



Programación de bases de datos con SQL

12-3

Valores DEFAULT, MERGE e Inserciones en Varias Tablas



Objetivos

En esta lección, aprenderá a:

- Comprender cuándo especificar un valor DEFAULT
- Crear y ejecutar una sentencia MERGE
- Construir y ejecutar sentencias DML utilizando subconsultas
- Construir y ejecutar inserciones en varias tablas



Objetivo

- Hasta ahora, ha estado actualizando los datos mediante una única sentencia INSERT.
- Ha sido relativamente fácil al agregar registros uno a uno, ¿y si la compañía es muy grande y utiliza un almacén de datos para almacenar los datos de registros de ventas, de clientes, de nóminas, de contabilidad y de personal?
- En este caso, es probable que los datos procedan de varios orígenes y estén gestionados por varias personas.
- La gestión de datos registro a registro puede ser muy confuso y llevar mucho tiempo.

Objetivo

- ¿Cómo determina lo que se ha insertado nuevo o qué se ha cambiado recientemente?
- En esta lección, conocerá un método más eficaz para actualizar e insertar datos utilizando una secuencia de comandos INSERT y UPDATE condicionales en una única sentencia atómica.
- También aprenderá a recuperar datos de una única subconsulta e INSERTAR las filas devueltas en más de una tabla de destino.
- A medida que amplíe sus conocimientos en SQL, apreciará formas eficaces para realizar su trabajo.

DEFAULT

- Cada columna de una tabla puede tener un valor por defecto especificado para él.
- En el caso de que se inserte una fila nueva y no haya ningún valor asignado para la columna, se asignará el valor por defecto en lugar de un valor nulo.
- El uso de valores por defecto permite controlar dónde y cuándo se debe aplicar el valor por defecto.

DEFAULT

- El valor por defecto puede ser un valor literal, una expresión o una función SQL como SYSDATE o USER, pero el valor no puede ser el nombre de otra columna.
- El valor por defecto debe coincidir con el tipo de dato de la columna.
- Se puede especificar DEFAULT para una columna al crear o modificar la tabla.

Ejemplo de DEFAULT

- En el ejemplo siguiente, se muestra un valor por defecto especificado para la columna hire_date al crear la tabla:

```
CREATE TABLE my_employees (  
    hire_date  DATE DEFAULT SYSDATE,  
    first_name  VARCHAR2(15),  
    last_name  VARCHAR2(15));
```

- Al agregar filas a esta tabla, SYSDATE se asignará a cualquier fila que no especifique explícitamente un valor hire_date.

DEFAULT explícito con INSERT

- Se pueden utilizar valores por defecto explícitos en las sentencias INSERT y UPDATE.
- El ejemplo de INSERT con la tabla my_employees muestra el uso explícito de DEFAULT:

```
INSERT INTO my_employees  
  (hire_date, first_name, last_name)  
VALUES  
  (DEFAULT, 'Angelina', 'Wright');
```

- Uso implícito de DEFAULT

```
INSERT INTO my_employees  
  (first_name, last_name)  
VALUES  
  ('Angelina', 'Wright');
```

DEFAULT explícito con UPDATE

- Se pueden utilizar valores por defecto explícitos en las sentencias INSERT y UPDATE.
- El ejemplo de UPDATE con la tabla my_employees muestra el uso explícito de DEFAULT:

```
UPDATE my_employees  
SET hire_date = DEFAULT  
WHERE last_name = 'Wright';
```

- Si se ha especificado un valor por defecto para la columna hire_date, se asigna a la columna el valor por defecto.
- Sin embargo, si no se ha especificado ningún valor por defecto al crear la columna, se asigna un valor nulo.

MERGE

- El uso de la sentencia MERGE logra dos tareas al mismo tiempo. MERGE INSERTARÁ y ACTUALIZARÁ simultáneamente. Si falta un valor, se inserta uno nuevo.
- Si existe un valor pero se debe cambiar, MERGE lo actualizará.
- Para realizar estos tipos de cambios en tablas de base de datos, debe tener privilegios INSERT y UPDATE en la tabla de destino y privilegios SELECT en la tabla de origen.
- Los alias se pueden utilizar con la sentencia MERGE.

Sintaxis de MERGE

- Se lee una fila cada vez de la tabla de origen y se compara con las filas de la tabla de destino utilizando la condición de coincidencia.
- Si existe una fila coincidente en la tabla de destino, la fila de origen se utiliza para actualizar una o más columnas de la fila de destino coincidente.
- Si no existe una fila coincidente, los valores de la fila de origen se utilizan para insertar una nueva fila en la tabla de destino.

```
MERGE INTO destination-table USING source-table
      ON matching-condition
      WHEN MATCHED THEN UPDATE
        SET .....
      WHEN NOT MATCHED THEN INSERT
        VALUES (.....);
```

Ejemplo de MERGE

- En este ejemplo, se utiliza la tabla EMPLOYEES (alias e) como origen de datos para insertar y actualizar filas en una copia de la tabla denominada COPY_EMP (alias c).

