

The logo for Oracle Academy is centered on a light gray background. It features the word "ORACLE" in a bold, orange, sans-serif font. Below it, the word "Academy" is written in a smaller, dark gray, sans-serif font. The entire logo is framed by two horizontal dark gray bars, one at the top and one at the bottom.

ORACLE

Academy

Database Programming with SQL

12-1

INSERT Statements

ORACLE
Academy



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Objectives

- This lesson covers the following objectives:
 - Explain the importance of being able to alter the data in a database
 - Construct and execute INSERT statements which insert a single row using a VALUES clause
 - Construct and execute INSERT statements that use special values, null values, and date values
 - Construct and execute INSERT statements that copy rows from one table to another using a subquery

Purpose

- Up to now, you have been learning how to access data in a database
- It's time to learn how to make changes to the data in the database
- In business, databases are dynamic
- They are constantly in the process of having data inserted, updated, and deleted

Purpose

- Think how many times the school's student database changes from day to day and year to year
- Unless changes are made, the database would quickly lose its usefulness
- In this lesson, you will begin to use data manipulation language (DML) statements to make changes to a database

Copy Tables Before Inserting

- You will be responsible for altering tables in your schema
- You will also be responsible for restoring them just as a real Database Administrator would assume that responsibility
- To keep your schema tables in their original state, you will make a copy of each table before completing the practice activities in this and later lessons
- In each practice activity, you will use the copy of the table that you create, not the original
- If you inadvertently alter a table copy, you can use the original table to restore the copy

Copy Tables Before Inserting

- You should name each copied table: `copy_tablename`
- The table copies will not inherit the associated primary-to-foreign-key integrity rules (relationship constraints) of the original tables
- The column data types, however, are inherited in the copied tables

Syntax to Create a Copy of a Table

- Create table syntax:

```
CREATE TABLE copy_tablename  
AS (SELECT * FROM tablename);
```

- For example:

```
CREATE TABLE copy_employees  
AS (SELECT * FROM employees);
```

```
CREATE TABLE copy_departments  
AS (SELECT * FROM departments);
```


Syntax to Create a Copy of a Table

- To verify and view the copy of the table, use the following DESCRIBE and SELECT statements:

```
DESCRIBE copy_employees;
```

```
SELECT * FROM copy_employees;
```

```
DESCRIBE copy_departments;
```

```
SELECT * FROM copy_departments;
```

The structure and content of the table can also be viewed Using the Object Browser in APEX.

INSERT

- The INSERT statement is used to add a new row to a table. The statement requires three values:
 - the name of the table
 - the names of the columns in the table to populate
 - corresponding values for each column
- How can we INSERT the data below to create a new department in the copy_departments table?

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
200	Human Resources	205	1500

INSERT

- The syntax below uses INSERT to add a new department to the copy_departments table
- This statement explicitly lists each column as it appears in the table
- The values for each column are listed in the same order
 - Note that number values are not enclosed in single quotation marks

```
INSERT INTO copy_departments
  (department_id, department_name, manager_id, location_id)
VALUES (200, 'Human Resources', 205, 1500);
```

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
200	Human Resources	205	1500

The column names can be listed in any order (although there is no benefit in doing this) as long as the column values are listed in the same order as the column names.

INSERT

- Another way to insert values in a table is to implicitly add them by omitting the column names
- One precaution: the values for each column must match exactly the default order in which they appear in the table (as shown in a DESCRIBE statement), and a value must be provided for each column



Note: At the time of writing, the DESCRIBE feature is not functioning as expected in APEX. Whilst this remains the case, use the Object Browser to see detailed table information.

INSERT

- The INSERT statement in this example was written without explicitly naming the columns
- For clarity, however, it is best to use the column names in an INSERT clause

```
INSERT INTO copy_departments  
VALUES  
(210, 'Estate Management', 102, 1700);
```

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
210	Estate Management	102	1700

Check The Table First

- Before inserting data into a table, you must check several table details
- The DESCRIBE tablename statement will return a description of the table structure in the table summary chart
- COPY_DEPARTMENTS TABLE SUMMARY:

Column	Data Type	Length	Precision	Scale	Primary Key	Nullable
DEPARTMENT_ID	NUMBER	-	4	0	-	✓
DEPARTMENT_NAME	VARCHAR2	30	-	-	-	-
MANAGER_ID	NUMBER	-	6	0	-	✓
LOCATION_ID	NUMBER	-	4	0	-	✓

Note: as this table was created as a copy of the departments table, the primary key constraint has not been copied from the original.

