CHAPTER 5

ADO Command Strategies

When I get to Chapters 6 and 7, which are all about Recordsets, I discuss lots of different ways to execute queries—many of which don't require use of the Command object to construct. The one important case where the Command object is required is when you have to capture parameters returned from stored procedures. ADO is very smart when it comes to handling stored procedures. If you aren't returning OUTPUT parameters, and if you don't care about the stored procedure return value, you don't have to construct a Command object—so don't. On the other hand, I encourage you to use OUTPUT parameters instead of Recordsets. A resultset with a dozen dozen OUTPUT parameters is still faster to manage than a single-row Recordset. But then, almost any approach is faster than constructing a Recordset.

Don't get me wrong, there are several advantages to using Command objects. In this chapter, you learn that the Command object leverages new SQL Server and MDAC technology to execute all kinds of queries more efficiently—especially with SQL Server 7.0 and later. This means that when you execute rowset-returning or action queries, setting up the Command object can make the process of managing the query and its parameters far more sane. After a Command object is created, you no longer have to worry about putting single quotes around strings or figuring out how to deal with embedded apostrophes. This means you won't have to remove all of the surnames from your database—such as O'Malley or O'Brien—that contain single quotes instead of apostrophes. This should make the Irish folks in your database happier. This chapter also addresses how to determine what your query is doing. I spend a significant amount of time poring over SQL Profiler logs to see exactly what unnatural acts SQL Server is being asked to perform.

When it comes time to execute your SQL query, the best object to use is often the ADO Command. However, as you'll learn, it's not always the best choice. Fortunately, due to ADO's flexibility, you have other alternatives to draw on, as I discuss when I get to the Recordset object in Chapter 6, “Recordset Strategies.” One thing you might not know—it's underdocumented—is that all ADO Command objects appear as methods on their associated Connection objects. This innovative technique (well, it was stolen from RDO) enables you to code the Command by name, followed by its parameters, followed by the Recordset to contain the rowset. Cool. I discuss how to set this up later in this chapter.

For those of you considering the move to ADO.NET (as I discuss beginning with Chapter 13, “Introducing ADO.NET”), you'll discover that the Command
object plays a pivotal role in ADO.NET—albeit in another form. The SqlCommand, OleDbCommand, and OdbcCommand objects all bear a similarity to the ADO Command object. This means that the time you invest today using the Command object to create ADO applications and components can be leveraged tomorrow (or perhaps next year) when you’re ready to migrate your code (or parts of it) to ADO.NET.

**Inner Workings of the Command Object**

The Command object’s biggest benefit is performance. Not only does it make your queries run efficiently, but it also makes you work more efficiently. That is, using the Command object can reduce the length of time it takes to code, debug, test, and deploy complex parameter-based queries—including queries executing stored procedures. For example, when accessing SQL Server 7.0, the ODBC and OLE DB providers have been tuned to access the new sp_executesql system stored procedure. There’s quite a write-up on this in SQL Server Books Online,¹ and I have summarized it here.

---

**TIP** You can easily squander the speed benefits of the Command object if you ask the back end to execute an intellectually-challenged query. In other words, if your SQL is dumb, you negate any performance benefit you would have gained by using the Command.

---

Basically, the Command object is driven from the CommandType argument that instructs ADO how to transmit your query to the data provider. Suppose you wish to execute a parameter-based ad hoc query:²

```
Select author, au_id, year_born from authors where year_born = ?
```

You placed a parameter marker (?) where you want ADO to insert the parameter, so you’re ready to have ADO execute this query. For this query, ADO manufactures an SQL statement that looks like this:

---

¹ Books Online is the copious set of Help topics and examples that ships with SQL Server in lieu of printed documentation. Most (if not all) of it is also available through MSDN via subscription or online.

² An ad hoc query is simply a hard-coded SQL query or action. Using these queries is not a good idea for performance, maintainability, and security reasons. If you can, use a stored procedure instead, but many developers depend on them, at least initially.