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

Ejemplo de MERGE

- Las filas 100 y 103 de EMPLOYEES tienen filas coincidentes en COPY_EMP y, por lo tanto, se actualizaron las filas coincidentes de COPY_EMP.
- EMPLOYEE 142 no tiene ninguna fila coincidente y, por lo tanto, se insertó en COPY_EMP.

EMPLOYEES (tabla de origen)

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
100	King	90
103	Hunold	60
142	Davies	50

Ejemplo de MERGE

EMPLOYEES (tabla de origen)

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
100	King	90
103	Hunold	60
142	Davies	50

COPY_EMP antes de ejecutar MERGE

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
100	Smith	40
103	Chang	30

COPY_EMP después de ejecutar MERGE

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
100	King	90
103	Hunold	60
142	Davies	50

Inserciones en Varias Tablas

- Las inserciones en varias tablas se utilizan cuando se deben insertar los mismos datos de origen en más de una tabla de destino.
- Esta funcionalidad resulta útil cuando trabaja en un entorno de almacén de datos donde es común mover con regularidad datos de los sistemas operativos a un almacén de datos para la generación de informes analíticos y análisis.
- La creación y la gestión de almacenes de datos es una forma de gestionar el a veces muy elevado número de filas insertadas en sistemas operativos durante un día laborable normal.

Inserciones en Varias Tablas

- Imagine, por ejemplo, ¿cuántas filas de datos debe crear su proveedor de telefonía diariamente para todas sus llamadas o mensajes de texto realizados en todos los dispositivos a los que tiene acceso a?
- A continuación, agregue a eso la navegación por Internet y las descargas de tonos, fondos de escritorio, juegos y otras aplicaciones móviles.
- Multiplique ese número por el número total de clientes y eso puede darle una idea de la cantidad de datos que las compañías de telecomunicaciones tienen que gestionar.

Inserciones en Varias Tablas

- Puede que estas filas se tengan que insertar en más de una tabla del almacén de datos, por lo que si simplemente tenemos que SELECCIONARLAS una vez y, a continuación, replicarlas, se mejorará el rendimiento.
- Las inserciones en varias pueden ser in condicionales o condicionales. En una inserción incondicional en varias tablas, Oracle insertará todas las filas devueltas por la subconsulta en todas las cláusulas de inserción de tabla encontradas en la sentencia.
- En una inserción condicional en varias tablas, puede especificar ALL o FIRST.

Inserciones en Varias Tablas

- Especificación de ALL:
 - Si especifica ALL, valor por defecto, la base de datos evalúa cada cláusula WHEN independientemente de los resultados de la evaluación de cualquier otra cláusula WHEN.
 - Para cada cláusula WHEN cuya condición se evalúe como true, la base de datos ejecuta la lista de cláusulas INTO correspondientes.
- Especificación de FIRST:
 - Si especifica FIRST, la base de datos evalúa cada cláusula WHEN en el orden en el que aparece en la sentencia.
 - Para la primera cláusula WHEN que se evalúe como true, la base de datos ejecuta la cláusula INTO correspondiente y omite las cláusulas WHEN posteriores de la fila determinada.

Inserciones en Varias Tablas

- Especificación de la cláusula ELSE:
 - Para una fila determinada, si la cláusula WHEN no se evalúa como true, la base de datos ejecuta la lista de cláusulas INTO asociada con la cláusula ELSE.
 - Si no especifica una cláusula ELSE, la base de datos no realiza ninguna acción para dicha fila.

Sintaxis de Inserciones en Varias Tablas

- La sintaxis de la sentencia de inserción en varias tablas es la siguiente:

```
INSERT ALL INTO clause VALUES clause SUBQUERY
```

- Un ejemplo de la sentencia de inserción en varias tablas es el siguiente:

```
INSERT ALL  
  INTO my_employees  
    VALUES (hire_date, first_name, last_name)  
  INTO copy_my_employees  
    VALUES (hire_date, first_name, last_name)  
SELECT hire_date, first_name, last_name  
FROM employees;
```

Inserciones Condicionales en Varias Tablas

```
INSERT ALL
  WHEN call_format IN ('tlk','txt','pic') THEN
  INTO all_calls
    VALUES (caller_id, call_timestamp, call_duration, call_format)
  WHEN call_format IN ('tlk','txt') THEN
  INTO police_record_calls
    VALUES (caller_id, call_timestamp, recipient_caller)
  WHEN call_duration < 50 AND call_type = 'tlk' THEN
  INTO short_calls
    VALUES (caller_id, call_timestamp, call_duration)
  WHEN call_duration >= 50 AND call_type = 'tlk' THEN
  INTO long_calls
    VALUES (caller_id, call_timestamp, call_duration)
SELECT caller_id, call_timestamp, call_duration, call_format,
       recipient_caller
FROM calls
WHERE TRUNC(call_timestamp) = TRUNC(SYSDATE);
```

Terminología

Entre los términos clave utilizados en esta lección se incluyen:

- DEFAULT
- MERGE
- Inserciones en Varias Tablas
- ALL, FIRST y ELSE

Resumen

En esta lección, debe haber aprendido lo siguiente:

- Comprender cuándo especificar un valor DEFAULT
- Crear y ejecutar una sentencia MERGE
- Construir y ejecutar sentencias DML utilizando subconsultas
- Construir y ejecutar inserciones en varias tablas

