Database Programming with SQL

13-3: Modifying a Table

Practice Activities

Objectives

* Explain why it is important to be able to modify a table
* •Explain and provide an example for each of the DDL statements—ALTER, DROP, RENAME, and TRUNCATE—and the effect each has on tables and columns
* •Construct a query and execute the ALTER TABLE commands ADD, MODIFY, and DROP
* •Explain and perform a FLASHBACK QUERY on a table
* •Explain and perform FLASHBACK table operations
* •Track the changes to data over a period of time
* •Explain the rationale for using TRUNCATE versus DELETE for tables
* •Add a comment to a table using the COMMENT ON TABLE command
* •Name the changes that can and cannot be made to modify a column
* •Explain when and why the SET UNUSED statement is advantageous

Try It / Solve It

Before beginning the practice exercises, execute a DESCRIBE for each of the following tables:

o\_employees, o\_departments and o\_jobs. These tables will be used in the exercises. If they do not

exist in your account, create them as follows:

***1. Create the three o\_tables – jobs, employees, and departments – using the syntax:***

***CREATE TABLE o\_jobs AS (SELECT \* FROM jobs);***

***CREATE TABLE o\_employees AS (SELECT \* FROM employees);***

***CREATE TABLE o\_departments AS (SELECT \* FROM departments);***

***2. Add the Human Resources job to the jobs table:***

***INSERT INTO o\_jobs (job\_id, job\_title, min\_salary, max\_salary)***

***VALUES('HR\_MAN', 'Human Resources Manager', 4500, 5500);***

***3. Add the three new employees to the employees table:***

***INSERT INTO o\_employees (employee\_id, first\_name, last\_name, email, hire\_date,***

***job\_id)***

***VALUES(210, 'Ramon', 'Sanchez', 'RSANCHEZ', SYSDATE, 'HR\_MAN');***

***4. Add Human Resources to the departments table:***

***INSERT INTO o\_departments(department\_id, department\_name)***

***VALUES (210,'Human Resources');***

You will need to know which columns do not allow null values.

1. Why is it important to be able to modify a table?

Ca sa poti face actualizari, if needed

2. CREATE a table called Artists.

a. Add the following to the table:

•artist ID

•first name

•last name

•band name

•email

•hourly rate

CREATE TABLE artists

(artist\_id NUMBER(5,0),

first\_name VARCHAR2(25) CONSTRAINT ait\_first\_name\_nn NOT NULL ENABLE,

last\_name VARCHAR2(30) CONSTRAINT ait\_last\_name\_nn NOT NULL ENABLE,

band\_name VARCHAR2(30),

email VARCHAR2(75) CONSTRAINT ait\_email\_nn NOT NULL ENABLE,

hr\_rate NUMBER(8,2) CONSTRAINT ait\_hr\_rate\_nn NOT NULL ENABLE,

song\_id NUMBER(5,0) CONSTRAINT ait\_song\_id\_nn NOT NULL ENABLE,

CONSTRAINT ait\_id\_pk PRIMARY KEY (artist\_id)

);

b. INSERT one artist from the d\_songs table.

Alter table artists

drop constraint alt\_email\_nn

alter table artists

drop constraint ait\_hr\_rate\_nn

INSERT INTO artists (artist\_id, first\_name, last\_name, band\_name, email, hr\_rate, song\_id)

SELECT 1 AS artist\_id,

CASE

WHEN artist IS NULL THEN 'first name unknown'

WHEN INSTR(artist,' ') = 0 THEN artist

ELSE SUBSTR(artist,1,INSTR(artist,' ') -1)

END

AS first\_name,

CASE

WHEN artist IS NULL THEN 'last name unknown'

WHEN INSTR(artist,' ') = 0 THEN artist

ELSE SUBSTR(artist,INSTR(artist,' '),LENGTH(artist))

END

AS last\_name,

artist as band\_name,

NULL as email,

NULL as hr\_rate,

id as song\_id

from d\_songs where rownum = 1

c. INSERT one artist of your own choosing.

