SQL para Principiantes (Parte 10): Las Declaraciones DELETE y TRUNCATE TABLE

by admin - domingo, octubre 04, 2020

https://dbandtech.com/sql-para-principiantes-parte-10-las-declaraciones-delete-y-truncate-table/

Esta es la décima parte de una serie de artículos que muestran los **conceptos básicos de SQL**. En este artículo echamos un vistazo a las declaraciones **DELETE y TRUNCATE TABLE**.

- **Preparar**
- **COMMIT y ROLLBACK**
- **DELETE Básico**
- **DELETE vía View**
- **0 Rows Deleted**
- **TRUNCATE TABLE**

Preparar

Puede realizar todas estas consultas en línea de forma gratuita utilizando SQL Fiddle.

https://youtu.be/KdE4LSIO2fM

Los ejemplos de este artículo requieren que estén presentes las siguientes tablas.

```
--DROP TABLE employees PURGE;
--DROP TABLE departments PURGE;

CREATE TABLE departments (
  department_id   NUMBER(2) CONSTRAINT departments_pk PRIMARY KEY,
  department_name VARCHAR2(14),
  location        VARCHAR2(13)
);
INSERT INTO departments VALUES (10,'ACCOUNTING','NEW YORK');
INSERT INTO departments VALUES (20,'RESEARCH','DALLAS');
INSERT INTO departments VALUES (30,'SALES','CHICAGO');
INSERT INTO departments VALUES (40,'OPERATIONS','BOSTON');
COMMIT;

CREATE TABLE employees (
  employee_id   NUMBER(4) CONSTRAINT employees_pk PRIMARY KEY,
  employee_name VARCHAR2(10),
  job           VARCHAR2(9),
  manager_id    NUMBER(4),
  hiredate      DATE,
```

```
  salary        NUMBER(7,2),
  commission    NUMBER(7,2),
  department_id NUMBER(2) CONSTRAINT emp_department_id_fk REFERENCES d
epartments(department_id)
);
INSERT INTO employees VALUES (7369,'SMITH','CLERK',7902,to_date('17-12
-1980','dd-mm-yyyy'),800,NULL,20);
INSERT INTO employees VALUES (7499,'ALLEN','SALESMAN',7698,to_date('20
-2-1981','dd-mm-yyyy'),1600,300,30);
INSERT INTO employees VALUES (7521,'WARD','SALESMAN',7698,to_date('22-
2-1981','dd-mm-yyyy'),1250,500,30);
INSERT INTO employees VALUES (7566,'JONES','MANAGER',7839,to_date('2-4
-1981','dd-mm-yyyy'),2975,NULL,20);
INSERT INTO employees VALUES (7654,'MARTIN','SALESMAN',7698,to_date('2
8-9-1981','dd-mm-yyyy'),1250,1400,30);
INSERT INTO employees VALUES (7698,'BLAKE','MANAGER',7839,to_date('1-5
-1981','dd-mm-yyyy'),2850,NULL,30);
INSERT INTO employees VALUES (7782,'CLARK','MANAGER',7839,to_date('9-6
-1981','dd-mm-yyyy'),2450,NULL,10);
INSERT INTO employees VALUES (7788,'SCOTT','ANALYST',7566,to_date('13-
JUL-87','dd-mm-rr')-85,3000,NULL,20);
INSERT INTO employees VALUES (7839,'KING','PRESIDENT',NULL,to_date('17
-11-1981','dd-mm-yyyy'),5000,NULL,10);
INSERT INTO employees VALUES (7844,'TURNER','SALESMAN',7698,to_date('8
-9-1981','dd-mm-yyyy'),1500,0,30);
INSERT INTO employees VALUES (7876,'ADAMS','CLERK',7788,to_date('13-JU
L-87', 'dd-mm-rr')-51,1100,NULL,20);
INSERT INTO employees VALUES (7900,'JAMES','CLERK',7698,to_date('3-12-
1981','dd-mm-yyyy'),950,NULL,30);
INSERT INTO employees VALUES (7902,'FORD','ANALYST',7566,to_date('3-12
-1981','dd-mm-yyyy'),3000,NULL,20);
INSERT INTO employees VALUES (7934,'MILLER','CLERK',7782,to_date('23-1
-1982','dd-mm-yyyy'),1300,NULL,10);
COMMIT;
```

Estas tablas son una variante de las tablas EMP y DEPT del esquema SCOTT. Verá muchos ejemplos de Oracle en Internet utilizando las tablas del esquema de SCOTT. Puede encontrar las definiciones de la tabla original en el script "$ORACLE_HOME/rdbms/admin/utlsampl.sql".

COMMIT y ROLLBACK

Todos los cambios en el lenguaje de **manipulación de datos (DML)** se realizan como parte de una transacción. No son permanentes hasta que se confirman mediante la declaración **COMMIT**. Una vez comprometido, la única forma de revertir un cambio es emitir una **nueva declaración DML** para alterar

los datos. Se pueden agrupar varios extractos para formar una sola transacción.

Los comandos del lenguaje de definición de datos (DDL) realizan una confirmación implícita, que también confirma todos los cambios de DML pendientes en la sesión actual.

