
ORACLE
Academy

Programación de bases de datos con SQL

13-3
Modificación de una Tabla

ORACLE Academy

Objetivos

En esta lección, aprenderá a:

- Explicar por qué es importante poder modificar una tabla
- Explicar y proporcionar un ejemplo para cada una de las sentencias DDL (ALTER, DROP, RENAME y TRUNCATE) y el efecto que tiene cada una en las tablas y columnas
- Construir una consulta y ejecutar los comandos ADD, MODIFY y DROP de ALTER TABLE
- Explicar y ejecutar FLASHBACK QUERY en una tabla
- Explicar y realizar operaciones FLASHBACK table

Objetivos

En esta lección, aprenderá a:

- Seguimiento de los Cambios en los Datos a lo Largo de un Período de Tiempo
- Explicar los motivos para utilizar TRUNCATE frente a DELETE en las tablas
- Agregar un comentario a una tabla utilizando el comando COMMENT ON TABLE
- Enumerar los cambios que se pueden y no se pueden realizar en una columna
- Explicar cuándo y por qué la sentencia SET UNUSED es ventajosa

Objetivo

- ¿Recuerde la frase "No hay nada permanente excepto el cambio"?
- ¿No sería fantástico si nunca cometiéramos errores ni necesitáramos cambiar nada?
- Como sabe hasta ahora, las bases de datos son entidades dinámicas.
- Probablemente no serían muy útiles si no se pudieran cambiar.

Objetivo

- Hasta ahora, ha creado tablas y realizado los cambios en los datos de fila de tablas, ¿pero cómo realiza cambios en las propias tablas?
- En esta lección se presentan los comandos DDL que se utilizan para modificar, renombrar, vaciar o simplemente eliminar una tabla por completo.

ALTER TABLE

- Las sentencias ALTER TABLE se utilizan para:
  - Agregar una nueva columna
  - Modificar una columna existente
  - Definir un valor DEFAULT para una columna.
  - Borrar una columna
- Puede agregar o modificar una columna de una tabla, pero no puede especificar donde aparece la columna.

ALTER TABLE

- Una columna recién agregada siempre se convierte en la última columna de la tabla.
- Además, si una tabla ya tiene filas de datos y agrega una nueva columna a la tabla, la nueva columna inicialmente será nula para todas las filas ya existentes.

ALTER TABLE: Adición de una Columna

- Para agregar una nueva columna, utilice la sintaxis SQL que se muestra a continuación:

```
ALTER TABLE tablename
ADD (column name data type [DEFAULT expression],
column name data type [DEFAULT expression], ...
```

- Por ejemplo:

```
ALTER TABLE my_cd_collection
ADD (release_date DATE DEFAULT SYSDATE);
```

```
ALTER TABLE my_friends
ADD (favorite_game VARCHAR2(30));
```

ALTER TABLE: Modificación de una Columna

- La modificación de una columna puede incluir cambios en el tipo de dato, tamaño y valor DEFAULT de una columna.
- Las reglas y las restricciones al modificar una columna son las siguientes:
  - Puede aumentar el ancho o la precisión de una columna numérica.
  - Puede aumentar el ancho de una columna de caracteres.
  - Puede reducir el ancho de una columna NUMBER si la columna contiene únicamente valores nulos o si la tabla no tiene filas.
  - Para tipos VARCHAR, puede reducir el ancho hasta el mayor valor incluido en la columna.

ALTER TABLE: Modificación de una Columna

- Puede cambiar el tipo de dato solo si la columna contiene valores nulos.
- Solo puede convertir una columna CHAR a VARCHAR2 o una columna VARCHAR2 a CHAR si la columna contiene valores nulos, o bien si no cambia el tamaño a algo más pequeño que cualquier valor de la columna.
- El cambio del valor DEFAULT de una columna silo afecta a las inserciones posteriores en la tabla.

