ITC 5104 Relational Database Design and SQL

LECTURE 3

CHAPTER 3 ORACLE 12C: SQL

TABLE CREATION AND MANAGEMENT

Objectives

Explanation for lab exercise 3

Create a table using the CREATE TABLE command

Identify the table and structure

Use a subquery to create a table

Add a column to an existing table

Modify the size of a column in an existing table

Mark a column as unused, then delete it at a later time

Rename a table

Truncate a table

Drop a table

Purge tables from recyclebin

Flashback table to recover a table

Introduction

We have already used some basic SQL to retrieve existing data. We will now use it to create and modify tables

Commands that are used to create and modify database tables are called data definition language (DDL) commands

These commands are SQL commands that are specifically used to *create* and *modify database objects*

A *database object* is a defined, self-contained structure in Oracle

The next few slides shows the SQL commands we'll be using for table creation and modification

Table Commands - Create

Creating Tables	
CREATE TABLE	Creates a new table in the database. The user names the columns and identifies the type of data to be stored. To view a table, use the SQL*PLUS command DESCRIBE.
CREATE TABLE...AS	Creates a table from existing database tables, using the AS clause and subqueries.

Table Commands - Modify

Modifying Tables	
ALTER TABLE... ADD	Adds a column to a table.
ALTER TABLE... MODIFY	Changes a column size, datatype, or default value.
ALTER TABLE... DROP COLUMN	Deletes one column from a table.
ALTER TABLE... SET UNUSED *or* SET UNUSED COLUMN	Marks a column for deletion at a later time.
DROP UNUSED COLUMNS	Completes the deletion of a column previously marked with SET UNUSED.
RENAME...TO	Changes a table name.
TRUNCATE TABLE	Deletes all table rows, but table name and column structure remain.

Table Commands – Deleting and Recovering

Deleting Tables	
DROP TABLE	Removes an entire table from the Oracle 10g database.
PURGE TABLE	Permanently deletes a table in the recyclebin.
Recovering Tables	
FLASHBACK TABLE . . . TO BEFORE DROP	Recovers a dropped table if PURGE option not used when table dropped.

Table Design

Before you can actually create a table you must choose the table's name, and determine its structure (which columns will be included in your table)

You also need to choose the width of the character and numeric columns

So, we'll assume that, at this point, we've already done an ERD to define our tables and the columns that will appear in each table, will discuss this shortly

Oracle imposes certain rules on names used for objects including tables and columns

Table Design

Once the contents of the table has been determined the columns can be designed

You must define each of the columns for a table

- Choose a name for each column
- Determine the type of data to store in each column
- Determine in some case the maximum width of a column

Oracle Naming Conventions - Refresher

Names of tables and columns can be up to 30 characters in length

Must *begin with a letter* A – Z or a – z

May also include numbers, the underscore (_) character and the # sign is table and column names

No blank spaces are permitted in a table or column name

Each table owned by a user will have a unique table name and the column names within should also be unique to that table (different users can have tables with the same names as yours)

In Oracle *reserved words* like, SELECT, DISTINCT, CHAR or NUMBER *cannot be used for names*

Oracle Datatypes – Including Other Oracle Less Used Datatypes

Datatype	Description
VARCHAR2(n)	**Var**iable-length **char**acter data, where n represents the maximum length of the column. Maximum size is 4000 characters. There is no default size for this datatype; a minimum value must be specified. *Example*: VARCHAR2(9) can contain up to nine letters, numbers, or symbols.
CHAR(n)	Fixed-length character column, where n represents the length of the column. Default size is 1. Maximum size is 2000 characters. *Example:* CHAR(9) can contain nine letters, numbers, or symbols. However, if fewer than nine are entered, spaces are added to the right to force the data to reach a length of nine.
NUMBER(p,s)	Numeric column, where p indicates **precision**, or the total number of digits to the left and right of the decimal position, to a maximum of 38 digits; and s, or **scale**, indicates the number of positions to the right of the decimal. *Example:* NUMBER(7, 2) can store a numeric value up to 99999.99. If precision or scale is not specified, the column defaults to a precision of 38 digits.
DATE	Stores date and time between January 1, 4712 B.C. and December 31, 9999 A.D. Seven bytes are allocated to the column to store the century, year, month, day, hour, minute, and second of a date. Oracle 10*g* displays the date in the format DD-MON-YY. Other aspects of a date can be displayed by using the TO_CHAR format. The width of the field is predefined by Oracle 10*g* as seven bytes.

Table Creation Statement

```
CREATE TABLE [schema] tablename
      (columnname datatype [DEFAULT value],
      [columnname datatype [DEFAULT value], …);
```

Schema	Is the owner's name, defaults to current user
Tablename	Is the name of the table
DEFAULT value	Specifies a default value if a value is omitted in the INSERT statement
Columnname	Is the name of the column
Datatype	Is the column's datatype and length

Table Creation

To create a table, you must have CREATE TABLE privileges (already given to you on your own account)

To create a table in someone else's schema (account) you have to be granted *permission* or have the *privilege* of the CREATE TABLE *command* for that user's schema

The *column list is enclosed in parentheses*

Commas separate the column definitions

The CREATE TABLE command also allows for a *default value to be assigned to a column.* This value will automatically be stored by Oracle if the user makes no entry into that column

Table Creation Example - Errors

Worksheet | Query Builder

```
1  CREATE TABLE employees1
2  (
3    empno       NUMBER(4),
4    ename       VARCHAR2(8),
5    init        VARCHAR2(5),
6    job         VARCHAR2(8),
7    mgr         NUMBER(4),
8    bdate       DATE,
9    msal        NUMBER(6,2),
10   comm        NUMBER(6,2),
11   deptno      NUMBER(2)
12 );
```