Table Summary

- As shown in the example, the table summary provides information about each column in the table, such as:
 - the allowance of duplicate values
 - the type of data allowed
 - the amount of data allowed
 - the allowance of NULL values

Column	Data Type	Length	Precision	Scale	Primary Key	Nullable
EMPLOYEE_ID	NUMBER	-	6	0	1	-
FIRST_NAME	VARCHAR2	20	-	-	-	✓
LAST_NAME	VARCHAR2	25	-	-	-	-
EMAIL	VARCHAR2	25	-	-	-	-
PHONE_NUMBER	VARCHAR2	20	-	-	-	✓
HIRE_DATE	DATE	7	-	-	-	-
JOB_ID	VARCHAR2	10	-	-	-	-
SALARY	NUMBER	-	8	2	-	✓
COMMISSION_PCT	NUMBER	-	2	2	-	✓
MANAGER_ID	NUMBER	-	6	0	-	✓
DEPARTMENT_ID	NUMBER	-	4	0	-	✓
BONUS	VARCHAR2	5	-	-	-	✓

Table Summary

- Notice where the “Data Type” column is a character data type the “Length” column specifies the maximum number of characters permitted

Column	Data Type	Length	Precision	Scale	Primary Key	Nullable
EMPLOYEE_ID	NUMBER	-	6	0	1	-
FIRST_NAME	VARCHAR2	20	-	-	-	✓
LAST_NAME	VARCHAR2	25	-	-	-	-
EMAIL	VARCHAR2	25	-	-	-	-
PHONE_NUMBER	VARCHAR2	20	-	-	-	✓
HIRE_DATE	DATE	7	-	-	-	-
JOB_ID	VARCHAR2	10	-	-	-	-
SALARY	NUMBER	-	8	2	-	✓
COMMISSION_PCT	NUMBER	-	2	2	-	✓
MANAGER_ID	NUMBER	-	6	0	-	✓
DEPARTMENT_ID	NUMBER	-	4	0	-	✓
BONUS	VARCHAR2	5	-	-	-	✓

Table Summary

- First_name has data type VARCHAR2, and Length 20, this means that up to 20 characters can be entered for this column

Column	Data Type	Length	Precision	Scale	Primary Key	Nullable
EMPLOYEE_ID	NUMBER	-	6	0	1	-
FIRST_NAME	VARCHAR2	20	-	-	-	✓
LAST_NAME	VARCHAR2	25	-	-	-	-
EMAIL	VARCHAR2	25	-	-	-	-
PHONE_NUMBER	VARCHAR2	20	-	-	-	✓
HIRE_DATE	DATE	7	-	-	-	-
JOB_ID	VARCHAR2	10	-	-	-	-
SALARY	NUMBER	-	8	2	-	✓
COMMISSION_PCT	NUMBER	-	2	2	-	✓
MANAGER_ID	NUMBER	-	6	0	-	✓
DEPARTMENT_ID	NUMBER	-	4	0	-	✓
BONUS	VARCHAR2	5	-	-	-	✓

Table Summary

- For Number data types the brackets specify the Precision and Scale
- Precision is the total number of digits, and Scale is the number of digits to the right of the decimal place

Column	Data Type	Length	Precision	Scale	Primary Key	Nullable
EMPLOYEE_ID	NUMBER	-	6	0	1	-
FIRST_NAME	VARCHAR2	20	-	-	-	✓
LAST_NAME	VARCHAR2	25	-	-	-	-
EMAIL	VARCHAR2	25	-	-	-	-
PHONE_NUMBER	VARCHAR2	20	-	-	-	✓
HIRE_DATE	DATE	7	-	-	-	-
JOB_ID	VARCHAR2	10	-	-	-	-
SALARY	NUMBER	-	8	2	-	✓
COMMISSION_PCT	NUMBER	-	2	2	-	✓
MANAGER_ID	NUMBER	-	6	0	-	✓
DEPARTMENT_ID	NUMBER	-	4	0	-	✓
BONUS	VARCHAR2	5	-	-	-	✓

Data types will be covered in more detail later in the course.

