Ediciones ENI

Oracle 12c
SQL, PL/SQL, SQL*Plus

Colección
Recursos Informáticos

Extracto del Libro

Capítulo 5
PL/SQL en objetos de la base de datos

1. Introducción

Además de los bloques PL/SQL anónimos utilizados por SQL*Plus o por las herramientas de desarrollo (Oracle*Forms, Oracle*Reports...), se puede emplear código PL/SQL en determinados objetos de la base de datos, como los procedimientos almacenados (PROCEDURE, FUNCTION, PACKAGE) y los triggers de base de datos.

2. Los triggers de bases de datos

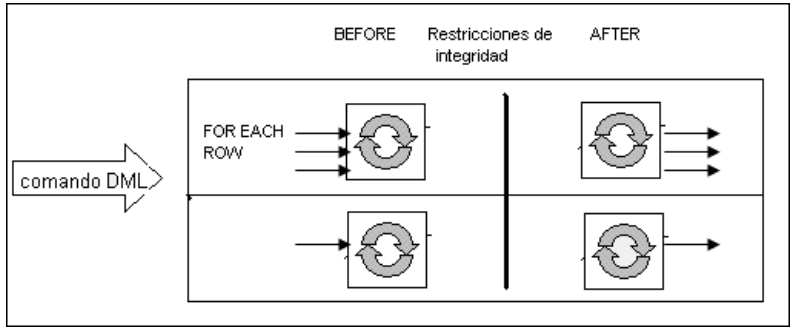
Un trigger es un bloque PL/SQL asociado a una tabla. Este bloque se ejecutará cuando se aplique a la tabla una instrucción DML (INSERT, UPDATE, DELETE).

El bloque PL/SQL que constituye el trigger puede ejecutarse antes o después de la actualización y, por lo tanto, antes o despues de la verificación de las restricciones de integridad.

Los triggers ofrecen una solución procedimental para definir restricciones complejas o que tengan en cuenta datos procedentes de varias filas o de varias tablas, como por ejemplo para garantizar el hecho de que un cliente no pueda tener más de dos pedidos no pagados. Sin embargo, los triggers no deben emplearse cuando sea posible establecer una restricción de integridad.

En efecto, las restricciones de integridad se definen en el nivel de tabla y forman parte de la estructura de la propia tabla, por lo que la verificación de estas restricciones es mucho más rápida. Además, las restricciones de integridad garantizan que todas las filas de las tablas respetan dichas restricciones, mientras que los triggers no tienen en cuenta los datos ya contenidos en la tabla en el momento de definirlos.

El bloque PL/SQL asociado a un trigger se puede ejecutar para cada fila afectada por la instrucción DML (opción FOR EACH ROW), o una única vez para cada instrucción DML ejecutada (opción predeterminada).



Ejecución antes o después de la comprobación de las restricciones de integridad para cada fila o cada sentencia

En los triggers BEFORE y FOR EACH ROW se pueden modificar los datos que van a insertarse en la tabla, de modo que respeten las restricciones de integridad. También es posible ejecutar consultas de tipo SELECT sobre la tabla a la que se aplica la instrucción DML, aunque únicamente en el marco de un trigger BEFORE INSERT. Todas estas operaciones no se pueden realizar en los triggers AFTER, ya que después de la verificación de las restricciones de integridad no es posible modificar los datos y, dado que la modificación (adición o eliminación) de la fila no se ha terminado, no es posible ejecutar consultas de tipo SELECT sobre la tabla.

También se pueden incluir triggers en las vistas (VIEW) con el fin de capturar las instrucciones DML que se pueden ejecutar sobre ellas. Estos triggers permiten controlar todas las operaciones realizadas sobre las vistas y, para el usuario final, la vista es completamente similar a una tabla, ya que puede realizar sobre ella operaciones INSERT, UPDATE y DELETE. Estos triggers son de tipo INSTEAD OF, es decir, que su ejecución va a reemplazar a la instrucción DML a la que estén asociados. Este tipo de trigger solo puede definirse en vistas y es el único tipo de trigger que puede implementarse en ellas.

