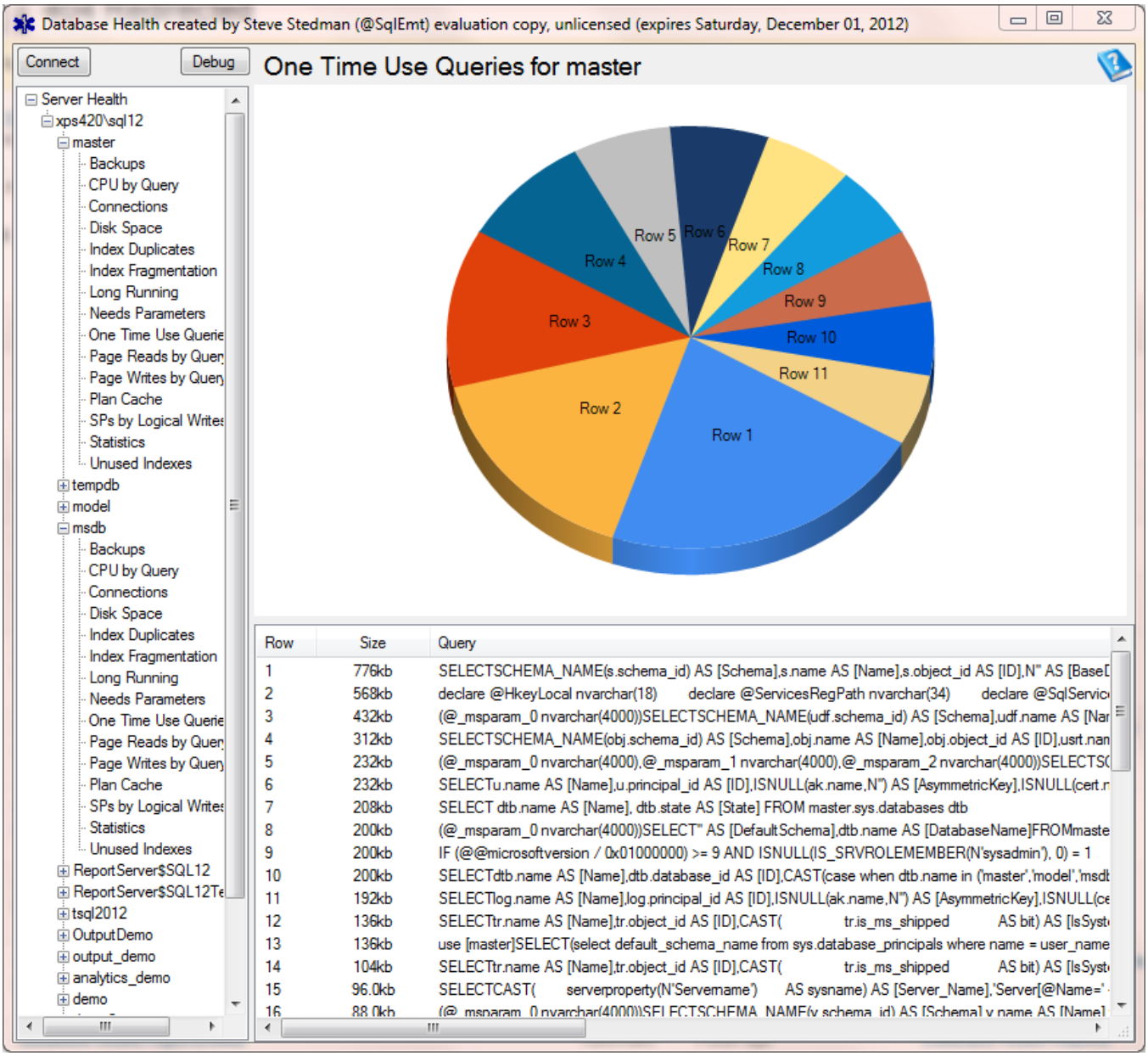
One Time Use Queries

This report shows you the biggest one time use queries in the plan cache. A one time use query is a query that has been parsed and is in the plan cache, but has only been used once.



You get to the One Time Use Queries page from the Database Overview Page or from the hierarchy tree.

Sometimes a one time use query is okay and will just get flushed out of the cache, other times that one time use query will just be a memory hog and wasteful on processor and other resources.

The general solution for one time use queries is parameterization, but first you should understand how SQL Server executes a query.

Steps SQL Server Uses To Execute a Query

- Parse and Normalize
- Compilation
- Optimization (At this point the complied query is stored in the plan cache for reuse)
- Execution

The first three steps can be skipped on the second run if using parameterized queries. Parameterization will lead to smaller plan cache, more of the right queries being available in the plan cache, and overall better performance.

Determining your plan cache size.

In SQL Server 2012, there was a change to the dynamic management view that reports on the plan cache, so the query has changed.

SQL 2012 Plan Cache Size

```
-- How big is the plan cache SQL 2012
select name, sum(pages_kb) /1024.0 MBUsed
from sys.dm_os_memory_clerks
where name = 'SQL PLans'
group by name;
```

Pre-SQL 2012 Plan Cache Size

```
-- How big is the plan cache pre SQL 2012
select name, SUM(single_pages_kb + multi_pages_kb)/1024.0 MBUsed
from sys.dm_os_memory_clerks
where name = 'SQL PLans'
group by name;
```

Although you don't have direct control over the size of the plan cache by any setting or server option, you do have the ability to control the plan cache in the following ways:

- Determining what goes into the plan cache
- Determining available memory on your SQL Server

Sometimes it is easier and cheaper to just add memory to your SQL Server, and not worry about the size of the plan cache, but if you are working on a system that needs to scale and grow, just adding memory only prolongs the issue.

Take the example of the following 3 queries, each of them will use up one occurrence in the plan cache, now consider if you had 40 different departments in the where clause, then you would have 40 occurrences of this in the plan cache. To complicate it ever more you might end up with 80 in the plan cache with 2 for each of the 40, one for parallel processing, and one for standard.

```
go
SELECT d.*
  FROM [Departments] d
 WHERE d.department like 'Swimming';

go
SELECT d.*
  FROM [Departments] d
 WHERE d.department like 'Clearance';

go
SELECT d.*
  FROM [Departments] d
 WHERE d.department like 'Gifts';
```

This can add up pretty quick. Now take a look at the following query

```
SELECT FirstName, LastName
  FROM [Users]
 WHERE user_id = 23439884
```

Consider this user query from the Users table for an online store that has thousands of visitors every day, and a total of over a million users. The above query would probably use up thousands to tens of thousands of rows in the plan cache, and when it is being run thousands of times a day it would have a huge amount of waste in re-parsing, re-normalizing, and re-optimizing this query thousands of times a day.

Now if that query had been written using a parameter as shown here

```
SELECT FirstName, LastName
  FROM [Users]
 WHERE user_id = @param1
```

This query would then only have 1 or 2 occurrences in the plan cache and every time it was run for a different user it would run faster, and not use up any additional space in the plan cache.

The whole point of the one-time use queries report is to let you find those queries that are being used only once, and are using up big chunks of the plan cache.

This is similar to the Queries Needing Parameters report, but the Queries Needing Parameters report shows you the queries that have multiple occurrences that could be specifically fixed by parameters.