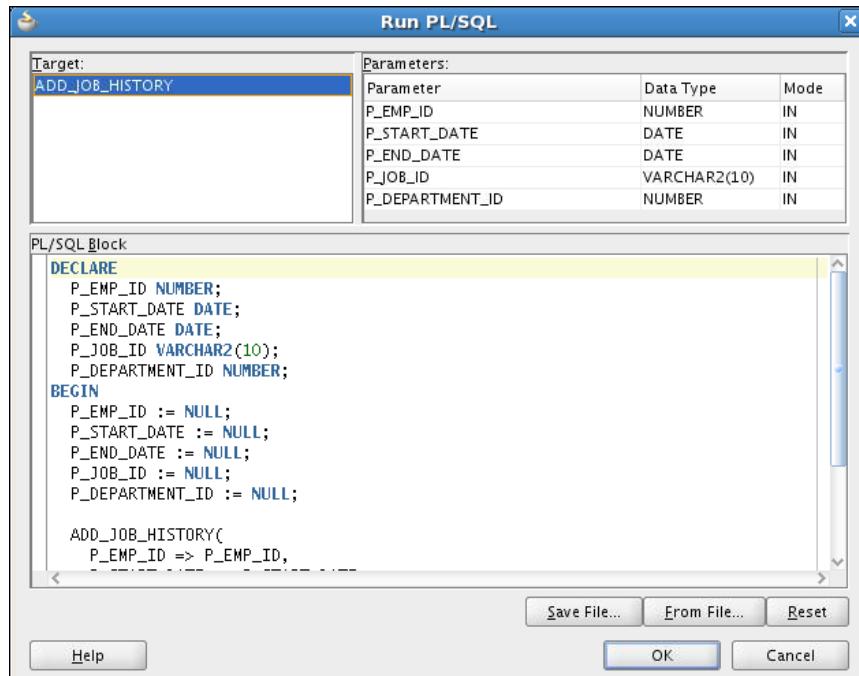
ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Compilación

Después de editar la definición del esqueleto, debe compilar la unidad de programa. Haga clic con el botón derecho en el objeto PL/SQL que necesita compilar en Connection Navigator y, a continuación, seleccione Compile. También puede pulsar CTRL + MAYÚS + F9 para compilar.

Ejecución de una Unidad de Programa



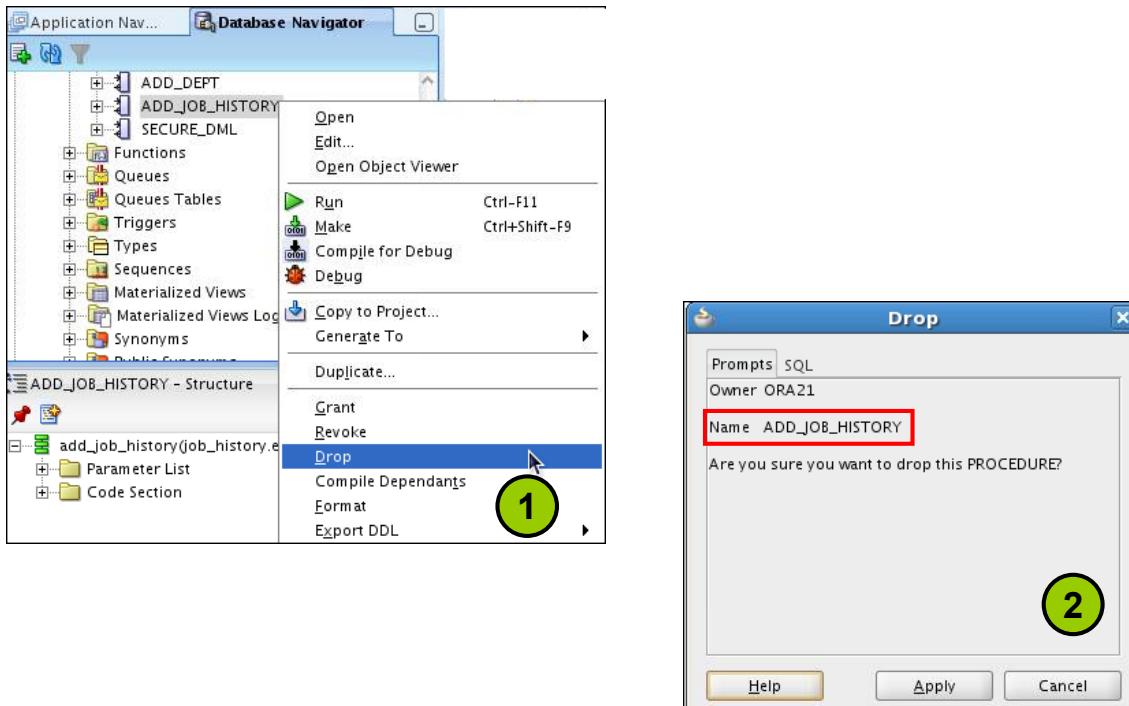
ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Ejecución de una Unidad de Programa

Para ejecutar la unidad de programa, haga clic con el botón derecho en el objeto y seleccione Run. Aparece el cuadro de diálogo Run PL/SQL. Puede que tenga que cambiar los valores NULL por valores razonables que se transfieren a la unidad de programa. Después de cambiar los valores, haga clic en OK. La salida se muestra en la ventana Message-Log.

Borrado de una Unidad de Programa



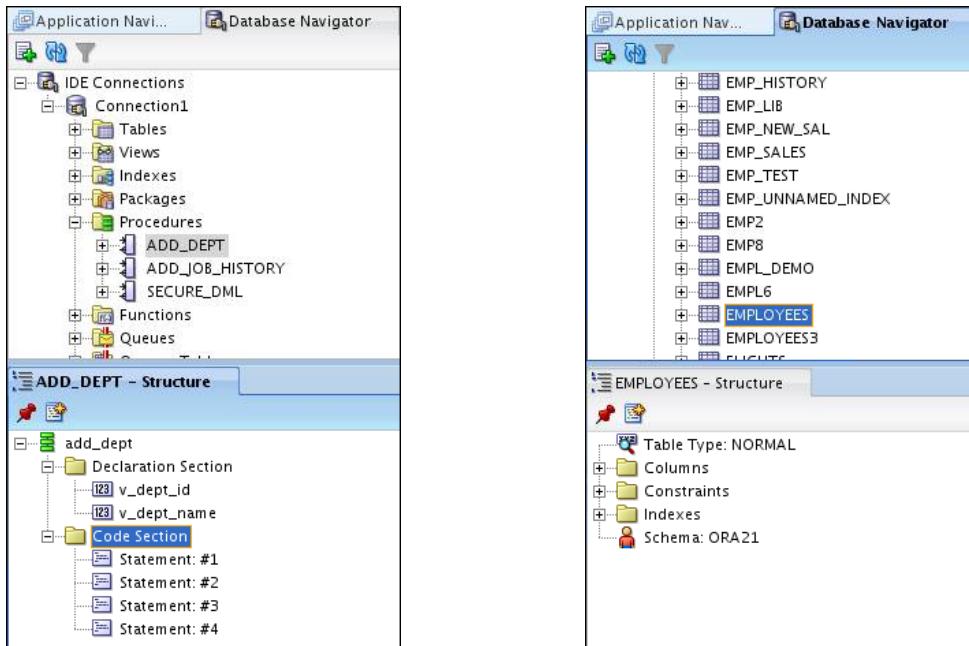
ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Borrado de una Unidad de Programa

Para borrar una unidad de programa, haga clic con el botón derecho en el objeto y seleccione Drop. Aparece el cuadro de diálogo Drop Confirmation. Haga clic en **Apply**. El objeto se borrará de la base de datos.