Ejemplo de ALTER TABLE: Modificación de una Columna

- Ejemplo: se ha creado una tabla con dos columnas:

```
CREATE TABLE mod_emp
  (last_name VARCHAR2(20),
 salary NUMBER(8,2));
```

- ¿Cuál de estas modificaciones estaría permitida y cuál no?
- Considere las respuestas con y sin filas de datos en la tabla.

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

```
ALTER TABLE mod_emp
   MODIFY (last_name VARCHAR2(10));
```

```
ALTER TABLE mod_emp
   MODIFY (salary NUMBER(10,2));
```

```
ALTER TABLE mod_emp
   MODIFY (salary NUMBER(8,2) DEFAULT 50);
```

Ejemplo de ALTER TABLE: Modificación de una Columna

- Solo estaría permitida si las columnas estuvieran vacías o el nombre más largo tuviera 10 caracteres o menos

```
ALTER TABLE mod_emp
   MODIFY (last_name VARCHAR2(10));
```

- Estaría permitida con o sin datos ya que aumenta el ancho de columna.

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

Ejemplo de ALTER TABLE: Modificación de una Columna

- Estaría permitida con o sin datos ya que aumenta la precisión de columna.

```
ALTER TABLE mod_emp
   MODIFY (salary NUMBER(10,2));
```

- Se permitiría con o sin datos, ya que solo se agrega un valor DEFAULT.

```
ALTER TABLE mod_emp
   MODIFY (salary NUMBER(8,2) DEFAULT 50);
```

ALTER TABLE: Borrado de una Columna

- Al borrar una columna se aplican las siguientes reglas:
  - Se puede borrar una columna que contenga datos.
  - Solo se puede borrar una columna cada vez.
  - No puede borrar todas las columnas de una tabla; debe permanecer al menos una columna.
  - Una vez borrada la columna, los valores de datos de esta no se pueden recuperar.

ALTER TABLE: Borrado de una Columna

- Sintaxis SQL:

```
ALTER TABLE tablename
DROP COLUMN column name;
```

- Por ejemplo:

```
ALTER TABLE my_cd_collection
DROP COLUMN release_date;
```

```
ALTER TABLE my_friends
DROP COLUMN favorite_game;
```

SET UNUSED para Columnas

- Borrar una columna de una tabla grande puede llevar mucho tiempo.
- Una alternativa más rápida es marcar la columna como no utilizable.
- Los valores de columna permanecen en la base de datos, pero no se puede acceder a ellos de ninguna forma, por lo que el efecto es el mismo que borrar la columna.
- De hecho, puede agregar una nueva columna a la base de datos con el mismo nombre que la columna no utilizada.
- Las columnas no utilizadas existen, pero son invisibles.
- Sintaxis:

```
ALTER TABLE tablename SET UNUSED (column name);
```

Ejemplo de SET UNUSED para Columnas

- Ejemplo:

```
ALTER TABLE copy_employees
 SET UNUSED (email);
```

- DROP UNUSED COLUMNS elimina todas las columnas marcadas actualmente como no utilizadas.
- Utilice esta sentencia cuando desee reclamar el espacio en disco adicional de las columnas no utilizadas en una tabla.
- Ejemplo:

```
ALTER TABLE copy_employees
 DROP UNUSED COLUMNS;
```

Resumen de ALTER TABLE

- En este gráfico se resumen los usos del comando ALTER TABLE:

Sintaxis	Resultados	Problemas
ALTER TABLE tablename ADD (column name data type [DEFAULT expression], column name data type [DEFAULT expression], …	Agrega una nueva columna a la tabla	No puede especificar dónde debe aparecer la columna en la tabla. Se convierte en la última columna.
ALTER TABLE tablename MODIFY (column name data type [DEFAULT expression], column name data type, …	Se utiliza para cambiar el tipo de dato, tamaño y valor por defecto de una columna.	El cambio del valor por defecto de una columna solo afecta a las inserciones posteriores en la tabla.
ALTER TABLE tablename DROP COLUMN column name;	Se utiliza para borrar una columna de una tabla	Debe quedar al menos una columna en la tabla después de modificarla. Una vez borrada, la columna no se puede recuperar.
ALTER TABLE tablename SET UNUSED (column name);	Se utiliza para marcar una o más columnas para que se puedan borrar más tarde	No restaura espacio en disco. Las columnas se tratan como si se hubieran borrado.
ALTER TABLE tablename DROP UNUSED COLUMNS	Elimina de la tabla todas las columnas marcadas actualmente como no utilizadas.	Una vez definidas como no utilizadas, no hay acceso a las columnas; no se muestra ningún dato con DESCRIBE. Eliminación permanente; sin rollback.