Principio de funcionamiento de los triggers instead of

Sintaxis

```
CREATE [OR REPLACE] TRIGGER nombre_trigger
{BEFORE/AFTER/INSTEAD OF}
{INSERT/UPDATE[OF col,...]/DELETE}
ON Nombre_tabla [FOR EACH ROW]
[FOLLOWS nombre_otro_trigger[,...]]
[ENABLE/DISABLE]
[WHEN (condition)]
Bloc PL/SQLv
```

`OR REPLACE`

Reemplaza la descripción del trigger si ya existe.

`BEFORE`

El bloque PL/SQL se ejecuta antes de la verificación de las restricciones de tabla y de actualizar los datos almacenados en la misma.

`AFTER`

El bloque PL/SQL se ejecuta después de la actualización de los datos contenidos en la tabla.

`INSTEAD OF`

El bloque PL/SQL siguiente reemplaza el procesamiento estándar asociado a la instrucción que ha activado al trigger (solo por una vista).

`INSERT/UPDATE [OF col,...]/DELETE`

Instrucción asociada a la activación del trigger. Varias instrucciones pueden activar un mismo trigger y se combinan mediante el operador OR.

`FOR EACH ROW`

El trigger se ejecuta para cada fila tratada por la instrucción asociada.

`FOLLOWS nombre_otro_trigger[,...]`

Oracle permite definir varios triggers para la misma tabla y el mismo evento. En este caso, el orden relativo de ejecución de estos triggers es indeterminado. Si el orden de ejecución de estos triggers es importante en su aplicación, puede utilizar la cláusula FOLLOWS disponible a partir de la versión 11. Esta cláusula permite indicar que el trigger debe ejecutarse después de los triggers enumerados.

`ENABLE/DISABLE`

Esta cláusula permite indicar si el trigger está o no activo desde el momento de su creación; por defecto, un trigger de nueva creación está activo. Crear un trigger desactivado permite verificar que se compila correctamente antes de ponerlo realmente en servicio. Un trigger creado desactivado puede activarse más adelante utilizando una sentencia ALTER TRIGGER...ENABLE.

`WHEN (condición)`

La condición especificada debe cumplirse para que se ejecute el código.

Los datos de la tabla a la que está asociado el trigger son inaccesibles desde las instrucciones del bloque. Solo la fila que se está modificando es accesible a través de dos variables de tipo RECORD: OLD y NEW, las cuales poseen la estructura de la tabla o de la vista asociada. Estas variables pueden utilizarse en la cláusula WHEN del trigger o en el bloque de instrucciones. En este último caso se referencian como variables host mediante el prefijo ":" (:OLD.nombre_campo, :NEW.nombre_campo).

La palabra OLD permite conocer qué fila se va a eliminar en un trigger DELETE o la fila que se va a modificar en un trigger UPDATE. La palabra NEW permite conocer cuál es la nueva fila insertada en un trigger INSERT o la fila tras su modificación en un trigger UPDATE.

Los nombres OLD y NEW están definidos de manera predeterminada, aunque es posible utilizar otros nombres empleando la cláusula `REFERENCING OLD AS nuevo_nombre NEW AS nuevo_nombre`. Esta cláusula se incluye justo antes de la cláusula FOR EACH ROW (si existe) en la definición del trigger.

Ejemplo

Ejecución de un bloque PL/SQL antes de una eliminación en la tabla CLIENTES por parte del usuario FLORENCIO:

```
CREATE TRIGGER antes_elim_cli
      BEFORE    DELETE
      ON FLORENCIO.CLIENTES
      DECLARE
         ...
      BEGIN
         ...
      END;
```

Ejecución de un bloque PL/SQL después de actualizar cada fila de la tabla ARTICULOS cuando el precio antiguo es mayor que el nuevo:

```
create or replace trigger post_actprecio
  after update of precio
  on articulos
  for each row
  when (old.precio > new.precio)
declare
  ...
begin
  ...
end;
```

Para cada pedido, se desea conocer el nombre del usuario de Oracle que lo ha introducido. La primera etapa consiste en añadir una nueva columna a la tabla de pedidos. Esta columna debe aceptar el valor NULL ya que, para las filas de los pedidos existentes, el nombre del usuario de Oracle es desconocido.

