

Visión General de PL/SQL

ORACLE®

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Objetivos del Curso

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

- **Describir el propósito de PL/SQL**
- **Describir el uso de PL/SQL para el desarrollador, así como para el DBA**
- **Explicar las ventajas de PL/SQL**
- **Crear, ejecutar y mantener procedimientos, funciones, paquetes y disparadores de bases de datos**
- **Gestionar disparadores y subprogramas PL/SQL**
- **Describir los paquetes de Oracle**
- **Manipular objetos de gran tamaño (LOB)**

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Finalidad de la Lección

En este curso conocerá las características y las ventajas del lenguaje PL/SQL. Aprenderá a acceder a la base de datos utilizando el lenguaje PL/SQL.

Se pueden desarrollar aplicaciones modulares con procedimientos de bases de datos utilizando objetos de bases de datos, como los siguientes:

- Procedimientos y funciones
- Paquetes
- Disparadores de bases de datos

Las aplicaciones modulares mejoran:

- La funcionalidad
- La seguridad
- El rendimiento global

Acerca de PL/SQL

- **PL/SQL es la extensión procedural de SQL con características de diseño de los lenguajes de programación.**
- **Las unidades de código procedurales incluyen sentencias de manipulación de datos y de consulta de SQL.**

ORACLE

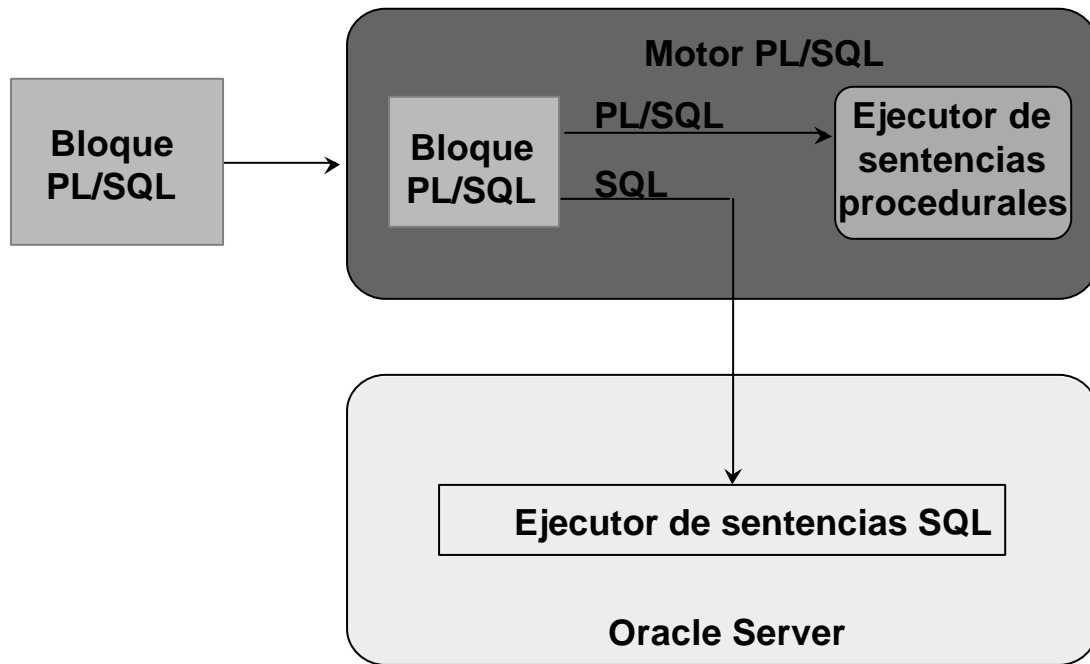
Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Acerca de PL/SQL

El lenguaje Procedural Language/SQL (PL/SQL) es la extensión del lenguaje procedural de SQL de Oracle Corporation, el lenguaje estándar de acceso a datos para bases de datos relacionales. PL/SQL incluye modernas funciones de ingeniería de software, como la encapsulación de datos, el manejo de excepciones, la ocultación de información y la orientación a objetos, y permite que Oracle Server y su conjunto de herramientas dispongan de lo último en programación.

PL/SQL incorpora muchas de las funciones avanzadas de los lenguajes de programación que se diseñaron durante las décadas de 1970 y 1980. Permite incluir las sentencias de manipulación de datos y de consulta de SQL en unidades de código estructuradas por bloques y procedurales, lo que hace que PL/SQL sea un lenguaje de procesamiento de transacciones muy potente. En PL/SQL, puede utilizar sentencias SQL para obtener datos de Oracle y sentencias de control PL/SQL para procesar los datos.

Entorno PL/SQL



ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Entorno PL/SQL

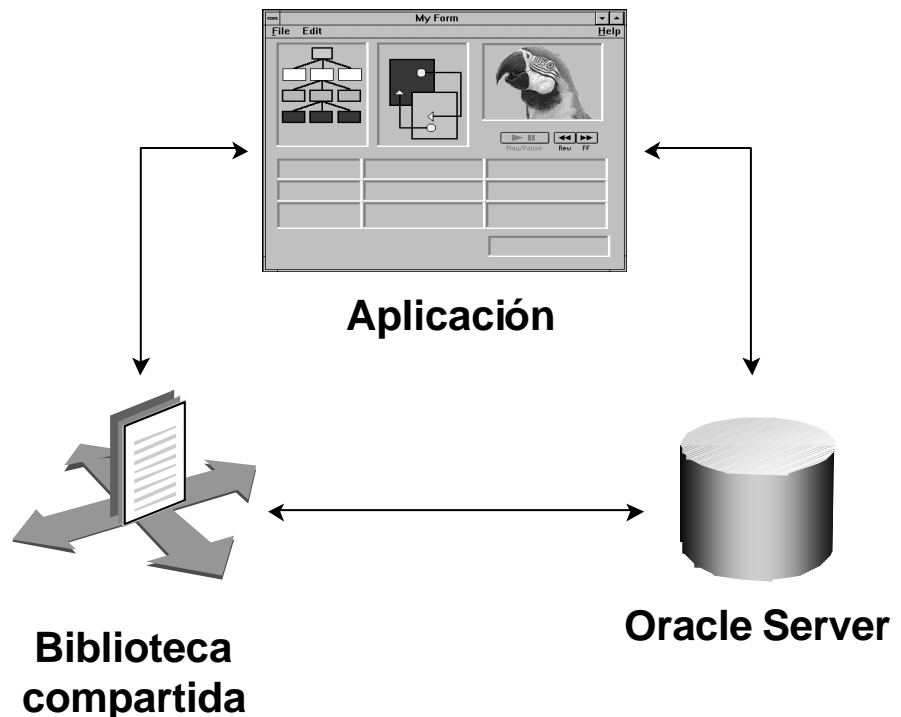
PL/SQL no es un producto de Oracle propiamente dicho. Se trata de una tecnología que emplea Oracle Server y ciertas herramientas de Oracle. Los bloques PL/SQL se transfieren a un motor PL/SQL que los procesa y que puede residir en la herramienta o en Oracle Server. El motor que se utiliza depende del lugar desde el cual se llama al bloque PL/SQL.

Cuando se ejecutan bloques PL/SQL desde un precompilador de Oracle como, por ejemplo, programas Pro*C o Pro*Cobol, rutinas de usuario, iSQL*Plus o Server Manager, el motor PL/SQL de Oracle Server los procesa. Se encarga de separar las sentencias SQL y las envía individualmente al ejecutor de sentencias SQL.

Sólo es necesaria una transferencia para enviar el bloque desde la aplicación a Oracle Server, mejorando así el rendimiento, especialmente en las redes cliente-servidor. Además, el código PL/SQL se puede almacenar en Oracle Server como subprogramas a los que puede hacer referencia cualquier número de aplicaciones conectadas a la base de datos.

Ventajas de PL/SQL

Integración



ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Ventajas de PL/SQL

Integración

PL/SQL desempeña un papel crucial tanto en Oracle Server (mediante procedimientos almacenados, funciones almacenadas, disparadores de bases de datos y paquetes), como en las herramientas de desarrollo de Oracle (a través de disparadores de componentes de Oracle Developer).

Las aplicaciones Oracle Forms Developer, Oracle Reports Developer y Oracle Graphics Developer utilizan bibliotecas compartidas en las que se guarda el código (procedimientos y funciones) y a las que se puede acceder de manera local o remota.

En PL/SQL también se pueden utilizar tipos de datos SQL. Combinados con el acceso directo que proporciona SQL, estos tipos de datos compartidos integran PL/SQL con el diccionario de datos de Oracle Server. PL/SQL es un puente entre un cómodo acceso a la tecnología de bases de datos y la necesidad de capacidades de programación procedurales.

PL/SQL en las Herramientas de Oracle

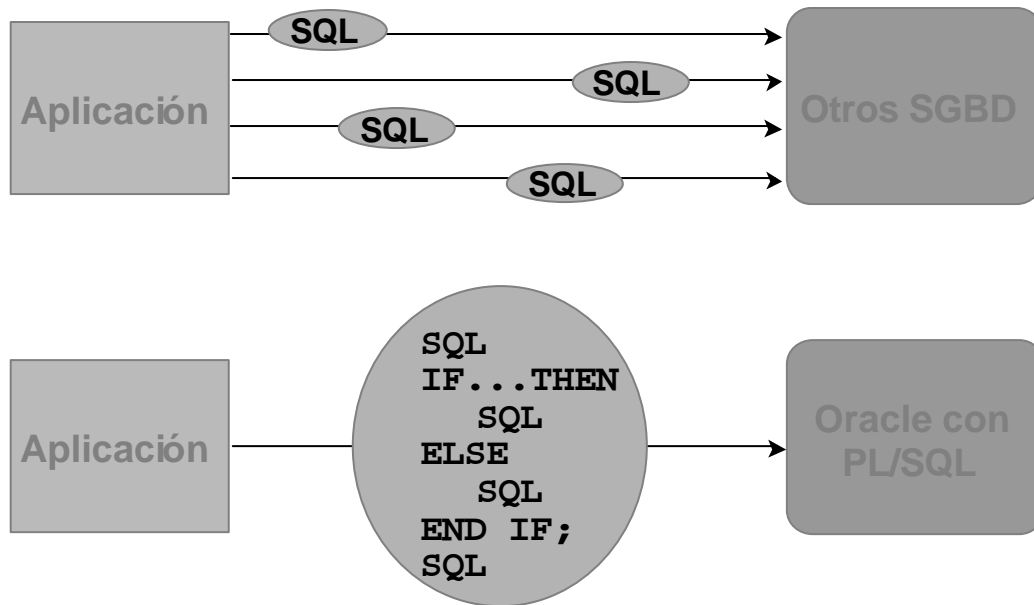
Muchas herramientas de Oracle, incluyendo Oracle Developer, tienen su propio motor PL/SQL, que es independiente del motor de Oracle Server.

Este motor filtra las sentencias SQL, las envía individualmente al ejecutor de sentencias SQL de Oracle Server y, después, procesa las sentencias procedurales restantes en el ejecutor de sentencias procedurales, que se encuentra en el motor PL/SQL.

El ejecutor de sentencias procedurales procesa los datos locales de la aplicación (es decir, los datos que ya se encuentran en el entorno cliente, en lugar de en la base de datos). De esta manera, se reduce el trabajo que se envía a Oracle Server y el número de cursores de memoria necesarios.

Ventajas de PL/SQL

Mejor rendimiento



ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Ventajas de PL/SQL (continuación)

Mejor rendimiento

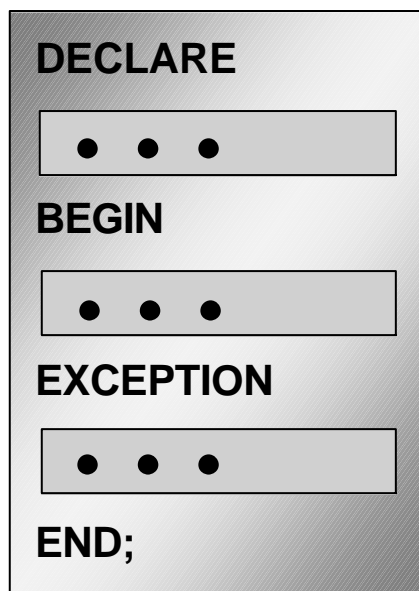
PL/SQL puede mejorar el rendimiento de una aplicación. Las ventajas varían en función de cada entorno de ejecución.

- PL/SQL se puede utilizar para agrupar sentencias SQL en un solo bloque y enviar el bloque completo al servidor con una sola llamada, reduciendo así el tráfico de red. Sin PL/SQL, las sentencias SQL se enviarían a Oracle Server de una en una. Cada sentencia SQL origina una llamada a Oracle Server y, por lo tanto, la sobrecarga de rendimiento es mayor. En un entorno de red, esta sobrecarga puede ser importante. Tal y como ilustra la transparencia, si la aplicación utiliza SQL de manera intensiva, se pueden usar subprogramas y bloques PL/SQL para agrupar sentencias SQL antes de enviarlas a Oracle Server para que las ejecute.
- PL/SQL también funciona con las herramientas de desarrollo de aplicaciones de Oracle Server como, por ejemplo, Oracle Forms y Oracle Reports. Al aumentar la capacidad de procesamiento procedural de estas herramientas, PL/SQL mejora el rendimiento.

Nota: Los procedimientos y las funciones que se declaran como parte de una aplicación Oracle Forms o Reports Developer son distintas de las almacenadas en la base de datos, aunque su estructura general sea la misma. Los subprogramas almacenados son objetos de base de datos y están almacenados en el diccionario de datos. Cualquier aplicación puede acceder a ellos, incluidas las aplicaciones Oracle Forms o Reports Developer.

Ventajas de PL/SQL

Desarrollo de programas modulares



ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Ventajas de PL/SQL (continuación)

Puede beneficiarse de las capacidades procedurales de PL/SQL, que no están disponibles en SQL.

Estructura de Bloques PL/SQL

Cada unidad de PL/SQL consta de uno o varios bloques. Estos bloques pueden estar completamente separados entre sí o anidados unos en otros. Las unidades básicas (procedimientos, funciones y bloques anónimos) que conforman un programa PL/SQL son bloques lógicos, que pueden contener cualquier número de subbloques anidados. Por lo tanto, un bloque puede representar una pequeña parte de otro bloque que, a su vez, puede formar parte de una unidad de código completa.

Desarrollo de Programas Modulares

- Agrupar lógicamente las sentencias relacionadas dentro de bloques.
- Anidar los subbloques en bloques de mayor tamaño para construir programas más potentes.
- Dividir un problema complejo en un conjunto de módulos lógicos, manejables y bien definidos e implementar los módulos con bloques.
- Colocar el código PL/SQL reutilizable en bibliotecas para que lo compartan las aplicaciones Oracle Forms y Oracle Reports o almacenarlo en un servidor Oracle Server para que sea accesible para cualquier aplicación que pueda interactuar con una base de datos de Oracle.

Ventajas de PL/SQL

- **PL/SQL es portable.**
- **Permite declarar variables.**

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Ventajas de PL/SQL (continuación)

Portabilidad

- Dado que PL/SQL es un lenguaje nativo de Oracle Server, se pueden desplazar los programas a cualquier entorno host (sistema operativo o plataforma) que sea compatible con Oracle Server y PL/SQL. Dicho de otro modo, los programas PL/SQL funcionan allí donde funcione Oracle Server, por lo que no es necesario adaptarlos a cada nuevo entorno.
- También puede mover código entre Oracle Server y la aplicación. Puede escribir paquetes de programas portables y crear bibliotecas que se puedan reutilizar en otros entornos.

Identificadores:

En PL/SQL, se pueden utilizar identificadores para hacer lo siguiente:

- Declarar variables, cursores, constantes y excepciones, y utilizarlos en sentencias procedurales y SQL.
- Declarar variables que pertenecen a los tipos de datos escalares, de referencia, compuestos y objetos de gran tamaño (LOB).
- Declarar variables dinámicamente basándose en la estructura de datos de las tablas y las columnas de la base de datos.

