# Do Stored Procedures really boost performance in MS SQL / .NET?

Asked 16 years, 1 month ago Modified 8 years, 5 months ago

Viewed 19k times

29

Jeff Atwood wrote about this here, and while I understand the theoretical performance boost a stored procedure could offer, it does seem like a tremendous pain.

What types of queries would you see the most performance increase using stored procedures, and what types of queries would you rather just build on the fly?

Any documentation one way or another would be greatly appreciated.

.net sql-server performance stored-procedures

Share
Improve this question
Follow

edited Jun 27, 2016 at 3:56
Pang
10.1k ● 146 ● 85 ● 124

asked Nov 8, 2008 at 5:41
Jon Smock
9,641 ● 10 ● 49 ● 49

## 11 Answers

Sorted by: Highest score (default)

50

The stored proc/no stored procs argument has become a religious issue. For every person that emphasizes optimized execution plans for procs, another points out common dynamic queries are cached and optimized in most modern DBMSes. For anyone that points out the security a proc might offer, another explains that dynamic queries can be made just as secure. Some like the flexibility of changing a proc without recompiling your app, while others argue queries should be captured in app code so they live and grow in the same code base.

I say...

Do what you like. I doubt we can come up with the correct answer. If procs are a hassle, don't use them. If they seem like a good idea, go for it. I've worked with both models and I honestly don't have a preference. I'm productive with or without 'em.

Share Improve this answer Follow

answered Nov 8, 2008 at 5:54

Corbin March
25.7k ● 6 ● 76 ● 100

That relieves a lot of pressure actually - thanks. I think in general I still want to know if there is any difference and what that difference is (documented), but this is a great philosophy....I wonder if my team will go for it though :-P
– Jon Smock Nov 8, 2008 at 6:16

1 I agree with you, though, about the religious nature of the question, and I almost didn't ask it. But, I do think as good programmers we need to understand exactly what's going on with each component of the system in case we need extreme performance. – Jon Smock Nov 8, 2008 at 6:19

3 Good answer. But it's better to understand the gory details so that you can justify whichever route you choose. See my answer for some of those details. – HTTP 410 Nov 8, 2008 at 18:29

---

17

In days-gone-by, there were *considerable* performance benefits from using stored procedures, but query plan re-use is now significantly better, such that the two are nearly the same in many cases.

If you are building dynamic SQL at the server, you can further increase query plan re-use (and injection safety) by using sp_ExecuteSQL (rather than just EXEC) to execute the SQL.

There are some advantages to using stored procedures:

- They fix exactly what will happen at the db
- For tight security scenarios, you can tightly control access to the stored procedure (rather than the table)
- You can sub-divide the query if the plan is going freaky in spots (sniffing)

However, there are advantages to SQL too:

- It allows ORM tools (LINQ, for example) to write composable queries (page 4, ordered by Foo,Bar) on the fly
- It allows you to write dynamic code in a more expressive language (TSQL is not really intended to be used to write TSQL!)
- You generally have better tooling

Some thing where I would definitely use an SP is in (for example) a data-migration step - i.e. for a bulk operation I might use SqlBulkCopy to push the data into a staging table, then a stored procedure to move the data over. Beyond that, I'm fairly flexible.

Share Improve this answer Follow

answered Nov 8, 2008 at 8:59

Marc Gravell
**1.1m** ● 272 ● 2.6k ● 3k

---

11

As is common with software, the SQL caching issue is more subtle than it looks. For example, let's look at caching of an **ad-hoc SQL** query:

```
-- First, clear the cache
DBCC FREEPROCCACHE

-- Look at what executable plans are in cache
SELECT sc.*
FROM master.dbo.syscacheobjects AS sc
WHERE sc.cacheobjtype = 'Executable Plan'

-- Execute the following statement
SELECT t.*
FROM pubs.dbo.titles AS t
```

```
WHERE t.price = 19.99

-- Look at what executable plans are in cache and you'
-- find that there's a plan for a NUMERIC(4,2)
SELECT sc.*
FROM master.dbo.syscacheobjects AS sc
WHERE sc.cacheobjtype = 'Executable Plan'

-- If you execute the EXACT same statement with a 4,2
-- then you will get THAT plan. But if you execute wit
-- then you'll get a new plan. Try this:
SELECT t.*
FROM pubs.dbo.titles AS t
WHERE price = 199.99

-- Look again at the cached executable plans, and you'
SELECT sc.*
FROM master.dbo.syscacheobjects AS sc
WHERE sc.cacheobjtype = 'Executable Plan'
```