Ventana Structure



ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Ventana Structure

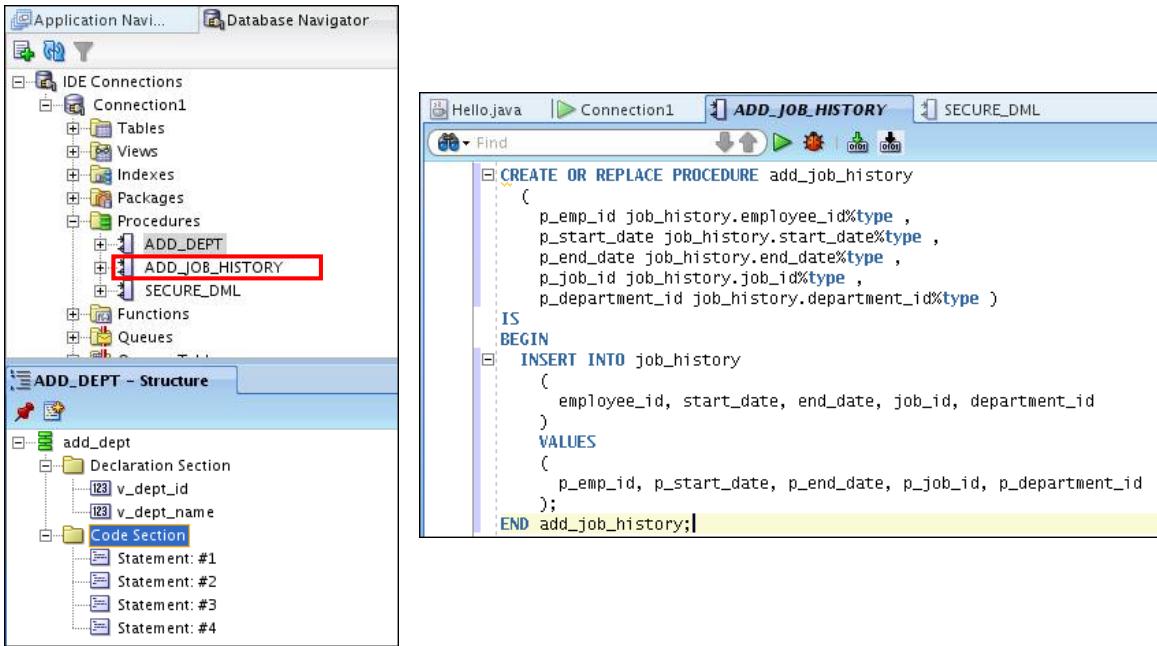
La ventana Structure ofrece una vista estructural de los datos del documento que está seleccionado actualmente en la ventana activa de las ventanas que proporcionan estructura: navegadores, editores, visores y Property Inspector.

Seleccione View > Structure para ver la ventana Structure.

En la ventana Structure, puede ver los datos del documento de distintas formas. Las estructuras que están disponibles para visualización están basadas en el tipo de documento. Para un archivo Java, puede ver la estructura del código, la estructura de la interfaz de usuario o los datos de modelo de interfaz de usuario. Para un archivo XML, puede ver la estructura XML, la estructura del diseño o los datos de modelo de interfaz de usuario.

La ventana Structure es dinámica y realiza siempre un seguimiento de la selección actual de la ventana activa (a menos que congele el contenido de la ventana en una vista concreta), ya que está relacionada con el editor que está actualmente activo. Cuando la selección actual es un nodo del navegador, se asume el editor por defecto. Para cambiar la vista en la estructura de la selección actual, haga clic en un separador de estructura distinto.

Ventana Editor



ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

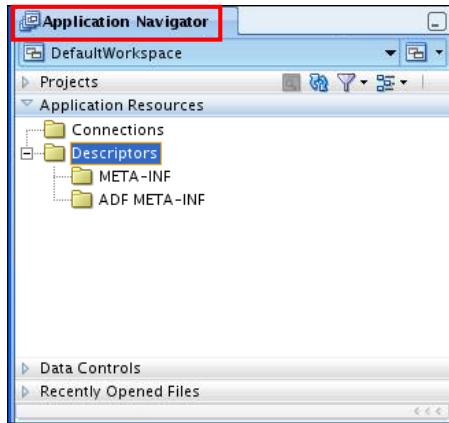
Ventana Editor

Al hacer clic dos veces en el nombre de una unidad de programa, se abre en la ventana Editor. Puede ver todos los archivos de proyectos en una única ventana del editor, abrir varias vistas del mismo archivo o abrir varias vistas de diferentes archivos.

Los separadores situados en la parte superior de la ventana del editor son los separadores del documento. Al hacer clic en un separador del documento, dicho documento se enfoca y se coloca en primer plano en la ventana del editor actual.

Los separadores situados en la parte inferior de la ventana del editor para un archivo concreto son los separadores del editor. Al hacer clic en un separador del editor, el archivo se abre en ese editor.

Application Navigator



ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

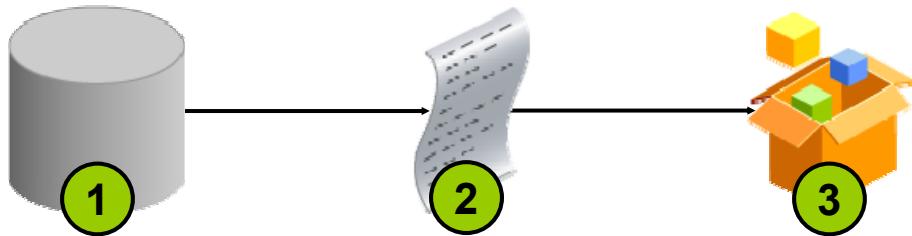
Application Navigator

Application Navigator proporciona una vista lógica de la aplicación y de los datos que contiene. Además proporciona una infraestructura a la que las distintas extensiones pueden conectarse y utilizar para organizar los datos y menús de forma abstracta y consistente. Aunque Application Navigator puede contener archivos individuales (como archivos de origen Java), está diseñado para consolidar datos complejos. Los tipos de dato complejos como los objetos de entidades, diagramas Unified Modeling Language (UML), Enterprise JavaBeans (EJB) o servicios web aparecen en el navegador como nodos únicos. Los archivos raw que componen estos nodos abstractos aparecen en la ventana Structure.

Despliegue de Procedimientos Java Almacenados

Antes de desplegar procedimientos Java almacenados, realice los siguientes pasos:

1. Cree una conexión de base de datos.
2. Cree un perfil de despliegue.
3. Despliegue los objetos.



ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Despliegue de Procedimientos Java Almacenados

Cree un perfil de despliegue para los procedimientos Java almacenados y, a continuación, despliegue las clases y, si lo desea, cualquier método estático público en JDeveloper con la configuración del perfil.

Cuando se despliega la base de datos, se utiliza la información proporcionada en Deployment Profile Wizard y dos utilidades de Oracle Database:

- `loadjava` carga la clase Java que contiene los procedimientos almacenados en una base de datos Oracle.
- `publish` genera los encapsuladores específicos de llamada PL/SQL para los métodos estáticos públicos cargados. La publicación permite que se llame a los métodos Java como funciones o procedimientos PL/SQL.

Publicación de Java en PL/SQL

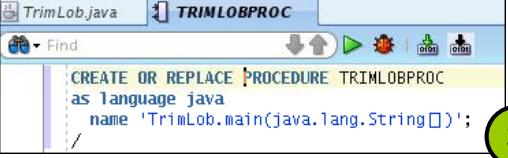


```
public class Trimlob
{
    public static void main (String args [] ) throws SQLException {
        Connection conn=null;
        if (System.getProperty("oracle.jserver.version") != null)
        {
            conn = DriverManager.getConnection("jdbc:default:connection:");
        }
        else
        {
            DriverManager.registerDriver(new oracle.jdbc.OracleDriver());
            conn = DriverManager.getConnection("jdbc:oracle:thin:scott/tiger");
        }
    }
}
```

[oracle@edrsr5pl-orcl ~]\$ cd wkdir
[oracle@edrsr5pl-orcl wkdir]\$ loadjava -u hr/hr Trimlob.java
[oracle@edrsr5pl-orcl wkdir]\$

1

2



```
CREATE OR REPLACE PROCEDURE TRIMLOBPROC
as Language java
name 'Trimlob.main(java.lang.String[])';
/
```

ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Publicación de Java en PL/SQL

En la diapositiva se muestra el código Java y se ilustra cómo publicar el código Java en un procedimiento PL/SQL.

1. Cree y compile una rutina Java.
2. Ejecute el comando mostrado en la diapositiva para cargar la rutina Java Trimlob Java en la base de datos.
3. Publique el método de clase Java mediante la creación de la especificación de llamada PL/SQL TrimlobProc que hace referencia a los métodos de clase Java. En la especificación PL/SQL TrimlobProc, identifique el nombre del método de clase Java y sus parámetros. Esto se denomina “publicar” el método de clase Java.