DROP TABLE

- La sentencia DROP TABLE elimina la definición de una tabla de Oracle.
- La base de datos pierde todos los datos de la tabla y los índices asociados a los mismos.
- Cuando se emite una sentencia DROP TABLE:
  - Se suprimen todos los datos de la tabla.
  - Se elimina la descripción de la tabla del diccionario de datos.
- Oracle Server no cuestiona la decisión y borra la tabla inmediatamente.

DROP TABLE

- En la siguiente diapositiva, verá que puede restaurar una tabla después borrarla, pero no está garantizado.
- Solo el creador de la tabla o un usuario con el privilegio DROP ANY TABLE (normalmente solo el DBA) puede eliminar una tabla.
- Sintaxis:

```
DROP TABLE tablename;
```

- Ejemplo:

```
DROP TABLE copy_employees;
```

FLASHBACK TABLE

- Si borra una tabla por error, puede recuperar esa tabla y sus datos.
- Cada usuario de base de datos tiene su propia papelera de reciclaje a la que se mueven los objetos borrados y que se pueden recuperar de aquí con el comando FLASHBACK TABLE.
- Este comando se puede utilizar para restaurar una tabla, una vista o un índice que se haya borrado por error.
- La sintaxis es la siguiente:

```
FLASHBACK TABLE tablename TO BEFORE DROP;
```

FLASHBACK TABLE

- Por ejemplo, si borra la tabla EMPLOYEES por error, puede recuperarla simplemente emitiendo el comando:

```
FLASHBACK TABLE copy_employees TO BEFORE DROP;
```

- Como propietario de una tabla, puede emitir el comando de flashback y si la tabla que está restaurando tenía índices, estos también se restauran.
- Es posible ver los objetos que se pueden restaurar consultando la vista del diccionario de datos USER_RECYCLEBIN.

FLASHBACK TABLE

- Se puede consultar la vista USER_RECYCLEBIN como todas las demás vistas del diccionario de datos:

```
SELECT original_name, operation, droptime
FROM user_recyclebin
```

ORIGINAL_NAME	OPERATION	DROPTIME
EMPLOYEES	DROP	2007-12-05:12.34.24
EMP_PK	DROP	2007-12-05:12.34.24

FLASHBACK TABLE

- Una vez restaurada una tabla con el comando FLASHBACK TABLE, ya no está visible en la vista USER_RECYCLEBIN.
- Cualquier índice que se borrara con la tabla original también se restaurará.
- Puede que sea necesario (por motivos de seguridad) borrar una tabla completamente, omitiendo la papelera de reciclaje.
- Esto se puede hacer agregando la palabra clave PURGE.

```
DROP TABLE copy_employees PURGE;
```

RENAME

- Para cambiar el nombre de una tabla, utilice la sentencia RENAME.
- Esto solo lo puede hacer el propietario del objeto o el DBA.
- Sintaxis:

```
RENAME old_name to new_name;
```

- Ejemplo:

```
RENAME my_cd_collection TO my_music;
```

- Veremos más adelante que podemos renombrar otros tipos de objetos como, por ejemplo, vistas, secuencias y sinónimos.

TRUNCATE

- Al truncar una tabla, se eliminan todas las filas de una tabla y se libera el espacio de almacenamiento utilizado por dicha tabla.
- Al utiliza la sentencia TRUNCATE TABLE:
  - No se puede realizar rollback de la eliminación de filas.
  - Debe ser el propietario de la tabla o tener privilegios de sistema DROP ANY TABLE.

