How much Procedure Cache memory does one SQL Statement take up?

I was recently remembered of a time I was working more with performance tuning on Oracle. The following is something that I had discovered to be very similar between Oracle and Microsoft SQL Server, but the names are different between the two systems.

So this brings me back to the question of how much procedure cache memory does one SQL Statement take up? The answer is either very little memory, or all of the memory available to the cache, depending on how you do it.

Take the following query as an example.

```
SELECT column1
FROM table1
WHERE id = ???;
```

Where ??? represents the actual id number being passed into a query. For instance:

```
SELECT column1FROM table1
WHERE id = 7;

SELECT column1
```

```
FROM table1
WHERE id = 928;

SELECT column1
FROM table1
WHERE id = 12341;
```

Now compare the above queries to the following parameterized query:

```
SELECT column1
FROM table1
WHERE id = @ID_TO_FIND;
```

Where @ID_TO_FIND is a parameter that gets set at runtime.

To start with we need to understand how SQL Server handles the processing of queries. There are 2 steps to running a query, the first is to compile the query (also known as creating the plan), and the second is to run the query. The process of creating the plan (compiling) is very expensive, so SQL Server keeps a cache of these compiled queries around. If SQL Server finds an already compiled query, it uses that, otherwise it is forced to recompile the query.

The difference between these two calling conventions is astounding. In the first calling convention where you pass hard coded strings in each time, the SQL procedure cache will have n unique copies of the first query format cached, where n is the number of unique id parameters that are

passed in.
In the second calling convention where the SQL statement is parameterized and is not changing each time, then we will have 1 copy of the SQL statement in the procedure cache, and this one cached copy is reused over and over again.

Given that the amount of memory for the procedure cache is limited (which it is), and that you could have hundreds or more queries running on your SQL server, and for each of those queries was not parameterized, you would multiply it by the number of unique parameters passed into it, you could easily in a day have hundreds of thousands of un-necessary plan compiling being done on your SQL Server.

Without using parameterized queries, you are just wasting the Procedure Cache. If you are using parameterized queries, then your system will run faster, and scale better.

If you are using Oracle, you have the same concept, but it is called Bind Variables instead of Parameterized Queries.