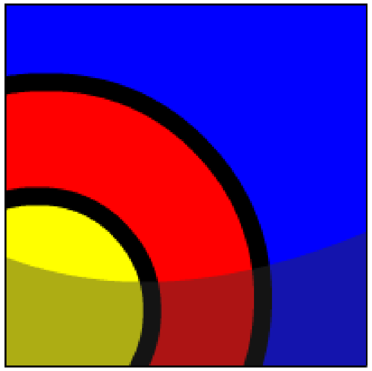
Modifying a Table

What Will I Learn?

In this lesson, you will learn to:

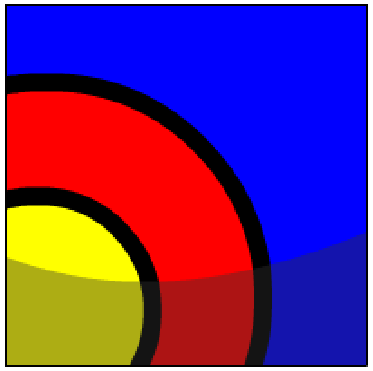


- Explain why it is important to be able to modify a table
- Explain and provide an example for each of the DDL statements ALTER, DROP, RENAME, and TRUNCATE and the effect each has on tables and columns
- Construct a query and execute the ALTER TABLE commands ADD, MODIFY, and DROP
- Explain and perform FLASHBACK TABLE operations
- Explain and perform FLASHBACK QUERY on a table

What Will I Learn?

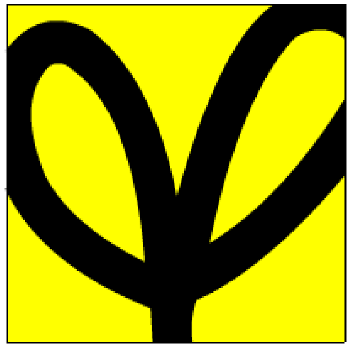
In this lesson, you will learn to:

- Explain the rationale for using TRUNCATE vs. 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



Why Learn It?

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.



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.

Tell Me / Show Me

ALTER TABLE

ALTER TABLE statements are used to:

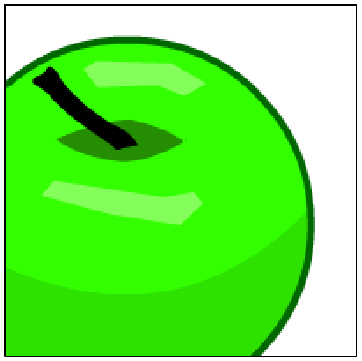
- Add a new column
- Modify an existing column
- Define a DEFAULT value for a column
- Drop a column.

You can add or modify a column in a table, but you cannot specify where the column appears. 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 the rows.

Tell Me / Show Me

ALTER TABLE: ADDING A COLUMN



To add a new column, use the SQL syntax shown.

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

For example:

```
ALTER TABLE copy_f_staffs
ADD (hire_date DATE DEFAULT SYSDATE);

ALTER TABLE copy_f_staffs
ADD (e_mail_address    VARCHAR2(80));
```

Tell Me / Show Me

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 column only if the column contains only null values or if the table has no rows.

Tell Me / Show Me

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.
- A change to the DEFAULT value of a column affects only later insertions to the table.

Tell Me / Show Me

ALTER TABLE: MODIFYING A COLUMN

Example: a table has been created with two columns:

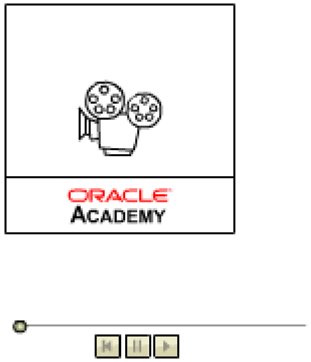
```
CREATE TABLE mod_emp
  (last_name       VARCHAR2(20),
   salary           NUMBER(8,2));
```

Which of these modifications would be allowed, and which would not?