Ventajas de PL/SQL

- **Se puede programar con estructuras de control de lenguaje procedural.**
- **PL/SQL puede manejar errores.**

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Ventajas de PL/SQL (continuación)

Estructuras de Control de Lenguaje Procedural:

Las estructuras de control de lenguaje procedural permiten hacer lo siguiente:

- Ejecutar una secuencia de sentencias de manera condicional
- Ejecutar una secuencia de sentencias de manera iterativa en un bucle
- Procesar individualmente las filas que devuelve una consulta de múltiples filas con un cursor explícito

Errores:

La funcionalidad del manejo de errores de PL/SQL permite hacer lo siguiente:

- Procesar los errores de Oracle Server con rutinas de manejo de excepciones
- Declarar condiciones de error definidas por el usuario y procesarlas con rutinas de manejo de errores

Ventajas de los Subprogramas

- **Fácil mantenimiento**
- **Mayor integridad y seguridad de los datos**
- **Mejor rendimiento**
- **Mayor claridad del código**

ORACLE

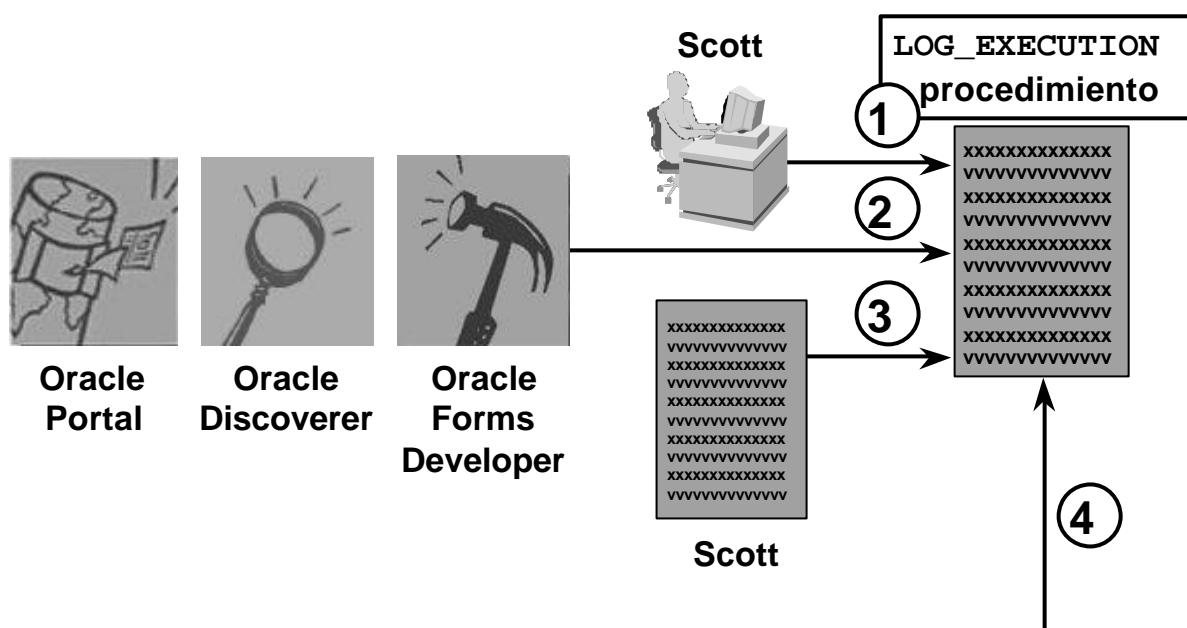
Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Ventajas de los Subprogramas

Las funciones y los procedimientos almacenados tienen muchas ventajas, además de permitir el desarrollo de aplicaciones modulares:

- Facilidad de mantenimiento que permite modificar:
 - Las rutinas en línea sin interferir con otros usuarios
 - Una rutina para que afecte a varias aplicaciones
 - Una rutina para no duplicar las pruebas
- Mayor integridad y seguridad de los datos, gracias a lo siguiente:
 - Control del acceso indirecto a los objetos de base de datos por parte de los usuarios sin privilegios utilizando privilegios de seguridad.
 - Garantía de que las acciones relacionadas se realizan juntas o no se realizan, al canalizar la actividad para tablas relacionadas a través de una sola ruta de acceso
- Mejor rendimiento que permite hacer lo siguiente:
 - Evitar repetir el análisis para múltiples usuarios utilizando el área de SQL compartido
 - Evitar el análisis de PL/SQL en tiempo de ejecución realizando el análisis en tiempo de compilación
 - Reducir el número de llamadas a la base de datos y disminuir el tráfico de red agrupando comandos
- Mayor claridad del código gracias a la utilización de nombres de identificadores adecuados para describir la acción de las rutinas, lo que reduce la necesidad de incluir comentarios y mejora la claridad del código.

Llamada a Funciones y Procedimientos Almacenados



ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Llamada a Funciones y Procedimientos Almacenados

Puede llamar a un procedimiento o función creado previamente desde una gran variedad de entornos como, por ejemplo, iSQL*Plus, Oracle Forms Developer, Oracle Discoverer, Oracle Portal, otro procedimiento almacenado y muchas otras herramientas de Oracle y precompiladores. La siguiente tabla explica cómo se puede llamar a un procedimiento creado previamente, log_execution, desde diversos entornos.

iSQL*Plus	EXECUTE log_execution
Herramientas de desarrollo de Oracle, como Oracle Forms Developer	log_execution;
Otro procedimiento	<pre> CREATE OR REPLACE PROCEDURE leave_emp (p_id IN employees.employee_id%TYPE) IS BEGIN DELETE FROM employees WHERE employee_id = p_id; log_execution; END leave_emp; </pre>

Resumen

- **PL/SQL es una extensión de SQL.**
- **Los bloques de código PL/SQL se transfieren a un motor PL/SQL, que los procesa.**
- **Ventajas de PL/SQL:**
 - Integración
 - Mejor rendimiento
 - Portabilidad
 - Desarrollo de programas modulares
- **Los subprogramas son bloques PL/SQL con nombre, declarados como procedimientos o funciones.**
- **Se puede llamar a los subprogramas desde diversos entornos.**

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Resumen

PL/SQL es un lenguaje que tiene funciones de programación que sirven de extensión a SQL. Ofrece la capacidad de controlar el flujo de construcciones y declarar y utilizar variables. Las aplicaciones PL/SQL funcionan en cualquier plataforma o sistema operativo en el que funcione Oracle.

Los bloques PL/SQL con nombre también se denominan subprogramas o unidades de programas. Los procedimientos, las funciones, los paquetes y los disparadores son diversas construcciones PL/SQL. Se puede llamar a los subprogramas desde diversos entornos.

1

Declaración de Variables

ORACLE®

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Objetivos

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

- **Reconocer el bloque PL/SQL básico y sus secciones**
- **Describir la importancia de las variables en PL/SQL**
- **Declarar variables PL/SQL**
- **Ejecutar un bloque PL/SQL**

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Finalidad de la Lección

Esta lección explica la estructura y las reglas básicas para escribir y ejecutar bloques de código PL/SQL. También le enseñará a declarar variables y asignarles tipos de dato.

Estructura de Bloques PL/SQL

DECLARE (Opcional)

Variables, cursores, excepciones definidas por el usuario

BEGIN (Obligatorio)

- Sentencias SQL
- Sentencias PL/SQL

EXCEPTION (Opcional)

Acciones que se deben realizar cuando ocurren errores

END; (Obligatorio)

DECLARE

...

BEGIN

...

EXCEPTION

...

END;

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Estructura de Bloques PL/SQL

PL/SQL es un lenguaje estructurado en bloques, lo que significa que los programas se pueden dividir en bloques lógicos. Los bloques PL/SQL constan de hasta tres secciones: la sección declarativa (opcional), la sección ejecutable (necesaria) y la sección de manejo de excepciones (opcional). La siguiente tabla describe estas tres secciones:

Sección	Descripción	Inclusión
Declarativa	Contiene todas las variables, constantes, cursores y excepciones definidas por el usuario a las que se hace referencia en las secciones ejecutable y declarativa.	Opcional
Ejecutable	Contiene sentencias SQL para manipular datos en la base de datos y sentencias PL/SQL para manipular datos en el bloque	Obligatoria
Manejo de excepciones	Especifica las acciones que hay que realizar cuando se producen errores y condiciones anormales en la sección ejecutable	Opcional

Ejecución de Sentencias y Bloques PL/SQL

```
DECLARE
  v_variable  VARCHAR2(5);
BEGIN
  SELECT nombre_columna
  INTO v_variable
  FROM nombre_tabla;
EXCEPTION
  WHEN nombre_excepción THEN
  ...
END;
```

```
DECLARE
  ...
BEGIN
  ...
EXCEPTION
  ...
END;
```

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Ejecución de Sentencias y Bloques PL/SQL

- Coloque un punto y coma (;) al final de las sentencias SQL o las sentencias de control PL/SQL.
- Cuando el bloque se ejecute correctamente, sin errores no tratados ni errores de compilación, se debería leer el siguiente mensaje:

PL/SQL procedure successfully completed.

- Las palabras clave de sección DECLARE, BEGIN y EXCEPTION no van seguidas de punto y coma.
- Las sentencias END y el resto de las sentencias PL/SQL necesitan un punto y coma para terminar la sentencia.
- Es posible encadenar sentencias en la misma línea, pero no se recomienda hacerlo por motivos de claridad y de edición.

Nota: En PL/SQL, los errores se *denominan excepciones*.

Gracias a la modularidad, es posible dividir una aplicación en módulos bien definidos y más manejables. A través de un análisis sucesivo, es posible reducir un problema complejo a un conjunto de problemas más sencillos que tienen soluciones fáciles de implementar. PL/SQL satisface esta necesidad con unidades de programa, que incluyen bloques, subprogramas y paquetes.

Tipos de Bloques

Anónimo

```
[DECLARE]

BEGIN
    --sentencias

[EXCEPTION]

END;
```

Procedimiento

```
PROCEDURE nombre
IS

BEGIN
    --sentencias

[EXCEPTION]

END;
```

Función

```
FUNCTION nombre
RETURN tipo_de_dato
IS
BEGIN
    --sentencias
    RETURN valor;
[EXCEPTION]

END;
```

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Tipos de Bloques

Los programas PL/SQL constan de uno o varios bloques. Estos bloques pueden estar completamente separados entre sí o anidados unos en otros. Las unidades básicas (procedimientos y funciones, también denominados subprogramas y bloques anónimos) que conforman un programa PL/SQL son bloques lógicos, que pueden contener cualquier número de subbloques anidados. Por lo tanto, un bloque puede representar una pequeña parte de otro bloque que, a su vez, puede formar parte de una unidad de código completa.

Bloques Anónimos

Los bloques anónimos son bloques sin nombre. Se declaran en el punto de la aplicación en el que tienen que ser ejecutados y se transfieren al motor PL/SQL para que los ejecute en tiempo de ejecución. Es posible embeber un bloque anónimo en un precompilador y en *iSQL*Plus* o *Server Manager*. Los disparadores de los componentes de *Oracle Developer* constan de dichos bloques.

Subprogramas

Los subprogramas son bloques PL/SQL con nombre que aceptan parámetros y se pueden llamar. Se pueden declarar como procedimientos o como funciones. Generalmente, los procedimientos se utilizan para realizar una acción y las funciones, para calcular un valor.

Los subprogramas se pueden almacenar en el nivel de aplicación o de servidor. Con los componentes de *Oracle Developer* (Forms, Reports y Graphics), los procedimientos y las funciones se pueden declarar como parte de la aplicación (una pantalla o un informe) y llamarlos desde otros procedimientos, funciones y disparadores (vea la página siguiente) dentro de la misma aplicación siempre que sea necesario.

Nota: Las funciones son similares a los procedimientos, excepto en que las funciones *deben* devolver un valor.

Construcciones de Programas

```

DECLARE
. . .
BEGIN
. . .
EXCEPTION
. . .
END ;
    
```

Construcciones de Herramientas

Bloques anónimos
Procedimientos o funciones de aplicaciones
Paquetes de aplicaciones
Disparadores de aplicaciones
Tipos de objetos

Construcciones de Servidores de Bases de Datos

Bloques anónimos
Procedimientos o funciones almacenados
Paquetes almacenados
Disparadores de bases de datos
Tipos de objetos

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Construcciones de Programas

En la siguiente tabla se describe una gran variedad de construcciones de programas PL/SQL diferentes que utilizan el bloque PL/SQL básico. Las construcciones de programas están disponibles en función del entorno en el que se ejecuten.

Construcción de programa	Descripción	Disponibilidad
Bloques anónimos	Bloques PL/SQL sin nombre que están embebidos en una aplicación o se ejecutan de manera interactiva	Todos los entornos PL/SQL
Procedimientos o funciones de aplicaciones	Bloques PL/SQL con nombre que están almacenados en una aplicación Oracle Forms Developer o en una biblioteca compartida; pueden aceptar parámetros y se pueden llamar repetidamente por su nombre	Componentes de herramientas de Oracle Developer como, por ejemplo, Oracle Forms Developer, Oracle Reports
Procedimientos o funciones almacenados	Bloques PL/SQL con nombre que están en Oracle Server; pueden aceptar parámetros y se pueden llamar repetidamente por su nombre	Oracle Server
Paquetes (de aplicación o almacenados)	Módulos PL/SQL con nombre que agrupan procedimientos, funciones e identificadores relacionados.	Componentes de las herramientas de Oracle Server y Oracle Developer como, por ejemplo, Oracle Forms Developer
Disparadores de bases de datos	Bloques PL/SQL que están asociados a una tabla de base de datos y se disparan automáticamente cuando los activan las sentencias DML	Oracle Server
Disparadores de aplicaciones	Bloques PL/SQL que están asociados a un evento de aplicación y se activan automáticamente	Componentes de herramientas de Oracle Developer como, por ejemplo, Oracle Forms Developer
Tipos de objetos	Tipos de datos compuestos definidos por el usuario que encapsulan una estructura de datos junto a las funciones y procedimientos necesarios para manipular los datos.	Oracle Server y herramientas de Oracle Developer

Uso de Variables

Las variables se utilizan para:

- **El almacenamiento temporal de los datos**
- **La manipulación de los valores almacenados**
- **Reutilización**
- **Facilidad de mantenimiento**

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Uso de Variables

Con PL/SQL, es posible declarar variables y utilizarlas en sentencias procedurales y SQL en cualquier lugar en el que se pueda utilizar una expresión. Las variables se utilizan para lo siguiente:

- **Almacenamiento temporal de los datos:** Los datos se pueden almacenar temporalmente en una o varias variables para utilizarlos cuando se valide la entrada de datos y para procesarlos posteriormente durante el flujo de datos.
- **Manipulación de valores almacenados:** Las variables pueden se pueden utilizar para realizar cálculos y otras manipulaciones de datos sin acceder a la base de datos.
- **Reutilización:** Después de ser declaradas, las variables se pueden utilizar repetidas veces en una aplicación simplemente haciendo referencia a ellas en otras sentencias, incluidas otras sentencias declarativas.
- **Facilidad de mantenimiento:** Cuando se utiliza %TYPE y %ROWTYPE (en una lección posterior encontrará más información acerca de %ROWTYPE), está declarando variables, basando las declaraciones en las definiciones de las columnas de la base de datos. Si una definición subyacente cambia, la declaración de la variable también cambia en tiempo de ejecución. De esta manera, se consigue la independencia de los datos, se reducen los costes de mantenimiento y permite que los programas se adapten a los cambios de la base de datos para satisfacer nuevas necesidades del negocio. Más adelante, en esta misma lección, encontrará más información acerca de %TYPE.

Manejo de Variables en PL/SQL

- **Declare e inicialice las variables en la sección de declaraciones.**
- **Asigne nuevos valores a las variables en la sección ejecutable.**
- **Tranfiera valores a los bloques PL/SQL mediante parámetros.**
- **Vea los resultados a través de variables de salida.**

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Manejo de Variables en PL/SQL

Declare e Inicialice las Variables en la Sección de Declaraciones

Puede declarar las variables en la parte declarativa de cualquier bloque, subprograma o paquete PL/SQL. Las declaraciones asignan espacio de almacenamiento para un valor, especifican su tipo de dato y dan un nombre a la ubicación de almacenamiento para que se le pueda hacer referencia. Las declaraciones también pueden asignar un valor inicial e imponer la restricción NOT NULL (no nulo) a la variable. Las referencias por adelantado no están permitidas. Debe declarar una variable antes de hacer referencia a ella en otras sentencias, incluidas otras sentencias declarativas.

