CHAPTER 9

Execution Plan Cache Analysis

The performance of any query depends on the effectiveness of the execution plan decided upon by the optimizer, as you learned in previous chapters. Because the overall time required to execute a query is the sum of the time required to generate the execution plan plus the time required to execute the query based on this execution plan, it is important that the cost of generating the execution plan itself is low. The cost incurred when generating the execution plan depends on the process of generating the execution plan, the process of caching the plan, and the reusability of the plan from the plan cache. In this chapter, you will learn how an execution plan is generated and how to analyze the execution plan cache for plan reusability.

In this chapter, I cover the following topics:

- Execution plan generation and caching
- The SQL Server components used to generate an execution plan
- Strategies to optimize the cost of execution plan generation
- Factors affecting parallel plan generation
- How to analyze execution plan caching
- Query plan hash and query hash as mechanisms for identifying queries to tune
- Execution plans gone wrong and parameter sniffing
- Ways to improve the reusability of execution plan caching

Execution Plan Generation

As you know by now, SQL Server uses a cost-based optimization technique to determine the processing strategy of a query. The optimizer considers both the metadata of the database objects and the current distribution statistics of the columns referred to in the query when deciding which index and join strategies should be used.

The cost-based optimization allows a database developer to concentrate on implementing a business rule, rather than on the exact syntax of the query. At the same time, the process of determining the query processing strategy remains quite complex and can consume a fair amount of resources. SQL Server uses a number of techniques to optimize resource consumption:

- Syntax-based optimization of the query
- Trivial plan match to avoid in-depth query optimization for simple queries
Index and join strategies based on current distribution statistics
• Query optimization in multiple phases to control the cost of optimization
• Execution plan caching to avoid the regeneration of query plans

The following techniques are performed in order, as shown in the flowchart in Figure 9-1.

• Parsing
• Binding
• Query optimizer
• Execution plan generation, caching, and hash plan generation
• Query execution

Let’s take a look at these steps in more detail.