You can however use **sp_executesql** to type the parameters and force the plan to be cached. All subsequent uses will get the same plan, but some people don't like the obscurity of this approach:

```
DECLARE @ExecStr nvarchar(4000)
SELECT @ExecStr = N'SELECT t.* FROM dbo.titles AS t WH
EXEC sp_executesql @ExecStr, N'@price money', 19.99
```

Now if you create a similar query as a **stored procedure** with a parameter for the price, the plan will be created and cached in memory (not disk) on the first execution, and reused regardless of the parameter's value.

The fact that the stored procedure plan is cached in memory and not on disk means that it will fall out of the cache on a server restart or due to low re-use. It can also fall out of cache if the data on which the procedure depends changes enough to cause the statistics to be invalidated. This causes SQL Server to invalidate the plan.

Share Improve this answer Follow

answered Nov 8, 2008 at 18:15

HTTP 410
17.6k ● 14 ● 79 ● 129

+1 Great Answer, I'd read a little of parameter sizes bloating the proc cache but that's a great example to show/beat people ;) – Meff Nov 14, 2008 at 17:20

8

We've found a benefit of Stored Procedures is that once initially created by the Developer, they can be handed over to the DBA or Database Tuning experts to be better written for performance reasons if needs be.

The DBA's obviously don't need access to the application code to do this, and can work alongside a pending application release.

Granted, the same is true of inline or embedded SQL queries, but I think Stored Procedures lend themselves better to this co-operative, larger team way of working.

Share Improve this answer Follow

answered Nov 8, 2008 at 9:24

Andrew
13.2k ● 15 ● 57 ● 87

6 Persoanlly, I'd want my "tuning experts" to be regular devs who understand the application code. In particular, I'd want them to have direct access to the same integration tests (which are probably in C# etc). Also, many times, refactoring the app code itself is the best way to improve performance. – Marc Gravell Nov 8, 2008 at 9:44

I agree with Marc, but I gave this a +1, because I think Andrew's stance is valid and useful. Nice. – Jon Smock Nov 13, 2008 at 21:12

1 Although I would want it tuned by the devs, the harsh cold reality is that most devs have zero clue how to create good queries. This is the number one reason for linq's existence. – ChrisLively Dec 8, 2008 at 20:00

That's a good point Marc, and I agree that most times there's more benefit it re-writing the app. code rather than the query. However, I've seen DBA's who really know their domain restructure a query so it returns the same data but in a fraction of the time of the original query! – Andrew Dec 15, 2008 at 11:33

6

The greatest advantages of stored procedures, in my opinion, are:

1/ They are centralized at the database management system so the DBAs know exactly what they're doing and can optimize the database to suit them. Ad-hoc queries from clients are much harder to do this for. I've seen databases bought to their knees because the DBAs allowed unfettered queries. They also force clients to think about what they need.

2/ They can do very complicated processing at the server minimizing the amount of data sent across the wire. This saves the applications from getting a bucketload of data, deciding something from it, then getting more data, and so on.

By all means, allow any random queries during development but, before you go to production, work out which DB actions should be moved to stored procs.

As a part-time DBA, I think that means all of them but it's not necessary as long as you keep a modicum of control over what queries the applications can send through.