Asigne Nuevos Valores a las Variables en la Sección Ejecutable

En la sección ejecutable, el valor existente de la variable se sustituye por el nuevo valor que se asigna a la variable.

Transfiera Valores a los Bloques PL/SQL Mediante Parámetros

Existen tres modos de parámetros: IN (por defecto), OUT e IN OUT. Utilice el parámetro IN para transferir valores al subprograma al que se está llamando. Use el parámetro OUT para devolver valores a quien llama a un subprograma. Y emplee el parámetro IN OUT para transferir valores iniciales al subprograma al que se llama y para devolver valores actualizados a quien realiza la llamada. Para transferir valores a un bloque anónimo, se utilizan variables de sustitución *SQL*PLUS*.

Nota: Más adelante, en esta misma lección, se explicará cómo se ven los resultados de un bloque PL/SQL mediante variables de salida.

Tipos de Variables

- **Variables PL/SQL:**
 - **Escalares**
 - **Compuestas**
 - **De referencia**
 - **LOB (objetos de gran tamaño)**
- **Variables que no son PL/SQL: Variables ligadas y de host**

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Tipos de Variables

Todas las variables PL/SQL tienen un tipo de dato que especifica un formato de almacenamiento, restricciones y un rango de valores válido. PL/SQL admite cuatro categorías de tipos de dato: escalares, compuestos, de referencia y LOB (objeto de gran tamaño), que se utilizan para declarar variables, constantes y punteros.

- Los tipos de dato escalares guardan un solo valor. Los tipos de dato principales son aquellos que se corresponden con tipos de columnas de las tablas de Oracle Server; PL/SQL también admite variables booleanas.
- Los tipos de dato compuestos, como los registros, permiten definir y manipular grupos de campos en bloques PL/SQL.
- Los tipos de dato de referencia guardan valores, denominados punteros, que designan otros elementos del programa. En este curso no se explicarán los tipos de dato de referencia.
- Los tipos de dato LOB guardan valores, denominados localizadores, que especifican la ubicación de los objetos de gran tamaño (como las imágenes) que se almacenan fuera de línea. Más adelante, en este mismo curso, hablaremos de los tipos de dato LOB.

Las variables que no son PL/SQL incluyen variables del lenguaje host que se declaran en los precompiladores, campos de pantalla de las aplicaciones Forms y variables de host de iSQL*Plus.

Para más información acerca de los tipos de dato LOB, consulte el capítulo “Fundamentals” de *PL/SQL User's Guide and Reference*.

Uso de Variables de iSQL*Plus en Bloques PL/SQL

- **PL/SQL no tiene la capacidad de entrada o salida en sí mismo.**
- **Para hacer referencia a variables de sustitución en un bloque PL/SQL, coloque delante el signo &.**
- **Se pueden utilizar variables de host (o "ligadas") de iSQL*Plus para transferir valores en tiempo de ejecución desde el bloque PL/SQL al entorno iSQL*Plus.**

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Uso de Variables de iSQL*Plus en Bloques PL/SQL

PL/SQL no tiene la capacidad de entrada o salida en sí mismo. Para transferir valores a/desde un bloque PL/SQL, hay que utilizar el entorno en el cual se está ejecutando PL/SQL.

En el entorno iSQL*Plus, se pueden utilizar variables de host (o "ligadas") para transferir valores en tiempo de ejecución a un bloque PL/SQL. Para hacer referencia a las variables de sustitución en un bloque PL/SQL, coloque delante el signo &, de la misma forma que para hacer referencia a las variables de sustitución iSQL*Plus en una sentencia SQL. Los valores de texto se sustituyen en el bloque PL/SQL antes de ejecutar el bloque. Por lo tanto, no es posible sustituir diferentes valores para las variables de sustitución utilizando un bucle. Sólo un valor reemplazará la variable de sustitución.

Se pueden utilizar variables de host de iSQL*Plus para transferir valores en tiempo de ejecución desde el bloque PL/SQL al entorno iSQL*Plus. Para hacer referencia a variables de host en un bloque PL/SQL, coloque delante dos puntos (:). Más adelante, en esta misma lección, se explicarán con más detalle las variables ligadas.

Tipos de Variables

TRUE



25-JAN-01

256120.08

“Hace ochenta y siete años
nuestros padres crearon en este
continente una nueva nación
concebida bajo el signo de la
LIBERTAD y consagrada al principio
de que todos los hombres nacen
iguales.”



Atlanta

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Tipos de Variables

Esta transparencia ilustra los siguientes tipos de dato de variables:

- TRUE representa un valor booleano.
- 25-JAN-01 representa un tipo de dato DATE.
- La fotografía representa un tipo de dato BLOB.
- El texto del discurso representa un tipo de dato LONG.
- 256120.08 representa un tipo de dato NUMBER con precisión y escala.
- La película representa un tipo de dato BFILE.
- El nombre de la ciudad, Atlanta, representa un tipo de dato VARCHAR2.

Declaración de Variables PL/SQL

Sintaxis:

```
identificador [CONSTANT] tipo_de_dato [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;
```

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Declaración de Variables PL/SQL

Debe declarar todos los identificadores PL/SQL en la sección de declaración antes de poder hacer referencia a ellos en el bloque PL/SQL. Tiene la opción de asignar un valor inicial a una variable. No obstante, para declarar una variable, no es necesario asignarle un valor. Si se refiere a otras variables en una declaración, asegúrese de declararlas por separado en una sentencia anterior.

En la sintaxis:

<i>identificador</i>	es el nombre de la variable.
CONSTANT	restringe la variable para que su valor no pueda cambiar; las constantes deben inicializarse.
<i>tipo_de_dato</i>	es un tipo de dato escalar, compuesto, de referencia o LOB. (En este curso sólo se explican los tipos de dato escalares, compuestos y LOB.)
NOT NULL	restringe la variable para que obligatoriamente contenga un valor. (Las variables NOT NULL deben inicializarse.)
<i>expr</i>	es cualquier expresión PL/SQL que pueda ser una expresión literal, otra variable o una expresión que implique el uso de operadores y funciones.

Instrucciones para Declarar Variables PL/SQL

- Siga las reglas de nomenclatura.
- Inicialice las variables designadas como **NOT NULL** y **CONSTANT**.
- Declare un identificador por línea.
- Utilice el operador de asignación (**:=**) o la palabra reservada **DEFAULT** para inicializar los identificadores.

```
identificador := expr;
```

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Instrucciones para Declarar Variables PL/SQL

He aquí algunas de las instrucciones que debe seguir para declarar variables PL/SQL:

- Asigne un nombre al identificador utilizando las mismas reglas que se aplican a los objetos SQL.
- Puede utilizar reglas de nomenclatura como, por ejemplo, *v_nombre* para representar una variable y *c_nombre* para representar una constante.
- Si utiliza la restricción **NOT NULL**, debe asignar un valor.
- Si declara un solo identificador por línea, el código será más fácil de leer y de mantener.
- En las declaraciones de constantes, la palabra clave **CONSTANT** debe preceder al especificador de tipo. La siguiente declaración define una constante del subtipo **REAL** de **NUMBER** y le asigna el valor de 50000. Las constantes se deben inicializar en su declaración; de lo contrario, se producirá un error de compilación al elaborar (compilar) la declaración.

```
v_sal      CONSTANT REAL := 50000.00;
```

- Inicialice la variable a una expresión con el operador de asignación (**:=**) o con la palabra reservada **DEFAULT**. Si no le asigna un valor inicial, la nueva variable contendrá el valor **NULL** por defecto hasta que le asigne un valor posteriormente. Para asignar o volver a asignar un valor a una variable, hay que escribir una sentencia de asignación PL/SQL. Para que reciba un nuevo valor, debe nombrar explícitamente la variable a la izquierda del operador de asignación (**:=**). La inicialización de todas las variables es una buena práctica de programación.

Reglas de Nomenclatura

- Dos variables pueden tener el mismo nombre, siempre y cuando estén en bloques diferentes.
- El nombre de la variable (identificador) no debería ser igual que el nombre de las columnas de tabla que se utilizan en el bloque.

```
DECLARE
    employee_id NUMBER(6);
BEGIN
    SELECT      employee_id
    INTO        employee_id
    FROM        employees
    WHERE       last_name = 'Kochhar';
END;
/
```

Utilice una regla de nomenclatura para los identificadores PL/SQL: por ejemplo, **v_employee_id**

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Reglas de Nomenclatura

Dos objetos pueden tener el mismo nombre, siempre y cuando estén definidos en bloques diferentes. Si coexisten, sólo se puede utilizar el objeto declarado en el bloque actual.

El nombre (identificador) de una variable no debería ser el mismo que el nombre de las columnas de la tabla utilizadas en el bloque. Si en las sentencias SQL hay variables PL/SQL que tienen el mismo nombre que una columna, Oracle Server da a entender que se está haciendo referencia a la columna. Aunque el código de ejemplo de la transparencia funciona, el código escrito utilizando el mismo nombre para una tabla de base de datos y una variable no es fácil de leer ni de mantener.

Considere el uso de reglas de nomenclatura para los diversos objetos que se declaran en la sección DECLARE del bloque PL/SQL. Si utiliza el prefijo v_ para representar *variable*, evitará conflictos de nomenclatura con objetos de la base de datos.

```
DECLARE
    v_hire_date date;
BEGIN
    ...
```

Nota: Los nombres de las variables no deben tener más de 30 caracteres. El primer carácter debe ser una letra y el resto pueden ser letras, números o símbolos especiales.

Inicialización de Variables y Palabras Clave

- Operador de asignación (**:=**)
- Palabra clave **DEFAULT**
- Restricción **NOT NULL**

Sintaxis:

```
identificador := expr;
```

Ejemplos:

```
v_hiredate := '01-JAN-2001';
```

```
v_ename := 'Maduro';
```

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Inicialización de Variables y Palabras Clave

En la sintaxis:

identificador es el nombre de la variable escalar.

expr puede ser una variable, literal o función, pero *no* una columna de base de datos.

Los ejemplos de asignación de valores a variables se definen de la siguiente manera:

- Define el identificador **V_HIREDATE** con el valor 01-JAN-2001.
- Almacena el nombre “Maduro” en el identificador **V_ENAME**.

Las variables se inicializan siempre que se accede a un bloque o subprograma. Por defecto, las variables se inicializan en **NULL**. A menos que inicialice explícitamente una variable, su valor será indefinido.

Utilice el operador de asignación (**:=**) para las variables que no tengan valores normales.

```
v_hire_date := '15-SEP-1999'
```

Nota: Esta asignación de un valor de cuatro dígitos para el año, **YYYY**, sólo es posible en Oracle8i y versiones posteriores. En las versiones anteriores, hay que utilizar la función **TO_DATE**.

DEFAULT: Puede utilizar la palabra clave **DEFAULT** en lugar del operador de asignación para inicializar variables. Utilice **DEFAULT** para variables que tengan valores normales.

```
v_mgr NUMBER(6) DEFAULT 100;
```

NOT NULL: Imponga la restricción **NOT NULL** cuando la variable deba contener un valor.

No puede asignar valores **NULL** a una variable que está definida como **NOT NULL**. La restricción **NOT NULL** debe ir seguida de una cláusula de inicialización.

```
v_city VARCHAR2(30) NOT NULL := 'Oxford'
```

Inicialización de Variables y Palabras Clave (continuación)

Nota: Los literales de cadena deben ir entre comillas simples. Por ejemplo, 'Hello, world'. Si en la cadena ya existe alguna comilla simple, utilice dos veces las comillas simples. Por ejemplo, para insertar el valor FISHERMAN'S DRIVE, la cadena sería 'FISHERMAN' 'S DRIVE'.

Otra manera de asignar valores a las variables es seleccionar o recuperar valores de la base de datos en ella. En el siguiente ejemplo se calcula una bonificación del 10% para el empleado EMPLOYEE_ID 176 y se asigna el valor obtenido a la variable v_bonus. Para ello, se utiliza la cláusula INTO.

```
DECLARE
    v_bonus NUMBER(8,2);
BEGIN
    SELECT  salary * 0.10
    INTO    v_bonus
    FROM    employees
    WHERE   employee_id = 176;
END;
/
```

Luego, puede utilizar la variable v_bonus en otro cálculo o insertar su valor en una tabla de base de datos.

Nota: Para asignar un valor a una variable desde la base de datos, utilice una sentencia SELECT o FETCH. Más adelante, en este mismo curso, se explicará la sentencia FETCH.

Tipos de Dato Escalares

- Guardan un solo valor
- No tienen componentes internos

25-OCT-99

256120.08

“Hace ochenta y siete años
nuestros padres crearon en este
continente una nueva nación
concebida bajo el signo de la
LIBERTAD y consagrada al
principio de que todos los
hombres nacen iguales.”

TRUE

Atlanta

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Tipos de Dato Escalares

Todas las constantes, variables y parámetros tienen un tipo de dato (o tipo) que especifica un formato de almacenamiento, restricciones y un rango de valores válido. PL/SQL tiene una gran variedad de tipos de dato predefinidos. Por ejemplo, puede elegir entre los tipos entero, de coma flotante, de carácter, booleano, de fecha, de recopilación, de referencia y LOB. Además, en este capítulo describiremos los tipos básicos que se utilizan con más frecuencia en los programas PL/SQL. En otros capítulos se explicarán los tipos más especializados.

El tipo de dato escalar guarda un solo valor y no tiene componentes internos. Los tipos de dato escalares se pueden clasificar en cuatro categorías: número, carácter, fecha y booleano. Los tipos de dato de carácter y número tienen subtipos que asocian un tipo base a una restricción. Por ejemplo, `INTEGER` y `POSITIVE` son subtipos del tipo base `NUMBER`.

Para obtener más información y la lista completa de los tipos de dato escalares, consulte el capítulo “Fundamentals” de *PL/SQL User's Guide and Reference*.

Tipos de Dato Escalares Base

- `CHAR [(longitud_máxima)]`
- `VARCHAR2 (longitud_máxima)`
- `LONG`
- `LONG RAW`
- `NUMBER [(precisión, escala)]`
- `BINARY_INTEGER`
- `PLS_INTEGER`
- `BOOLEAN`

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Tipos de Dato Escalares Base

Tipo de Dato	Descripción
<code>CHAR</code> <code>[(longitud_máxima)]</code> <code>)</code>	Tipo base para datos de caracteres de longitud fija de hasta 32.767 bytes. Si no se especifica una <i>longitud_máxima</i> , la longitud por defecto es 1.
<code>VARCHAR2</code> <code>(longitud_máxima)</code>	Tipo base para datos de caracteres de longitud variable de hasta 32.767 bytes. No existe un tamaño por defecto para las constantes y variables <code>VARCHAR2</code> .
<code>LONG</code>	Tipo base para datos con una longitud de caracteres variable de hasta 32.760 bytes. Utilice el tipo de dato <code>LONG</code> para almacenar cadenas de caracteres de longitud variable. Puede insertar cualquier valor <code>LONG</code> en una columna de base de datos <code>LONG</code> , porque el ancho máximo de las columnas <code>LONG</code> es de 2**31 bytes. Sin embargo, no puede recuperar un valor mayor que 32760 bytes desde una columna <code>LONG</code> en una variable <code>LONG</code> .
<code>LONG RAW</code>	Tipo base para datos binarios y cadenas de bytes de hasta 32.760 bytes. PL/SQL no interpreta los datos <code>LONG RAW</code> .
<code>NUMBER</code> <code>[(precisión, escala)]</code>	Número que tiene una precisión <i>p</i> y una escala <i>s</i> . La precisión <i>p</i> oscila entre 1 y 38, mientras que la escala <i>s</i> puede oscilar entre -84 y 127.

Tipos de Dato Escalares Base (continuación)

Tipo de Dato	Descripción
<code>BINARY_INTEGER</code>	Tipo base para enteros entre -2.147.483.647 y 2.147.483.647.
<code>PLS_INTEGER</code>	Tipo base para enteros con signo entre -2.147.483.647 y 2.147.483.647. Los valores <code>PLS_INTEGER</code> requieren menos espacio de almacenamiento y son más rápidos que los valores <code>NUMBER</code> y <code>BINARY_INTEGER</code> .
<code>BOOLEAN</code>	Tipo base que almacena uno de los tres valores posibles que se usan para los cálculos lógicos: <code>TRUE</code> , <code>FALSE</code> o <code>NULL</code> .