1. **ALTER TABLE mod_emp MODIFY (last_name VARCHAR2(30));**
2. **ALTER TABLE mod_emp MODIFY (last_name VARCHAR2(10));**
3. **ALTER TABLE mod_emp MODIFY (salary NUMBER(10,2));**
4. **ALTER TABLE mod_emp MODIFY (salary NUMBER(8,2) DEFAULT 50);**

Tell Me / Show Me

ALTER TABLE: **DROPPING A COLUMN**



When dropping a column the following rules apply:

- A column to be dropped may or may not contain data.
- 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.

SQL syntax:

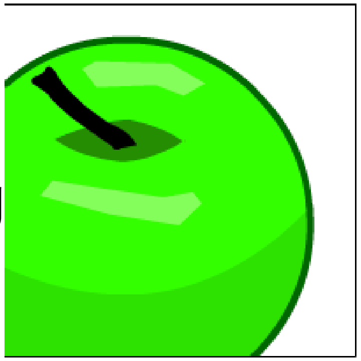
ALTER TABLE tablename DROP COLUMN column name;

For example:

ALTER TABLE copy_f_staffs DROP COLUMN manager_target;

Tell Me / Show Me

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 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:

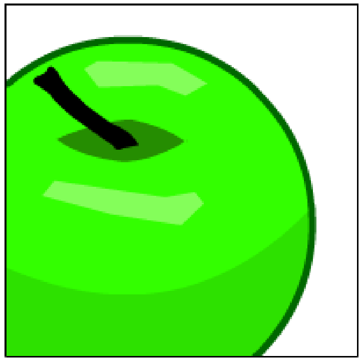
```
ALTER TABLE tablename SET UNUSED (column name);
```

Tell Me / Show Me

SET UNUSED COLUMNS

Example:

```
ALTER TABLE copy_f_staffs
    SET UNUSED (manager_budget);
```



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.

```
ALTER TABLE tablename DROP UNUSED
COLUMNS;
```

Example:

```
ALTER TABLE copy_f_staffs DROP UNUSED
COLUMNS;
```

Tell Me / Show Me

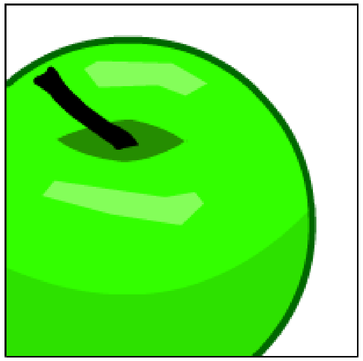
ALTER TABLE

This chart summarizes the uses of the ALTER TABLE command.

Syntax	**Outcomes**	**Concerns**
ALTER TABLE tablename ADD (column name datatype [DEFAULT expression], column name datatype [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 datatype [DEFAULT expression], column name datatype, …	Used to change a column's datatype, 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.

Tell Me / Show Me

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
- It may be irreversible! The Oracle Server does not question your decision.

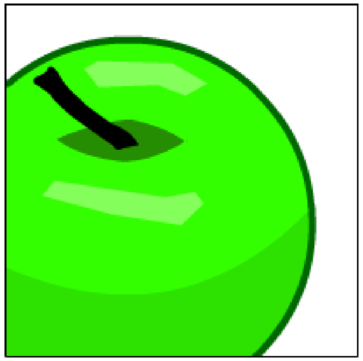
Tell Me / Show Me

DROP TABLE

Only the creator of the table or a user with DROP ANY TABLE privilege (usually only the DBA) can remove a table.

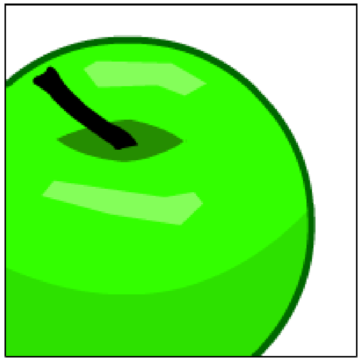
```
DROP TABLE tablename;
```

```
DROP TABLE copy_f_staffs;
```



Tell Me / Show Me

FLASHBACK TABLE



If you drop a table by mistake, in Oracle 10g, there is a way of bringing that table and its data back.