Para obtener más información sobre la ejecución de rutinas Java en la base de datos, consulte el siguiente OBE: http://www.oracle.com/technology/obe/hol08/11gR1_JDBC_Java_otn.htm

¿Cómo Puedo Obtener Más Información sobre JDeveloper 11g?

Tema	Sitio web
Página del Producto Oracle JDeveloper	http://www.oracle.com/technology/products/jdev/index.html
Tutoriales de Oracle JDeveloper 11g	http://www.oracle.com/technology/obe/obe11jdev/11/index.html
Documentación del Producto Oracle JDeveloper 11g	http://www.oracle.com/technology/documentation/jdev.html
Foro de Discusión de Oracle JDeveloper 11g	http://forums.oracle.com/forums/forum.jspa?forumID=83



Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.



ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Variables de Cursor

- Las variables de cursor son como punteros C o Pascal, que incluyen la ubicación (dirección) de la memoria de un elemento en lugar del elemento en sí.
- En PL/SQL, un puntero se declara como `REF X`, donde `REF` es la abreviatura de `REFERENCE` y `X` indica una clase de objetos.
- Una variable de cursor tiene el tipo de dato `REF CURSOR`.
- Un cursor es estático, pero una variable de cursor es dinámica.
- Las variables de cursor aportan mayor flexibilidad.



Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Variables de Cursor

Las variables de cursor son como punteros C o Pascal, que incluyen la ubicación (dirección) de la memoria de un elemento en lugar del elemento en sí. Por lo tanto, al declarar una variable de cursor se crea un puntero y no un elemento. En PL/SQL, un puntero tiene el tipo de dato `REF X`, donde `REF` es la abreviatura de `REFERENCE` y `X` indica una clase de objetos. Una variable de cursor tiene el tipo de dato `REF CURSOR`.

Al igual que un cursor, una variable de cursor apunta a la fila actual del juego de resultados de una consulta de varias filas. Sin embargo, los cursores difieren de las variables de cursor al igual que las constantes difieren de las variables. Un cursor es estático, pero una variable de cursor es dinámica porque no está ligada a una consulta específica. Puede abrir una variable de cursor para cualquier consulta compatible con el tipo. Esto aporta mayor flexibilidad.

Las variables de cursor están disponibles para todos los clientes PL/SQL. Por ejemplo, puede declarar una variable de cursor en un entorno del host PL/SQL como, por ejemplo, un programa OCI o Pro*C, y transferirla como variable de host de entrada (variable de enlace) a PL/SQL. Además, las herramientas de desarrollo de aplicaciones como Oracle Forms y Oracle Reports, que disponen de un motor PL/SQL, pueden utilizar totalmente variables de cursor en el cliente. Oracle Server también tiene un motor PL/SQL. Puede transferir variables una y otra vez entre una aplicación y el servidor mediante llamadas a procedimientos remotos (RPC).

Uso de Variables de Cursor

- Puede utilizar variables de cursor para transferir juegos de resultados de consultas entre los subprogramas almacenados PL/SQL y distintos clientes.
- PL/SQL puede compartir un puntero al área de trabajo de consulta en la que esté almacenado el juego de resultados.
- Puede transferir el valor de una variable de cursor libremente de un ámbito a otro.
- Puede reducir el tráfico de red con un bloque PL/SQL que abra (o cierre) varias variables de cursor de host en un solo recorrido de ida y vuelta.



Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Uso de Variables de Cursor

Puede utilizar variables de cursor para transferir juegos de resultados de consultas entre los subprogramas almacenados PL/SQL y distintos clientes. Ni PL/SQL ni ninguno de sus clientes es propietario de un juego de resultados; simplemente comparten un puntero al área de trabajo de consulta en la que está almacenado dicho juego. Por ejemplo, un cliente OCI, una aplicación Oracle Forms y Oracle Server pueden hacer referencia a la misma área de trabajo.

Un área de trabajo de consulta permanece accesible mientras una variable de cursor apunte a ella. Por lo tanto, puede transferir el valor de una variable de cursor libremente de un ámbito a otro. Por ejemplo, si transfiere una variable de cursor de host a un bloque PL/SQL embebido en un programa Pro*C, el área de trabajo a la que apunta dicha variable permanece accesible después de que termine el bloque.

Si dispone de un motor PL/SQL en el cliente, las llamadas desde el cliente al servidor no imponen ninguna restricción. Por ejemplo, puede declarar una variable de cursor en el cliente, abrir y recuperar de ella en el servidor y, a continuación, seguir recuperando de ella en el cliente. También puede reducir el tráfico de red con un bloque PL/SQL que abra (o cierre) varias variables de cursor de host en un solo recorrido de ida y vuelta.

Una variable de cursor incluye una referencia al área de trabajo del cursor en el Área Global de Programa (PGA) en lugar de utilizar un nombre estático. Puesto que se utiliza una referencia, obtiene la flexibilidad de una variable.

Definición de Tipos REF CURSOR

Defina un tipo REF CURSOR:

```
Define a REF CURSOR type
TYPE ref_type_name IS REF CURSOR [RETURN return_type];
```

Declare una variable de cursor de dicho tipo:

```
ref_cv ref_type_name;
```

Ejemplo:

```
DECLARE
  TYPE DeptCurTyp IS REF CURSOR RETURN
    departments%ROWTYPE;
  dept_cv DeptCurTyp;
```

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Definición de Tipos REF CURSOR

Para definir un tipo REF CURSOR, debe realizar dos pasos. En primer lugar, defina un tipo REF CURSOR y, a continuación, declare las variables de cursor de dicho tipo. Puede definir los tipos REF CURSOR en cualquier bloque, subprograma o paquete PL/SQL mediante la siguiente sintaxis:

```
TYPE ref_type_name IS REF CURSOR [RETURN return_type];
```

donde:

ref_type_name Es un especificador de tipo utilizado en declaraciones posteriores de variables de cursor.

return_type Representa un registro o una fila de una tabla de base de datos.

En este ejemplo, se especifica un tipo de retorno que representa una fila de la tabla de base de datos DEPARTMENT.

Los tipos REF CURSOR pueden ser fuertes (restrictivos) o débiles (no restrictivos). Como se muestra en el siguiente ejemplo, una definición de un tipo REF CURSOR fuerte especifica un tipo de retorno, pero una definición de un tipo débil no:

```
DECLARE
```

```
  TYPE EmpCurTyp IS REF CURSOR RETURN employees%ROWTYPE; --  
strong
```

```
  TYPE GenericCurTyp IS REF CURSOR; -- weak
```

Definición de Tipos REF CURSOR (continuación)

Los tipos REF CURSOR fuertes son menos proclives a errores porque el compilador PL/SQL permite asociar una variable de cursor de tipo fuerte sólo a consultas compatibles con el tipo. Sin embargo, los tipos REF CURSOR débiles son más flexibles porque el compilador permite asociar una variable de cursor de tipo débil a cualquier consulta.

Declaración de Variables de Cursor

Después de definir un tipo REF CURSOR, puede declarar variables de cursor de dicho tipo en cualquier bloque o subprograma PL/SQL. En el siguiente ejemplo, se declara la variable de cursor DEPT_CV:

```
DECLARE
    TYPE DeptCurTyp IS REF CURSOR RETURN departments%ROWTYPE;
    dept_cv DeptCurTyp; -- declare cursor variable
```

Nota: no puede declarar variables de cursor en un paquete. A diferencia de las variables empaquetadas, las variables de cursor no tienen estados persistentes. Recuerde que, al declarar una variable de cursor, se crea un puntero y no un elemento. Las variables de cursor no se pueden guardar en la base de datos; siguen las reglas normales de ámbitos e instanciación.