Script Output

Task completed in 0.078 seconds

```
table EMPLOYEES1 created.
```

- If the creation is successful, Oracle will respond with the message CREATE TABLE succeeded.
- I used the name EMPLOYEES1 since EMPLOYEES already exists in my schema

Table Creation Example - Errors

Worksheet | Query Builder

```
1  CREATE TABLE employees1
2  (
3    empno       NUMBER(4),
4    ename       VARCHAR2(8),
5    init        VARCHAR2(5),
6    job         VARCHAR2(8),
7    mgr         NUMBER(4),
8    bdate       DATE,
9    msal        NUMBER(6,2),
10   comm        NUMBER(6,2),
11   deptno      NUMBER(2)
12 );
```

Script Output

Task completed in 0.172 seconds

```
Error starting at line : 1 in command -
CREATE TABLE employees1
(
  empno       NUMBER(4),
  ename       VARCHAR2(8),
  init        VARCHAR2(5),
  job         VARCHAR2(8),
  mgr         NUMBER(4),
  bdate       DATE,
  msal        NUMBER(6,2),
  comm        NUMBER(6,2),
  deptno      NUMBER(2)
)
Error at Command Line : 1 Column : 14
Error report -
SQL Error: ORA-00955: name is already used by an existing object
00955. 00000 -  "name is already used by an existing object"
*Cause:
*Action:
```

- If you try to create the table a second time, or create a table with the same name, you will receive the above error
- Several of you managed to do this, not a problem just tells you the object already exists in your schema

Table Creation Example - Errors

Worksheet | Query Builder

```
1  CREATE TABLE employees1
2  (
3    empno       NUMBER(4),
4    ename       VARCHAR2(8)
5    init        VARCHAR2(5),
6    job         VARCHAR2(8),
7    mgr         NUMBER(4),
8    bdate       DATE,
9    msal        NUMBER(6,2),
10   comm        NUMBER(6,2),
11   deptno      NUMBER(2)
12 );
```

Script Output | Task completed in 0.156 seconds

```
Error starting at line : 1 in command -
CREATE TABLE employees1
(
  empno       NUMBER(4),
  ename       VARCHAR2(8)
  init        VARCHAR2(5),
  job         VARCHAR2(8),
  mgr         NUMBER(4),
  bdate       DATE,
  msal        NUMBER(6,2),
  comm        NUMBER(6,2),
  deptno      NUMBER(2)
)
Error at Command Line : 5 Column : 3
Error report -
SQL Error: ORA-00907: missing right parenthesis
00907. 00000 -  "missing right parenthesis"
*Cause:
*Action:
```

- On line 4 I omitted the comma at the end of the line, this is the error produced
- Notice the red squiggly line at the end of line 4
- Some error messages can be a little misleading

Viewing a List of Tables: Data Dictionary

Worksheet | Query Builder

```
1 SELECT table_name
2 FROM user_tables;
```

Query Result

SQL | All Rows Fetched: 10 in 0.328 seconds

	TABLE_NAME
1	COURSES
2	CUSTOMERS
3	DEPARTMENTS
4	EMPLOYEES
5	EMPLOYEES1
6	HISTORY
7	OFFERINGS
8	REGISTRATIONS
9	SALES
10	SALGRADES

You can query the data dictionary to verify all existing tables in your schema

Good method to begin exploring an existing database

Viewing a List of Tables: Data Dictionary

Worksheet | Query Builder

```
1  SELECT *
2  FROM tab;
```

Query Result

SQL | All Rows Fetched: 10 in 0.234 seconds

	TNAME	TABTYPE	CLUSTERID
1	SALGRADES	TABLE	(null)
2	SALES	TABLE	(null)
3	REGISTRATIONS	TABLE	(null)
4	OFFERINGS	TABLE	(null)
5	HISTORY	TABLE	(null)
6	EMPLOYEES1	TABLE	(null)
7	EMPLOYEES	TABLE	(null)
8	DEPARTMENTS	TABLE	(null)
9	CUSTOMERS	TABLE	(null)
10	COURSES	TABLE	(null)

Another method is to use the TAB pseudo table

This table exists for all users and will allow you to view the TABLES, and another object called a VIEW

In addition we will see it will also show tables we drop from our schema, this is coming later

Viewing the Table Structure: DESCRIBE

To determine whether the table structure was created correctly you can use the SQL*Plus command DESCRIBE tablename to display the structure of the table

Since this is an *SQL*Plus command* it can be abbreviated to DESC

With this command, no ; is necessary…

Viewing the Table Structure: DESCRIBE

Worksheet | Query Builder

```
1  DESC employees1;
2
3  DESCRIBE employees1;
```

Script Output

Task completed in 1.109 seconds

```
DESC employees1
Name   Null Type
------ ---- -----------
EMPNO       NUMBER(4)
ENAME       VARCHAR2(8)
INIT        VARCHAR2(5)
JOB         VARCHAR2(8)
MGR         NUMBER(4)
BDATE       DATE
MSAL        NUMBER(6,2)
COMM        NUMBER(6,2)
DEPTNO      NUMBER(2)

DESCRIBE employees1
Name   Null Type
------ ---- -----------
EMPNO       NUMBER(4)
ENAME       VARCHAR2(8)
INIT        VARCHAR2(5)
JOB         VARCHAR2(8)
MGR         NUMBER(4)
BDATE       DATE
MSAL        NUMBER(6,2)
COMM        NUMBER(6,2)
DEPTNO      NUMBER(2)
```

