# C:\Users\stefano.maggi.CONBIPELSPA\Desktop\prom38.png

**Date**: 06/12/2012

**Procedure:** How Do I Record Execution Time Of My T-Sql Code?

**Source:** [**LINK**](http://sqlserver2000.databases.aspfaq.com/how-do-i-time-my-t-sql-code.html)

**Permalink:** [**LINK**](http://heelpbook.altervista.org/2012/how-do-i-record-execution-time-of-my-t-sql-code/)

**Created by:** HeelpBook Staff

**Document Version:** 1.0

# [**How do I record execution time of my T-SQL code?**](http://heelpbook.altervista.org/2012/how-do-i-record-execution-time-of-my-t-sql-code/)

This is for **SQL Server** specifically.

The first thing you can do is simply compare the difference between the timestamp **BEFORE** your query, and the timestamp **AFTER**. For example:

|  |
| --- |
| DECLARE @a DATETIME, @b DATETIME  SET @a = CURRENT\_TIMESTAMP    DECLARE @i INT  SET @i = 0  WHILE @i < 10000  BEGIN      SET @i = @i + 1  END  SET @b = CURRENT\_TIMESTAMP  SELECT DATEDIFF(MS, @a, @b) |

You can achieve similar results by running **SQL Profiler**, setting appropriate filters, and watching the **Duration** column as your query runs.

Finally, you can alter the above code slightly so that you see all of the durations on the messages tab of **Query Analyzer**:

|  |
| --- |
| SET STATISTICS TIME ON      — query here  SET STATISTICS TIME OFF |

Then if you look in **Query Analyzer’s Messages** tab, you will see the number of *milliseconds* taken by each step in your query.

Obviously, this is much more useful for queries with a reasonable amount of unique queries, and doesn’t do much good for code with loops. This particular **STATISTICS** option prints durations for every single operation (*each iteration of a loop is recorded*), so there will be 10,000 messages, likely all stating **0 ms**. This could really impact the time it takes to execute a query.

What you can do after this, to compare two queries (and perhaps get to the bottom of why one takes longer than the other), is to turn on **Show Execution Plan** (**CTRL+K**) and view that tab after your queries are finished.

You will be able to spot table scans and other operations with high I/O or CPU costs.