Modificación de la tabla PEDIDOS:

```
SQL> alter table pedidos
  2    add (usuario varchar2(30));

Tabla modificada.

SQL>
```

En el trigger se cambia el nombre de la nueva fila, y el trigger se ejecuta antes de la verificación de las restricciones de integridad para cada fila insertada en la tabla de pedidos.

Definición del trigger:

```
SQL> create or replace trigger bf_ins_pedidos
  2     before insert
  3     on pedidos
  4     referencing new as nuevo
  5     for each row
  6  begin
  7     select user into :nuevo.usuario from dual;
  8  end;
  9  /

Trigger creado.

SQL>
```

Se desea saber el número de pedidos introducido por el usuario de Oracle. Para evitar escribir una consulta que recorra la tabla de pedidos completa, operación que puede resultar muy pesada, el trigger se limita a actualizar una tabla de estadísticas.

Ediciones ENI

Oracle 12c
Administración

Colección
Recursos Informáticos

Extracto del Libro

Capítulo 7
Creación de una nueva base de datos

1. Descripción general

1.1 Etapas en la creación de una nueva base de datos para una aplicación

El proceso completo de creación de una nueva base de datos para una aplicación tiene las siguientes etapas importantes:

Diseño de la plantilla física

- Definir todos los objetos (Oracle) de la aplicación: tablas, restricciones de integridad (claves principales/únicas/extranjeras), índices, vistas, programas almacenados (triggers, procedimientos/funciones almacenadas, paquetes).
- Estudiar el volumen de la aplicación (número de usuarios, número de registros esperados en las tablas).

Creación de la base de datos propiamente dicha (este capítulo)

- Crear una nueva instancia.
- Crear una nueva base de datos (archivos de control, de traza y de datos de los tablespaces "técnicos" de Oracle).
- Hacer el diccionario de datos explotable.
- En este estado, la base de datos se puede ver como una " caja vacía" en la que se van a crear estructuras para una o varias aplicaciones.

Creación de estructuras de almacenamiento adaptadas (capítulo Tablespaces y archivos de datos)

– Crear los tablespaces (con sus archivos de datos) destinados a almacenar los datos de la aplicación (tablas e índices).
– Dimensionarlas en función del estudio del volumen que se ha realizado inicialmente.

Creación de la cuenta Oracle que va a contener los objetos de la aplicación (capítulo Gestión de usuarios y sus permisos)

– Crear la cuenta.
– Darle permisos suficientes para crear los objetos.
– Autorizarla a utilizar el espacio en los tablespaces de la aplicación.

Creación de los objetos de la aplicación en esta cuenta Oracle (capítulo Gestión de las tablas e índices)

– Crear los objetos Oracle de la aplicación (generalmente en forma de uno o varios scripts).

Creación de los usuarios finales de la aplicación (capítulo Gestión de usuarios y sus permisos)

– Crear los usuarios.
– Darles permisos adaptados a los objetos de la aplicación (es decir, sobre los objetos creados anteriormente, en la cuenta propietario de la aplicación).

Copia de seguridad de la base de datos (capítulo Copia de seguridad y restauración)

– Copia de seguridad de referencia de la base de datos.

Como puede comprobar, la creación de la base de datos propiamente dicha que se presenta en este capítulo solo es una pequeña etapa del proceso completo (pero una etapa fundamental).

1.2 Etapas en la creación de la base de datos propiamente dicha

Las grandes etapas en la creación de la base de datos propiamente dicha son las siguientes:

– Crear los repositorios en los discos, si es posible, respetando las recomendaciones del estándar OFA.