Table Summary

- The SALARY column allows numbers with a Precision of 8 and a Scale of 2
- The maximum value allowed in this column is 999999.99

Column	Data Type	Length	Precision	Scale	Primary Key	Nullable
EMPLOYEE_ID	NUMBER	-	6	0	1	-
FIRST_NAME	VARCHAR2	20	-	-	-	✓
LAST_NAME	VARCHAR2	25	-	-	-	-
EMAIL	VARCHAR2	25	-	-	-	-
PHONE_NUMBER	VARCHAR2	20	-	-	-	✓
HIRE_DATE	DATE	7	-	-	-	-
JOB_ID	VARCHAR2	10	-	-	-	-
SALARY	NUMBER	-	8	2	-	✓
COMMISSION_PCT	NUMBER	-	2	2	-	✓
MANAGER_ID	NUMBER	-	6	0	-	✓
DEPARTMENT_ID	NUMBER	-	4	0	-	✓
BONUS	VARCHAR2	5	-	-	-	✓

Inserting Rows With Null Values

- The INSERT statement need not specify every column—the Nullable columns may be excluded
- If every column that requires a value is assigned a value, the insert works



Inserting Rows With Null Values

- In our example, the EMAIL column is defined as a NOT NULL column
- An implicit attempt to add values to the table as shown would generate an error

```
INSERT INTO copy_employees
  (employee_id, first_name, last_name, phone_number,
  hire_date,
  job_id, salary)
VALUES
  (302, 'Grigorz', 'Polanski', '8586667641', '15-Jun-2017',
  'IT_PROG', 4200);
```

```
ORA-01400: cannot insert NULL into
("US_A009EMEA815_PLSQL_T01"."COPY_EMPLOYEES"."EMAIL")
```

The EMAIL column has been omitted from the insert, so the insert fails as NULL values are not permitted for the EMAIL column.

Inserting Rows With Null Values

- An implicit insert will automatically insert a null value in columns that allow nulls
- To explicitly add a null value to a column that allows nulls, use the NULL keyword in the VALUES list



Inserting Rows With Null Values

- To specify empty strings and/or missing dates, use empty single quotation marks (with no spaces between them like '') for the missing data

```
INSERT INTO copy_employees
(employee_id, first_name, last_name, email, phone_number,
hire_date, job_id, salary)
VALUES
(302, 'Grigorz', 'Polanski', 'gpolanski', '', '15-Jun-2017',
'IT_PROG', 4200);
```

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY
302	Grigorz	Polanski	gpolanski	-	15-Jun-2017	IT_PROG	4200

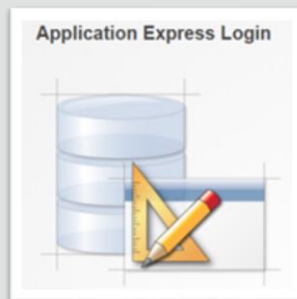
COMM_PCT	MGR_ID	DEPT_ID	BONUS
-	-	-	-

.....

NOTE: the column names in the output have been abbreviated in order to fit on the slide.

Inserting Special Values

- Special values such as SYSDATE and USER can be entered in the VALUES list of an INSERT statement
- SYSDATE will put the current date and time in a column
- USER will insert the current session's username, which is OAE_PUBLIC_USER in Oracle Application Express



Inserting Special Values

- This example adds USER as the last name, and SYSDATE for hire date

```
INSERT INTO copy_employees
  (employee_id, first_name, last_name, email, phone_number,
  hire_date,
  job_id, salary)
VALUES
  (304, 'Test', USER, 't_user', 4159982010, SYSDATE,
  'ST_CLERK', 2500);
```

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY
304	Test	APEX_PUBLIC_USER	t_user	4159982010	15-Jun-2017	ST_CLERK	2500

COMM_PCT	MGR_ID	DEPT_ID	BONUS
-	-	-	-

.....

Some columns have been omitted from the output for display purposes.