Each database user has their own recyclebin into which dropped objects are now moved, and they can be recovered from here with the FLASHBACK TABLE command.

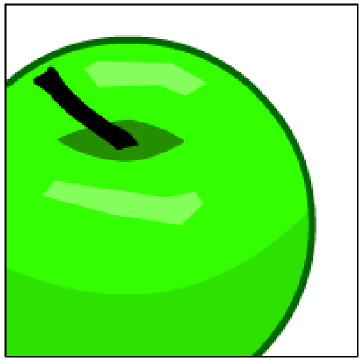
This command can be used to restore a table, a view or an index that was dropped in error.

The Syntax is quite simple:

FLASHBACK TABLE tablename TO BEFORE DROP;

Tell Me / Show Me

FLASHBACK TABLE



So if you have dropped the EMPLOYEES table in error, you can restore it by simply issuing the command:

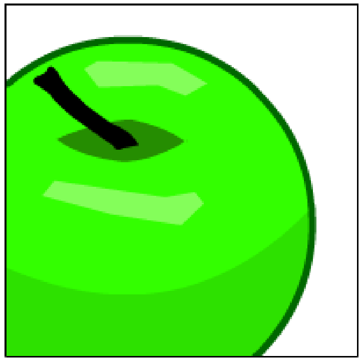
```
FLASHBACK TABLE employees TO BEFORE DROP;
```

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 what objects you have that you can restore, by simply querying the data dictionary view USER_RECYCLEBIN.

Tell Me / Show Me

FLASHBACK TABLE



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

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

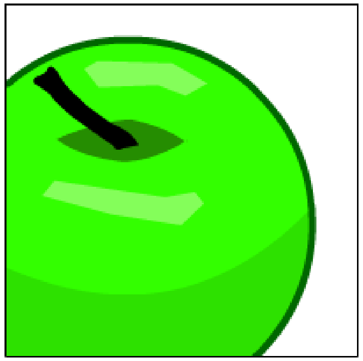
ORIGINAL_NAME	OPERATION	DROPTIME
EMPLOYEES	DROP	2007-12-05:12.34.24
EMP_PK	DROP	2007-12-05:12.34.24

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 along with the original table, will also have been restored.

Tell Me / Show Me

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.

```
RENAME old_name to new_name;
```

Example:

```
RENAME copy_f_staffs to copy_fastfood_staffs;
```

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

Tell Me / Show Me

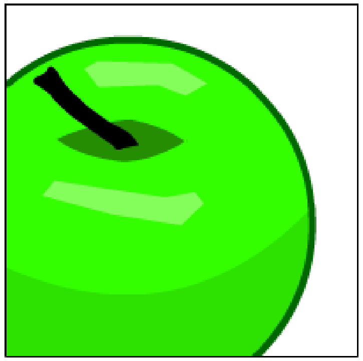
TRUNCATE

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.

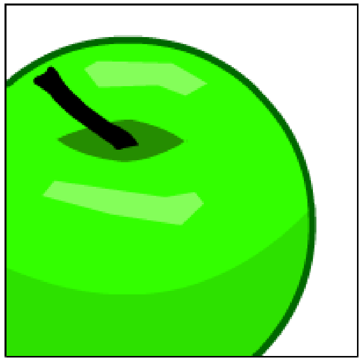
Syntax: TRUNCATE TABLE tablename;

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



Tell Me / Show Me

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. The comment is stored in the data dictionary and can be viewed in one of the following data dictionary views in the COMMENTS column:

Comments on columns:

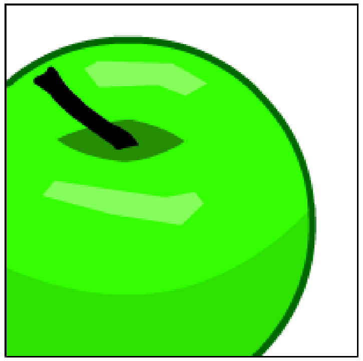
- ALL_COL_COMMENTS
- USER_COL_COMMENTS

Comments on tables:

- ALL_TAB_COMMENTS
- USER_TAB_COMMENTS

Tell Me / Show Me

COMMENT ON TABLE: EXAMPLES



Syntax:

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

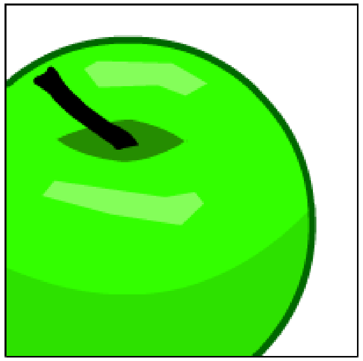
Example:

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

Tell Me / Show Me

COMMENT ON TABLE: EXAMPLES

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



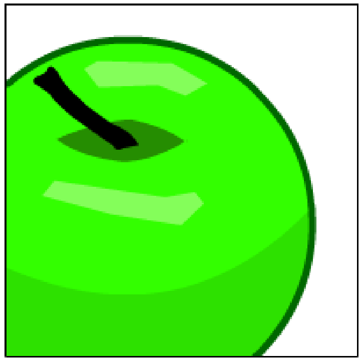
```
TABLE_NAME       COMMENTS
--------------------     ------------------------------
EMPLOYEES        Western Region only
```

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

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

Tell Me / Show Me

FLASHBACK QUERY



You may discover that somehow data in a table has been inappropriately changed.

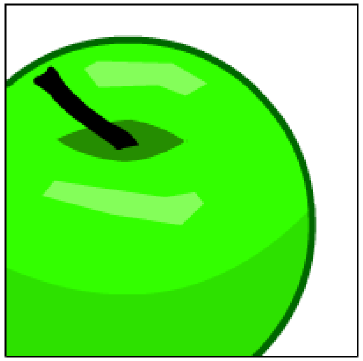
Luckily, Oracle has a facility that allows us to view row data at specific points in time, so we can compare different versions of a row over time.

This facility is very useful. Imagine for instance that someone accidently performed some DML on a table, and COMMIT'ed those changes. Oracle Application Express commits automatically, so mistakes are easily made.

We can use this facility to look at what the rows looked like BEFORE those changes were applied.

Tell Me / Show Me

FLASHBACK QUERY

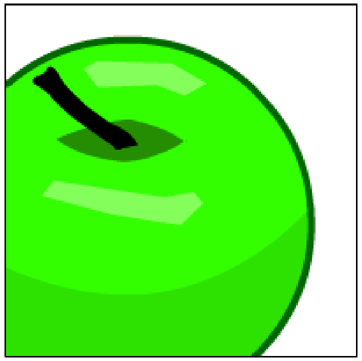


When Oracle changes data, it always keeps a copy of what the amended data looked like before any changes were made. So it keeps a copy of the old column value for a column update, it keeps the entire row for a delete and it keeps nothing for a insert statement.

These old copies are held in a special place called the UNTO tablespace. Users have no normal access to this area of the Database, but we can use the data it contains for FLASHBACK QUERIES.

Tell Me / Show Me

FLASHBACK QUERY



A new clause is used in a SELECT statement to allow us to look at older versions of the rows: VERSIONS.

A typical flashback query might look like this:

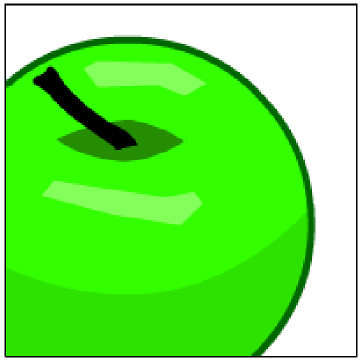
```
SELECT versions_starttime AS "START_DATE",
       versions_endtime  AS "END_DATE",
       salary
FROM employees
  VERSIONS BETWEEN SCN MINVALUE AND MAXVALUE
WHERE  department_id = 90;
```

Tell Me / Show Me

FLASHBACK QUERY

This is easiest demonstrated via an example:

The contents of the employees is as follows for department_id 90.