En la cláusula RETURN de una definición de un tipo REF CURSOR, puede utilizar %ROWTYPE para especificar un tipo de registro que representa una fila devuelta por una variable de cursor de tipo fuerte (no débil), de la siguiente forma:

```
DECLARE
    TYPE TmpCurTyp IS REF CURSOR RETURN employees%ROWTYPE;
    tmp_cv TmpCurTyp; -- declare cursor variable
    TYPE EmpCurTyp IS REF CURSOR RETURN tmp_cv%ROWTYPE;
    emp_cv EmpCurTyp; -- declare cursor variable
```

De modo similar, puede utilizar %TYPE para proporcionar el tipo de dato de una variable de registro, como se muestra en el siguiente ejemplo:

```
DECLARE
    dept_rec departments%ROWTYPE; -- declare record variable
    TYPE DeptCurTyp IS REF CURSOR RETURN dept_rec%TYPE;
    dept_cv DeptCurTyp; -- declare cursor variable
```

En el ejemplo final, se especifica un tipo RECORD definido por el usuario en la cláusula RETURN:

```
DECLARE
    TYPE EmpRecTyp IS RECORD (
        empno NUMBER(4),
        ename VARCHAR2(10),
        sal    NUMBER(7,2));
    TYPE EmpCurTyp IS REF CURSOR RETURN EmpRecTyp;
    emp_cv EmpCurTyp; -- declare cursor variable
```

Variables de Cursor como Parámetros

Puede declarar las variables de cursor como parámetros formales de funciones y procedimientos. En el siguiente ejemplo, se define REF CURSOR del tipo EmpCurTyp y, a continuación, se declara una variable de cursor de dicho tipo como parámetro formal de un procedimiento:

```
DECLARE  
    TYPE EmpCurTyp IS REF CURSOR RETURN emp%ROWTYPE;  
    PROCEDURE open_emp_cv (emp_cv IN OUT EmpCurTyp) IS ...
```

Uso de las Sentencias OPEN-FOR, FETCH y CLOSE

- La sentencia OPEN-FOR asocia una variable de cursor a una consulta de varias filas, ejecuta la consulta, identifica el juego de resultados y coloca el cursor de forma que apunte a la primera fila del juego de resultados.
- La sentencia FETCH devuelve una fila del juego de resultados de una consulta de varias filas, asigna los valores de los elementos de la lista select a las variables o los campos correspondientes de la cláusula INTO, aumenta el recuento mantenido por %ROWCOUNT y avanza el cursor a la siguiente fila.
- La sentencia CLOSE desactiva una variable de cursor.

ORACLE

Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Uso de las Sentencias OPEN-FOR, FETCH y CLOSE

Se utilizan tres sentencias para procesar una consulta de varias filas dinámica: OPEN-FOR, FETCH y CLOSE. En primer lugar, abra una variable de cursor para una consulta de varias filas. A continuación, recupere las filas del juego de resultados de una en una. Una vez procesadas todas las filas, cierre la variable de cursor.

Apertura de la Variable de Cursor

La sentencia OPEN-FOR asocia una variable de cursor a una consulta de varias filas, ejecuta la consulta, identifica el juego de resultados, coloca el cursor de forma que apunte a la primera fila del juego de resultados y, a continuación, define en cero el recuento de filas procesadas mantenido por %ROWCOUNT. A diferencia de la forma estática de OPEN-FOR, su forma dinámica tiene una cláusula USING opcional. En tiempo de ejecución, los argumentos enlazados en la cláusula USING sustituyen a los correspondientes marcadores de posición en la sentencia SELECT dinámica. La sintaxis es la siguiente:

```
OPEN {cursor_variable | :host_cursor_variable} FOR
dynamic_string
[USING bind_argument [, bind_argument]...];
```

donde CURSOR_VARIABLE es una variable de cursor de tipo débil (sin tipo de retorno), HOST_CURSOR_VARIABLE es una variable de cursor declarada en un entorno del host PL/SQL como un programa OCI y dynamic_string es una expresión de cadena que representa una consulta de varias filas.

Uso de las Sentencias OPEN-FOR, FETCH y CLOSE (continuación)

En el siguiente ejemplo, la sintaxis declara una variable de cursor y, a continuación, la asocia a una sentencia SELECT dinámica que devuelve filas de la tabla employees:

```

DECLARE
  TYPE EmpCurTyp IS REF CURSOR; -- define weak REF CURSOR      type
  emp_cv  EmpCurTyp; -- declare cursor variable
  my_ename VARCHAR2(15);
  my_sal   NUMBER := 1000;
BEGIN
  OPEN emp_cv FOR -- open cursor variable
    'SELECT last_name, salary FROM employees WHERE salary >
     :s'
    USING my_sal;
  ...
END;

```

Los argumentos enlazados de la consulta se evalúan sólo cuando la variable de cursor está abierta. Por lo tanto, para recuperar filas del cursor con distintos valores de enlace, debe volver a abrir la variable de cursor con los argumentos enlazados definidos en sus valores nuevos cada vez.

Recuperación de la Variable de Cursor

La sentencia `FETCH` devuelve una fila del juego de resultados de una consulta de varias filas, asigna los valores de los elementos de la lista select a las variables o los campos correspondientes de la cláusula `INTO`, aumenta el recuento mantenido por `%ROWCOUNT` y avanza el cursor a la siguiente fila. Utilice la siguiente sintaxis:

```

  FETCH {cursor_variable | :host_cursor_variable}
    INTO {define_variable[, define_variable]... | record};

```

Siguiendo con el ejemplo, recupere filas de la variable de cursor `emp_cv` en las variables de definición `MY_ENAME` y `MY_SAL`:

```

LOOP
  FETCH emp_cv INTO my_ename, my_sal; -- fetch next row
  EXIT WHEN emp_cv%NOTFOUND; -- exit loop when last row is
    fetched
  -- process row
END LOOP;

```

Para cada valor de columna devuelto por la consulta asociada a la variable de cursor, debe haber una variable o un campo compatible con el tipo correspondiente en la cláusula `INTO`. Puede utilizar otra cláusula `INTO` en recuperaciones independientes con la misma variable de cursor. Cada recuperación recupera otra fila del mismo juego de resultados. Si intenta recuperar de una variable de cursor cerrada o que no se ha abierto nunca, PL/SQL produce la excepción predefinida `INVALID_CURSOR`.

Uso de las Sentencias OPEN-FOR, FETCH y CLOSE (continuación)

Cierre de la Variable de Cursor

La sentencia CLOSE desactiva una variable de cursor. Después, se anula la definición del juego de resultados asociado. Utilice la siguiente sintaxis:

```
CLOSE {cursor_variable | :host_cursor_variable};
```

En este ejemplo, cuando se procese la última fila, cierre la variable de cursor emp_cv:

```
LOOP  
  FETCH emp_cv INTO my_ename, my_sal;  
  EXIT WHEN emp_cv%NOTFOUND;  
  -- process row  
END LOOP;  
CLOSE emp_cv;  -- close cursor variable
```

Si intenta cerrar una variable de cursor ya cerrada o que no se ha abierto nunca, PL/SQL emite INVALID_CURSOR.

Ejemplo de Recuperación

```
DECLARE
    TYPE EmpCurTyp IS REF CURSOR;
    emp_cv    EmpCurTyp;
    emp_rec   employees%ROWTYPE;
    sql_stmt  VARCHAR2(200);
    my_job    VARCHAR2(10) := 'ST_CLERK';
BEGIN
    sql_stmt := 'SELECT * FROM employees
                 WHERE job_id = :j';
    OPEN emp_cv FOR sql_stmt USING my_job;
    LOOP
        FETCH emp_cv INTO emp_rec;
        EXIT WHEN emp_cv%NOTFOUND;
        -- process record
    END LOOP;
    CLOSE emp_cv;
END;
/
```



Copyright © 2010, Oracle y/o sus filiales. Todos los derechos reservados.

Ejemplo de Recuperación

En el ejemplo de la diapositiva se muestra que puede recuperar filas del juego de resultados de una consulta de varias filas dinámicas en un registro. En primer lugar, debe definir un tipo REF CURSOR, EmpCurTyp. A continuación, defina una variable de cursor emp_cv, del tipo EmpcurTyp. En la sección ejecutable del bloque PL/SQL, la sentencia OPEN-FOR asocia la variable de cursor emp_cv a la consulta de varias filas, sql_stmt. La sentencia FETCH devuelve una fila del juego de resultados de una consulta de varias filas y asigna los valores de los elementos de la lista select a EMP_REC en la cláusula INTO. Una vez procesada la última fila, cierre la variable de cursor emp_cv.