Share Improve this answer Follow

answered Nov 8, 2008 at 8:39

paxdiablo
880k ● 241 ● 1.6k ● 2k

#2 is not a factor people who say use dynamic will agree with you and point to that as examples of where stored procedures are the proper tool. – Will Dieterich Dec 8, 2008 at 20:00

5

Stored Procedures providing a performance benefit is a myth held over from earlier versions of SQL Server. SQL Server > version 7 treats all queries the same and will cache execution plans for all commonly used queries, no matter their genesis.

I wonder how many more years this myth will live on.

Share Improve this answer Follow

answered Nov 8, 2008 at 5:58

John Sheehan
**78.1k** ● 30 ● 161 ● 194

Can you provide documentation of this? – Jon Smock Nov 8, 2008 at 6:14

sommarskog.se/dynamic_sql.html: If you use EXEC to run SQL, e.g.'EXEC @dynamic_sql', it will cache a plan but will not parametrise (or normalise it), meaning that unless exactly the same query is sent it will not re-use the cached query plan. – Mitch Wheat Nov 8, 2008 at 6:16

If you're using an ORM or similar method, the same queries are being sent all the time. – John Sheehan Nov 8, 2008 at 6:20

This "myth" will live on for many years, because it's not really a myth. See my answer for the gory details. – HTTP 410 Nov 8, 2008 at 18:23

---

2

Frans Bouma caused a stir a while back with this topic.

On SQL Server, it's more important to understand that any performance increase is due to the query-plan being cached. A query plan is quite expensive to compile, so it makes sense to cache them. EDIT: Ad-hoc SQL is less likely to be matched to a cached query plan.. Dynamic SQL can be cached if it is run by calling sp_executesql.

If you use EXEC to run SQL, e.g.'EXEC @dynamic_sql', it will cache a plan but will not parametrise (or normalise it), meaning that unless exactly the same query is sent it will not re-use the cached query plan.

Share Improve this answer Follow

edited Nov 8, 2008 at 6:15

answered Nov 8, 2008 at 5:46

Mitch Wheat
300k ● 44 ● 477 ● 550

But use sp_ExecuteSQL and it will – Marc Gravell Nov 8, 2008 at 8:51

But sp_executesql is rather obscure, and complete overkill just for the purpose of caching an execution plan.
– HTTP 410 Nov 8, 2008 at 18:22

2

One of the ideas behind a stored procedure is that it allows you to do several things in a single trip to the database.

MSSQL allows query batches to be submitted in a single request; this is roughly equivalent.

However, the principle is the same. In a large scale distributed system, latency to the database (rather than time the query takes on the server) can be a performance issue. Issuing fewer queries can be of benefit.

So using query batches or stored procedures vs issuing single queries is a potential performance win in a distributed system.

Share Improve this answer

answered Nov 8, 2008 at 11:41

Follow

MarkR
63.5k ● 15 ● 119 ● 154

0

The idea behind Stored Procedures being faster is that they can store cached query plans, kinda like precompiling the stored procedure. So if the procedure is very complex with lots of joins between lots of tables you may see a speed benefit.

I think stored procs are desired for more than their speed, they help abstract the database schema and help to prevent sql injection attacks.

Share Improve this answer

Follow

edited Nov 8, 2008 at 5:47

Mitch Wheat
300k ● 44 ● 477 ● 550

answered Nov 8, 2008 at 5:44

vfilby
10k ● 9 ● 51 ● 62

I know this is 3 years old at this point, but for posterity's sake: there is nothing about a sproc that helps it prevent SQL injection. – Bryan B Dec 12, 2011 at 18:17

0

A badly written stored procedure will give no performance boost. The same as a bad inline query will have worse performance than a well thought out one.

Share Improve this answer

answered Nov 8, 2008 at 6:09

Follow

wonderchook
1,200 ● 10 ● 18

0

One thing people are skipping are that all those CRUD stored procedures are filled with coalesce and if statements and those are painfully slow.

If you are using parametrized statements vs a coalesce using stored procedure the dynamic statements are going to win.

Share Improve this answer

Follow

answered Dec 8, 2008 at 19:55

Will Dieterich
504 ● 3 ● 9