- The 2 different variations of the DESCRIBE command are shown
- DESC and DESCRIBE
- The semi-colon was needed here since I used the execute script button to execute both commands in the same window

Table Creation Through Subqueries

Previously, we created a table “from scratch”

It is also possible to create a table based on data contained in existing tables

To create a table that will contain data from existing tables you can use the CREATE TABLE command with an AS clause that contains a subquery

The CREATE TABLE … AS is sometimes referred to as CTAS

The CREATE TABLE command tells Oracle to create a table

The AS keyword tells Oracle to retrieve the columns from the specified query

You have the option of providing new column names or using the columns that are retrieved from the subquery

Table Creation Through Subqueries

```
CREATE TABLE tablename [(columnname, …)]
AS (subquery);
```

The commands above will create a new table based on data from another table

The columns returned by the subquery will provide the structure of the new table, this includes column names and datatypes

You have the option to provide your own column names and override the columns provided through the subquery

Table Creation Through Subqueries

Worksheet | Query Builder

```
1  CREATE TABLE employees_job_sal
2  AS (SELECT empno, ename, job, msal
3      FROM employees);
```

Script Output

Task completed in 0.203 seconds

```
table EMPLOYEES_JOB_SAL created.
```

The new table has been created based on the returned columns from the customers table

The column names are retrieved from the original table, compare the two DESC commands on the two tables

Table Creation Through Subqueries

Worksheet | Query Builder

```
1  DESC employees;
2
3  DESC employees_job_sal;
```

Query Result | Script Output

Task completed in 0.094 seconds

```
DESC employees
Name   Null     Type
------ -------- -----------
EMPNO  NOT NULL NUMBER(4)
ENAME  NOT NULL VARCHAR2(8)
INIT   NOT NULL VARCHAR2(5)
JOB             VARCHAR2(8)
MGR             NUMBER(4)
BDATE  NOT NULL DATE
MSAL   NOT NULL NUMBER(6,2)
COMM            NUMBER(6,2)
DEPTNO          NUMBER(2)

DESC employees_job_sal
Name  Null     Type
----- -------- -----------
EMPNO          NUMBER(4)
ENAME NOT NULL VARCHAR2(8)
JOB            VARCHAR2(8)
MSAL  NOT NULL NUMBER(6,2)
```

You will notice that the column names of the new table are the same names as the ones in the original CUSTOMERS table

The datatypes identified are the same for both tables since the original table provides these to the newly defined column names

It is possible to provide a column list to define new names for the columns of the new table

Table Creation Through Subqueries

Worksheet | Query Builder

```
1  SELECT *
2  FROM employees_job_sal;
```

Script Output | Query Result

SQL | All Rows Fetched: 14 in 0.062 seconds

	EMPNO	ENAME	JOB	MSAL
1	7369	SMITH	TRAINER	800
2	7499	ALLEN	SALESREP	1600
3	7521	WARD	SALESREP	1250
4	7566	JONES	MANAGER	2975
5	7654	MARTIN	SALESREP	1250
6	7698	BLAKE	MANAGER	2850
7	7782	CLARK	MANAGER	2450
8	7788	SCOTT	TRAINER	3000
9	7839	KING	DIRECTOR	5000
10	7844	TURNER	SALESREP	1500
11	7876	ADAMS	TRAINER	1100
12	7900	JONES	ADMIN	800
13	7902	FORD	TRAINER	3000
14	7934	MILLER	ADMIN	1300

As you can see the data that was stored in the EMPLOYEES table is also placed into the new table

It is only the columns we requested that show for the new table

Modifying Existing Tables

There are times when you need to make structural changes to a table

You may need to add, delete, or resize a column

All of these changes are accomplished through the ALTER TABLE command

A table can be modified without having to shutdown the database

Even if a user is accessing a table, it can still be modified with no disruption of service

Modifying Existing Tables

```
ALTER TABLE tablename
ADD|MODIFY|DROP COLUMN| columnname [definition];
```

The ADD, MODIFY or DROP COLUMN clause you use depends on the type of change being made

ALTER TABLE … ADD Command

Using an ADD clause with the ALTER TABLE command adds a new column to a table

The same rules that apply to defining a column during table creation apply to creating a new column:

- The new column must be defined by a column name and a datatype with width if applicable
- A default value can also be assigned

The new column is always added at the end of the existing table, so it will be the last column

ALTER TABLE … ADD Command

```
ALTER TABLE tablename
ADD (columnname datatype, [DEFAULT] …);
```

As indicated in the syntax, more than one column can be added to the table with a single statement

Add the next column to the column list by separating it from the previous one with a comma (same format as the CREATE TABLE command)

ALTER TABLE … ADD Command

Worksheet | Query Builder

```
1  ALTER TABLE employees1
2  ADD (gender CHAR);
3
4  DESC employees1;
```

Script Output | Query Result

Task completed in 0.109 seconds

```
table EMPLOYEES1 altered.
DESC employees1
Name    Null Type
------ ---- -----------
EMPNO        NUMBER(4)
ENAME        VARCHAR2(8)
INIT         VARCHAR2(5)
JOB          VARCHAR2(8)
MGR          NUMBER(4)
BDATE        DATE
MSAL         NUMBER(6,2)
COMM         NUMBER(6,2)
DEPTNO       NUMBER(2)
GENDER       CHAR(1)
```

