

User name:

Book: SQL in a Nutshell, 2nd Edition

---

No part of any chapter or book may be reproduced or transmitted in any form by any means without the prior written permission for reprints and excerpts from the publisher of the book or chapter. Redistribution or other use that violates the fair use privilege under U.S. copyright laws (see 17 USC107) or that otherwise violates these Terms of Service is strictly prohibited. Violators will be prosecuted to the full extent of U.S. Federal and Massachusetts laws.

---

## 5.5. Executing Statements

The primary purpose of programming to a specific database API is to execute SQL statements. The next section covers the steps required to successfully execute a simple SQL statement, such as *INSERT* or *UPDATE*, that does not return results. The section following focuses on the more complicated task of executing statements that do return results.

### 5.5.1. Executing an ADO.NET Statement

The following code fragment provides the syntax for executing a SQL *INSERT* statement using ADO.NET:

```
{Odbc|OleDb|Sql}Command statement = connection.CreateCommand( );
statement.CommandText = "INSERT INTO authors(au_id, au_lname, " +
                        "                        au_fname, contract) " +
                        "VALUES ('xyz', 'Brown', 'Emmit', 1)";
int rowsInserted = statement.ExecuteNonQuery( );
statement.Close( );
```

To execute a statement using ADO.NET, take the following steps:

1. Create an ADO.NET `Command` object. `SqlCommand` is used for access to Microsoft SQL Server, `OdbcCommand` is used for ODBC data sources, and `OleDbCommand` for OLE-DB data sources.

```
{Odbc|OleDb|Sql}Command statement = connection.CreateCommand( );
```

2. After creating the appropriate `Command` object, a SQL statement needs to be associated with the `Command` before it can be executed.

```
statement.CommandText = "INSERT INTO authors(au_id, au_lname, " +
                        "                        au_fname, contract) " +
                        "VALUES ('xyz', 'Brown', 'Emmit', 1)";
```

3. After assignment of the SQL statement, the `Command` object is ready for execution.

```
int rowsInserted = statement.ExecuteNonQuery( );
```

There are three methods that can be used to execute ADO.NET `Command` objects:

- a. `ExecuteReader`: Executes SQL statements that return rows, such as *SELECT* statements. The return value is an *ADO.NET DataReader* object.
- b. `ExecuteNonQuery`: Executes SQL statements that do not typically return result sets, such as *INSERT*, *DELETE*, or *UPDATE* statements. The return value is an integer value equal to the number of rows affected by the statement execution.
- c. `ExecuteScalar`: Executes SQL statements that return a single value, such as statements containing a single aggregate function.
- d. After the `Command` object is executed, it should be explicitly closed to free up resources that

have been allocated to execute statements and to communicate with the database server. To free up these resources, invoke the `Command` object's `Dispose` method:

```
statement.Dispose( );
```



The resources held by ADO.NET `Statement` objects are freed using the `Dispose` method, which differs from the `Close` method used by all other ADO.NET types.

### 5.5.2. Executing a JDBC Statement

The following code fragment provides the syntax for executing a SQL `INSERT` statement using JDBC:

```
java.sql.Statement statement = connection.createStatement( );
int rowsInserted = statement.executeUpdate(
    "INSERT INTO authors(au_id, au_lname, au_fname, contract) " +
    " VALUES ('xyz', 'Brown', 'Emmit', 1)" );
statement.close( );
```

1. Executing a SQL statement first requires the creation of a JDBC `Statement` object. A `Statement` object is produced by invoking the `createStatement` method on a valid JDBC `Connection` object:

```
java.sql.Statement statement = connection.createStatement( );
```

2. After a `Statement` object has been created, a SQL statement can be executed by invoking one of the execute methods found on the `Statement` object. For non-query statements, the `executeUpdate` method is the best method to use:

```
statement.executeUpdate( "INSERT INTO authors(au_id, au_lname, " +
    "au_fname, contract) VALUES " +
    "('xyz', 'Brown', 'Emmit', 1)" );
```

The `executeUpdate` method returns an `int` value indicating the number of rows inserted, updated, or deleted during the execution of the SQL statement. If the execution fails, you will get a `SQLException`.

3. After the `Statement` object has been executed, it may be executed again, or freed by invoking the `close` method.

```
statement.close( );
```