Apéndice AP

Prácticas y Soluciones

Tabla de Contenido

Visión General	3
Prácticas Adicionales y Soluciones: Lecciones 1 y 2	4
Práctica 1: Evaluación de Declaraciones	4
Práctica 2: Evaluación de Expresiones	4
Solución 1: Evaluación de Declaraciones	5
Solución 2: Evaluación de Expresiones	5
Práctica Adicional y Solución: Lección 3.....	6
Práctica 3: Evaluación de Sentencias Ejecutables	6
Solución 3: Evaluación de Sentencias Ejecutables	7
Prácticas Adicionales y Soluciones de la Lección 4	8
Práctica 4-1: Interacción con Oracle Server	8
Práctica 4-2: Interacción con Oracle Server	8
Solución 4-1: Interacción con Oracle Server	9
Solución 4-2: Interacción con Oracle Server	10
Prácticas Adicionales y Soluciones de la Lección 5	12
Práctica 5-1: Escritura de las Estructuras de Control.....	12
Práctica 5-2: Escritura de las Estructuras de Control.....	12
Solución 5-1: Escritura de las Estructuras de Control	13
Solución 5-2: Escritura de las Estructuras de Control	14
Prácticas Adicionales y Soluciones de las Lecciones 6 y 7	15
Práctica 6/7-1: Recuperación de Datos con un Cursor Explícito.....	15
Práctica 6/7-2: Uso de Matrices Asociativas y Cursores Explícitos	15
Solución 6/7-1: Recuperación de Datos con un Cursor Explícito	16
Solución 6/7-2: Uso de Matrices Asociativas y Cursores Explícitos	17
Práctica Adicional y Solución de la Lección 8	19
Práctica 8-1: Manejo de Excepciones	19
Solución 8-1: Manejo de Excepciones.....	20

Visión General

Estas prácticas adicionales se proporcionan como complemento del curso *Oracle Database: Conceptos Fundamentales de PL/SQL*. En estas prácticas se aplican los conceptos que ha aprendido a lo largo del curso.

Estas prácticas adicionales proporcionan prácticas complementarias en declaración de variables, escritura de sentencias ejecutables, interacción con Oracle Server, escritura de estructuras de control y trabajo con tipos de dato compuestos, cursores y manejo de excepciones. Las tablas utilizadas en esta parte de las prácticas adicionales son employees, jobs, job_history y departments.

Prácticas Adicionales y Soluciones: Lecciones 1 y 2

Estos ejercicios que se responden en papel se utilizan como práctica adicional para la declaración de variables y la escritura de sentencias ejecutables.

Práctica 1: Evaluación de Declaraciones

Evalué cada una de las siguientes declaraciones. Determine cuál de ellas no es válida y explique el porqué.

1. DECLARE
name,dept VARCHAR2 (14);
2. DECLARE
test NUMBER (5);
3. DECLARE
MAXSALARY NUMBER (7, 2) = 5000;
4. DECLARE
JOINDATE BOOLEAN := SYSDATE;

Práctica 2: Evaluación de Expresiones

En cada una de las siguientes asignaciones, determine el tipo de dato de la expresión resultante.

1. email := firstname || to_char(empno);
2. confirm := to_date('20-JAN-1999', 'DD-MON-YYYY');
3. sal := (1000*12) + 500
4. test := FALSE;
5. temp := temp1 < (temp2/ 3);
6. var := sysdate;

Solución 1: Evaluación de Declaraciones

Evalúe cada una de las siguientes declaraciones. Determine cuál de ellas no es válida y explique el porqué.

1. DECLARE
name,dept VARCHAR2 (14) ;

No es válida porque sólo se permite un identificador por declaración.

2. DECLARE
test NUMBER (5) ;

Es válida.

3. DECLARE
MAXSALARY NUMBER (7, 2) = 5000;

No es válida porque el operador de asignación no es correcto. Debe ser :=.

4. DECLARE
JOINDATE BOOLEAN := SYSDATE;

No es válida porque no hay coincidencia en los tipos de dato. Un tipo de dato booleano no se puede asignar a un valor de fecha. El tipo de dato debe ser date.

Solución 2: Evaluación de Expresiones

En cada una de las siguientes asignaciones, determine el tipo de dato de la expresión resultante.

1. email := firstname || to_char(empno);

Cadena de caracteres

2. confirm := to_date('20-JAN-1999', 'DD-MON-YYYY');

Fecha

3. sal := (1000*12) + 500

Número

4. test := FALSE;

Booleano

5. temp := temp1 < (temp2/ 3);

Booleano

6. var := sysdate;

Fecha

Práctica Adicional y Solución: Lección 3

Práctica 3: Evaluación de Sentencias Ejecutables

En este ejercicio que se responde en papel, evaluará el bloque PL/SQL y responderá a las preguntas para determinar el tipo de dato y valor de cada variable, según las reglas de los ámbitos.

```
DECLARE
    v_custid      NUMBER(4) := 1600;
    v_custname VARCHAR2(300) := 'Women Sports Club';
    v_new_custid  NUMBER(3) := 500;
BEGIN
    DECLARE
        v_custid      NUMBER(4) := 0;
        v_custname   VARCHAR2(300) := 'Shape up Sports Club';
        v_new_custid NUMBER(3) := 300;
        v_new_custname VARCHAR2(300) := 'Jansports Club';
    BEGIN
        v_custid := v_new_custid;
        v_custname := v_custname || ' ' || v_new_custname;
    → END;
    v_custid := (v_custid *12) / 10;
    → END;
```

Evalúe el bloque PL/SQL anterior y determine tanto el *valor* como el *tipo de dato* de cada una de las siguientes variables, según las reglas de los ámbitos:

1. `v_custid` en la posición 1:
2. `v_custname` en la posición 1:
3. `v_new_custid` en la posición 1:
4. `v_new_custname` en la posición 1:
5. `v_custid` en la posición 2:
6. `v_custname` en la posición 2:

Solución 3: Evaluación de Sentencias Ejecutables

Evalúe el siguiente bloque PL/SQL. A continuación, responda a las siguientes preguntas para determinar tanto el tipo de dato como el valor de cada una de las siguientes variables, según las reglas de los ámbitos.

```

DECLARE
    v_custid      NUMBER(4) := 1600;
    v_custname    VARCHAR2(300) := 'Women Sports Club';
    v_new_custid  NUMBER(3) := 500;
BEGIN
    DECLARE
        v_custid      NUMBER(4) := 0;
        v_custname    VARCHAR2(300) := 'Shape up Sports Club';
        v_new_custid  NUMBER(3) := 300;
        v_new_custname VARCHAR2(300) := 'Jansports Club';
    BEGIN
        v_custid := v_new_custid;
        v_custname := v_custname || ' ' || v_new_custname;
    1 →
        END;
        v_custid := (v_custid *12) / 10;
    2 →
        END;
    
```

Evalúe el bloque PL/SQL anterior y determine tanto el *valor* como el *tipo de dato* de cada una de las siguientes variables, según las reglas de los ámbitos:

1. v_custid en la posición 1:

300 y el tipo de dato es NUMBER.

2. v_custname en la posición 1:

Shape up Sports Club Jansports Club y el tipo de dato es VARCHAR2.

3. v_new_custid en la posición 1:

500 y el tipo de dato es NUMBER (o INTEGER).

4. v_new_custname en la posición 1:

Jansports Club y el tipo de dato es VARCHAR2.

5. v_custid en la posición 2:

1920 y el tipo de dato es NUMBER.

6. v_custname en la posición 2:

Women Sports Club y el tipo de dato es VARCHAR2.

Prácticas Adicionales y Soluciones de la Lección 4

Práctica 4-1: Interacción con Oracle Server

Para este ejercicio, se necesita una tabla temporal para almacenar los resultados.

- Ejecute el script lab_ap_04.sql que crea la tabla descrita aquí:

Column Name	NUM_STORE	CHAR_STORE	DATE_STORE
Key Type			
Nulls/Unique			
FK Table			
FK Column			
Data Type	Number	VARCHAR2	Date
Length	7,2	35	

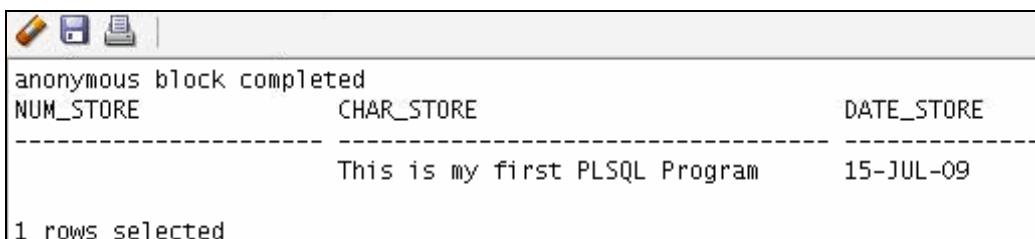
- Escriba un bloque PL/SQL que realice las siguientes acciones:

- Declare dos variables y asigne los siguientes valores a estas variables:

Variable	Tipo de Dato	Contenido
V_MESSAGE	VARCHAR2 (35)	This is my first PL/SQL program
V_DATE_WRITTEN	DATE	Fecha actual

- Almacene los valores de estas variables en las columnas adecuadas de la tabla TEMP.