TRUNCATE

- Sintaxis:

```
TRUNCATE TABLE tablename;
```

- Con la sentencia DELETE también se eliminan todas las filas de una tabla, pero no se libera el espacio de almacenamiento.
- TRUNCATE es más rápido que DELETE porque no genera información de rollback.

COMMENT ON TABLE

- Puede agregar un comentario de hasta 2.000 caracteres sobre una columna, tabla o vista mediante la sentencia COMMENT.
- Sintaxis:

```
COMMENT ON TABLE tablename | COLUMN table.column
   IS 'place your comment here';
```

- Ejemplo:

```
COMMENT ON TABLE employees
   IS 'Western Region only';
```

COMMENT ON TABLE

- Para ver los comentarios de la tabla en el diccionario de datos:

```
SELECT table_name, comments
FROM user_tab_comments;
```

TABLE_NAME	COMMENTS
EMPLOYEES	Western Region Only

- Si desea borrar un comentario realizado anteriormente en una tabla o columna, utilice la cadena vacía (''):

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

FLASHBACK QUERY

- Puede que detecte que los datos en una tabla se han cambiado, de algún modo, de forma incorrecta.
- Afortunadamente, Oracle tiene una función que permite ver los datos de fila en momentos concretos en el tiempo, por lo que puede comparar diferentes versiones de una fila a lo largo del tiempo.
- Esta función es muy útil.
- Imagine, por ejemplo, que alguien por error ha realizado algún DML en una tabla y, a continuación, ejecuta COMMIT en dichos cambios.
- Oracle Application Express confirma automáticamente, por lo que se cometen fácilmente errores.

FLASHBACK QUERY

- Puede utilizar la función FLASHBACK QUERY para examinar qué aspecto tenían las filas ANTES de que se aplicaran esos cambios.
- Cuando Oracle cambia datos, siempre mantiene una copia del aspecto de los datos corregidos antes de que se realizara cualquier cambio.
- Por lo que mantiene una copia del valor anterior de la columna para una actualización de columna, mantiene toda la fila para una supresión y no mantiene nada para una sentencia de inserción.

FLASHBACK QUERY

- Estas copias antiguas se mantienen en un lugar especial denominado el tablespace UNDO.
- Los usuarios pueden acceder a esta zona especial de la base de datos mediante una consulta de flashback.
- Puede consultar las versiones anteriores de los datos utilizando la cláusula VERSIONS en una sentencia SELECT.

FLASHBACK QUERY

- Por ejemplo:

```
SELECT employee_id,first_name ||' '|| last_name AS "NAME",
  versions_operation AS "OPERATION",
  versions_starttime AS "START_DATE",
  versions_endtime  AS "END_DATE", salary
FROM employees
  VERSIONS BETWEEN SCN MINVALUE AND MAXVALUE
WHERE  employee_id = 1;
```

- El número SCN al que se hacía referencia en la consulta anterior significa que el número de cambio del sistema es una identificación precisa de tiempo en la base de datos.
- Se trata de un número secuencial que incrementa y mantiene la propia base de datos.

FLASHBACK QUERY

- La mejor forma de demostrar FLASHBACK QUERY es con un ejemplo.
- El contenido es el siguiente para employee_id 1 en la tabla de empleados.

```
SELECT employee_id,first_name ||' '|| last_name AS "NAME",
     versions_operation AS "OPERATION",
     versions_starttime AS "START_DATE",
     versions_endtime  AS "END_DATE", salary
FROM copy_employees
  VERSIONS BETWEEN SCN MINVALUE AND MAXVALUE
WHERE  employee_id = 1;
```

no se han encontrado datos

FLASHBACK QUERY

- A continuación, creamos el empleado:

```
INSERT INTO copy_employees
VALUES (1, 'Natacha', 'Hansen', 'NHANSEN', '4412312341234',
    '07-SEP-1998', 'AD_VP', 12000, null, 100, 90, NULL);
```

```
SELECT employee_id,first_name ||' '|| last_name AS "NAME",
     versions_operation AS "OPERATION",
     versions_starttime AS "START_DATE",
     versions_endtime  AS "END_DATE", salary
FROM copy_employees
  VERSIONS BETWEEN SCN MINVALUE AND MAXVALUE
WHERE  employee_id = 1;
```