– Preparar un nuevo archivo de argumentos de texto, generalmente copiando uno antiguo.
– Dar valor a la variable de entorno `ORACLE_SID`.
– Crear el servicio asociado a la instancia (plataforma Windows) o crear el archivo de contraseñas para la identificación `SYSDBA` (plataforma Unix o Linux).
– Ejecutar SQL*Plus y conectarse `AS SYSDBA`.
– Crear un archivo de argumentos de servidor (no es obligatorio, pero sí aconsejable).
– Iniciar la instancia en estado `NOMOUNT`.
– Crear la base de datos (sentencia SQL `CREATE DATABASE`).
– Finalizar la creación del diccionario (algunos scripts que se van a ejecutar).
– Configurar Oracle Net para la nueva base de datos.
– Registrar la nueva instancia en el archivo `oratab` (plataforma Unix o Linux).
– Configurar EM Express.

La creación de una nueva base de datos supone la instalación inicial de Oracle (consulte el capítulo Instalación).

Observación

Si el servidor ya ha abierto las bases de datos Oracle, es muy aconsejable realizar una copia de seguridad de estas bases de datos antes de arrancar el proceso de creación.

Después de estas etapas, la nueva base de datos está abierta y contiene:

– los tablespaces `SYSTEM` y `SYSAUX` con su(s) archivo(s) de datos asociado(s);
– eventualmente, un tablespace de anulación y un tablespace temporal, según las opciones utilizadas;
– los archivos de control y de traza;
– las dos cuentas DBA estándar (`SYS` y `SYSTEM`);
– el segmento de anulación `SYSTEM`;
– el diccionario de datos.

En este estado, la base de datos está preparada para recibir las estructuras complementarías que van a formar la aplicación.

1.3 Métodos disponibles

La nueva base de datos se puede crear a mano, con las herramientas del sistema operativo y SQL*Plus; en este caso, es muy sencillo escribir o recuperar los scripts y reutilizarlo cada vez. Las etapas de creación de la base de datos propiamente dicha siempre son las mismas y dependen (relativamente) poco de las características de la aplicación (y, en cualquier caso, los argumentos se pueden ajustar más adelante, en función de las características de la aplicación); por lo tanto, es posible utilizar scripts "genéricos" de creación de bases.

La nueva base de datos también se puede crear con ayuda de un asistente gráfico, el asistente **Configuración de base de datos**. Este asistente facilita la creación de la base de datos, ofreciendo la posibilidad de utilizar plantillas de base de datos listas para usar y/o que permiten definir, de manera muy precisa, las características de la nueva base de datos con ayuda de varias pantallas. Por otra parte, es posible definir sus propias plantillas de base de datos, incluyendo o no archivos de datos listos para usar y utilizarlos después durante la creación posterior de una nueva base de datos. El asistente gráfico también ofrece la posibilidad de generar scripts de creación de la base de datos sin crear la base de datos; es un buen sistema para crear scripts "genéricos".

Observación

Utilizar el asistente gráfico es el método recomendado por Oracle para crear una nueva base de datos.

2. Creación de la base de datos manualmente

2.1 Crear los repositorios en los discos

Para respetar las recomendaciones del estándar OFA (consulte el capítulo Instalación) debe crear:

– un repositorio de administración con el nombre de la base de datos ubicado en el repositorio `%ORACLE_BASE%\admin` (Windows) o bien `$ORACLE_BASE/admin` (Linux/Unix),

– un repositorio de datos con el nombre de la base de datos ubicado en un repositorio `oradata`, ubicado a su vez en `ORACLE_BASE` o en otra unidad.

Observación

Desde la versión 11 y la aparición del Repositorio de diagnóstico automático, el repositorio de administración contiene menos repositorios y archivos.

El repositorio de administración generalmente contiene las siguientes carpetas:

`adump`	Carpeta para los archivos de auditoría.
`create` o `scripts`	Carpeta de scripts de creación de la base de datos.
`dpdump`	Carpeta para los archivos de exportación.
`pfile`	Carpeta para los archivos de texto de argumentos.