- Verifique los resultados consultando la tabla TEMP. La salida resultante debe ser como la siguiente:



The screenshot shows the output of an anonymous block. At the top, there are icons for a pencil, a floppy disk, and a printer. Below that, the text "anonymous block completed" is displayed. The results are presented in a table with three columns: NUM_STORE, CHAR_STORE, and DATE_STORE. A dashed line separates the header from the data. The data row contains the value "This is my first PLSQL Program" for CHAR_STORE and "15-JUL-09" for DATE_STORE. At the bottom, the message "1 rows selected" is shown.

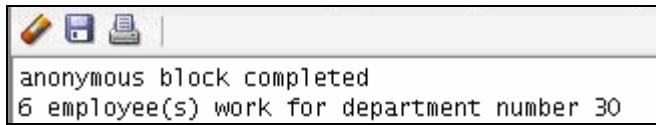
NUM_STORE	CHAR_STORE	DATE_STORE
	This is my first PLSQL Program	15-JUL-09

1 rows selected

Práctica 4-2: Interacción con Oracle Server

En este ejercicio, utilizará datos de la tabla employees.

- Escriba un bloque PL/SQL para determinar el número de empleados que trabajan en un determinado departamento. El bloque PL/SQL debería:
 - Utilizar una variable de sustitución para almacenar un número de departamento
 - Imprimir el número de personas que trabajan en el departamento especificado
- Cuando se ejecute el bloque, aparecerá una ventana de variable de sustitución. Introduzca un número de departamento válido y haga clic en OK. La salida resultante debe tener un aspecto similar al siguiente:



Solución 4-1: Interacción con Oracle Server

Para este ejercicio, se necesita una tabla temporal para almacenar los resultados.

- Ejecute el script lab_ap_04.sql que crea la tabla descrita aquí:

Column Name	NUM_STORE	CHAR_STORE	DATE_STORE
Key Type			
Nulls/Unique			
FK Table			
FK Column			
Data Type	Number	VARCHAR2	Date
Length	7 , 2	35	

- Escriba un bloque PL/SQL que realice las siguientes acciones:

- Declare dos variables y asigne los siguientes valores a estas variables:

Variable	Tipo de Dato	Contenido
V_MESSAGE	VARCHAR2 (35)	This is my first PL/SQL program
V_DATE_WRITTEN	DATE	Fecha actual

- Almacene los valores de estas variables en las columnas adecuadas de la tabla TEMP.

```

DECLARE
    V_MESSAGE VARCHAR2(35);
    V_DATE_WRITTEN DATE;
BEGIN
    V_MESSAGE := 'This is my first PLSQL Program';
    V_DATE_WRITTEN := SYSDATE;
    INSERT INTO temp(CHAR_STORE,DATE_STORE)
        VALUES (V_MESSAGE,V_DATE_WRITTEN);
END;
/

```

- Verifique los resultados consultando la tabla TEMP. La salida resultante debe tener un aspecto similar al siguiente:

```
SELECT * FROM TEMP;
```

```

anonymous block completed
NUM_STORE          CHAR_STORE          DATE_STORE
-----
This is my first PLSQL Program      15-JUL-09
                                        

1 rows selected

```

Solución 4-2: Interacción con Oracle Server

En este ejercicio, utilizará datos de la tabla employees.

- Escriba un bloque PL/SQL para determinar el número de empleados que trabajan en un determinado departamento. El bloque PL/SQL debería:

- Utilizar una variable de sustitución para almacenar un número de departamento
- Imprimir el número de personas que trabajan en el departamento especificado

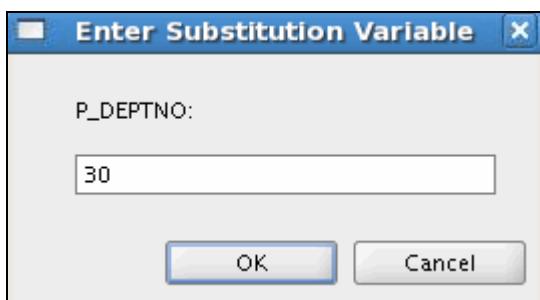
```

SET SERVEROUTPUT ON;

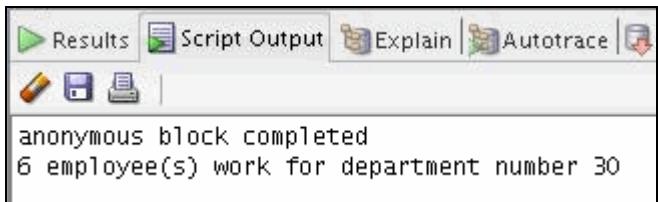
DECLARE
    V_HOWMANY NUMBER(3);
    V_DEPTNO DEPARTMENTS.department_id%TYPE := &P_DEPTNO;
BEGIN
    SELECT COUNT(*) INTO V_HOWMANY FROM employees
    WHERE department_id = V_DEPTNO;
    DBMS_OUTPUT.PUT_LINE (V_HOWMANY || ' employee(s)'
                           work for department number ' || V_DEPTNO);
END;
/

```

- Cuando se ejecute el bloque, aparecerá una ventana de variable de sustitución. Introduzca un número de departamento válido y haga clic en OK.



La salida resultante debe tener un aspecto similar al siguiente:



The screenshot shows a window from Oracle SQL Developer. At the top, there are tabs: 'Results' (which is selected), 'Script Output', 'Explain', and 'Autotrace'. Below the tabs are three small icons: a pencil, a floppy disk, and a printer. The main area of the window contains the following text:
anonymous block completed
6 employee(s) work for department number 30

Prácticas Adicionales y Soluciones de la Lección 5

En estas prácticas, utilizará estructuras de control para dirigir la lógica del flujo de programa.

Práctica 5-1: Escritura de las Estructuras de Control

1. Escriba un bloque PL/SQL que acepte la entrada de un año y compruebe si es un año bisiesto. **Indicación:** el año debe ser divisible por 4 pero no por 100 o debe ser divisible por 400.
2. Compruebe la solución con la siguiente tabla. Por ejemplo, si el año introducido es 1990, el resultado debe ser “1990 is not a leap year”.

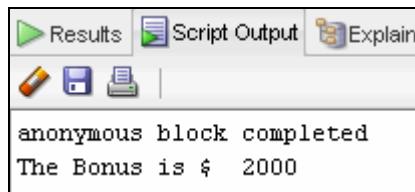
1990	Not a leap year
2000	Leap year
1996	Leap year
1886	Not a leap year
1992	Leap year
1824	Leap year

Práctica 5-2: Escritura de las Estructuras de Control

1. Escriba un bloque PL/SQL para almacenar el salario mensual de un empleado en una variable de sustitución. El bloque PL/SQL debería:
 - Calcular el salario anual como salario * 12.
 - Calcular el incentivo como se indica en la siguiente tabla:

Salario anual	Incentivo
>= 20,000	2,000
19,999–10,000	1,000
<= 9,999	500

- Muestre la cantidad del incentivo en la ventana Script Output con el siguiente formato:



2. Pruebe el bloque PL/SQL para los siguientes casos:

Salario mensual	Incentivo
3000	2000
1200	1000
800	500

Solución 5-1: Escritura de las Estructuras de Control

- Escriba un bloque PL/SQL que acepte la entrada de un año y compruebe si es un año bisiesto. **Indicación:** el año debe ser divisible por 4 pero no por 100 o debe ser divisible por 400.

```

SET SERVEROUTPUT ON;
DECLARE
    v_YEAR NUMBER(4) := &P_YEAR;
    v_REMAINDER1 NUMBER(5,2);
    v_REMAINDER2 NUMBER(5,2);
    v_REMAINDER3 NUMBER(5,2);
BEGIN
    v_REMAINDER1 := MOD(v_YEAR,4);
    v_REMAINDER2 := MOD(v_YEAR,100);
    v_REMAINDER3 := MOD(v_YEAR,400);
    IF ((v_REMAINDER1 = 0 AND v_REMAINDER2 <> 0 ) OR
        v_REMAINDER3 = 0) THEN
        DBMS_OUTPUT.PUT_LINE(v_YEAR || ' is a leap year');
    ELSE
        DBMS_OUTPUT.PUT_LINE(v_YEAR || ' is not a leap
year');
    END IF;
END;
/

```

- Compruebe la solución con la siguiente tabla. Por ejemplo, si el año introducido es 1990, el resultado debe ser “1990 is not a leap year”.