A new column called GENDER is to be added to the EMPLOYEES1 table of our schema

It is to have a CHAR datatype with a defualt length of 1, so a size is not required it will default to 1

New column added to the EMPLOYEES1 table, this is verified after with DESCRIBE

Notice it is added as the last column in the table

ALTER TABLE … MODIFY Command

A MODIFY clause can be used with the ALTER TABLE command to change the definition of an existing column

The changes that can be made to a column include:

- Changing the size of a column (increase or decrease)
- Changing the datatype (VARCHAR2 to CHAR)
- Changing or adding a default value to a column

ALTER TABLE … MODIFY Command

```
ALTER TABLE tablename
MODIFY (columnname datatype [DEFAULT],…);
```

ALTER TABLE … MODIFY Command

There are three rules that you need to be aware of when modifying existing columns:

1. A column must be as wide as the data values it already contains
2. If a NUMBER column already contains data, you cannot decrease the precision or scale of the column
3. Changing the default value of a column does not change the values of data already in the table

ALTER TABLE … MODIFY Command

Rule # 1 applies when you want to decrease the size of a column that already contains data

You can only decrease the size of a column to a size that is not less than the largest width of existing data

For example if a column had been declared as a VARCHAR2(15) and the longest value was a width of 12 characters, you would not be able to decrease the width to less than 12 characters

If you attempt it, Oracle will return an error

ALTER TABLE … MODIFY Command

Worksheet | Query Builder

```
1  ALTER TABLE employees1
2  MODIFY ename VARCHAR2(5);
```

Script Output | Query Result

Task completed in 0.156 seconds

```
Error starting at line : 1 in command -
ALTER TABLE employees1
MODIFY ename VARCHAR2(5)
Error report -
SQL Error: ORA-01441: cannot decrease column length because some value is too big
01441. 00000 -  "cannot decrease column length because some value is too big"
*Cause:
*Action:
```

Error generated when attempting to decrease the width of a column to a size smaller than the length of the current data

ALTER TABLE … MODIFY Command

Rule # 2 specifies that Oracle will not allow you to decrease the precision or scale of a NUMBER column if the column contains data

This is regardless of whether the current values stored in the NUMBER column will be affected

Oracle will return an error message and the statement will fail unless the column is empty

ALTER TABLE … MODIFY Command

Worksheet | Query Builder

```
1  ALTER TABLE employees1
2  MODIFY msal NUMBER(5,2);
```

Script Output | Query Result

Task completed in 0.124 seconds

```
Error starting at line : 1 in command -
ALTER TABLE employees1
MODIFY msal NUMBER(5,2)
Error report -
SQL Error: ORA-01440: column to be modified must be empty to decrease precision or scale
01440. 00000 -  "column to be modified must be empty to decrease precision or scale"
*Cause:
*Action:
```

Error generated when attempting to resize a NUMBER column that already contains data

ALTER TABLE … MODIFY Command

Rule # 3 applies when you modify existing columns and change the default value assigned to a column

When a default value of a column is changed, it will only change the default value assigned to future rows inserted into the table, the default value assigned to existing rows remains the same

Any changes to any default values previously inserted must be done manually within the table

ALTER TABLE … MODIFY Command

Worksheet | Query Builder

```
ALTER TABLE employees1
MODIFY (gender DEFAULT 'F');

SELECT *
FROM employees1;
```

Script Output | Query Result

SQL | All Rows Fetched: 14 in 0.062 seconds

	EMPNO	ENAME	INIT	JOB	MGR	BDATE	MSAL	COMM	DEPTNO	GENDER
1	7369	SMITH	N	TRAINER	7902	17-DEC-65	800	(null)	20	(null)
2	7499	ALLEN	JAM	SALESREP	7698	20-FEB-61	1600	300	30	(null)
3	7521	WARD	TF	SALESREP	7698	22-FEB-62	1250	500	30	(null)
4	7566	JONES	JM	MANAGER	7839	02-APR-67	2975	(null)	20	(null)
5	7654	MARTIN	P	SALESREP	7698	28-SEP-56	1250	1400	30	(null)
6	7698	BLAKE	R	MANAGER	7839	01-NOV-63	2850	(null)	30	(null)
7	7782	CLARK	AB	MANAGER	7839	09-JUN-65	2450	(null)	10	(null)
8	7788	SCOTT	SCJ	TRAINER	7566	26-NOV-59	3000	(null)	20	(null)
9	7839	KING	CC	DIRECTOR	(null)	17-NOV-52	5000	(null)	10	(null)
10	7844	TURNER	JJ	SALESREP	7698	28-SEP-68	1500	0	30	(null)
11	7876	ADAMS	AA	TRAINER	7788	30-DEC-66	1100	(null)	20	(null)
12	7900	JONES	R	ADMIN	7698	03-DEC-69	800	(null)	30	(null)
13	7902	FORD	MG	TRAINER	7566	13-FEB-59	3000	(null)	20	(null)
14	7934	MILLER	TJA	ADMIN	7782	23-JAN-62	1300	(null)	10	(null)

Our new GENDER column that was added to the EMPLOYEES1 table has had a default value of 'F' given to it, The SELECT shows the column with no data in the GENDER column

ALTER TABLE … MODIFY Command

Worksheet | Query Builder

```
1  ALTER TABLE employees1
2  MODIFY (ename VARCHAR2(25));
```

Script Output | Query Result

Task completed in 0.062 seconds

```
table EMPLOYEES1 altered.
```

Alter table EMPLOYEES1 to lengthen the ENAME column from VARCHAR2(8) to VARCHAR2(25)