Si el servidor tiene varios discos, es buena idea repartir los diferentes archivos de la base de datos en estos discos con el objetivo de optimizar las entradas/salidas y evitar las limitaciones; en este caso, hay que crear otros repositorios de datos en los discos implicados.

Se puede crear un repositorio adicional para la zona de restauración rápida (consulte el capítulo Copia de seguridad y restauración).

Observación

Generalmente, la base de datos y la instancia tienen el mismo nombre.

2.2 Preparar un nuevo archivo de argumentos de texto

2.2.1 Principios generales

Como que se ha indicado en la sección La base de datos, del capítulo Las bases de la arquitectura Oracle, se aconseja utilizar un archivo de argumentos de servidor. Este archivo se crea inicialmente a partir de un archivo de argumentos de texto.

Para respetar el estándar OFA, este archivo de argumentos de texto se debe llamar `init.ora` y estar ubicado en la carpeta `pfile` del repositorio de administración. Generalmente, este archivo de argumentos de texto se crea duplicando un archivo existente o un archivo modelo, que se haya definido.

No crearemos un archivo `init<SID>.ora` (con una inclusión del archivo `init.ora`), en la ubicación por defecto de la plataforma (`dbs` en Unix/Linux, `database` en Windows); de esta manera, no evitaremos correr el riesgo de arrancar la instancia sin darnos cuenta con un archivo de argumentos de texto.

Hay más de 250 argumentos documentados por Oracle. Evidentemente, no es cuestión de especificarlos todos. De la totalidad de los argumentos hay que conocer unos treinta, suficientes para la mayor parte de las bases de datos.

Algunos argumentos se describirán brevemente en esta parte y se presentarán con más detalle en los capítulos posteriores.

2.2.2 Los principales argumentos

Los argumentos no se enumeran en orden alfabético, sino en orden temático.

DB_NAME

Nombre de la base de datos (hasta 8 caracteres). Generalmente DB_NAME es igual al nombre de la instancia (ORACLE_SID).

Ejemplo:

```
DB_NAME = NuestraBase
```

DB_DOMAIN

Localización lógica de la base de datos en la red (hasta 128 caracteres). Este argumento, asociado al argumento DB_NAME, permite a Oracle construir el nombre global de la base de datos, en forma DB_NAME.DB_DOMAIN. Este argumento es importante si la base de datos pertenece a un sistema distribuido (o susceptible de serlo); en caso contrario, se puede ignorar.

Ejemplo:

```
DB_DOMAIN = ivan-mendoza.es
```

DB_UNIQUE_NAME

Nombre único de la base de datos (hasta 30 caracteres). Las bases de datos con el mismo DB_NAME dentro del mismo DB_DOMAIN (por ejemplo, una base de producción y otra de prueba), deben tener un DB_UNIQUE_NAME diferente. Este argumento apareció en la versión 10. Por defecto, es igual a DB_NAME.

Este argumento se debe especificar si desea abrir en un servidor, al mismo tiempo, dos bases de datos con el mismo nombre (mismo DB_NAME); permite diferenciarlas.

Ejemplo:

```
DB_UNIQUE_NAME = NuestraBase_demo
```

COMPATIBLE

Indica un número de versión de Oracle con la que la base de datos debe ser compatible. Valores posibles: 11.0.0 hasta el número de la versión actual (12.1.0.1). Valor por defecto: 12.0.0.

Este argumento permite utilizar una nueva versión de Oracle manteniendo la compatibilidad con una versión más antigua y, por tanto, sin tener necesidad de probar nuevas las funcionalidades en la base de datos. Algunas funcionalidades de la nueva versión pueden estar restringidas. El valor del argumento se puede aumentar posteriormente, aunque generalmente es imposible disminuirlo más tarde (hay que partir de nuevo de una copia de seguridad anterior al cambio).