1990	Not a leap year
2000	Leap year
1996	Leap year
1886	Not a leap year
1992	Leap year
1824	Leap year

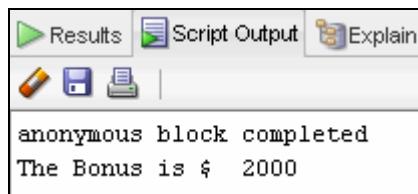
Solución 5-2: Escritura de las Estructuras de Control

1. Escriba un bloque PL/SQL para almacenar el salario mensual de un empleado en una variable de sustitución. El bloque PL/SQL debería:

- Calcular el salario anual como salario * 12.
- Calcular el incentivo como se indica en la siguiente tabla:

Salario anual	Incentivo
>= 20,000	2,000
19,999–10,000	1,000
<= 9,999	500

- Muestre la cantidad del incentivo en la ventana Script Output con el siguiente formato:



```

SET SERVEROUTPUT ON;
DECLARE
    V_SAL          NUMBER(7,2) := &M_SALARY;
    V_BONUS        NUMBER(7,2);
    V_ANN_SALARY   NUMBER(15,2);
BEGIN
    V_ANN_SALARY := V_SAL * 12;
    IF V_ANN_SALARY >= 20000 THEN
        V_BONUS := 2000;
    ELSIF V_ANN_SALARY <= 19999 AND V_ANN_SALARY >=10000 THEN
        V_BONUS := 1000;
    ELSE
        V_BONUS := 500;
    END IF;
    DBMS_OUTPUT.PUT_LINE ('The Bonus is $ ' ||
                          TO_CHAR(V_BONUS));
END;
/

```

2. Pruebe el bloque PL/SQL para los siguientes casos:

Salario mensual	Incentivo
3000	2000
1200	1000
800	500

Prácticas Adicionales y Soluciones de las Lecciones 6 y 7

En los siguientes ejercicios, practicará con el uso de matrices asociativas (este tema se trata en la lección 6) y cursos explícitos (este tema se trata en la lección 7). En el primer ejercicio, definirá y utilizará un cursor explícito para recuperar datos. En el segundo ejercicio, combinará el uso de matrices asociativas con un cursor explícito para generar la salida de datos que cumplan determinados criterios.

Práctica 6/7-1: Recuperación de Datos con un Cursor Explícito

En esta práctica, creará un bloque PL/SQL que realice las siguientes acciones:

1. Declare un cursor llamado EMP_CUR para seleccionar el apellido, el salario y la fecha de contratación del empleado de la tabla EMPLOYEES
2. Procese todas las filas del cursor y, si el salario es mayor que 15.000 y la fecha de contratación es posterior al 01-FEB-1988, muestre el nombre, el salario y la fecha de contratación del empleado con el formato que se muestra en la siguiente salida de ejemplo:

```

Results | Script Output | Explain | Autotrace | DBMS Output | OW
anonymous block completed
Kochhar earns 17000 and joined the organization on 21-SEP-89
De Haan earns 17000 and joined the organization on 13-JAN-93
  
```

Práctica 6/7-2: Uso de Matrices Asociativas y Cursosores Explícitos

En esta práctica, creará un bloque PL/SQL para recuperar y generar la salida del apellido e identificador de departamento de cada empleado de la tabla EMPLOYEES para aquellos empleados cuyo EMPLOYEE_ID sea menor que 115.

En el bloque PL/SQL, utilice una estrategia de bucle FOR de cursor en lugar de los métodos de cursor OPEN / FETCH / CLOSE utilizados en la práctica anterior.

1. En la sección de declaraciones:
 - Cree dos matrices asociativas. La columna de clave única de ambas matrices debe ser de tipo de dato BINARY_INTEGER. Una matriz contiene el apellido del empleado y la otra el identificador de departamento.
 - Declare un cursor que seleccione el apellido e identificador de departamento de los empleados con un identificador inferior a 115.
 - Declare la variable de contador adecuada que se utilizará en la sección ejecutable.
2. En la sección ejecutable, utilice un bucle FOR de cursor (descrito en la lección 7) para acceder a los valores del cursor, asígnelos a las matrices asociativas adecuadas y genere una salida de dichos valores desde las matrices. La salida correcta debe devolver 15 filas, con el siguiente formato:

```

Results Script Output Explain Autotrace DBMS
| | |
anonymous block completed
Employee: King is in department number: 90
Employee: Kochhar is in department number: 90
Employee: De Haan is in department number: 90

```

Solución 6/7-1: Recuperación de Datos con un Cursor Explícito

En esta práctica, creará un bloque PL/SQL que realice las siguientes acciones:

- Declare un cursor llamado EMP_CUR para seleccionar el apellido, el salario y la fecha de contratación del empleado de la tabla EMPLOYEES

```

SET SERVEROUTPUT ON;
DECLARE
    CURSOR C_EMP_CUR IS
        SELECT last_name,salary,hire_date FROM EMPLOYEES;
    V_ENAME VARCHAR2(25);
    V_SAL    NUMBER(7,2);
    V_HIREDATE DATE;

```

- Procese todas las filas del cursor y, si el salario es mayor que 15.000 y la fecha de contratación es posterior al 01-FEB-1988, muestre el nombre, el salario y la fecha de contratación del empleado con el formato que se muestra en la siguiente salida de ejemplo:

```

BEGIN
    OPEN C_EMP_CUR;
    FETCH C_EMP_CUR INTO V_ENAME,V_SAL,V_HIREDATE;
    WHILE C_EMP_CUR%FOUND
    LOOP
        IF V_SAL > 15000 AND V_HIREDATE >=
            TO_DATE('01-FEB-1988','DD-MON-YYYY') THEN
            DBMS_OUTPUT.PUT_LINE (V_ENAME || ' earns '
                || TO_CHAR(V_SAL) || ' and joined the organization on '
                || TO_DATE(V_HIREDATE,'DD-Mon-YYYY'));
        END IF;
        FETCH C_EMP_CUR INTO V_ENAME,V_SAL,V_HIREDATE;
    END LOOP;
    CLOSE C_EMP_CUR;
END;
/

```

```

Results Script Output Explain Autotrace DBMS Output
| | |
anonymous block completed
Kochhar earns 17000 and joined the organization on 21-SEP-89
De Haan earns 17000 and joined the organization on 13-JAN-93

```

Solución 6/7-2: Uso de Matrices Asociativas y Cursos Explícitos

En esta práctica, creará un bloque PL/SQL para recuperar y generar la salida del apellido e identificador de departamento de cada empleado de la tabla EMPLOYEES para aquellos empleados cuyo EMPLOYEE_ID sea menor que 115.

En el bloque PL/SQL, utilice una estrategia de bucle FOR de cursor en lugar de los métodos de cursor OPEN / FETCH / CLOSE utilizados en la práctica anterior.