Tipos de Dato Escalares Base

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

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Tipos de Dato Escalares Base (continuación)

Tipo de Dato	Descripción
DATE	Tipo base para fechas y horas. Los valores DATE incluyen la hora del día en segundos, contados a partir de media noche. El rango de las fechas oscila entre 4712 a.C y 9999 d.C.
TIMESTAMP	El tipo de dato TIMESTAMP, que es una extensión del tipo de dato DATE, almacena el año, el mes, el día, la hora, el minuto y el segundo. La sintaxis es: <code>TIMESTAMP[(precisión)]</code> donde el parámetro opcional precisión especifica el número de dígitos de la parte fraccional del campo de los segundos. No puede utilizar una constante o variable simbólica para especificar la precisión, sino un literal entero en el rango 0 .. 9. Por defecto, es 6.
TIMESTAMP WITH TIME ZONE	El tipo de dato TIMESTAMP WITH TIME ZONE, que es una extensión del tipo de dato TIMESTAMP, incluye el desfase de una zona horaria. El desfase de la zona horaria es la diferencia (en horas y minutos) entre la hora local y el UTC (Coordinated Universal Time), antes denominado GMT (Greenwich Mean Time). La sintaxis es: <code>TIMESTAMP[(precisión)] WITH TIME ZONE</code> donde el parámetro opcional precisión especifica el número de dígitos de la parte fraccional del campo de los segundos. No puede utilizar una constante simbólica o variable para especificar la precisión, sino un literal entero en el rango 0 .. 9. Por defecto, es 6.

Tipos de Dato Escalares Base (continuación)

Tipo de Dato	Descripción
TIMESTAMP WITH LOCAL TIME ZONE	<p>El tipo de dato <code>TIMESTAMP WITH LOCAL TIME ZONE</code>, que es una extensión del tipo de dato <code>TIMESTAMP</code>, incluye el desfase de una zona horaria. El desfase de la zona horaria es la diferencia (en horas y minutos) entre la hora local y el UTC (Coordinated Universal Time), antes denominado GMT (Greenwich Mean Time). La sintaxis es:</p> <pre>TIMESTAMP[(precisión)] WITH LOCAL TIME ZONE</pre> <p>donde el parámetro opcional <code>precisión</code> especifica el número de dígitos de la parte fraccional del campo de los segundos. No puede utilizar una constante o variable simbólica para especificar la precisión, sino un literal entero en el rango 0 .. 9. Por defecto, es 6.</p> <p>Este tipo de dato se diferencia de <code>TIMESTAMP WITH TIME ZONE</code> en que cuando se inserta un valor en una columna de base de datos, ese valor se normaliza con la zona horaria de la base de datos y el desfase de la zona horaria no se almacena en la columna. Al recuperar el valor, Oracle devuelve dicho valor en la zona horaria de su sesión local.</p>
INTERVAL YEAR TO MONTH	<p>El tipo de dato <code>INTERVAL YEAR TO MONTH</code> sirve para almacenar y manipular intervalos de años y meses. La sintaxis es:</p> <pre>INTERVAL YEAR[(precisión)] TO MONTH</pre> <p>donde <code>years_precision</code> especifica el número de dígitos del campo de los años. No puede utilizar una constante o variable simbólica para especificar la precisión, sino un literal entero en el rango 0 .. 4. Por defecto, es 2.</p>
INTERVAL DAY TO SECOND	<p>El tipo de dato <code>INTERVAL YEAR TO SECOND</code> sirve para almacenar y manipular intervalos de días, horas, minutos y segundos. La sintaxis es:</p> <pre>INTERVAL DAY[(precisión1)] TO SECOND[(precisión2)]</pre> <p>donde <code>precisión1</code> y <code>precisión2</code> especifican el número de dígitos del campo de los días y el campo de los segundos, respectivamente. En ninguno de los casos, se puede utilizar una constante o variable simbólica para especificar la precisión, sino un literal entero en el rango 0 .. 9. Por defecto es 2 y 6, respectivamente.</p>

Declaraciones de Variables Escalares

Ejemplos:

```
DECLARE
  v_job          VARCHAR2(9);
  v_count        BINARY_INTEGER := 0;
  v_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

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Declaración de Variables Escalares

Los ejemplos de declaraciones de variables que se muestran en la transparencia se definen de la siguiente manera:

- `v_job`: variable para almacenar el puesto de un empleado en la empresa
- `v_count`: variable que cuenta las iteraciones de un bucle y que se inicializa en 0
- `v_total_sal`: variable que suma el total de los sueldos de un departamento y que se inicializa en 0
- `v_orderdate`: variable que almacena la fecha de envío de un pedido y se inicializa a una semana del día actual
- `c_tax_rate`: una variable constante para el tipo de impuestos, que no cambia nunca en el bloque PL/SQL
- `v_valid`: indicador que señala si un dato es válido o no y que se inicializa en `TRUE`

El Atributo %TYPE

- **Declare una variable de acuerdo con:**
 - La definición de una columna de base de datos
 - Otra variable declarada anteriormente
- **Prefije %TYPE con:**
 - La tabla y la columna de base de datos
 - El nombre de la variable declarada anteriormente

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

El Atributo %TYPE

Cuando declare variables PL/SQL para guardar valores de columnas, debe asegurarse de que la variable es del tipo de dato y precisión correctos. Si no es así, se producirá un error PL/SQL durante la ejecución.

En lugar de codificar el tipo de dato y la precisión de una variable, puede utilizar el atributo %TYPE para declarar la variable de acuerdo con otra variable que se haya declarado anteriormente, o en función de una columna de base de datos. El atributo %TYPE se suele utilizar cuando el valor almacenado en la variable se va a extraer de una tabla de base de datos. Para utilizar el atributo en lugar del tipo de dato que es necesario en la declaración de la variable, prefíjelo con el nombre de la tabla y la columna de base de datos. Si está haciendo referencia a una variable declarada anteriormente, prefije el nombre de la variable al atributo.

PL/SQL determina el tipo de dato y el tamaño de la variable al compilar el bloque, de manera que dichas variables sean siempre compatibles con la columna que se utiliza para rellenarla. Esta ventaja es muy importante a la hora de escribir y mantener el código, ya que no es necesario preocuparse por los cambios de tipos de dato de las columnas que se realizan en la base de datos. También puede declarar una variable de acuerdo con otra variable declarada anteriormente prefijando el atributo con el nombre de la variable.

Declaración de Variables con el Atributo %TYPE

Sintaxis:

```
identificador      Tabla.nombre_columna%TYPE;
```

Ejemplos:

```
...  
  v_name              employees.last_name%TYPE;  
  v_balance          NUMBER(7,2);  
  v_min_balance      v_balance%TYPE := 10;  
...
```

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Declaración de Variables con el Atributo %TYPE

Declare variables para almacenar el apellido de un empleado. La variable `v_name` se define con el mismo tipo de dato que la columna `LAST_NAME` de la tabla `EMPLOYEES`. `%TYPE` proporciona el tipo de dato de una columna de base de datos:

```
...  
  v_name              employees.last_name%TYPE;  
...
```

Declare variables para almacenar el saldo de una cuenta bancaria y su saldo mínimo, que se inicia como 10. La variable `v_min_balance` se define con el mismo tipo de dato que la variable `v_balance`. `%TYPE` proporciona el tipo de dato de una variable:

```
...  
  v_balance              NUMBER(7,2);  
  v_min_balance          v_balance%TYPE := 10;  
...
```

La restricción `NOT NULL` para columnas de bases de datos no se aplica 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`, es posible asignar un valor `NULL` a la variable.

Declaración de Variables Booleanas

- **A una variable booleana sólo se le pueden asignar los valores TRUE, FALSE y NULL.**
- **Las variables se comparan por medio de los operadores lógicos AND, OR y NOT .**
- **Las variables siempre devuelven TRUE, FALSE o NULL.**
- **Se pueden utilizar expresiones aritméticas, de carácter y de fecha para devolver un valor booleano.**

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Declaración de Variables Booleanas

En PL/SQL, puede comparar variables tanto en las sentencias SQL como en las procedurales. Estas comparaciones, denominadas expresiones booleanas, constan de expresiones simples o complejas separadas por operadores relacionales. En las sentencias SQL, puede utilizar expresiones booleanas para especificar las filas de una tabla a las que afecta la sentencia. En las sentencias procedurales, las expresiones booleanas son la base del control condicional. NULL representa un valor perdido, no aplicable o desconocido.

Ejemplos

```
v_sal1 := 50000;  
v_sal2 := 60000;
```


La siguiente expresión devuelve TRUE:

```
v_sal1 < v_sal2
```

Declare e inicialice una variable booleana:

```
DECLARE  
    v_flag BOOLEAN := FALSE;  
BEGIN  
    v_flag := TRUE;  
END;
```

Tipos de Dato Compuestos

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

Estructura de tablas PL/SQL

1	SMITH
2	JONES
3	NANCY
4	TIM

↑
↑
BINARY_INTEGER
VARCHAR2

Estructura de tablas PL/SQL

1	5000
2	2345
3	12
4	3456

↑
↑
BINARY_INTEGER
NUMBER

ORACLE

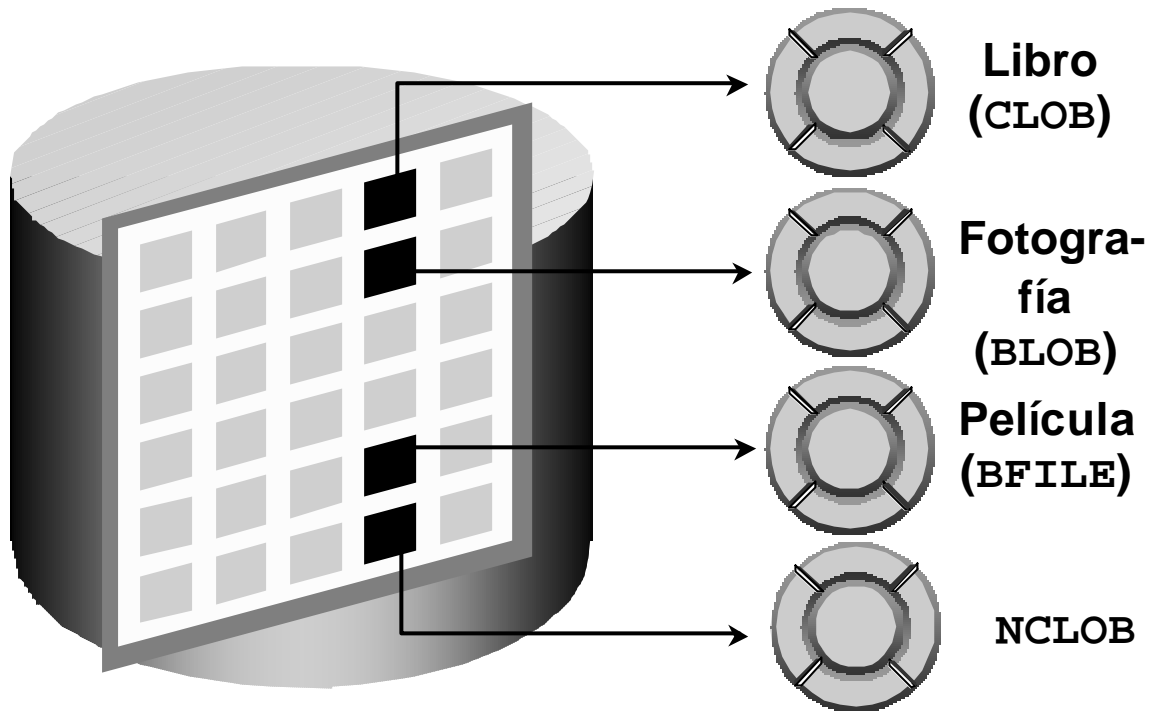
Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Tipos de Dato Compuestos

Los tipos escalares no tienen componentes internos. No obstante, los tipos compuestos tienen componentes internos que se pueden manipular de manera individual. Los tipos de dato compuestos (también denominados recopilaciones) son TABLE, RECORD, NESTED TABLE y VARRAY. El tipo de dato RECORD se utiliza para tratar los datos relacionados, pero distintos, como una unidad lógica. Utilice el tipo de dato TABLE para hacer referencia y manipular las recopilaciones de datos como un objeto completo. En una lección posterior se explicarán todos los detalles de los tipos de dato RECORD y TABLE. Los tipos de dato NESTED TABLE y VARRAY se explican en el curso *Advanced PL/SQL*.

Para más información, consulte el capítulo “Collections and Records” de *PL/SQL User’s Guide and Reference*.

Variables del Tipo de Dato LOB



ORACLE

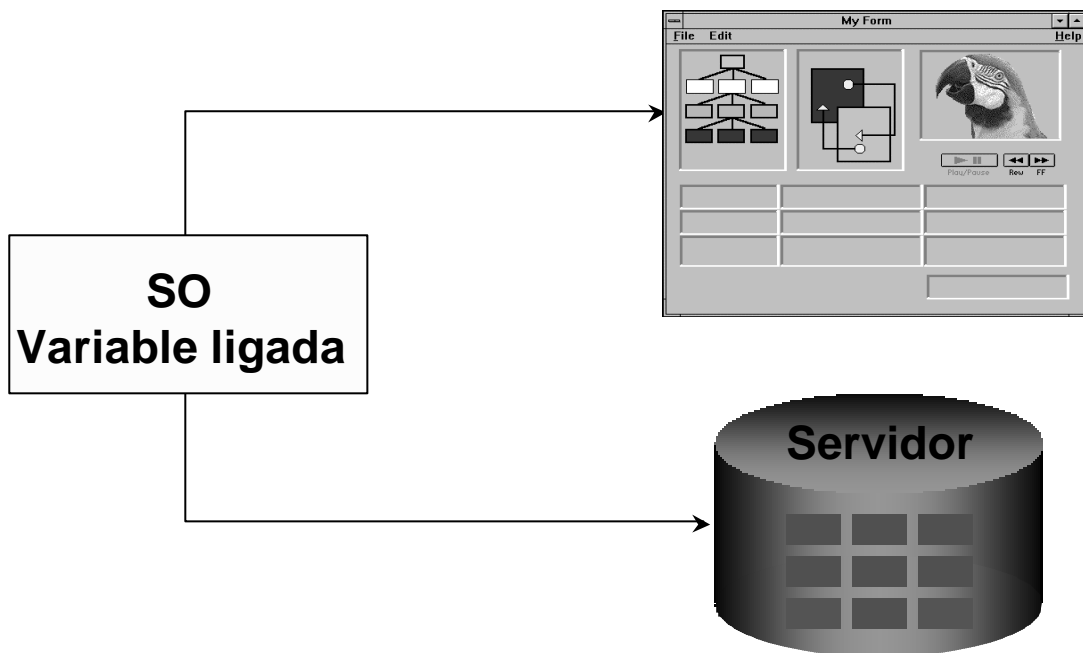
Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Variables del Tipo de Dato LOB

En los tipos de dato LOB (objeto de gran tamaño), puede almacenar bloques de datos no estructurados (como textos, imágenes, vídeo clips y formas de sonido) con un tamaño de hasta 4 GB. Los tipos de dato LOB permiten acceder a los datos de una manera eficiente, aleatoria y por partes, y pueden ser atributos de un tipo de objeto. Los LOB también permiten el acceso aleatorio a los datos.

- El tipo de dato CLOB (objetos de gran tamaño de caracteres) se utiliza para almacenar grandes bloques de datos de caracteres de un solo byte en la base de datos en línea (en el interior de la fila) o fuera de línea (en el exterior de la fila).
- El tipo de dato BLOB (objetos de gran tamaño binarios) se utiliza para almacenar objetos binarios de gran tamaño en la base de datos en línea (en el interior de la fila) o fuera de línea (en el exterior de la fila).
- El tipo de dato BFILE (archivo binario) se utiliza para almacenar objetos binarios de gran tamaño en archivos del sistema operativo fuera de la base de datos.
- El tipo de dato NCLOB (objeto de gran tamaño de caracteres de idioma nacional) se utiliza para almacenar grandes bloques de datos unicode NCHAR de un solo byte o multibyte de ancho fijo en la base de datos, ya sea en línea o fuera de línea.

Variables Ligadas



ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Variables Ligadas

Las variables ligadas son aquellas que se declaran en un entorno host. Las variables ligadas se pueden utilizar para transferir valores en tiempo de ejecución, ya sean números o caracteres, hacia o desde uno o varios programas PL/SQL. Los programas PL/SQL utilizan variables ligadas de la misma manera que utilizan cualquier otra variable. Puede hacer referencia a las variables declaradas en el host o en el entorno que realiza la llamada en sentencias PL/SQL, a menos que la sentencia esté en un procedimiento, función o paquete. Esto incluye las variables del lenguaje host declaradas en precompiladores, campos de pantalla de las aplicaciones Oracle Developer Forms y variables ligadas de *iSQL*Plus*.

Creación de Variables Ligadas

Para declarar una variable ligada en el entorno *iSQL*Plus*, utilice el comando `VARIABLE`. Por ejemplo, declare una variable del tipo `NUMBER` y `VARCHAR2` de la siguiente forma:

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

Tanto SQL como *iSQL*Plus* pueden hacer referencia a la variable ligada, y *iSQL*Plus* permite mostrar su valor con el comando `PRINT` de *iSQL*Plus*.

Visualización de Variables Ligadas

Utilice el comando `PRINT` para mostrar el valor actual de las variables ligadas en el entorno *iSQL*Plus*. No obstante, `PRINT` no se puede utilizar dentro de un bloque PL/SQL porque es un comando de *iSQL*Plus*. El siguiente ejemplo ilustra el comando `PRINT`:

```
VARIABLE g_n NUMBER
```

```
...
```

```
PRINT g_n
```

Puede hacer referencia a variables de host en los programas PL/SQL. Estas variables deben ir precedidas por dos puntos (:).