ALTER TABLE artists

DROP CONSTRAINT ait\_song\_id\_nn;

**INSERT INTO artists (artist\_id, first\_name, last\_name, band\_name, email, hr\_rate, song\_id)**

**VALUES(2,'David','Gray','david''s band','some.email@somedomain.com','999999.99',NULL);**

**SELECT \* FROM artists;**

d. Give an example how each of the following may be used on the table that you have created:

1) ALTER TABLE

**ALTER TABLE artists**

**ADD (specialty VARCHAR2(100), college VARCHAR2(100));**

**ALTER TABLE artists**

**MODIFY (specialty VARCHAR2(99), college VARCHAR2(98));**

**ALTER TABLE artists**

**DROP COLUMN specialty;**

**ALTER TABLE artists**

**RENAME COLUMN college to university;**

**ALTER TABLE artists**

**RENAME TO artists\_new\_name;**

**ALTER TABLE artists\_new\_name**

**MODIFY (university VARCHAR2(98) DEFAULT 'Great College');**

**ALTER TABLE artists\_new\_name**

**SET UNUSED (university);**

SELECT column\_name FROM user\_tab\_columns WHERE LOWER(table\_name) = 'artists\_new\_name';

**ALTER TABLE artists\_new\_name**

**DROP UNUSED COLUMNS;**

2) DROP TABLE

3) RENAME TABLE

**RENAME artists\_new\_name TO artists\_new2;**

4) TRUNCATE

**TRUNCATE TABLE artists\_new2;**

SELECT \* FROM artists\_new2;

5) COMMENT ON TABLE

**COMMENT ON TABLE artists\_new2 IS 'comentariu';**

3. In your o\_employees table, enter a new column called “Termination.” The datatype for the new

column should be VARCHAR2. Set the DEFAULT for this column as SYSDATE to appear as

character data in the format: February 20th, 2003.

**ALTER TABLE o\_employees**

**ADD ("Termination" VARCHAR2(100)  DEFAULT  TO\_CHAR(SYSDATE,'Month ddth, YYYY') );**

INSERT INTO o\_employees (employee\_id, first\_name, last\_name, email, hire\_date, job\_id)

VALUES(213, 'Ramon213', 'Sanchez213', 'RSANCHEZ213', SYSDATE, 'HR\_MAN');

SELECT "Termination" FROM o\_employees WHERE employee\_id = 213;

4. Create a new column in the o\_employees table called start\_date. Use the TIMESTAMP WITH

LOCAL TIME ZONE as the datatype.

**ALTER TABLE o\_employees**

**ADD (start\_date TIMESTAMP WITH LOCAL TIME ZONE);**

**default fractional\_seconds\_precision is 6 here**

5. Truncate the o\_jobs table. Then do a SELECT \* statement.

Are the columns still there? Is the data still there?

DESCRIBE o\_jobs;

SELECT \* FROM  o\_jobs;

**TRUNCATE TABLE o\_jobs;**

DESCRIBE o\_jobs;

SELECT \* FROM  o\_jobs

coloanele sunt tot acolo, datele s-au dus.

6. What is the distinction between TRUNCATE, DELETE, and DROP for tables?

Drop> sterge definitia tabelului si datele din acesta

Truncate> Sterge toate LINIILE (atentie) si elibereaza spatiul fara sa mai poti face rollback; e mai rapid ca Delete

Delete> Sterge liniile, nu elibereaza spatiul.

7. List the changes that can and cannot be made to a column.

**-1. Newly created column is always put at last. But select can be written to return in desired manner, so no issues.**

**0. Adding new column to table will always give null to pre-existing row's new field, even if, default value is assigned to new column.**

**1. I can increase precision of a number column.**

**2. In can increase length of character column.**

**3. I can decrease precision of number column if: it contains only nulls till now or there is no row in table. Otherwise I will get ORA-00940: invalid ALTER command**

**4. varchar2 can be decreased to length down to the largest value present currently in all rows.**

**5. Datatype can be changed altogether if all values in this column are nulls.**

**6. char can become varchar2 if column contain nulls or the size given is not less than any existing field for that column.**

**6. varchar2  can become char if column contain nulls or the size given is not less than any existing field for that column.**

**7. Change in default value is effective to new inserts only not the already present rows.**

**8. A column containing values can be dropped if this is not referenced as foreign key in further tables. Also, data values in it not recovered after column drop.**

**9. I can drop only one column at a time. Also, at least one column must remain, I cannot drop last column.**

**10. Since dropping column may take time, it does modify each row before deleting, I can use SET UNUSED command as a replacement for practical purposes and DROP UNUSED later on.**

**11. I can rename a column if I want.**

8. Add the following comment to the o\_jobs table:

"New job description added"

View the data dictionary to view your comments.