Inserting Specific Date Values

- The default format model for date data types is DD-Mon-YYYY
- With this date format, the default time of midnight (00:00:00) is also included
- In an earlier section, we learned how to use the TO_CHAR function to convert a date to a character string when we want to retrieve and display a date value in a non-default format
- Here is a reminder of TO_CHAR:

```
SELECT first_name, TO_CHAR(hire_date, 'Month, fmdd, yyyy')  
FROM employees  
WHERE employee_id = 101;
```

ORACLE
Academy

FIRST_NAME	TO_CHAR(HIRE_DATE, 'MONTH, FMDD, YYYY')
Neena	September, 21, 1989

DP 12-1
INSERT Statements

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

26

Inserting Specific Date Values

- Similarly, if we want to INSERT a row with a non-default format for a date column, we must use the TO_DATE function to convert the date value (a character string) to a date

```
INSERT INTO copy_employees
  (employee_id, first_name, last_name, email, phone_number,
  hire_date, job_id, salary)
VALUES
  (301, 'Katie', 'Hernandez', 'khernandez', '8586667641',
   TO_DATE('July 8, 2017', 'Month fmdd, yyyy'),
  'MK_REP', 4200);
```

Inserting Specific Date Values

- A second example of TO_DATE allows the insertion of a specific time of day, overriding the default time of midnight

```
INSERT INTO copy_employees
(employee_id, first_name, last_name, email, phone_number,
hire_date, job_id, salary)
VALUES
(303, 'Angelina', 'Wright', 'awright', '4159982010',
TO_DATE('July 10, 2017 17:20', 'Month fmdd, yyyy HH24:MI'),
'MK_REP', 3600);
```

```
SELECT first_name, last_name,
TO_CHAR(hire_date, 'dd-Mon-YYYY HH24:MI') As "Date and Time"
FROM copy_employees
WHERE employee_id = 303;
```

FIRST_NAME	LAST_NAME	Date and Time
Angelina	Wright	10-Jul-2017 17:20

Using A Subquery To Copy Rows

- Each INSERT statement we have seen so far adds only one row to the table
- But suppose we want to copy 100 rows from one table to another
- We do not want to have to write and execute 100 separate INSERT statements, one after the other
- That would be very time-consuming
- Fortunately, SQL allows us to use a subquery within an INSERT statement

NOTE: In lesson 3 of this section, we will look at the INSERT ALL Statement. This allows multiple rows to be added in one SQL statement.

Using A Subquery To Copy Rows

- All the results from the subquery are inserted into the table
- So we can copy 100 rows – or 1000 rows – with one multiple-row subquery within the INSERT
- As you would expect, you don't need a VALUES clause when using a subquery to copy rows because the inserted values will be exactly the values returned by the subquery

Using A Subquery To Copy Rows

- In the example shown, a new table called SALES_REPS is being populated with copies of some of the rows and columns from the EMPLOYEES table
- The WHERE clause is selecting those employees that have job IDs like '%REP%'

```
INSERT INTO sales_reps(id, name, salary, commission_pct)
  SELECT employee_id, last_name, salary, commission_pct
 FROM   employees
 WHERE  job_id LIKE '%REP%';
```

NOTE: in order to run the above INSERT statement, the sales_reps table would need to be created first.

Using A Subquery To Copy Rows

- The number of columns and their data types in the column list of the INSERT clause must match the number of columns and their data types in the subquery
- The subquery is not enclosed in parentheses as is done with the subqueries in the WHERE clause of a SELECT statement



Using A Subquery To Copy Rows

- If we want to copy all the data – all rows and all columns – the syntax is even simpler
- To select all rows from the EMPLOYEES table and insert them into the SALES_REPS table, the statement would be written as shown:

```
INSERT INTO sales_reps  
  SELECT *  
  FROM   employees;
```

- Again, this will work only if both tables have the same number of columns with matching data types, and they are in the same order

Terminology

- Key terms used in this lesson included:
 - INSERT INTO
 - USER
 - Transaction
 - Explicit

Summary

- In this lesson, you should have learned how to:
 - Explain the importance of being able to alter the data in a database
 - Construct and execute INSERT statements which insert a single row using a VALUES clause
 - Construct and execute INSERT statements that use special values, null values, and date values
 - Construct and execute INSERT statements that copy rows from one table to another using a subquery

The logo for Oracle Academy is centered on a light gray background. It features the word "ORACLE" in a bold, orange, sans-serif font. Below it, the word "Academy" is written in a smaller, dark gray, sans-serif font. The entire logo is framed by two horizontal dark gray bars, one at the top and one at the bottom.

ORACLE

Academy