```
VARIABLE RESULT NUMBER
```

He aquí un ejemplo de cómo utilizar una variable de host en un bloque PL/SQL:

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

Uso de Variables Ligadas

Para hacer referencia a una variable ligada en PL/SQL, debe prefijar el nombre con dos puntos (:).

Ejemplo:

```
VARIABLE      g_salary NUMBER
BEGIN
  SELECT      salary
  INTO        :g_salary
  FROM        employees
  WHERE       employee_id = 178;
END;
/
PRINT g_salary
```

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Impresión de Variables Ligadas

En *iSQL*Plus* puede visualizar el valor de la variable ligada utilizando el comando `PRINT`.

G_SALARY	
	7000

Referencia a Variables No PL/SQL

Almacene el sueldo anual en una variable de host de *iSQL*Plus*.

```
:g_monthly_sal := v_sal / 12;
```

- Haga referencia a las variables que no son PL/SQL como variables de host.
- Prefije las referencias con dos puntos (:).

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Referencia a Variables No PL/SQL

Para hacer referencia a variables de host, debe prefijar las referencias con dos puntos (:) para distinguirlas de las variables PL/SQL declaradas.

Ejemplo

En este ejemplo se calcula el sueldo mensual, basándose en el sueldo anual que ha introducido el usuario. Este archivo de comandos contiene tanto comandos *iSQL*Plus* como un bloque PL/SQL completo.

```
VARIABLE g_monthly_sal NUMBER
DEFINE p_annual_sal = 50000

SET VERIFY OFF
DECLARE
    v_sal NUMBER(9,2) := &p_annual_sal;
BEGIN
    :g_monthly_sal := v_sal/12;
END;
/
PRINT g_monthly_sal
```

El comando DEFINE especifica una variable de usuario y le asigna un valor CHAR. Aunque introduzca el número 50000, *iSQL*Plus* asigna un valor CHAR a p_annual_sal que está compuesto por los caracteres 5,0,0,0 y 0.

DBMS_OUTPUT.PUT_LINE

- Un procedimiento empaquetado de Oracle
- Una alternativa para ver los datos de un bloque PL/SQL
- Se debe activar en *iSQL*Plus* con
`SET SERVEROUTPUT ON`

```
SET SERVEROUTPUT ON
DEFINE p_annual_sal = 60000
```

```
DECLARE
    v_sal NUMBER(9,2) := &p_annual_sal;
BEGIN
    v_sal := v_sal/12;
    DBMS_OUTPUT.PUT_LINE ('The monthly salary is ' ||
                           TO_CHAR(v_sal));
END;
/
```

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

DBMS_OUTPUT.PUT_LINE

Ha comprobado que puede declarar una variable de host, hacer referencia a ella en un bloque PL/SQL y luego ver su contenido en *iSQL*Plus* utilizando el comando `PRINT`. Otra opción para ver la información de un bloque PL/SQL es `DBMS_OUTPUT.PUT_LINE`. `DBMS_OUTPUT` es un paquete de Oracle y `PUT_LINE` es un procedimiento contenido en ese paquete.

Haga referencia a `DBMS_OUTPUT.PUT_LINE` dentro de un bloque PL/SQL y, entre paréntesis, especifique la cadena que desea imprimir en pantalla. Primero, hay que activar el paquete en la sesión de *iSQL*Plus*. Para ello, ejecute el comando `SET SERVEROUTPUT ON` de *iSQL*Plus*.

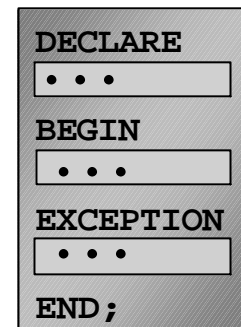
El ejemplo de esta transparencia calcula el sueldo mensual y lo imprime en pantalla utilizando `DBMS_OUTPUT.PUT_LINE`. La salida es la que se muestra a continuación:

```
The monthly salary is 5000
PL/SQL procedure successfully completed.
```

Resumen

En esta lección ha aprendido que:

- **Los bloques PL/SQL se componen de las siguientes secciones:**
 - **Declarativa (opcional)**
 - **Ejecutable (obligatoria)**
 - **Manejo de excepciones (opcional)**
- **Un bloque PL/SQL puede ser una función, un procedimiento o un bloque anónimo.**



ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Resumen

Los bloques PL/SQL son unidades básicas sin nombre de un programa PL/SQL. Estos bloques constan de un conjunto de sentencias SQL o PL/SQL y realizan una sola función lógica. La sección declarativa es la primera parte del bloque PL/SQL y se utiliza para declarar objetos como variables, constantes, cursores y definiciones de situaciones de error que se denominan excepciones. La sección ejecutable es la parte obligatoria de los bloques PL/SQL y contiene sentencias SQL y PL/SQL para consultar y manipular datos. La sección de manejo de excepciones está embebida en la sección ejecutable del bloque y se coloca al final de ella.

Los bloques PL/SQL anónimos son las unidades básicas sin nombre de un programa PL/SQL. Los procedimientos y las funciones se pueden compilar por separado y almacenar permanentemente en una base de datos Oracle, listos para ser ejecutados.

Resumen

En esta lección ha aprendido que:

- **Los identificadores PL/SQL:**
 - Se definen en la sección declarativa
 - Pueden ser del tipo de dato escalar, compuesto, de referencia o LOB
 - Se pueden basar en la estructura de otra variable u objeto de base de datos
 - Se pueden inicializar
- Las variables declaradas en un entorno externo como *iSQL*Plus* se denominan variables de host.
- Se puede utilizar `DBMS_OUTPUT.PUT_LINE` para ver los datos desde un bloque PL/SQL.

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Resumen (continuación)

Todos los tipos de dato PL/SQL son escalares, compuestos, de referencia o LOB. Los tipos de dato escalares no tienen ningún componente, mientras que los tipos de dato compuestos contienen otros tipos de dato. Las variables PL/SQL se declaran y se inicializan en la sección declarativa.

Cuando se escribe y se ejecuta un programa PL/SQL con *iSQL*Plus*, *iSQL*Plus* se convierte en el entorno host del programa PL/SQL. Las variables declaradas en *iSQL*Plus* se denominan variables de host. Luego, el programa PL/SQL se escribe y se ejecuta utilizando, por ejemplo, Oracle Forms. Forms se convierte en un entorno host y las variables declaradas en Oracle Forms se denominan variables de host. Las variables de host también se denominan variables ligadas.

Para ver información de un bloque PL/SQL, utilice `DBMS_OUTPUT.PUT_LINE`. `DBMS_OUTPUT` es un paquete de Oracle y `PUT_LINE` es un procedimiento contenido en ese paquete. Haga referencia a `DBMS_OUTPUT.PUT_LINE` dentro de un bloque PL/SQL y, entre paréntesis, especifique la cadena que desea imprimir en pantalla.

Visión General de la Práctica 1

Esta práctica cubre los siguientes temas:

- **Determinación de la validez de las declaraciones**
- **Declaración de un bloque PL/SQL simple**
- **Ejecución de un bloque PL/SQL simple**

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Visión General de la Práctica 1

Esta práctica refuerza los conceptos básicos de PL/SQL que se han explicado en esta lección como, por ejemplo, los tipos de dato, las definiciones de identificadores y la validación de expresiones. Debe unir todos estos elementos para crear un bloque PL/SQL simple.

Preguntas sobre Papel

Las preguntas 1 y 2 se realizan sobre papel.

Práctica 1

1. Evalúe cada una de las siguientes declaraciones. Determine cuáles *no* son válidas y explique por qué.

- a. DECLARE
 v_id NUMBER(4);

- b. DECLARE
 v_x, v_y, v_z VARCHAR2(10);

- c. DECLARE
 v_birthdate DATE NOT NULL;

- d. DECLARE
 v_in_stock BOOLEAN := 1;

Práctica 1 (continuación)

2. Para cada una de las siguientes asignaciones, indique si la sentencia es válida y cuál será el tipo de dato válido del resultado.

a. `v_days_to_go := v_due_date - SYSDATE;`

b. `v_sender := USER || ':' || TO_CHAR(v_dept_no);`

c. `v_sum := $100,000 + $250,000;`

d. `v_flag := TRUE;`

e. `v_n1 := v_n2 > (2 * v_n3);`

f. `v_value := NULL;`

3. Cree un bloque anónimo para hacer que aparezca la frase “My PL/SQL Block Works” en la pantalla.

G_MESSAGE
My PL/SQL Block Works

Práctica 1 (continuación)

Si le queda tiempo, realice el siguiente ejercicio:

4. Cree un bloque que declare dos variables. Asigne el valor de estas variables PL/SQL a las variables de host de *iSQL*Plus* e imprima los resultados de las variables PL/SQL en pantalla. Ejecute el bloque PL/SQL. Guarde dicho bloque PL/SQL en un archivo con el nombre `p1q4.sql`. Para ello, haga clic en el botón `Save Script`. No se olvide de guardar el archivo de comandos con la extensión `.sql`.

V_CHAR	Character (longitud variable)
V_NUM	Number

Asigne valores a estas variables de la siguiente manera:

Variable	Valor
V_CHAR	El literal '42 es la respuesta'
V_NUM	Los primeros dos caracteres de V_CHAR

G_CHAR
42 is the answer

G_NUM
42

2

Escritura de Sentencias Ejecutables

ORACLE®

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Objetivos

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

- **Describir la importancia de la sección ejecutable**
- **Utilizar correctamente los identificadores**
- **Escribir sentencias en la sección ejecutable**
- **Describir las reglas de los bloques anidados**
- **Ejecutar y probar un bloque PL/SQL**
- **Utilizar las convenciones de codificación**

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Finalidad de la Lección

En esta lección, aprenderá a escribir código ejecutable en el bloque PL/SQL. También aprenderá las reglas para anidar bloques de código PL/SQL, así como a ejecutar y probar dicho código.

Instrucciones y Sintaxis de los Bloques PL/SQL

- Las sentencias pueden ocupar varias líneas.
- Las unidades léxicas se pueden clasificar como:
 - Delimitadores
 - Identificadores
 - Literales
 - Comentarios

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Instrucciones y Sintaxis de los Bloques PL/SQL

Puesto que PL/SQL es una extensión de SQL, las reglas generales de la sintaxis de SQL también se aplican al lenguaje PL/SQL.

- Una línea de texto PL/SQL contiene unos grupos de caracteres denominados unidades léxicas, que se pueden clasificar de la siguiente manera:
 - Delimitadores (símbolos simples y compuestos)
 - Identificadores, que incluyen palabras reservadas
 - Literales
 - Comentarios
- Para facilitar la lectura, puede separar las unidades léxicas con espacios. De hecho, los identificadores adyacentes se deben separar por un espacio o un signo de puntuación.
- No se pueden embeber espacios en las unidades léxicas, excepto en el caso de los literales y los comentarios.
- Las sentencias se pueden dividir en varias líneas, pero las palabras clave no se pueden dividir.

Instrucciones y Sintaxis de los Bloques PL/SQL (continuación)

Delimitadores

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

Símbolos Simples

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

Símbolos Compuestos

Símbolo	Significado
<>	Operador relacional
!=	Operador relacional
	Operador de concatenación
--	Indicador de comentarios de una línea
/*	Delimitador del principio del comentario
*/	Delimitador del final del comentario
:=	Operador de asignación

Nota: Las palabras reservadas no se pueden utilizar como identificadores a menos que vayan entre comillas dobles (por ejemplo, "SELECT").

Identificadores

- Pueden contener hasta 30 caracteres
- Deben comenzar por un carácter alfabético
- Pueden contener números, signos de dólar, subrayados y almohadillas (o numerales)
- No pueden contener caracteres como, por ejemplo, guiones, barras inclinadas y espacios
- No deberían tener el mismo nombre que una columna de tabla de base de datos
- No deberían ser palabras reservadas

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Identificadores

Los identificadores sirven para poner nombre a elementos y unidades de programas PL/SQL como, por ejemplo, constantes, variables, excepciones, cursores, variables de cursores, subprogramas y paquetes.

- Los identificadores pueden contener hasta 30 caracteres, pero deben comenzar por un carácter alfabético.
- No ponga al identificador el mismo nombre que a las columnas de una tabla que se utiliza en el bloque. Si en las sentencias SQL hay identificadores PL/SQL que tienen el mismo nombre que una columna, Oracle da por hecho que se está haciendo referencia a la columna.
- Las palabras reservadas se deberían escribir en mayúsculas para facilitar la lectura.
- Los identificadores se componen de una letra, que opcionalmente va seguida de otras letras, números, signos de dólar, subrayados y almohadillas (o numerales). Hay otros caracteres como, por ejemplo, los guiones, las barras inclinadas y los espacios, que no son válidos, tal y como demuestran los siguientes ejemplos:

```
dots&dashes    -- ampersand no válido
debit-amount   -- guión no válido
on/off         -- barra inclinada no válida
user id        -- espacio no válido
```

money\$\$\$\$tree, SN##, try_again_ son ejemplos que demuestran que está permitido utilizar signos de dólar, subrayados y almohadillas (o numerales).

Instrucciones y Sintaxis de los Bloques PL/SQL

- **Literales**
 - Los literales de fecha y carácter deben ir entre comillas simples.

