
ORACLE
Academy

Database Programming with SQL

13-3
Modifying a Table

ORACLE Academy

Objectives

This lesson covers the following 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 FLASHBACK QUERY on a table
- Explain and perform FLASHBACK table operations

Objectives

This lesson covers the following objectives:

- 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

Purpose

- Remember the statement, "There is nothing permanent except change"?
- Wouldn't it be nice if we never made mistakes or needed to change anything?
- As you know by now, databases are dynamic entities.
- They probably wouldn't be very useful if they couldn't be changed.

Purpose

- Up to now, you've created tables and made changes to the row data inside tables, but how do you make changes to the tables themselves?
- This lesson presents the DDL commands that are used to alter, rename, empty, or simply eliminate a table altogether.

ALTER TABLE

- ALTER TABLE statements are used to:
  - Add a new column
  - Modify an existing column
  - Define a DEFAULT value for a column
  - Drop a column
- You can add or modify a column in a table, but you cannot specify where the column appears.

ALTER TABLE

- A newly added column always becomes the last column of the table.
- Also, if a table already has rows of data and you add a new column to the table, the new column is initially null for all of the pre-existing the rows.

ALTER TABLE: Adding a Column

- To add a new column, use the SQL syntax shown:

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

- For example:

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

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

ALTER TABLE: Modifying a Column

- Modifying a column can include changes to a column's data type, size, and DEFAULT value.
- Rules and restrictions when modifying a column are:
  - You can increase the width or precision of a numeric column.
  - You can increase the width of a character column.
  - You can decrease the width of a NUMBER column if the column contains only null values, or if the table has no rows.
  - For VARCHAR types, you can decrease the width down to the largest value contained in the column.

ALTER TABLE: Modifying a Column

- You can change the data type only if the column contains null values.
- You can convert a CHAR column to VARCHAR2 or convert a VARCHAR2 COLUMN to CHAR only if the column contains null values, or if you do not change the size to something smaller than any value in the column.
- A change to the DEFAULT value of a column affects only later insertions to the table.

ALTER TABLE: Modifying a Column Example

- Example: a table has been created with two columns:

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

- Which of these modification would be allowed, and which would not? (Consider your answers both with and without rows of data in the table.)

```
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);
```

ALTER TABLE: Modifying a Column Example

- Would be permitted only if columns were empty, or the largest name was 10 or less characters

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

- Would be permitted with or without data as column width increased.

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

ALTER TABLE: Modifying a Column Example

- Would be permitted with or without data as column precision increased.

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

- Would be permitted with or without data as only a DEFAULT value added.

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

ALTER TABLE: Dropping a Column

- When dropping a column the following rules apply:
  - A column containing data may be dropped.
  - Only one column can be dropped at a time.
  - You can't drop all of the columns in a table; at least one column must remain.
  - Once a column is dropped, the data values in it cannot be recovered.

ALTER TABLE: Dropping a Column

- SQL Syntax:

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

- For Example:

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

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

SET UNUSED Columns

- Dropping a column from a large table can take a long time.
- A quicker alternative is to mark the column as unusable.
- The column values remain in the database but cannot be accessed in any way, so the effect is the same as dropping the column.
- In fact, you could add a new column to the database with the same name as the unused column.
- The unused columns are there, but invisible!
- Syntax:

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

SET UNUSED Columns Example

- Example:

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

- DROP UNUSED COLUMNS removes all columns currently marked as unused.
- You use this statement when you want to reclaim the extra disk space from unused columns in a table.
- Example:

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

ALTER TABLE Summarized

- This chart summarizes the uses of the ALTER TABLE command:

Syntax	Outcomes	Concerns
ALTER TABLE tablename ADD (column name data type [DEFAULT expression], column name data type [DEFAULT expression], …	Adds a new column to a table	You cannot specify where the column is to appear in the table. It becomes the last column.
ALTER TABLE tablename MODIFY (column name data type [DEFAULT expression], column name data type, …	Used to change a column's data type, size, and default value	A change to the default value of a column affects only subsequent insertions to the table.
ALTER TABLE tablename DROP COLUMN column name;	Used to drop a column from a table	The table must have at least one column remaining in it after it is altered. Once dropped, the column cannot be recovered.
ALTER TABLE tablename SET UNUSED (column name);	Used to mark one or more columns so they can be dropped later	Does not restore disk space. Columns are treated as if they were dropped.
ALTER TABLE tablename DROP UNUSED COLUMNS	Removes from the table all columns currently marked as unused	Once set unused, there is no access to the columns; no data displayed using DESCRIBE. Permanent removal; no rollback.

DROP TABLE

- The DROP TABLE statement removes the definition of an Oracle table.
- The database loses all the data in the table and all the indexes associated with it.
- When a DROP TABLE statement is issued:
  - All data is deleted from the table.
  - The table's description is removed from the Data Dictionary.
- The Oracle Server does not question your decision and it drops the table immediately.