LAST_NAME	SALARY	DEPARTMENT_ID
King	24000	90
Kochhar	17000	90
De Haan	17000	90

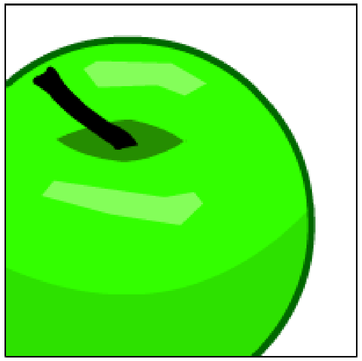
```
update employees
set salary = 1
where department_id = 90;
commit;
```

LAST_NAME	SALARY	DEPARTMENT_ID
King	1	90
Kochhar	1	90
De Haan	1	90

Tell Me / Show Me

FLASHBACK QUERY

```
SELECT versions_starttime "START_DATE",
          versions_endtime   "END_DATE",
          salary
FROM   employees
  VERSIONS BETWEEN SCN MINVALUE AND MAXVALUE
WHERE  department_id=90
```

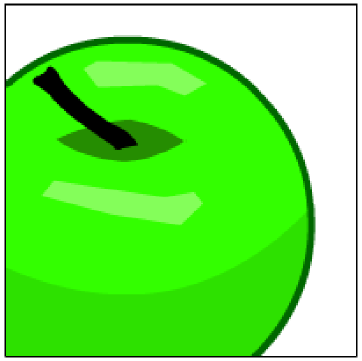


Rows from the ‘old’ updated version are in red in the output table below:

START_DATE	END_DATE	SALARY
11-DEC-07 05.44.24 AM	-	1
11-DEC-07 05.44.24 AM	-	1
11-DEC-07 05.44.24 AM	-	1
-	11-DEC-07 05.44.24 AM	2400
-	11-DEC-07 05.44.24 AM	17000
-	11-DEC-07 05.44.24 AM	17000

Tell Me / Show Me

FLASHBACK QUERY



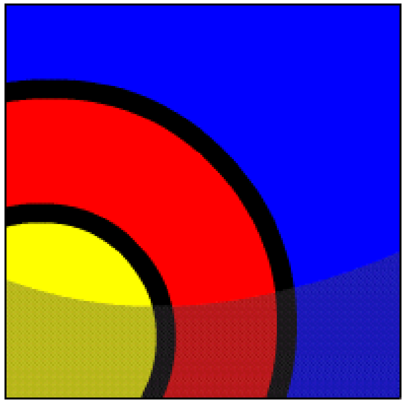
So we could use a FLASHBACK QUERY to look at the data, as it was BEFORE the update took place, and use that to manually update the rows back to the way they were before the update was committed.

Oracle is getting the FLASHBACK QUERY data from the “before images” held in the database in the UNDO tablespace. Typically, this UNDO data is only available for up to 15 minutes after the transaction was committed. Once the UNDO data has been overwritten by Oracle, Flashback queries are no longer possible, so if you want to use this functionality, you will have to do it quickly after the changes were committed.

Summary

In this lesson you have learned to:

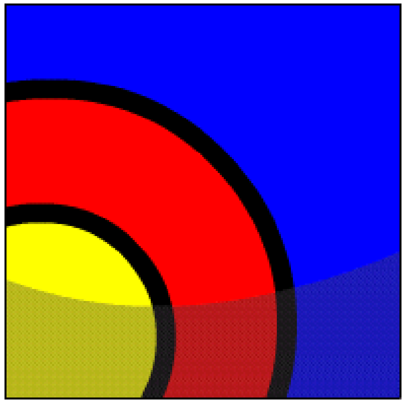
- Explain why it is important to be able to modify a table
- Explain and provide an example for each of the DDL statements ALTER, DROP, RENAME and TRUNCATE and the effect each has on tables and columns
- Construct a query and execute the ALTER TABLE commands ADD, MODIFY and DROP
- Explain and perform FLASHBACK TABLE operations
- Explain and perform FLASHBACK QUERY on a table



Summary

In this lesson you have learned to:

- Explain the rationale for using TRUNCATE vs. 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



Summary

Practice Guide

The link for the lesson practice guide can be found in the course resources in Section 0.