```
v_name := 'Henderson';
```
 - Los números pueden ser valores simples o notaciones científicas.
- La barra inclinada (/) se utiliza en los bloques PL/SQL de los archivos de comandos o en algunas herramientas como *iSQL*PLUS*.

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Instrucciones y Sintaxis de los Bloques PL/SQL

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

- Los literales de carácter son todos los caracteres imprimibles del conjunto 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 por una notación científica (por ejemplo, 2E5, que significa 2×10 a la potencia de 5 = 200000).

Los programas PL/SQL finalizan y se ejecutan por medio de una línea que contiene únicamente una barra inclinada (/).

Comentarios del Código

- Prefije con dos guiones (--) los comentarios de una sola línea.
- Coloque los comentarios de varias líneas entre los símbolos /* y */.

Ejemplo:

```
DECLARE
...
  v_sal NUMBER (9,2);
BEGIN
  /* Calcula el sueldo anual basándose en
    el sueldo mensual introducido por el
    usuario */
  v_sal := :g_monthly_sal * 12;
END;      -- Este es el final del bloque
```

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Comentarios del Código

Incluya comentarios en el código para documentar cada fase y para que sirvan de ayuda en el proceso de depuración. Comente el código PL/SQL con dos guiones (--) si el comentario ocupa una sola línea, o colóquelo entre los símbolos /* y */ si ocupa varias líneas. Los comentarios son estrictamente informativos y no imponen ninguna condición ni comportamiento de los datos o de la lógica. Un comentario bien colocado puede ser muy valioso para facilitar la lectura del código y su mantenimiento futuro.

Ejemplo

En el ejemplo de la transparencia, la línea situada entre /* y */ es el comentario que explica el código que hay a continuación.

Funciones SQL en PL/SQL

- **Disponibles en sentencias procedurales:**
 - Número de una fila
 - Carácter de una fila
 - Conversión de tipos de dato
 - Fecha
 - Registro de hora
 - GREATEST y LEAST
 - Otras funciones
- **No disponibles en sentencias procedurales:**
 - DECODE
 - Funciones de grupo

} Igual que en SQL

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Funciones SQL en PL/SQL

La mayoría de las funciones disponibles en SQL también son válidas en las expresiones PL/SQL:

- Funciones de números de una fila
- Funciones de caracteres de una fila
- Funciones de conversión de tipos de dato
- Funciones de fecha
- Funciones de registro de hora
- GREATEST, LEAST
- Otras funciones

Las siguientes funciones no están disponibles en las sentencias procedurales:

- DECODE.
- Funciones de grupo: AVG, MIN, MAX, COUNT, SUM, STDDEV y VARIANCE. Las funciones de grupo se aplican a los grupos de filas de una tabla y, por lo tanto, sólo están disponibles en las sentencias SQL de un bloque PL/SQL.

Funciones SQL en PL/SQL: Ejemplos

- Cree la lista de correo de una empresa.

```
v_mailing_address := v_name || CHR(10) ||  
                    v_address || CHR(10) || v_state ||  
                    CHR(10) || v_zip;
```

- Pase el nombre de un empleado a minúsculas.

```
v_ename          := LOWER(v_ename);
```

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Funciones SQL en PL/SQL: Ejemplos

En PL/SQL, se puede utilizar la mayoría de las funciones SQL. Estas funciones incorporadas ayudan a manipular los datos y pertenecen a las siguientes categorías:

- Número
- Carácter
- Conversión
- Fecha
- Otros

Los ejemplos de funciones de la transparencia se definen de la siguiente manera:

- Cree la dirección de correo de una empresa.
- Pase el nombre a minúsculas.

CHR es la función SQL que convierte un código ASCII en su carácter correspondiente; 10 es el código de un salto de línea.

PL/SQL tiene sus propias funciones de gestión de errores, que son las siguientes:

- SQLCODE
- SQLERRM (Estas funciones de gestión de errores se explicarán más adelante, en este mismo curso)

Conversión de Tipos de Dato

- Convierta los datos en tipos de dato equivalentes.
- La mezcla de tipos de dato puede producir un error y afectar al rendimiento.
- Funciones de conversión:
 - TO_CHAR
 - TO_DATE
 - TO_NUMBER

```
DECLARE
    v_date DATE := TO_DATE('12-JAN-2001', 'DD-MON-YYYY');
BEGIN
    . . .
```

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Conversión de Tipos de Dato

PL/SQL intenta convertir los tipos de dato dinámicamente si se mezclan en una sentencia. Por ejemplo, si asigna un valor NUMBER a una variable CHAR, PL/SQL traducirá dinámicamente el número en una representación de un carácter, para poderlo almacenar en la variable CHAR. También se podría producir la situación inversa, siempre que la expresión de carácter represente un valor numérico.

Si son compatibles, también puede asignar caracteres a variables DATE y viceversa.

Se debería asegurar de que los tipos de dato de una expresión son los mismos. Si se produce una mezcla de tipos de dato en la expresión, debería utilizar la función de conversión apropiada para convertir los datos.

Sintaxis

TO_CHAR (valor, fmt)

TO_DATE (valor, fmt)

TO_NUMBER (valor, fmt)

donde: *valor* es una cadena de caracteres, un número o una fecha.
 fmt es el modelo de formato que se utiliza para convertir un valor.

Conversión de Tipos de Dato

Esta sentencia produce un error de compilación si la variable `v_date` se declara como un tipo de dato `DATE`.

```
v_date := 'January 13, 2001';
```

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Conversión de Tipos de Dato (continuación)

El ejemplo de conversión de la transparencia se define de la siguiente manera:

Almacene una cadena de caracteres que represente una fecha en una variable que se declare como un tipo de dato `DATE`. *Este código produce un error de sintaxis.*

Conversión de Tipos de Dato

Para corregir el error, utilice la función de conversión TO_DATE.

```
v_date := TO_DATE ('January 13, 2001',  
                  'Month DD, YYYY');
```

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Conversión de Tipos de Dato (continuación)

El ejemplo de conversión de la transparencia para corregir el error de la transparencia anterior se define de la siguiente manera:

Para corregir el error de la transparencia anterior, convierta la cadena en una fecha con la función de conversión TO_DATE.

PL/SQL intenta hacer la conversión, si es posible, pero su éxito depende de las operaciones que se realicen. Una buena práctica de programación consiste en realizar explícitamente las conversiones de datos, dado que pueden afectar favorablemente al rendimiento y siguen siendo válidas incluso si se cambian las versiones del software.

Bloques Anidados y Ámbito de las Variables

- Los bloques PL/SQL se pueden anidar donde se permita una sentencia ejecutable.
- Un bloque anidado se convierte en una sentencia.
- Una sección de excepciones puede contener bloques anidados.
- El ámbito de un identificador es la región de una unidad de programa (bloque, subprograma o paquete) desde la cual se puede hacer referencia al identificador.

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Bloques Anidados

Una de las ventajas de PL/SQL con respecto a SQL es su capacidad de anidar sentencias. Es posible anidar bloques en los lugares en los que se permita una sentencia ejecutable, con lo que se convierte el bloque anidado en una sentencia. Por lo tanto, se puede dividir la parte ejecutable de un bloque en bloques de menor tamaño. La sección de excepciones también puede contener bloques anidados.

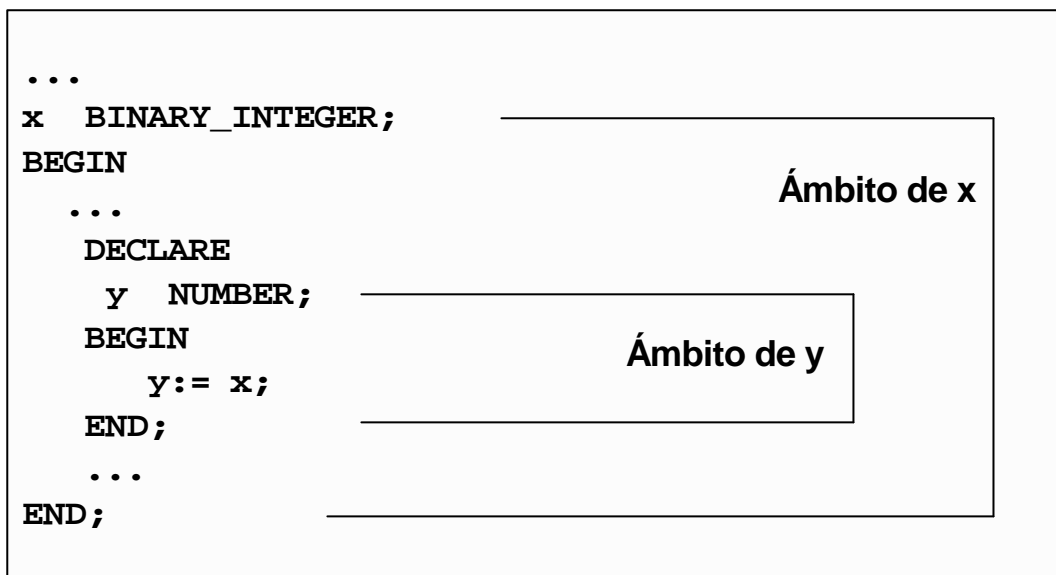
Ámbito de las Variables

Las referencias a un identificador se resuelven de acuerdo con su ámbito y su visibilidad. El ámbito de un identificador es la región de una unidad de programa (bloque, subprograma o paquete) desde la cual se puede hacer referencia al identificador. Los identificadores sólo son visibles en las regiones desde las cuales se puede hacer referencia a ellos utilizando un nombre no cualificado. Los identificadores que se declaran en un bloque PL/SQL se consideran locales para ese bloque y globales para todos sus subbloques. Si se vuelve a declarar un identificador global en un subbloque, ambos identificadores permanecen en el ámbito. En el subbloque, no obstante, sólo es visible el identificador local, porque hay que utilizar un nombre cualificado para hacer referencia al identificador global.

Aunque no es posible declarar dos veces un identificador en el mismo bloque, se puede declarar el mismo identificador en dos bloques diferentes. Los dos elementos representados por el identificador son distintos y los cambios realizados en uno no afectan al otro. Sin embargo, un bloque no puede hacer referencia a identificadores declarados en otros bloques en el mismo nivel, porque esos identificadores no son locales ni globales para el bloque.

Bloques Anidados y Ámbito de las Variables

Ejemplo:



ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Bloques Anidados y Ámbito de las Variables

En el bloque anidado de la transparencia, la variable *y* puede hacer referencia a la variable *x*. Sin embargo, la variable *x* no puede hacer referencia a la variable *y* fuera del ámbito de *y*. Si la variable *y* del bloque anidado recibe el mismo nombre que la variable *x* del bloque exterior, su valor sólo es válido el tiempo que dure el bloque anidado.

Ámbito

El ámbito de un identificador es la región de una unidad de programa (bloque, subprograma o paquete) desde la cual se puede hacer referencia al identificador.

Visibilidad

Los identificadores sólo son visibles en las regiones desde las cuales se puede hacer referencia a ellos utilizando un nombre no cualificado.

Ámbito de los Identificadores

Los identificadores son visibles en las regiones desde las que se puede hacer referencia a ellos sin tenerlos que cualificar:

- **Un bloque puede consultar el bloque en el que está contenido.**
- **Un bloque no puede consultar los bloques contenidos en su interior.**

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Ámbito de los Identificadores

Los identificadores son visibles en el bloque en el que se declaran y en todos los subbloques, procedimientos y funciones anidadas. Si un bloque no encuentra el identificador declarado de manera local, consulta la sección declarativa de los bloques en los que está contenido (bloques principales). El bloque nunca consulta los bloques contenidos en él (bloques secundarios) ni sus bloques hermanos.

El ámbito se aplica a todos los objetos declarados, ya sean variables, cursores, excepciones definidas por el usuario o constantes.

Cualificación de un Identificador

- El cualificador puede ser la etiqueta de un bloque que contiene otros bloques.
- Para cualificar un identificador, hay que utilizar el prefijo de la etiqueta del bloque.

```
<<outer>>
  DECLARE
    birthdate DATE;
  BEGIN
    DECLARE
      birthdate DATE;
    BEGIN
      ...
      outer.birthdate :=
        TO_DATE( '03-AUG-1976',
                  'DD-MON-YYYY' );
    END;
  ...
END;
```

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Cualificación de un Identificador

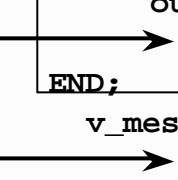
Para cualificar un identificador, hay que utilizar el prefijo de la etiqueta del bloque. En el ejemplo de la transparencia, el bloque exterior tiene la etiqueta `outer`. En el bloque interior, se declara una variable que tiene el mismo nombre, `birthdate`, que la variable del bloque exterior. Para hacer referencia a la variable `birthdate` del bloque exterior en el bloque interior, hay que prefijar la variable con el nombre del bloque, `outer.birthdate`.

Para obtener más información acerca de las etiquetas de los bloques, consulte el capítulo “Fundamentals” de *PL/SQL User’s Guide and Reference*.

Determinación del Ámbito de las Variables

Ejercicio de Clase

```
<<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
    v_message := 'CLERK not' || v_message;
    outer.v_comm := v_sal * 0.30;
  END;
  v_message := 'SALESMAN' || v_message;
END;
```



ORACLE


Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Ejercicio de Clase

Examine el bloque PL/SQL de la transparencia. Determine cada uno de los siguientes valores de acuerdo con las reglas del ámbito:

1. El valor de V_MESSAGE en la posición 1.
2. El valor de V_TOTAL_COMP en la posición 2.
3. El valor de V_COMM en la posición 1.
4. El valor de outer.V_COMM en la posición 1.
5. El valor de V_COMM en la posición 2.
6. El valor de V_MESSAGE en la posición 2.

Operadores de PL/SQL

- Lógicos
 - Aritméticos
 - Concatenaciones
 - Paréntesis para controlar el orden de las operaciones
- 
- Igual que en SQL
- Operador exponencial (**)

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Orden de las Operaciones

Las operaciones de una expresión se realizan en un orden en particular que depende de su precedencia (prioridad). La siguiente tabla muestra el orden que tienen por defecto las operaciones, desde la prioridad más alta hasta la más baja.

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

Nota: No es necesario utilizar paréntesis con las expresiones booleanas, pero facilitan la lectura del texto.

Operadores de PL/SQL

Ejemplos:

- **Incremente el contador para un bucle.**

```
v_count      := v_count + 1;
```

- **Especifique el valor de un indicador booleano.**

```
v_equal      := (v_n1 = v_n2);
```

- **Valide si el número de un empleado contiene un valor.**

```
v_valid      := (v_empno IS NOT NULL);
```

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Operadores de PL/SQL

A la hora de trabajar con valores nulos, es posible evitar algunos de los errores más habituales si se tienen en cuenta las siguientes reglas:

- Las comparaciones que implican valores nulos siempre devuelven NULL.
- Si se aplica el operador lógico NOT a un valor nulo, se devolverá NULL.
- En las sentencias de control condicional, si la condición devuelve NULL, su secuencia de sentencias asociada no se ejecuta.

Instrucciones de Programación

Facilite el mantenimiento del código haciendo lo siguiente:

- **Documente el código con comentarios**
- **Cree una convención sobre el uso de mayúsculas/minúsculas en el código**
- **Cree reglas de nomenclatura para los identificadores y otros objetos**
- **Facilite la lectura utilizando sangrados**

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Instrucciones de Programación

A la hora de desarrollar un bloque PL/SQL, siga las instrucciones de programación de la transparencia para hacer que el código sea claro y reducir su mantenimiento.

Convenciones del Código

En la siguiente tabla se enumeran las instrucciones para escribir código en mayúsculas o en minúsculas con el objetivo de ayudarle a distinguir las palabras clave de los objetos con nombre.

Categoría	Convención	Ejemplos
Sentencias SQL	Mayúscula	SELECT, INSERT
Palabras clave de PL/SQL	Mayúscula	DECLARE, BEGIN, IF
Tipos de datos	Mayúscula	VARCHAR2, BOOLEAN
Identificadores y parámetros	Minúscula	v_sal, emp_cursor, g_sal, p_empno
Tablas y columnas de base de datos	Minúscula	employees, employee_id, department_id

Sangrado del Código

Para mejorar la claridad del código, utilice sangrados en cada nivel de código.

Ejemplo:

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

```
DECLARE
  v_deptno      NUMBER(4);
  v_location_id NUMBER(4);
BEGIN
  SELECT  department_id,
          location_id
  INTO    v_deptno,
          v_location_id
  FROM    departments
  WHERE   department_name
          = 'Sales';