DROP TABLE

- In the next slide, you will see that you may be able to restore a table after it is dropped, but it is not guaranteed.
- Only the creator of the table or a user with DROP ANY TABLE privilege (usually only the DBA) can remove a table.
- Syntax:

```
DROP TABLE tablename;
```

- Example:

```
DROP TABLE copy_employees;
```

FLASHBACK TABLE

- If you drop a table by mistake, you may be able to bring that table and its data back.
- Each database user has his own recycle bin into which dropped objects are moved, and they can be recovered from here with the FLASHBACK TABLE command.
- This command can be used to restore a table, a view, or an index that was dropped in error.
- The Syntax is:

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

FLASHBACK TABLE

- For example, if you drop the EMPLOYEES table in error, you can restore it by simply issuing the command:

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

- As the owner of a table, you can issue the flashback command, and if the table that you are restoring had any indexes, then these are also restored.
- It is possible to see which objects can be restored by querying the data dictionary view USER_RECYCLEBIN.

FLASHBACK TABLE

- The USER_RECYCLEBIN view can be queried like all other data dictionary views:

```
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

- Once a table has been restored by the FLASHBACK TABLE command, it is no longer visible in the USER_RECYCLEBIN view.
- Any indexes that were dropped with the original table will also be restored.
- It may be necessary (for security reasons) to completely drop a table, bypassing the recycle bin.
- This can be done by adding the keyword PURGE.

```
DROP TABLE copy_employees PURGE;
```

RENAME

- To change the name of a table, use the RENAME statement.
- This can be done only by the owner of the object or by the DBA.
- Syntax:

```
RENAME old_name to new_name;
```

- Example:

```
RENAME my_cd_collection TO my_music;
```

- We will see later that we can rename other types of objects such as views, sequences, and synonyms.

TRUNCATE

- Truncating a table removes all rows from a table and releases the storage space used by that table.
- When using the TRUNCATE TABLE statement:
  - You cannot roll back row removal.
  - You must be the owner of the table or have been given DROP ANY TABLE system privileges.

TRUNCATE

- Syntax:

```
TRUNCATE TABLE tablename;
```

- The DELETE statement also removes rows from a table, but it does not release storage space.
- TRUNCATE is faster than DELETE because it does not generate rollback information.

COMMENT ON TABLE

- You can add a comment of up to 2,000 characters about a column, table, or view by using the COMMENT statement.
- Syntax:

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

- Example:

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

COMMENT ON TABLE

- To view the table comments in the data dictionary:

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

TABLE_NAME	COMMENTS
EMPLOYEES	Western Region Only

- If you want to drop a comment previously made on a table or column, use the empty string(''):

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

FLASHBACK QUERY

- You may discover that data in a table has somehow been inappropriately changed.
- Luckily, Oracle has a facility that allows you to view row data at specific points in time, so you can compare different versions of a row over time.
- This facility is very useful.
- Imagine, for instance, that someone accidently performs some DML on a table, and then executes a COMMIT on those changes.
- Oracle Application Express commits automatically, so mistakes are easily made.

FLASHBACK QUERY

- You can use the FLASHBACK QUERY facility to examine what the rows looked like BEFORE those changes were applied.
- When Oracle changes data, it always keeps a copy of what the amended data looked like before any changes were made.
- So it keeps a copy of the old column value for a column update, it keeps the entire row for a delete, and it keeps nothing for an insert statement.

FLASHBACK QUERY

- These old copies are held in a special place called the UNDO tablespace.
- Users can access this special area of the Database using a flashback query.
- You can look at older versions of data by using the VERSIONS clause in a SELECT statement.

FLASHBACK QUERY

- For example:

```
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;
```

- The SCN number referred to in the above query means the System Change Number and is a precise identification of time in the database.
- It is a sequential number incremented and maintained by the database itself.

FLASHBACK QUERY

- The best way to demonstrate FLASHBACK QUERY is with an example.
- The contents are as follows for employee_id 1 in the employees table.

```
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 data found

FLASHBACK QUERY

- Then we create the employee:

```
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

- Then you can update the row:

```
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

- Then you can delete the row:

```
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

- The result from the last query on the previous slide is only available when using Flashback query, i.e. the VERSIONS clause.
- If you attempt a normal search from employee_id = 1 following the delete statement, you would have received the normal error, No Data Found.

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

Terminology

Key terms used in this lesson included:

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

Terminology

Key terms used in this lesson included:

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

Summary

In this lesson you have learned to:

- 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 FLASHBACK QUERY on a table
- Explain and perform FLASHBACK table operations

Summary

In this lesson you should have learned how to:

- 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


ORACLE
Academy