ALTER TABLE … DROP COLUMN Command

The DROP COLUMN command can be used with the ALTER TABLE command to delete an existing column from a table

The command will delete both the column and is its contents

ALTER TABLE … DROP COLUMN Command

Cautions when using the DROP COLUMN clause:

- Unlike ALTER TABLE with the ADD or MODIFY, a DROP COLUMN clause can reference only one column
- If you drop a column from a table, the deletion is permanent. You may not "undo" the damage if you accidentally delete the wrong column from a table. The only option will be to add the column back to the table and then manually re-enter all deleted data
- You can't delete a column if there is only one column left in the table

ALTER TABLE … DROP COLUMN Command

Worksheet | Query Builder

```
1  ALTER TABLE employees1
2  DROP COLUMN gender;
```

Script Output | Query Result

Task completed in 0.078 seconds

```
table EMPLOYEES1 altered.
```

The ALTER TABLE command with the DROP COLUMN command is used to drop the GENDER column

ALTER TABLE … SET UNUSED/ DROP UNUSED COLUMN Command

While the Oracle server drops a column from a very large table, it can slow down the processing of queries or other SQL commands from users

To avoid this problem, a SET UNUSED clause can be included in the ALTER TABLE command to *mark the column for deletion at a later time*

If a column is marked for deletion, *it is unavailable and will not be displayed in the table structure or in the results of any queries*

Nor can any other operation except ALTER TABLE … DROP UNUSED be performed on the column

ALTER TABLE … SET UNUSED/ DROP UNUSED COLUMN Command

In other words, once a column is set as "unused" the column and all its *contents are no longer available and cannot be recovered at a later time*

It postpones the physical erasing of the data from the storage device until a later time, usually after business hours

A DROP UNUSED clause is used with the ALTER TABLE command to complete the deletion process for any column that has been marked as unused

ALTER TABLE … SET UNUSED/DROP UNUSED COLUMN Command

```
ALTER TABLE tablename
SET UNUSED (columnname);
     OR
ALTER TABLE tablename
SET UNUSED COLUMN columnname;
```

There are two options for the SET UNUSED option. Regardless of the syntax used, only one column can be marked for deletion

ALTER TABLE … SET UNUSED/DROP UNUSED COLUMN Command

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

This syntax is used to drop a column previously identified as “unused”. When it is used, any column previously set as “unused” is deleted and storage previously occupied by the data contained in the column becomes available

ALTER TABLE … SET UNUSED/DROP UNUSED COLUMN Command

Worksheet | Query Builder

```
1  ALTER TABLE employees1
2  SET UNUSED (init);
```

Script Output | Query Result

Task completed in 0.234 seconds

```
table EMPLOYEES1 altered.
```

INIT column set to "unused", can use DESC to show it is gone, then column can be actually removed with the DROP UNUSED

ALTER TABLE … SET UNUSED/DROP UNUSED COLUMN Command

Worksheet | Query Builder

```
1  DESC employees1;
```

Script Output | Query Result

Task completed in 0.063 seconds

```
DESC employees1
Name    Null Type
------ ---- ------------
EMPNO        NUMBER(4)
ENAME        VARCHAR2(25)
JOB          VARCHAR2(8)
MGR          NUMBER(4)
BDATE        DATE
MSAL         NUMBER(6,2)
COMM         NUMBER(6,2)
DEPTNO       NUMBER(2)
```

As you can see the column is n o longer visible in the table description

You can no longer use the column with the table

ALTER TABLE … SET UNUSED/DROP UNUSED COLUMN Command

Worksheet | Query Builder

```
1  ALTER TABLE employees1
2  DROP UNUSED COLUMNS;
```

Script Output | Query Result

Task completed in 0.312 seconds

```
table EMPLOYEES1 altered.
```

The DROP UNUSED COLUMNS option now removes the unused columns from the table

Renaming a Table

Oracle will allow you to change the name of any table you own using the RENAME … TO command

```
RENAME oldtablename TO newtablename;
```

Renaming a Table

Worksheet | Query Builder

```
1  RENAME employees_job_sal TO employees_sal_job;
```

Script Output | Query Result

Task completed in 0.078 seconds

```
employees_job_sal TO succeeded.
```

Table is renamed, try to access the old table and an error occurs, you can now access the new tablename

Renaming a Table

Worksheet | Query Builder

```
1  SELECT *
2  FROM employees_job_sal;
```

Script Output | Query Result

SQL | Executing:SELECT *FROM employees_job_sal in 0 seconds

```
ORA-00942: table or view does not exist
00942. 00000 - "table or view does not exist"
*Cause:
*Action:
Error at Line: 2 Column: 6
```

If you try to access the old table name an error occurs, you can now only access the new table name you changed it to

Rename a Table

The renamed table shows the contents of the table

Worksheet | Query Builder

```
1 SELECT *
2 FROM employees_sal_job;
```

Script Output | Query Result

SQL | All Rows Fetched: 14 in 0.046 seconds

	EMPNO	ENAME	JOB	MSAL
1	7369	SMITH	TRAINER	800
2	7499	ALLEN	SALESREP	1600
3	7521	WARD	SALESREP	1250
4	7566	JONES	MANAGER	2975
5	7654	MARTIN	SALESREP	1250
6	7698	BLAKE	MANAGER	2850
7	7782	CLARK	MANAGER	2450
8	7788	SCOTT	TRAINER	3000
9	7839	KING	DIRECTOR	5000
10	7844	TURNER	SALESREP	1500
11	7876	ADAMS	TRAINER	1100
12	7900	JONES	ADMIN	800
13	7902	FORD	TRAINER	3000
14	7934	MILLER	ADMIN	1300

