Chapter 1 - Introduction
Oracle Database 11g SQL
by Jason Price
Oracle Press © 2008 *Citation*

Previous Next

Adding, Modifying, and Removing Rows

In this section, you'll learn how to add, modify, and remove rows in database tables by using the SQL `INSERT`, `UPDATE`, and `DELETE` statements. You can make your row changes permanent in the database using the `COMMIT` statement, or you can undo them using the `ROLLBACK` statement. This section doesn't exhaustively cover all the details of using these statements; you'll learn more about them in Chapter 8.

Adding a Row to a Table

You use the `INSERT` statement to add new rows to a table. You can specify the following information in an `INSERT` statement:

- The table into which the row is to be inserted
- A list of columns for which you want to specify column values
- A list of values to store in the specified columns

When inserting a row, you need to supply a value for the primary key and all other columns that are defined as `NOT NULL`. You don't have to specify values for the other columns if you don't want to; those columns will be automatically set to null if you omit values for them.

You can tell which columns are defined as `NOT NULL` using the SQL*Plus `DESCRIBE` command. The following example `DESCRIBE`s the `customers` table:

```
SQL> DESCRIBE customers
 Name                                      Null?    Type
 ----------------------------------------- -------- ------------
 CUSTOMER_ID                               NOT NULL NUMBER(38)
 FIRST_NAME                                NOT NULL VARCHAR2(10)
 LAST_NAME                                 NOT NULL VARCHAR2(10)
 DOB                                                DATE
 PHONE                                              VARCHAR2(12)
```

As you can see, the `customer_id`, `first_name`, and `last_name` columns are `NOT NULL`, meaning that you must supply a value for these columns. The `dob` and `phone` columns don't require a value; you could omit the values if you wanted, and they would be automatically set to null.

Go ahead and run the following `INSERT` statement, which adds a row to the `customers` table; notice that the order of values in the `VALUES` list matches the order in which the columns are specified in the column list:

```
SQL> INSERT INTO customers (
  2    customer_id, first_name, last_name, dob, phone
  3  ) VALUES (
  4    6, 'Fred', 'Brown', '01-JAN-1970', '800-555-1215'
  5  );
1 row created.
```

Note SQL*Plus automatically numbers lines after you hit ENTER at the end of each line.

In the previous example, SQL*Plus responds that one row has been created after the INSERT statement is executed. You can verify this by running the following SELECT statement:

```
SELECT *
FROM customers;

CUSTOMER_ID FIRST_NAME LAST_NAME  DOB       PHONE
----------- ---------- ---------- --------- ------------
          1 John       Brown      01-JAN-65 800-555-1211
          2 Cynthia    Green      05-FEB-68 800-555-1212
          3 Steve      White      16-MAR-71 800-555-1213
          4 Gail       Black                800-555-1214
          5 Doreen     Blue       20-MAY-70
          6 Fred       Brown      01-JAN-70 800-555-1215
```

Notice the new row that has been added to the end of the table.

By default, the Oracle database displays dates in the format DD-MON-YY, where DD is the day number, MON is the first three characters of the month (in uppercase), and YY is the last two digits of the year. The database actually stores all four digits for the year, but by default it only displays the last two digits.

When a row is added to the customers table, a unique value for the customer_id column must be given. The Oracle database will prevent you from adding a row with a primary key value that already exists in the table; for example, the following INSERT statement causes an error because a row with a customer_id of 1 already exists:

```
SQL> INSERT INTO customers (
  2    customer_id, first_name, last_name, dob, phone
  3  ) VALUES (
  4    1, 'Lisa', 'Jones', '02-JAN-1971', '800-555-1225'
  5  );

INSERT INTO customers (
*
ERROR at line 1:
ORA-00001: unique constraint (STORE.CUSTOMERS_PK) violated
```

Notice that the name of the constraint is shown in the error (CUSTOMERS_PK). That's why you should always name your primary key constraints; otherwise, the Oracle database assigns an unfriendly system-generated name to a constraint (for example, SYS_C0011277).

Modifying an Existing Row in a Table

You use the UPDATE statement to change rows in a table. Normally, when you use the UPDATE statement, you specify the following information:

- The table containing the rows that are to be changed
- A WHERE clause that specifies the rows that are to be changed
- A list of column names, along with their new values, specified using the SET clause

You can change one or more rows using the same UPDATE statement. If more than one row is specified, the same change will be made for all the rows. The following example updates customer #2's last_name to Orange:

```
UPDATE customers
SET last_name = 'Orange'
WHERE customer_id = 2;

1 row updated.
```

SQL*Plus confirms that one row was updated.

Caution If you forget to add a `WHERE` clause, then all the rows will be updated.

The following query confirms the update worked:

```
SELECT *
FROM customers
WHERE customer_id = 2;

CUSTOMER_ID FIRST_NAME LAST_NAME  DOB       PHONE
----------- ---------- ---------- --------- ------------
          2 Cynthia    Orange     05-FEB-68 800-555-1212
```

Removing a Row from a Table

You use the `DELETE` statement to remove rows from a table. You typically use a `WHERE` clause to limit the rows you wish to delete; if you don't, *all* the rows will be deleted from the table.

The following `DELETE` statement removes customer #2:

```
DELETE FROM customers
WHERE customer_id = 2;

1 row deleted.
```

To undo the changes you've made to the rows, you use `ROLLBACK`:

```
ROLLBACK;

Rollback complete.
```

Go ahead and run the `ROLLBACK` to undo any changes you've made so far. That way, your results will match those shown in subsequent chapters.

Note You can make changes to rows permanent using `COMMIT`. You'll see how to do that in Chapter 8.

Previous Next