EMPLOYEE_ID	NAME	OPERATION	START_DATE	END_DATE	SALARY
1	Natacha Hansen	I	07-SEP-1998 06.51.58 AM	-	12000

FLASHBACK QUERY

- A continuación, puede actualizar la fila:

```
UPDATE copy_employees
SET salary = 1
WHERE employee_id = 1;
```

```
SELECT employee_id,first_name ||' '|| last_name AS "NAME",
     versions_operation AS "OPERATION",
     versions_starttime AS "START_DATE",
     versions_endtime  AS "END_DATE", salary
FROM copy_employees
  VERSIONS BETWEEN SCN MINVALUE AND MAXVALUE
WHERE  employee_id = 1;
```

EMPLOYEE_ID	NAME	OPERATION	START_DATE	END_DATE	SALARY
1	Natacha Hansen	U	07-SEP-1998 06.57.01 AM	-	1
1	Natacha Hansen	I	07-SEP-1998 06.51.58 AM	07-SEP-1998 06.57.01 AM	12000

FLASHBACK QUERY

- A continuación, puede suprimir la fila:

```
DELETE from copy_employees
WHERE employee_id = 1;
```

```
SELECT employee_id,first_name ||' '|| last_name AS "NAME",
     versions_operation AS "OPERATION",
     versions_starttime AS "START_DATE",
     versions_endtime  AS "END_DATE", salary
FROM copy_employees
  VERSIONS BETWEEN SCN MINVALUE AND MAXVALUE
WHERE  employee_id = 1;
```

EMPLOYEE_ID	NAME	OPERATION	START_DATE	END_DATE	SALARY
1	Natacha Hansen	D	07-SEP-1998 07.00.10 AM	-	1
1	Natacha Hansen	U	07-SEP-1998 06.57.01 AM	07-SEP-1998 07.00.10 AM	1
1	Natacha Hansen	I	07-SEP-1998 06.51.58 AM	07-SEP-1998 06.57.01 AM	12000

FLASHBACK QUERY

- El resultado de la última consulta de la diapositiva anterior solo está disponible si se utiliza Flashback query, es decir, la cláusula VERSIONS.
- Si intenta realizar una búsqueda normal de employee_id = 1 tras la sentencia delete, recibirá el error normal, No Data Found.

```
SELECT employee_id, salary
FROM copy_employees
WHERE employee_id = 1;
```

Terminología

Entre los términos clave utilizados en esta lección se incluyen:

- ALTER TABLE
  - ADD
  - MODIFY
  - DROP
- DROP TABLE
- RENAME
- TRUNCATE

Terminología

Entre los términos clave utilizados en esta lección se incluyen:

- COMMENT ON TABLE
- FLASHBACK TABLE
- FLASHBACK QUERY
- SET UNUSED

Resumen

En esta lección, ha aprendido lo siguiente:

- Explicar por qué es importante poder modificar una tabla
- Explicar y proporcionar un ejemplo para cada una de las sentencias DDL (ALTER, DROP, RENAME y TRUNCATE) y el efecto que tiene cada una en las tablas y columnas
- Construir una consulta y ejecutar los comandos ADD, MODIFY y DROP de ALTER TABLE
- Explicar y ejecutar FLASHBACK QUERY en una tabla
- Explicar y realizar operaciones FLASHBACK table

Resumen

En esta lección, ha aprendido lo siguiente:

- Seguimiento de los Cambios en los Datos a lo Largo de un Período de Tiempo
- Explicar los motivos para utilizar TRUNCATE frente a DELETE en las tablas
- Agregar un comentario a una tabla utilizando el comando COMMENT ON TABLE
- Enumerar los cambios que se pueden y no se pueden realizar en una columna
- Explicar cuándo y por qué la sentencia SET UNUSED es ventajosa


ORACLE
Academy