Truncating a Table

To delete all the rows stored in a table **and** free up the storage space that was occupied by those rows, use the TRUNCATE TABLE command

When a table is truncated all *the rows in the table are removed but the table itself remains*

The columns still exist even though no values are stored in them

It is basically the same as deleting all the rows in a table, however, if you *delete all the rows in a table the storage space occupied by the rows will still be allocated to the table*

Truncating a Table

Worksheet | Query Builder

```
1  TRUNCATE TABLE employees_sal_job;
```

Script Output | Query Result

Task completed in 0.093 seconds

```
table EMPLOYEES_SAL_JOB truncated.
```

Truncating a Table

Worksheet | Query Builder

```
1  SELECT *
2  FROM employees_sal_job;
```

Script Output | Query Result

SQL | All Rows Fetched: 0 in 0.047 seconds

EMPNO	ENAME	JOB	MSAL

- The table still exists but the data in the table has been removed

Deleting a Table

A table can be removed from an Oracle database by issuing a DROP TABLE command

Deleting a Table

```
DROP TABLE tablename [PURGE];
```

Always exercise caution when deleting especially when it is a table

Once a table is deleted the table and all its data are gone (can be recovered though as we shall see)

In addition, any index that has been created based on the table is also dropped (we'll discuss indexes later)

Dropping a Table

Worksheet | Query Builder

```
1  DROP TABLE employees_sal_job;
2
3  DESC employees_job_sal;
```

Script Output | Query Result

Task completed in 0.062 seconds

```
table EMPLOYEES_SAL_JOB dropped.
DESC employees_job_sal
ERROR:
------------------------------------------------
ERROR: object EMPLOYEES_JOB_SAL does not exist
```

Table is dropped, so the table name is no longer valid and the error message is displayed

Introduction to Purge

Since the DROP TABLE command is a DDL command, any dropped table was permanently removed, and could only be recovered from backup

Effective with the 10g database release, Oracle has included a recycle bin for holding dropped tables

This is similar to the recycle bin used by Microsoft Windows Explorer when you delete a file

The Recyclebin allows you to recovered deleted files, or, in this case, tables

RECYCLEBIN

Several new commands have been added to work with RECYCLEBIN

- FLASHBACK
- PURGE
- DROP TABLE … PURGE

You may have already noticed that when you drop a table, an additional entry is added into the tables you own

List the Tables You Own

Worksheet | Query Builder

```
1  SELECT *
2  FROM tab;
3
```

Script Output | Query Result

SQL | All Rows Fetched: 11 in 0.234 seconds

	TNAME	TABTYPE	CLUSTERID
1	SALGRADES	TABLE	(null)
2	SALES	TABLE	(null)
3	REGISTRATIONS	TABLE	(null)
4	OFFERINGS	TABLE	(null)
5	HISTORY	TABLE	(null)
6	EMPLOYEES_SAL_JOB	TABLE	(null)
7	EMPLOYEES1	TABLE	(null)
8	EMPLOYEES	TABLE	(null)
9	DEPARTMENTS	TABLE	(null)
10	CUSTOMERS	TABLE	(null)
11	COURSES	TABLE	(null)

- This is the normal appearance of the tables you own after issuing the command SELECT * FROM tab;
- Please note I replaced the EMPLOYEES_SAL_JOB table that was dropped previously

Drop A Table

Worksheet | Query Builder

```
1  DROP TABLE employees_sal_job;
```

Script Output | Query Result

Task completed in 0.063 seconds

```
table EMPLOYEES_SAL_JOB dropped.
```

The EMPLOYEES_SAL_JOB table is dropped.

List Tables Again

Worksheet | Query Builder

```
1 SELECT *
2 FROM tab;
```

Script Output | Query Result

SQL | All Rows Fetched: 11 in 0.047 seconds

	TNAME	TABTYPE	CLUSTERID
1	SALGRADES	TABLE	(null)
2	SALES	TABLE	(null)
3	REGISTRATIONS	TABLE	(null)
4	OFFERINGS	TABLE	(null)
5	HISTORY	TABLE	(null)
6	EMPLOYEES1	TABLE	(null)
7	EMPLOYEES	TABLE	(null)
8	DEPARTMENTS	TABLE	(null)
9	CUSTOMERS	TABLE	(null)
10	COURSES	TABLE	(null)
11	BIN$DknU6Q21RHfgVQAAAAAAAQ==$0	TABLE	(null)

- You will now notice a new table starting with BIN$ has been added to the list of tables that you own
- The table that was dropped is now missing, so EMPLOYEES_SAL_JOB is not listed now

RECYCLEBIN

The RECYCLEBIN is a pseudo table

You can discard any entries in your recycle bin

You can list the entries or describe the recycle bin the same way as any other table you own

RECYCLEBIN

Worksheet | Query Builder

```
1  DESC recyclebin;
```

Script Output | Query Result

Task completed in 0.561 seconds

```
DESC recyclebin
Name            Null     Type
--------------- -------- -------------
OBJECT_NAME     NOT NULL VARCHAR2(128)
ORIGINAL_NAME            VARCHAR2(128)
OPERATION                VARCHAR2(9)
TYPE                     VARCHAR2(25)
TS_NAME                  VARCHAR2(30)
CREATETIME               VARCHAR2(19)
DROPTIME                 VARCHAR2(19)
DROPSCN                  NUMBER
PARTITION_NAME           VARCHAR2(128)
CAN_UNDROP               VARCHAR2(3)
CAN_PURGE                VARCHAR2(3)
RELATED         NOT NULL NUMBER
BASE_OBJECT     NOT NULL NUMBER
PURGE_OBJECT    NOT NULL NUMBER
SPACE                    NUMBER
```