1. En la sección de declaraciones:

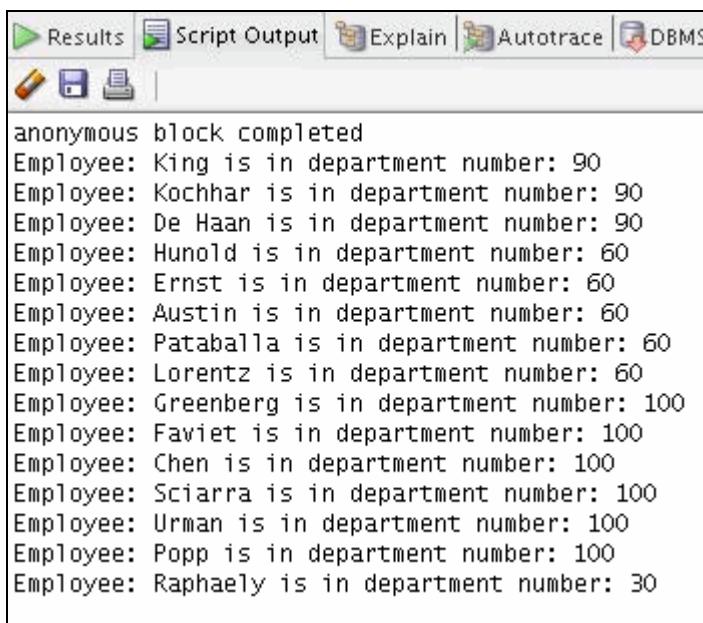
- Cree dos matrices asociativas. La columna de clave única de ambas matrices debe ser de tipo de dato BINARY_INTEGER. Una matriz contiene el apellido del empleado y la otra el identificador de departamento.
- Declare la variable de contador que se utilizará en la sección ejecutable.
- Declare un cursor que seleccione el apellido e identificador de departamento de los empleados con un identificador inferior a 115.

```
SET SERVEROUTPUT ON;
DECLARE
  TYPE Table_Ename IS table of employees.last_name%TYPE
    ..... INDEX BY BINARY_INTEGER;
  TYPE Table_dept IS table of employees.department_id%TYPE
    ..... INDEX BY BINARY_INTEGER;
  Tename Table_Ename;
  Tdept Table_dept;
  i BINARY_INTEGER :=0;
  CURSOR Namedept IS SELECT last_name,department_id
    FROM employees WHERE employee_id < 115;
```

2. En la sección ejecutable, utilice un bucle FOR de cursor (descrito en la lección 7) para acceder a los valores del cursor, asígnelos a las matrices asociativas adecuadas y genere una salida de dichos valores desde las matrices.

```
BEGIN
  FOR emprec in Namedept
  LOOP
    ..... i := i +1;
    Tename(i) := emprec.last_name;
    Tdept(i) := emprec.department_id;
    DBMS_OUTPUT.PUT_LINE ('Employee: ' || Tename(i) ||
      ' is in department number: ' || Tdept(i));
  END LOOP;
END;
/
```

La salida correcta debe devolver 15 filas, que sería similar a la siguiente:



The screenshot shows a software interface for executing SQL code. At the top, there is a menu bar with tabs: 'Results' (which is selected), 'Script Output', 'Explain', 'Autotrace', and 'DBMS'. Below the menu bar are three icons: a pencil for edit, a floppy disk for save, and a printer for print. The main area displays the output of an anonymous block. The output consists of 15 lines, each starting with 'Employee:' followed by the employee's name and their department number. The output is as follows:

```
anonymous block completed
Employee: King is in department number: 90
Employee: Kochhar is in department number: 90
Employee: De Haan is in department number: 90
Employee: Hunold is in department number: 60
Employee: Ernst is in department number: 60
Employee: Austin is in department number: 60
Employee: Pataballa is in department number: 60
Employee: Lorentz is in department number: 60
Employee: Greenberg is in department number: 100
Employee: Faviet is in department number: 100
Employee: Chen is in department number: 100
Employee: Sciarra is in department number: 100
Employee: Urman is in department number: 100
Employee: Popp is in department number: 100
Employee: Raphaely is in department number: 30
```

Práctica Adicional y Solución de la Lección 8

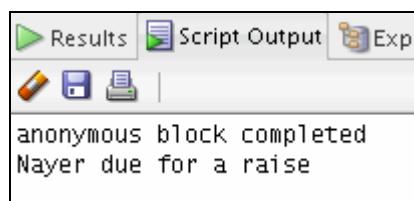
Práctica 8-1: Manejo de Excepciones

Para este ejercicio, primero debe crear una tabla para almacenar algunos resultados. Ejecute el script `lab_ap_08.sql` que crea la tabla. El script es parecido al siguiente:

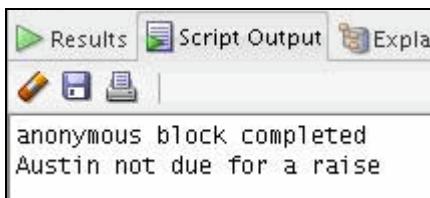
```
CREATE TABLE analysis
  (ename Varchar2(20), years Number(2), sal Number(8,2)
  );
```

En esta práctica, escribirá un bloque PL/SQL que maneje una excepción, de la siguiente forma:

1. Declare variables para el apellido, el salario y la fecha de contratación del empleado. Utilice una variable de sustitución para el apellido del empleado. A continuación, consulte la tabla `employees` para `last_name`, `salary` y `hire_date` del empleado especificado.
2. Si el empleado lleva en la organización más de cinco años y su salario es inferior a 3.500, emita una excepción. En el manejador de excepciones, realice lo siguiente:
 - Genere la salida de la siguiente información: el apellido del empleado y el mensaje “due for a raise”, similar al siguiente:



- Inserte el apellido, los años de servicio y el salario en la tabla `analysis`.
- 3. Si no hay ninguna excepción, genere la salida del apellido del empleado y el mensaje “not due for a raise”, similar al siguiente:



4. Verifique los resultados consultando la tabla `analysis`. Utilice los siguientes casos de prueba para probar el bloque PL/SQL.

LAST_NAME	MENSAJE
Austin	Not due for a raise
Nayer	Due for a raise
Fripp	Not due for a raise
Khoo	Due for a raise

Solución 8-1: Manejo de Excepciones

Para este ejercicio, primero debe crear una tabla para almacenar algunos resultados. Ejecute el script lab_ap_08.sql que crea la tabla. El script es parecido al siguiente:

```
CREATE TABLE analysis
    (ename Varchar2(20), years Number(2), sal Number(8,2)
 );
```

En esta práctica, escribirá un bloque PL/SQL que maneje una excepción, de la siguiente forma:

1. Declare variables para el apellido, el salario y la fecha de contratación del empleado. Utilice una variable de sustitución para el apellido del empleado. A continuación, consulte la tabla employees para last_name, salary y hire_date del empleado especificado.
2. Si el empleado lleva en la organización más de cinco años y su salario es inferior a 3.500, emita una excepción. En el manejador de excepciones, realice lo siguiente:
 - Genere la salida de la siguiente información: el apellido del empleado y el mensaje “due for a raise”.
 - Inserte el nombre, los años de servicio y el salario en la tabla analysis.
3. Si no hay ninguna excepción, genere la salida del apellido del empleado y el mensaje “not due for a raise”.

```
SET SERVEROUTPUT ON;
DECLARE
    E_DUE_FOR_RAISE EXCEPTION;
    V_HIREDATE EMPLOYEES.HIRE_DATE%TYPE;
    V_ENAME EMPLOYEES.LAST_NAME%TYPE := INITCAP(' & B_ENAME');
    V_SAL EMPLOYEES.SALARY%TYPE;
    V_YEARS NUMBER(2);
BEGIN
    SELECT LAST_NAME, SALARY, HIRE_DATE
    INTO V_ENAME, V_SAL, V_HIREDATE
    FROM employees WHERE last_name = V_ENAME;
    V_YEARS := MONTHS_BETWEEN(SYSDATE, V_HIREDATE)/12;
    IF V_SAL < 3500 AND V_YEARS > 5 THEN
        RAISE E_DUE_FOR_RAISE;
    ELSE
        DBMS_OUTPUT.PUT_LINE (' not due for a raise');
    END IF;
EXCEPTION
    WHEN E_DUE_FOR_RAISE THEN
        BEGIN
            DBMS_OUTPUT.PUT_LINE (V_NAME || ' due for a raise');
            INSERT INTO ANALYSIS(ENAME, YEARS, SAL)
            VALUES (V_ENAME, V_YEARS, V_SAL);
        END;
END;
/
```

4. Verifique los resultados consultando la tabla `analysis`. Utilice los siguientes casos de prueba para probar el bloque PL/SQL.

LAST_NAME	MENSAJE
Austin	Not due for a raise
Nayer	Due for a raise
Fripp	Not due for a raise
Khoo	Due for a raise

```
SELECT * FROM analysis;
```