**COMMENT ON TABLE o\_jobs IS 'New job description added';**

**SELECT table\_name, comments**

**FROM user\_tab\_comments WHERE LOWER(table\_name) = 'o\_jobs';**

9. Rename the o\_jobs table to o\_job\_description.

**ALTER TABLE o\_jobs**

**RENAME TO o\_job\_description;**

**RENAME o\_job\_description TO o\_job\_description2**

10. F\_staffs table exercises:

a. Create a copy of the f\_staffs table called copy\_f\_staffs and use this copy table for the

remaining labs in this lesson.

**CREATE TABLE copy\_f\_staffs**

**AS ( SELECT \* FROM f\_staffs);**

DESCRIBE f\_staffs;

DESCRIBE copy\_f\_staffs;

SELECT \* FROM f\_staffs;

SELECT \* FROM copy\_f\_staffs;

b. Describe the new table to make sure it exists.

**DESCRIBE copy\_f\_staffs;**

c. Drop the table.

**DROP TABLE copy\_f\_staffs;**

d. Try to select from the table.

**SELECT \* FROM copy\_f\_staffs;**

e. Investigate your recyclebin to see where the table went.

**DESCRIBE user\_recyclebin ;**

**Please note droptime is varchar2 here.**

**SELECT \* FROM**

**(SELECT \* FROM user\_recyclebin ORDER BY droptime DESC)**

**WHERE ROWNUM <= 100;**

**SELECT object\_name,droptime FROM user\_recyclebin  WHERE LOWER(original\_name) = 'copy\_f\_staffs';**

apoi

**SELECT object\_name FROM user\_recyclebin  WHERE LOWER(original\_name) = 'copy\_f\_staffs' AND droptime = '2016-11-02:20:14:25';**

f. Try to select from the dropped table by using the value stored in the OBJECT\_NAME column.

You will need to copy and paste the name as it is exactly, and enclose the new name in “ “

(double quotes). So if the dropped name returned to you is

BIN$Q+x1nJdcUnngQESYELVIdQ==$0, you need to write a query that refers to

“BIN$Q+x1nJdcUnngQESYELVIdQ==$0”.

**SELECT \* FROM "BIN$QF30ctmEV7jgU81jFJDpGA==$0";**

g. Undrop the table.

**FLASHBACK TABLE copy\_f\_staffs TO BEFORE DROP;**

h. Describe the table.

**DESCRIBE copy\_f\_staffs;**

11. Still working with the copy\_f\_staffs table, perform an update on the table.

a. Issue a select statement to see all rows and all columns from the copy\_f\_staffs table;

**SELECT \* FROM copy\_f\_staffs;**

b. Change the salary for Sue Doe to 12 and commit the change.

**UPDATE copy\_f\_staffs**

**SET salary = 12**

**WHERE first\_name = 'Sue' AND last\_name = 'Doe';**

c. Issue a select statement to see all rows and all columns from the copy\_f\_staffs table;

**SELECT \* FROM copy\_f\_staffs;**

d. For Sue Doe, update the salary to 2 and commit the change.

**UPDATE copy\_f\_staffs**

**SET salary = 2**

**WHERE first\_name = 'Sue' AND last\_name = 'Doe';**

e. Issue a select statement to see all rows and all columns from the copy\_f\_staffs table;

**SELECT \* FROM copy\_f\_staffs;**

f. Now, issue a FLASHBACK QUERY statement against the copy\_f\_staffs table, so you can see

all the changes made.

**SELECT versions\_operation, versions\_starttime,  versions\_endtime, id, first\_name, last\_name, birthdate, salary,overtime\_rate,training,staff\_type,manager\_id, manager\_budget,manager\_target**

**FROM copy\_f\_staffs**

**VERSIONS BETWEEN SCN MINVALUE AND MAXVALUE**

**WHERE id = 12;**

g. Investigate the result of f), and find the original salary and update the copy\_f\_staffs table salary

column for Sue Doe back to her original salary.**UPDATE copy\_f\_staffs**

**SET salary = (SELECT salary**

**FROM copy\_f\_staffs**

**VERSIONS BETWEEN SCN MINVALUE AND MAXVALUE**

**WHERE first\_name = 'Sue' AND last\_name = 'Doe' AND versions\_operation IS NULL AND versions\_starttime IS NULL)**

**WHERE first\_name = 'Sue' AND last\_name = 'Doe';**

might be incomplete (only g)