- This is a description of your recycle bin
- You will notice there is a column called original_name, this will retain the original name of your table

FLASHBACK Command

The table that was dropped a few slides back can be recovered

To do this, a new command has been added to allow the user to recover the table

This new command is called FLASHBACK

The following slides will detail the recovery process for a file

FLASHBACK Command

Worksheet | Query Builder

```
1  SELECT object_name, original_name, droptime, dropSCN
2  FROM recyclebin;
```

Script Output | Query Result

SQL | All Rows Fetched: 1 in 0.234 seconds

	OBJECT_NAME	ORIGINAL_NAME	DROPTIME	DROPSCN
1	BIN$DknU6Q21RHfgVQAAAAAAAQ==$0	EMPLOYEES_SAL_JOB	2015-02-04:15:19:50	21995188

- The SELECT shows the newly created entry starting with BIN$
- Also shows that the original table name has been retained
- The DROPTIME is also captured
- The DROPSCN is a system number for the action that took place

FLASHBACK Command

Worksheet | Query Builder

```
1  FLASHBACK TABLE employees_sal_job
2  TO BEFORE DROP;
```

Script Output | Query Result

Task completed in 0.078 seconds

```
table EMPLOYEES_SAL_JOB succeeded.
```

- The FLASHBACK TABLE command with the table name is issued with the TO BEFORE DROP option

FLASHBACK Command

Worksheet | Query Builder

```
1  SELECT *
2  FROM tab;
```

Script Output | Query Result

SQL | All Rows Fetched: 11 in 0.047 seconds

	TNAME	TABTYPE	CLUSTERID
1	SALGRADES	TABLE	(null)
2	SALES	TABLE	(null)
3	REGISTRATIONS	TABLE	(null)
4	OFFERINGS	TABLE	(null)
5	HISTORY	TABLE	(null)
6	EMPLOYEES_SAL_JOB	TABLE	(null)
7	EMPLOYEES1	TABLE	(null)
8	EMPLOYEES	TABLE	(null)
9	DEPARTMENTS	TABLE	(null)
10	CUSTOMERS	TABLE	(null)
11	COURSES	TABLE	(null)

- The TAB table is queried to see that the original table is displayed again. The BIN$ entry is now removed, and the original table is restored
- This will recover all the data that was in the table as well