Si decide que no desea mantener algunos cambios sin confirmar, puede deshacerse de ellos usando la instrucción ROLLBACK. Muchos de los ejemplos de este artículo emitirán declaraciones ROLLBACK después de la prueba, para revertir los datos a su estado original.

Algunas herramientas y lenguajes de programación se comprometen automáticamente de forma predeterminada, por lo que emiten automáticamente una declaración COMMIT después de cada declaración DML que procesan. No dejes que esto te engañe haciéndote pensar que es un comportamiento predeterminado. No lo es.

DELETE Básico

La instrucción **DELETE** se usa para eliminar filas de la tabla. Sin una cláusula **WHERE**, todas las filas de la tabla se eliminan mediante una única declaración.

El siguiente ejemplo elimina todas las filas de la tabla **EMPLOYEES** y luego emite un **ROLLBACK** para cancelar la eliminación.

```
DELETE FROM employees;

14 rows deleted.

SQL> ROLLBACK;
```

La cláusula **WHERE** le permite limitar las filas que se eliminarán.

```
DELETE FROM employees
WHERE employee_id = 7369;

1 row deleted.

SQL> ROLLBACK;
```

DELETE vía View

Es posible eliminar de la tabla base asociada con una vista. Hay algunas restricciones asociadas con esto, pero están un poco fuera del alcance de un artículo de nivel principiante. En el siguiente ejemplo, creamos una vista simple en la tabla **EMPLOYEES** y luego la eliminamos.

```
CREATE OR REPLACE VIEW employees_v AS
SELECT * FROM employees;

DELETE FROM employees_v
WHERE employee_id = 7369;

1 row updated.

SQL> ROLLBACK;
```

No lo verá muy a menudo, pero también puede eliminarlo mediante vistas en línea. Esto se puede usar para controlar el número de filas eliminadas, en lugar de usar un filtro en la cláusula **WHERE** de la propia instrucción **DELETE**.

```
DELETE FROM (SELECT employee_id, salary
             FROM   employees
             WHERE  department_id = 20);

5 rows deleted.

SQL> ROLLBACK;
0 Rows Deleted
```

Una eliminación de cero filas es una eliminación válida y no genera un error. Esto puede resultar bastante confuso para los principiantes.

```
DELETE FROM employees
WHERE  employee_id = 9999;

0 rows deleted.

SQL>
```

Como resultado, no puede probar si no se eliminan filas con la excepción **NO_DATA_FOUND en PL/SQL**, ya que no se genera.

```
SET SERVEROUTPUT ON
BEGIN
  DELETE FROM employees
  WHERE  employee_id = 9999;
```

```
  DBMS_OUTPUT.put_line('NO_DATA_FOUND Not Raised');
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.put_line('NO_DATA_FOUND Raised');
END;
/
NO_DATA_FOUND Not Raised

PL/SQL procedure successfully completed.

SQL>
```

En su lugar, debe probar manualmente el número de filas eliminadas mediante **SQL%ROWCOUNT**.

```
SET SERVEROUTPUT ON
BEGIN
  DELETE FROM employees
  WHERE  employee_id = 9999;

  IF SQL%ROWCOUNT = 0 THEN
    -- Manually raise the NO_DATA_FOUND exception.
    RAISE NO_DATA_FOUND;
  END IF;
  DBMS_OUTPUT.put_line('NO_DATA_FOUND Not Raised');
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.put_line('NO_DATA_FOUND Raised');
END;
/
NO_DATA_FOUND Raised

PL/SQL procedure successfully completed.

SQL>
```

TRUNCATE TABLE

Si desea eliminar todas las filas de una tabla, la instrucción **TRUNCATE TABLE** es mucho más eficiente que la instrucción **DELETE**. La instrucción **TRUNCATE TABLE** es un **comando DDL**, por lo que incluye un **COMMIT implícito**, por lo que no hay forma de emitir un **ROLLBACK** si decide que no desea eliminar las filas.

En el siguiente ejemplo, verificamos el número de filas en la tabla, emitimos la instrucción **TRUNCATE TABLE**, inmediatamente **ROLLBACK** y verificamos nuevamente el número de filas en la tabla. Verá en la salida, **ROLLBACK** no cancela la instrucción **TRUNCATE TABLE**.

```
SELECT COUNT(*)
FROM   employees;

  COUNT(*)
----------
        14

1 row selected.

SQL> TRUNCATE TABLE employees;

Table truncated.

SQL> ROLLBACK;

Rollback complete.

SQL>

SELECT COUNT(*)
FROM   employees;

  COUNT(*)
----------
         0

1 row selected.

SQL>
```

La sentencia **TRUNCATE TABLE** puede eliminar el almacenamiento asociado con la tabla o dejarlo para reutilizarlo más tarde.

```
-- Elimina almacenamiento.
TRUNCATE TABLE employees;
TRUNCATE TABLE employees DROP STORAGE;

-- Mantener el almacenamiento.
TRUNCATE TABLE employees REUSE STORAGE;
```

PDF generated by Kalin's PDF Creation Station