  ...
END;
/
```

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Sangrado del Código

Para mejorar la claridad del código y facilitar su lectura, utilice sangrados en cada nivel de código. Para que la estructura quede clara, puede dividir las líneas con retornos de carro y sangrarlas con espacios o tabuladores. Compare la legibilidad de las siguientes sentencias IF:

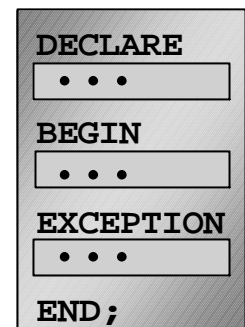
```
IF x>y THEN v_max:=x;ELSE v_max:=y;END IF;
```

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

Resumen

En esta lección, ha aprendido:

- **Las instrucciones y la sintaxis de los bloques PL/SQL**
- **Cómo utilizar correctamente los identificadores**
- **La estructura de los bloques PL/SQL: bloques anidados y reglas del ámbito**
- **Programación PL/SQL:**
 - **Funciones**
 - **Conversiones de tipos de dato**
 - **Operadores**
 - **Convenciones e instrucciones**



ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Resumen

Puesto que PL/SQL es una extensión de SQL, las reglas generales de la sintaxis de SQL también se aplican al lenguaje PL/SQL.

Los identificadores sirven para poner nombre a elementos y unidades de programas PL/SQL como, por ejemplo, constantes, variables, excepciones, cursores, variables de cursores, subprogramas y paquetes.

Los bloques pueden tener cualquier número de bloques anidados definidos en su sección ejecutable. Los bloques que están definidos en el interior de otro bloque se denominan subbloques. Los bloques sólo se pueden anidar en la sección ejecutable de un bloque.

La mayoría de las funciones disponibles en SQL también son válidas en las expresiones PL/SQL. Las funciones de conversión convierten un valor de un tipo de dato a otro. Generalmente, la forma de la función utiliza la convención *tipo de dato* TO *tipo de dato*. El primero es el tipo de dato de entrada, mientras que el segundo es el tipo de dato de salida.

Los operadores de comparación comparan dos expresiones entre sí. El resultado es siempre TRUE, FALSE o NULL. Normalmente, los operadores de comparación se utilizan en las sentencias de control condicional y en la cláusula WHERE de las sentencias de manipulación de datos SQL. Los operadores relacionales permiten comparar arbitrariamente expresiones complejas.

Las variables declaradas en iSQL*Plus se denominan variables ligadas. Para hacer referencia a estas variables en los programas PL/SQL, deben ir precedidas de dos puntos (:).

Visión General de la Práctica 2

Esta práctica cubre los siguientes temas:

- **Revisión de las reglas del ámbito y el anidamiento**
- **Desarrollo y prueba de bloques PL/SQL**

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Visión General de la Práctica 2

Esta práctica sirve para reforzar los conceptos básicos de PL/SQL que se han explicado en esta lección. En las prácticas, se utilizan bloques PL/SQL de ejemplo para poner a prueba su conocimiento de las reglas del ámbito. Además, los alumnos deberán escribir y probar bloques PL/SQL.

Preguntas sobre Papel

Las preguntas 1 y 2 se realizan sobre papel.

Práctica 2

Bloque PL/SQL

```
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;

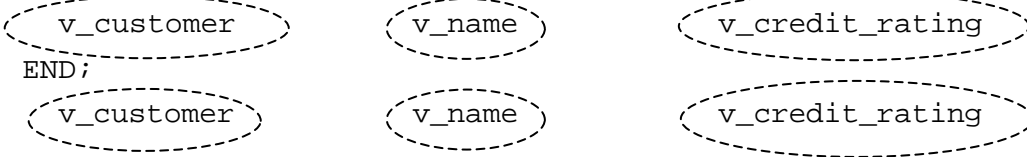
2 → END;
/

1. Examine el bloque PL/SQL anterior y determine el tipo de dato y el valor de cada una de las siguientes variables, de acuerdo con las reglas del ámbito.
 - a. El valor de V_WEIGHT en la posición 1 es:
 - b. El valor de V_NEW_LOCN en la posición 1 es:
 - c. El valor de V_WEIGHT en la posición 2 es:
 - d. El valor de V_MESSAGE en la posición 2 es:
 - e. El valor de V_NEW_LOCN en la posición 2 es:

Práctica 2 (continuación)

Ejemplo de Ámbito

```
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_customer          v_customer      v_credit_rating
  END;
    v_customer          v_name          v_credit_rating
  END;
/
```



2. Suponga que embebe un subbloque en un bloque, tal y como se ha mostrado anteriormente. Declara dos variables, V_CUSTOMER y V_CREDIT_RATING, en el bloque principal. También declara dos variables, V_CUSTOMER y V_NAME, en el subbloque. Determine los valores y los tipos de dato para cada uno de los siguientes casos:
 - a. El valor de V_CUSTOMER en el subbloque es:
 - b. El valor de V_NAME en el subbloque es:
 - c. El valor de V_CREDIT_RATING en el subbloque es:
 - d. El valor de V_CUSTOMER en el bloque principal es:
 - e. El valor de V_NAME en el bloque principal es:
 - f. El valor de V_CREDIT_RATING en el bloque principal es:

Práctica 2 (continuación)

3. Cree y ejecute un bloque PL/SQL que acepte dos números mediante variables de sustitución de *iSQL*Plus*.
 - a. Utilice el comando DEFINE para proporcionar dos valores.

```
DEFINE p_num1 = 2
DEFINE p_num2 = 4
```
 - b. Transfiera los dos valores definidos en el paso anterior al bloque PL/SQL mediante variables de sustitución *iSQL*Plus*. El primer número se debe dividir por el segundo número y hay que sumar el segundo número al resultado. Luego, hay que guardar el resultado en una variable PL/SQL e imprimirlo en pantalla.

Nota: Especifique SET VERIFY OFF en el bloque PL/SQL.

4.5

PL/SQL procedure successfully completed.

4. Cree un bloque PL/SQL que calcule la compensación total de un año.
 - a. Los valores del sueldo anual y del porcentaje de bonificación anual se definen con el comando DEFINE.
 - b. Transfiera los dos valores definidos en el paso anterior al bloque PL/SQL mediante variables de sustitución *iSQL*Plus*. Hay que convertir la bonificación de un número entero a un decimal (por ejemplo, de 15 a .15). Si el sueldo es null, defínalo en cero antes de calcular la compensación total. Ejecute el bloque PL/SQL. *Recordatorio:* Utilice la función NVL para gestionar valores null.

Nota: La compensación total es la suma del sueldo anual y la bonificación anual.

Para probar la función NVL, defina la variable DEFINE en igual a NULL.

```
DEFINE p_salary = 50000
DEFINE p_bonus = 10
```

PL/SQL procedure successfully completed.

G_TOTAL	
	55000

3

Interacción con Oracle Server

ORACLE®

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Objetivos

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

- **Escribir una sentencia `SELECT` correcta en PL/SQL**
- **Escribir sentencias `DML` en PL/SQL**
- **Controlar las transacciones en PL/SQL**
- **Determinar el resultado de las sentencias `SQL DML` (Lenguaje de Manipulación de Datos)**

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Finalidad de la Lección

En esta lección, aprenderá a embeber las sentencias SQL estándar `SELECT`, `INSERT`, `UPDATE` y `DELETE` en bloques PL/SQL. También aprenderá a controlar las transacciones y a determinar el resultado en PL/SQL de las sentencias `SQL DML` (Lenguaje de Manipulación de Datos).

Sentencias SQL en PL/SQL

- **Extraiga una fila de datos de la base de datos utilizando el comando `SELECT`.**
- **Realice cambios en las filas de la base de datos utilizando comandos `DML`.**
- **Controle una transacción con los comandos `COMMIT`, `ROLLBACK` o `SAVEPOINT`.**
- **Determine el resultado `DML` con atributos de cursores implícitos.**

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Sentencias SQL en PL/SQL

Para extraer información desde la base de datos o aplicar cambios en ella, hay que utilizar el lenguaje SQL. El lenguaje PL/SQL es compatible con el lenguaje de manipulación de datos y con los comandos de control de transacciones de SQL. Se pueden utilizar sentencias `SELECT` para rellenar las variables con los valores que se han consultado en una fila de una tabla. Se pueden utilizar comandos `DML` para modificar los datos de una tabla de base de datos. Sin embargo, no se olvide de seguir estas reglas de los bloques PL/SQL al utilizar las sentencias `DML` y los comandos de control de transacciones en los bloques PL/SQL.

- La palabra clave `END` señala el final de un bloque PL/SQL, no el final de una transacción. Al igual que un bloque puede abarcar múltiples transacciones, una transacción puede abarcar múltiples bloques.
- PL/SQL no admite directamente sentencias `DDL` (Lenguaje de Definición de Datos) como, por ejemplo, `CREATE TABLE`, `ALTER TABLE` o `DROP TABLE`.
- PL/SQL no admite sentencias `DCL` (Lenguaje de Control de Datos) como, por ejemplo, `GRANT` o `REVOKE`.

Sentencias SELECT en PL/SQL

Recupere datos de la base de datos con una sentencia SELECT.

Sintaxis:

```
SELECT  lista_select
INTO    {nombre_variable[, nombre_variable]...
        | nombre_registro}
FROM    tabla
[WHERE  condición];
```

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Recuperación de Datos con PL/SQL

Utilice la sentencia SELECT para recuperar datos de la base de datos. En la sintaxis:

lista_select es una lista de al menos una columna que puede incluir expresiones SQL, funciones de filas o funciones de grupos.

nombre_variable es la variable escalar que guarda el valor recuperado.

nombre_registro es el registro (RECORD) PL/SQL que guarda los valores recuperados.

tabla especifica el nombre de la tabla de base de datos.

condición se compone de nombres de columnas, expresiones, constantes y operadores de comparación, incluidos variables y constantes PL/SQL.

Instrucciones para Recuperar Datos en PL/SQL

- Termine cada sentencia SQL con un punto y coma (;).
- Es necesario utilizar la cláusula INTO para la sentencia SELECT cuando está embebida en PL/SQL.
- La cláusula WHERE es opcional y se puede utilizar para especificar variables de entrada, constantes, literales o expresiones PL/SQL.

Recuperación de Datos con PL/SQL (continuación)

- Especifique el mismo número de variables en la cláusula `INTO` que columnas de base de datos de la cláusula `SELECT`. Asegúrese de que sus posiciones se corresponden y de que sus tipos de dato son compatibles.
- Utilice funciones de grupos como, por ejemplo, `SUM`, en las sentencias SQL, porque este tipo de funciones se aplica a los grupos de filas de una tabla.

Sentencias `SELECT` en PL/SQL

- Es necesario utilizar la cláusula `INTO`.
- Las consultas deben devolver sólo una fila.

Ejemplo:

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

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Sentencias `SELECT` en PL/SQL

Cláusula `INTO`

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

Utilice la cláusula `INTO` para rellenar variables PL/SQL o variables del host.

Las Consultas Deben Devolver Sólo una Fila

Las sentencias `SELECT` de los bloques PL/SQL se engloban en la categoría ANSI de SQL embebido, para las que se aplica la siguiente regla: las consultas deben devolver sólo una fila. Si las consultas devuelven más de una fila o ninguna, se generará un error.

Para gestionar estos errores, PL/SQL emite excepciones estándar, que se pueden interrumpir en la sección de excepciones del bloque con las excepciones `NO_DATA_FOUND` y `TOO_MANY_ROWS` (la gestión de excepciones se cubre en una lección posterior). Codifique sentencias `SELECT` que devuelvan sólo una fila.

Recuperación de Datos en PL/SQL

Recupere la fecha de contratación y el sueldo del empleado especificado.

Ejemplo:

```
DECLARE
  v_hire_date    employees.hire_date%TYPE;
  v_salary       employees.salary%TYPE;
BEGIN
  SELECT    hire_date, salary
  INTO      v_hire_date, v_salary
  FROM employees
  WHERE     employee_id = 100;
  ...
END;
/
```

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Recuperación de Datos en PL/SQL

En el ejemplo de la transparencia, las variables `v_hire_date` y `v_salary` se declaran en la sección `DECLARE` del bloque PL/SQL. En la sección ejecutable, se recuperan los valores de las columnas `HIRE_DATE` y `SALARY` del empleado `EMPLOYEE_ID 100` de la tabla `EMPLOYEES` y se almacenan en las variables `v_hire_date` y `v_salary`, respectivamente. Observe cómo la cláusula `INTO`, junto con la sentencia `SELECT`, recupera los valores de la columna de base de datos en las variables PL/SQL.

Recuperación de Datos en PL/SQL

Obtenga la suma de los sueldos de todos los empleados del departamento especificado.

Ejemplo:

```
SET SERVEROUTPUT ON
DECLARE
  v_sum_sal    NUMBER(10,2);
  v_deptno     NUMBER NOT NULL := 60;
BEGIN
  SELECT      SUM(salary)  -- función de grupo
  INTO        v_sum_sal
  FROM        employees
  WHERE       department_id = v_deptno;
  DBMS_OUTPUT.PUT_LINE ('The sum salary is ' ||
                        TO_CHAR(v_sum_sal));
END;
/
```

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Recuperación de Datos en PL/SQL

En el ejemplo de la transparencia, las variables `v_sum_sal` y `v_deptno` se declaran en la sección `DECLARE` del bloque PL/SQL. En la sección ejecutable, se calcula el sueldo total del departamento `DEPARTMENT_ID` 60 utilizando la función SQL de agregación `SUM` y luego se asigna a la variable `v_sum_sal`. Tenga en cuenta que en la sintaxis PL/SQL no se pueden utilizar funciones de grupos. Estas funciones se utilizan en las sentencias SQL dentro de un bloque PL/SQL.

Este es el resultado del bloque PL/SQL de la transparencia:

```
The sum salary is 28800
PL/SQL procedure successfully completed.
```

Reglas de Nomenclatura

```
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;
/
```

```
DECLARE
*
ERROR at line 1:
ORA-01422: exact fetch returns more than requested number of rows
ORA-06512: at line 6
```

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Reglas de Nomenclatura

En las sentencias SQL potencialmente ambiguas, los nombres de las columnas de base de datos tienen precedencia con respecto a los nombres de las variables locales. El ejemplo de la transparencia se define de la siguiente manera: Recuperar la fecha de contratación y la fecha de hoy de la tabla EMPLOYEES del empleado EMPLOYEE_ID 176. Este ejemplo emite una excepción no tratada en tiempo de ejecución, porque en la cláusula WHERE las variables PL/SQL tienen el mismo nombre que las columnas de base de datos de la tabla EMPLOYEES.

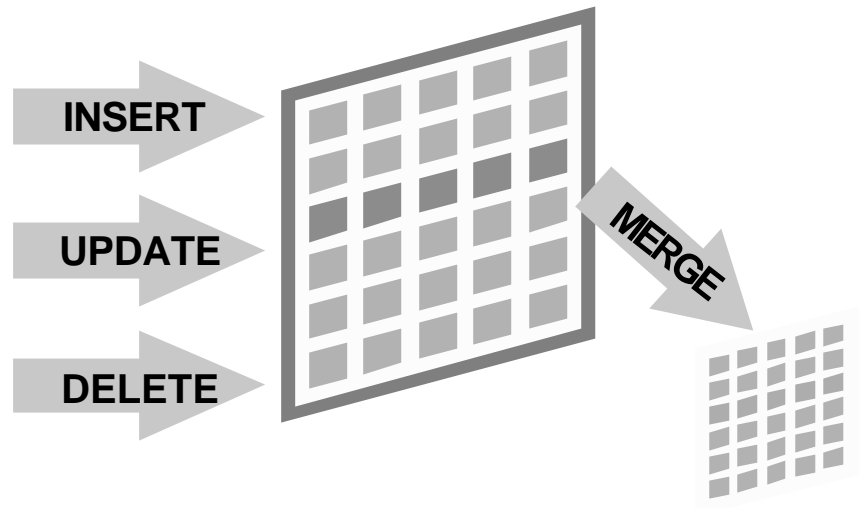
La siguiente cláusula DELETE suprime a todos los empleados de la tabla EMPLOYEES cuyo apellido no es nulo, y no sólo 'King', porque Oracle Server da por hecho que los dos LAST_NAME de la cláusula WHERE se refieren a la columna de base de datos:

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

Manipulación de Datos con PL/SQL

Realice cambios en las tablas de base de datos utilizando comandos DML:

- INSERT
- UPDATE
- DELETE
- MERGE



ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Manipulación de Datos con PL/SQL

Los datos de la base de datos se manipulan con los comandos DML. En PL/SQL, se pueden emitir los comandos DML INSERT, UPDATE, DELETE y MERGE sin ninguna restricción. Para liberar los bloqueos de filas (y de tablas), hay que incluir sentencias COMMIT o ROLLBACK en el código PL/SQL.