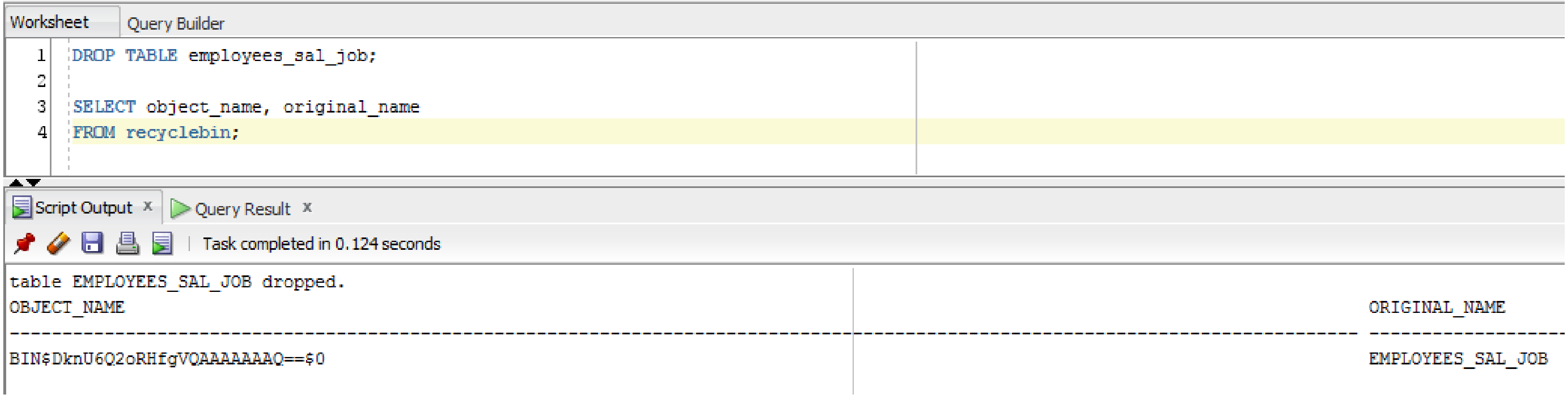
PURGE RECYCLEBIN

The PURGE command is similar to emptying the recyclebin in Windows Explorer

You may only PURGE tables you have dropped

You can no longer recover tables once you have purged them

PURGE Tables From the RECYCLEBIN



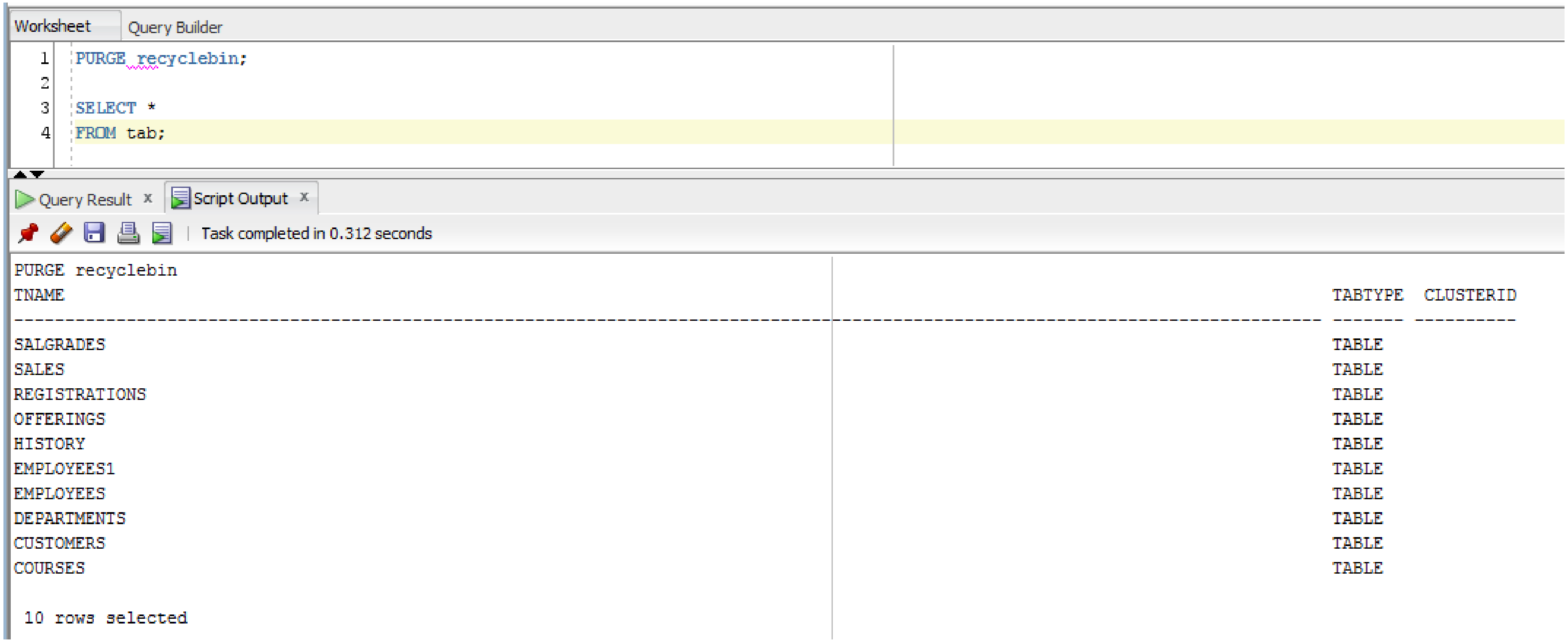
```
DROP TABLE employees_sal_job;

SELECT object_name, original_name
FROM recyclebin;
```

Task completed in 0.124 seconds

```
table EMPLOYEES_SAL_JOB dropped.
OBJECT_NAME                                                                                                                                  ORIGINAL_NAME
---------------------------------------------------------------------------------------------------------------------------------- --------------------
BIN$DknU6Q2oRHfgVQAAAAAAAQ==$0                                                                                                               EMPLOYEES_SAL_JOB
```

- The table was dropped again, the recycle bin was queried to show that the table entry is now in the recyclebin
- It shows the BIN$ name that has been assigned to it, as well as its original name

PURGE Tables From the RECYCLEBIN



```sql
PURGE recyclebin;

SELECT *
FROM tab;
```

```
PURGE recyclebin
TNAME                                                                                                                              TABTYPE  CLUSTERID
------------------------------------------------------------------------------------------------------------------------------- ------- ----------
SALGRADES                                                                                                                          TABLE
SALES                                                                                                                              TABLE
REGISTRATIONS                                                                                                                      TABLE
OFFERINGS                                                                                                                          TABLE
HISTORY                                                                                                                            TABLE
EMPLOYEES1                                                                                                                         TABLE
EMPLOYEES                                                                                                                          TABLE
DEPARTMENTS                                                                                                                        TABLE
CUSTOMERS                                                                                                                          TABLE
COURSES                                                                                                                            TABLE

 10 rows selected
```

- You now see both the original table name is no longer visible
- Also you can see there are no tables starting with BIN$

DROP TABLE with the PURGE Option

If you know you want to drop the table and do not need the requirement to recover it, the PURGE option can be added to the DROP TABLE command

With this option, no entry is added to the RECYCLEBIN and the table cannot be recovered

DROP TABLE with the PURGE Option

Worksheet | Query Builder

```
1  DROP TABLE employees1 PURGE;
```

Query Result x | Script Output x

Task completed in 0.078 seconds

```
table EMPLOYEES1 dropped.
```

The PURGE option is used with the DROP TABLE

DROP TABLE with the PURGE Option

Worksheet | Query Builder

```
1 SELECT *
2 FROM tab;
```

Script Output | Query Result

SQL | All Rows Fetched: 9 in 0.047 seconds

	TNAME	TABTYPE	CLUSTERID
1	SALGRADES	TABLE	(null)
2	SALES	TABLE	(null)
3	REGISTRATIONS	TABLE	(null)
4	OFFERINGS	TABLE	(null)
5	HISTORY	TABLE	(null)
6	EMPLOYEES	TABLE	(null)
7	DEPARTMENTS	TABLE	(null)
8	CUSTOMERS	TABLE	(null)
9	COURSES	TABLE	(null)

- You will notice the table that was dropped does not appear
- There is also no BIN$ entry that was created
- The table is dropped without creating an entry on the recyclebin