- La sentencia INSERT agrega nuevas filas de datos a la tabla.
- La sentencia UPDATE modifica las filas ya existentes de la tabla.
- La sentencia DELETE elimina las filas no deseadas de la tabla.
- La sentencia MERGE selecciona filas de una tabla para actualizarlas o insertarlas en otra tabla. La decisión de actualizarlas o insertarlas en la tabla destino se basa en una condición de la cláusula ON.

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

Inserción de Datos

Agregue la nueva información del empleado a la tabla EMPLOYEES.

Ejemplo:

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

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Inserción de Datos

En el ejemplo de la transparencia, se utiliza una sentencia INSERT dentro de un bloque PL/SQL para insertar un registro en la tabla EMPLOYEES. Si utiliza el comando INSERT en el bloque PL/SQL, puede:

- Utilizar funciones SQL como, por ejemplo, USER y SYSDATE
- Generar valores de clave primaria utilizando secuencias de base de datos
- Derivar valores en el bloque PL/SQL
- Agregar valores por defecto de las columnas

Nota: No hay ninguna posibilidad de que se produzcan ambigüedades con los nombres de las columnas y los identificadores en la sentencia INSERT. Cualquier identificador de la cláusula INSERT debe ser un nombre de columna de la base de datos.

Actualización de Datos

Aumente el sueldo a todos los empleados que tengan el puesto de administrativos.

Ejemplo:

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

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Actualización de Datos

Se podría producir una ambigüedad en la cláusula SET de la sentencia UPDATE porque, aunque el identificador situado a la izquierda del operador de asignación es siempre una columna de base de datos, el identificador situado a la derecha puede ser una columna de base de datos o una variable PL/SQL.

Recuerde que la cláusula WHERE sirve para determinar qué filas se ven afectadas. Si no se modifica ninguna fila, no se produce ningún error, a diferencia de lo que ocurre con la sentencia SELECT en PL/SQL.

Nota: Las asignaciones de variables PL/SQL siempre utilizan :=, y las asignaciones de columnas SQL siempre usan =.

Recuerde que si en la cláusula WHERE los nombres de las columnas y los identificadores son idénticos, Oracle Server busca primero el nombre en la base de datos.

Supresión de Datos

Suprima las filas que pertenecen al departamento 10 de la tabla EMPLOYEES.

Ejemplo:

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

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Supresión de Datos

La sentencia DELETE sirve para eliminar las filas no deseadas de una tabla. Si no se utilizara la cláusula WHERE, se podría eliminar todo el contenido de la tabla, siempre y cuando no hubiera restricciones de integridad.

Fusión de Filas

Inserte o actualice las filas de la tabla COPY_EMP para que coincida con la tabla EMPLOYEES.

```
DECLARE
    v_empno employees.employee_id%TYPE := 100;
BEGIN
MERGE INTO copy_emp c
    USING employees e
    ON (e.employee_id = v_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 © Oracle Corporation, 2002. Todos los Derechos Reservados.

Fusión de Filas

La sentencia MERGE inserta o actualiza las filas de una tabla utilizando los datos de otra tabla. Cada una de las filas se inserta o se actualiza en la tabla destino, dependiendo de una condición de unión de igualdad.

Este ejemplo hace coincidir employee_id de la tabla COPY_EMP con employee_id de la tabla EMPLOYEES. Si encuentra una coincidencia, la fila se actualiza para que sea igual que la de la tabla EMPLOYEES. Si no encuentra la fila, la inserta en la tabla COPY_EMP.

El ejemplo completo para utilizar MERGE en un bloque PL/SQL se encuentra en la siguiente página.

Fusión de Datos

```
DECLARE
    v_empno EMPLOYEES.EMPLOYEE_ID%TYPE := 100;
BEGIN
MERGE INTO copy_emp c
    USING employees e
    ON (e.employee_id = v_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;
/
```

Reglas de Nomenclatura

- **Utilice una regla de nomenclatura para evitar ambigüedades en la cláusula `WHERE`.**
- **Los identificadores y las columnas de base de datos deberían tener nombres diferentes.**
- **Se pueden producir errores de sintaxis, porque PL/SQL comprueba primero en la base de datos la columna de una tabla.**
- **Los nombres de las variables locales y los parámetros formales tienen precedencia con respecto a los nombres de las tablas de base de datos.**
- **Los nombres de las columnas de tablas de base de datos tienen precedencia con respecto a los nombres de las variables locales.**

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Reglas de Nomenclatura

Evite las ambigüedades en la cláusula `WHERE` utilizando una regla de nomenclatura que distinga los nombres de las columnas de base de datos de los nombres de las variables PL/SQL.

- Los identificadores y las columnas de base de datos deberían tener nombres diferentes.
- Se pueden producir errores de sintaxis, porque PL/SQL comprueba primero en la base de datos la columna de una tabla.

Reglas de Nomenclatura (continuación)

La siguiente tabla muestra un conjunto de prefijos y sufijos que distinguen los identificadores de otros identificadores, de objetos de base de datos y de otros objetos con nombre.

Identificador	Regla de Nomenclatura	Ejemplo
Variable	v_nombre	v_sal
Constante	c_nombre	c_nombre_de_compañía
Cursor	nombre_cursor	emp_cursor
Excepción	e_nombre	e_demasiados
Tipo de tabla	nombre_table_type	cantidad_table_type
Tabla	nombre_tabla	países
Tipo de registro	nombre_record_type	emp_record_type
Registro	nombre_record	cliente_record
Variable de sustitución de iSQL*Plus (también llamada parámetro de sustitución)	p_nombre	p_sal
Variable del host o ligada de iSQL*Plus	g_nombre	g_salario_anual

En tales casos, para evitar ambigüedades, prefije los nombres de las variables locales y los parámetros formales con `v_`, de la siguiente manera:

```
DECLARE  
    v_last_name VARCHAR2(25);
```

Nota: No existe la posibilidad de que se produzcan ambigüedades en la cláusula `SELECT`, porque cualquier identificador de la cláusula `SELECT` debe ser un nombre de columna de base de datos. Tampoco hay posibilidad de que se produzcan ambigüedades en la cláusula `INTO`, porque los identificadores de la cláusula `INTO` deben ser variables PL/SQL. Sólo hay posibilidades de que se produzcan confusiones en la cláusula `WHERE`.

Cursores SQL

- Un cursor es un área de trabajo SQL privada.
- Existen dos tipos de cursores:
 - Cursores implícitos
 - Cursores explícitos
- Oracle Server utiliza los cursores implícitos para analizar y ejecutar las sentencias SQL.
- El programador declara explícitamente los cursores explícitos.

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Cursores SQL

Siempre que se emite una sentencia SQL, Oracle Server abre un área de memoria en la que se analiza y se ejecuta el comando. Esta área se denomina cursor.

Cuando la parte ejecutable de un bloque emite una sentencia SQL, PL/SQL crea un cursor implícito, que PL/SQL gestiona automáticamente. El programador declara y pone un nombre explícitamente al cursor explícito. Existen cuatro atributos disponibles en PL/SQL que se pueden aplicar a los cursores.

Nota: En una lección posterior, encontrará más información acerca de los cursores explícitos.

Para obtener más información, consulte el capítulo “Interaction with Oracle” de *PL/SQL User’s Guide and Reference*.

Atributos de los Cursores SQL

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

SQL%ROWCOUNT	Número de filas afectadas por la sentencia SQL más reciente (un valor entero)
SQL%FOUND	Atributo booleano que se evalúa como TRUE si la sentencia SQL más reciente afecta a una o varias filas
SQL%NOTFOUND	Atributo booleano que se evalúa como TRUE si la sentencia SQL más reciente no afecta a ninguna fila
SQL%ISOPEN	Siempre se evalúa como FALSE porque PL/SQL cierra los cursores implícitos inmediatamente después de ejecutarlos

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Atributos de los Cursores SQL

Los atributos de los cursores SQL le permiten evaluar qué ocurrió la última vez que se utilizó un cursor implícito. Utilice estos atributos en las sentencias PL/SQL, pero no en las sentencias SQL.

Puede utilizar los atributos SQL%ROWCOUNT, SQL%FOUND, SQL%NOTFOUND y SQL%ISOPEN en la sección de excepciones de un bloque para conseguir información acerca de la ejecución de una sentencia DML. PL/SQL no devuelve un error si una sentencia DML no afecta a ninguna fila de la tabla subyacente. Sin embargo, si una sentencia SELECT no recupera ninguna fila, PL/SQL devuelve una excepción.

Atributos de los Cursores SQL

Suprima las filas que tengan el número de identificación de empleado especificado de la tabla EMPLOYEES. Imprima el número de filas suprimidas.

Ejemplo:

```
VARIABLE rows_deleted VARCHAR2(30)
DECLARE
  v_employee_id employees.employee_id%TYPE := 176;
BEGIN
  DELETE FROM employees
  WHERE      employee_id = v_employee_id;
  :rows_deleted := (SQL%ROWCOUNT ||
                    ' row deleted.');
```

END;
/
PRINT rows_deleted

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Atributos de los Cursores SQL (continuación)

El ejemplo de la transparencia suprime las filas del empleado EMPLOYEE_ID 176 de la tabla EMPLOYEES. Con el atributo SQL%ROWCOUNT, puede imprimir el número de filas suprimidas.

Sentencias de Control de Transacciones

- **Inicie una transacción con el primer comando DML para seguir una sentencia COMMIT o ROLLBACK.**
- **Utilice las sentencias SQL COMMIT y ROLLBACK para terminar explícitamente una transacción.**

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Sentencias de Control de Transacciones

Las sentencias SQL COMMIT y ROLLBACK sirven para controlar la lógica de las transacciones, haciendo que algunos grupos de cambios de la base de datos sean permanentes, mientras que otros se descartan. Al igual que en Oracle Server, las transacciones DML se inician en el primer comando que sigue a una sentencia COMMIT o ROLLBACK y terminan en la siguiente sentencia COMMIT o ROLLBACK realizada con éxito. Estas acciones se pueden producir en el interior de un bloque PL/SQL o como resultado de los eventos del entorno host (por ejemplo, en la mayoría de los casos, si se termina una sesión de iSQL*Plus automáticamente, se valida la transacción pendiente). Para marcar un punto intermedio en el procesamiento de la transacción, utilice SAVEPOINT.

```
COMMIT [WORK];
```

```
SAVEPOINT nombre_puntograbación;
```

```
ROLLBACK [WORK];
```

```
ROLLBACK [WORK] TO [SAVEPOINT] nombre_puntograbación;
```

donde: WORK sirve para cumplir los estándares ANSI.

Nota: Todos los comandos de control de transacciones son válidos en PL/SQL, aunque el entorno host puede establecer alguna restricción sobre su uso.

También puede incluir en un bloque comandos de bloqueo explícitos (como LOCK TABLE y SELECT . . . FOR UPDATE), que siguen siendo efectivos hasta el final de la transacción (en una lección posterior se dará más información acerca del comando FOR UPDATE). Además, un bloque PL/SQL no implica necesariamente una transacción.

Resumen

En esta lección, ha aprendido a:

- **Embeber SQL en el bloque PL/SQL utilizando `SELECT`, `INSERT`, `UPDATE`, `DELETE` y `MERGE`**
- **Embeber sentencias de control de transacciones en un bloque PL/SQL mediante `COMMIT`, `ROLLBACK` y `SAVEPOINT`**

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Resumen

En los programas PL/SQL, se pueden utilizar los comandos DML `INSERT`, `UPDATE`, `DELETE` y `MERGE` sin ninguna restricción. La sentencia `COMMIT` finaliza la transacción actual y hace permanente cualquier cambio realizado durante esa transacción. La sentencia `ROLLBACK` finaliza la transacción actual y cancela cualquier cambio realizado durante esa transacción. `SAVEPOINT` pone un nombre y marca el punto actual del procesamiento de una transacción. Con la sentencia `ROLLBACK TO SAVEPOINT`, puede deshacer partes de una transacción, en lugar de la transacción entera.

Resumen

En esta lección, ha aprendido que:

- **Hay dos tipos de cursores: implícitos y explícitos.**
- **Los atributos de los cursores implícitos sirven para verificar el resultado de las sentencias DML:**
 - `SQL%ROWCOUNT`
 - `SQL%FOUND`
 - `SQL%NOTFOUND`
 - `SQL%ISOPEN`
- **El programador es quien define los cursores explícitos.**

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Resumen (continuación)

PL/SQL declara un cursor implícito por cada sentencia de manipulación de datos SQL. Cada cursor implícito posee cuatro atributos: `%FOUND`, `%ISOPEN`, `%NOTFOUND` y `%ROWCOUNT`. Cuando se agregan al cursor o variable de cursor, estos atributos devuelven información muy útil acerca de la ejecución de una sentencia DML. Los atributos de cursores se pueden utilizar en las sentencias procedurales, pero no en las sentencias SQL. El programador es quien define los cursores explícitos.

Visión General de la Práctica 3

Esta práctica trata sobre la creación de un bloque PL/SQL para:

- **Seleccionar datos de una tabla**
- **Insertar datos en una tabla**
- **Actualizar datos en una tabla**
- **Suprimir un registro de una tabla**

ORACLE

Copyright © Oracle Corporation, 2002. Todos los Derechos Reservados.

Visión General de la Práctica 3

En esta práctica, tiene que escribir bloques PL/SQL para seleccionar, introducir, actualizar y suprimir información de una tabla utilizando sentencias de consulta SQL y DML básicas en un bloque PL/SQL.

Práctica 3

1. Cree un bloque PL/SQL que seleccione el número de departamento más alto en la tabla DEPARTMENTS y lo almacene en una variable de iSQL*Plus. Imprima los resultados en pantalla. Guarde el bloque PL/SQL en un archivo con el nombre p3q1.sql. Para ello, haga clic en el botón Save Script. Guarde el archivo de comandos con la extensión .sql.

G_MAX_DEPTNO	
	270

2. Modifique el bloque PL/SQL que ha creado en el ejercicio 1 para insertar un nuevo departamento en la tabla DEPARTMENTS. Guarde el bloque PL/SQL en un archivo con el nombre p3q2.sql haciendo clic en el botón Save Script. Guarde el archivo de comandos con la extensión .sql.
 - a. Utilice el comando DEFINE para proporcionar el nombre del departamento. Llame al nuevo departamento Education.
 - b. Transfiera el valor definido para el nombre del departamento al bloque PL/SQL mediante una variable de sustitución de iSQL*Plus. En lugar de imprimir el número de departamento que recuperó en el ejercicio 1, súmele 10 y utilícelo como número de departamento del departamento nuevo.
 - c. Por ahora, deje el número de la ubicación como nulo.
 - d. Ejecute el bloque PL/SQL.
 - e. Muestre el nuevo departamento que ha creado.

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
280	Education		

3. Cree un bloque PL/SQL que actualice el número de identificación de ubicación del nuevo departamento que agregó en la práctica anterior. Guarde el bloque PL/SQL en un archivo con el nombre p3q3.sql haciendo clic en el botón Save Script. Guarde el archivo de comandos con la extensión .sql.
 - a. Utilice una variable de iSQL*Plus para el número de identificación del departamento que agregó en la práctica anterior.
 - b. Utilice el comando DEFINE para proporcionar el número de identificación de la ubicación. Llame a este nuevo número de identificación 1700.

```
DEFINE p_deptno = 280
DEFINE p_loc = 1700
```
 - c. Transfiera el valor al bloque PL/SQL mediante una variable de sustitución de iSQL*Plus. Pruebe el bloque PL/SQL.
 - d. Muestre el nuevo departamento que ha actualizado.

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
280	Education		1700

Práctica 3 (continuación)

4. Cree un bloque PL/SQL que suprima el departamento que creó en el ejercicio 2. Luego, guarde el bloque PL/SQL en un archivo llamado `p3q4.sql` haciendo clic en el botón `Save Script`. Guarde el archivo de comandos con la extensión `.sql`.
 - a. Utilice el comando `DEFINE` para proporcionar el número de identificación del departamento.
`DEFINE p_deptno=280`
 - b. Transfiera el valor al bloque PL/SQL mediante una variable de sustitución de `iSQL*Plus`. Imprima en pantalla el número de filas afectadas.
 - c. Pruebe el bloque PL/SQL.

G_RESULT
1 row(s) deleted.

- d. Confirme que el departamento se ha suprimido.

no rows selected