What is your naming convention for stored procedures? [closed]

118

57

I have seen various rules for naming stored procedures.

Some people prefix the sproc name with usp_, others with an abbreviation for the app name, and still others with an owner name. You shouldn't use sp_ in SQL Server unless you really mean it.

Some start the proc name with a verb (Get, Add, Save, Remove). Others emphasize the entity name(s).

On a database with hundreds of sprocs, it can be very hard to scroll around and find a suitable sproc when you think one already exists. Naming conventions can make locating a sproc easier.

Do you use a naming convention? Please describe it, and explain why you prefer it over other choices.

Summary of replies:

- Everybody seems to advocate consistency of naming, that it might be more important for everyone to use the same naming convention than which particular one is used.
- Prefixes: While a lot of folks use usp_ or something similar (but rarely sp_), many others use database or app name. One clever DBA uses gen, rpt and tsk to distinguish general CRUD sprocs from those used for reporting or tasks.
- Verb + Noun seems to be slightly more popular than Noun + Verb. Some people use the SQL keywords (Select, Insert, Update, Delete) for the verbs, while others use non-SQL verbs (or abbreviations for them) like Get and Add. Some distinguish between singluar and plural nouns to indicate whether one or many records are being retrieved.
- An additional phrase is suggested at the end, where appropriate. GetCustomerById, GetCustomerBySaleDate.
- Some people use underscores between the name segments, and some avoid underscores. app_ Get_Customer vs. appGetCustomer -- I guess it's a matter of readability.
- Large collections of sprocs can be segregated into Oracle packages or Management Studio (SQL Server) solutions and projects, or SQL Server schemas.
- Inscrutable abbreviations should be avoided.

inadvertently creates a duplicate sproc with another name.

Since I generally work on very large apps with hundreds of sprocs, I have a preference for the easiest-to-find naming method. For a smaller app, I might advocate Verb + Noun, as it follows the general coding convention for method names.

He also advocates prefixing with app name instead of the not very useful usp_. As several people pointed out, sometimes the database contains sprocs for multiple apps. So, prefixing with app name helps to segregate the sprocs AND helps DBAs and others to determine which app the sproc is used for.

sql stored-procedures naming-conventions

edited Oct 26 '08 at 23:50

Mitch Wheat
261k 36 411 503

asked Oct 26 '08 at 17:03

DOK
28.5k 7 53 89

closed as not constructive by casperOne Mar 28 '12 at 17:45

As it currently stands, this question is not a good fit for our Q&A format. We expect answers to be supported by facts, references, or expertise, but this question will likely solicit debate, arguments, polling, or extended discussion. If you feel that this question can be improved and possibly reopened, visit the help center for guidance.

If this question can be reworded to fit the rules in the help center, please edit the question.

1 What does usp stand for? – Midhat May 4 '10 at 13:09

1 I believe that usp is short for "user procedure". That distinguishes it from the system procedures prefixed "sp_". That is an important distinction, as you can read in the answers. – DOK May 5 '10 at 14:37

thanks dok. grazie mille – Midhat May 6 '10 at 8:43

sqlblog.com/blogs/aaron_bertrand/archive/2008/10/30/… (also see: stackoverflow.com/questions/871612/…) – mg1075 Mar 21 '13 at 18:00

1 I'm upvoting this just because it's closed, hopefully to show the powers that be that questions like this are useful to the community. – tsilb Feb 10 '16 at 22:59

64

For my last project i used usp_[Action][Object][Process] so for example, usp_AddProduct or usp_GetProductList, usp_GetProductDetail. However now the database is at 700 procedures plus, it becomes a lot harder to find all procedures on a specific object. For example i now have to search 50 odd Add procedures for the Product add, and 50 odd for the Get etc.

Because of this in my new application I'm planning on grouping procedure names by object, I'm also dropping the usp as I feel it is somewhat redundant, other than to tell me its a procedure, something I can deduct from the name of the procedure itself.

The new format is as follows

```
[App]_[Object]_[Action][Process]

App_Tags_AddTag
App_Tags_AddTagRelations
App_Product_Add
App_Product_GetList
App_Product_GetSingle
```

It helps to group things for easier finding later, especially if there are a large amount of sprocs.

Regarding where more than one object is used, I find that most instances have a primary and secondary object, so the primary object is used in the normal instance, and the secondary is refered to in the process section, for example App_Product_AddAttribute.

edited Oct 26 '08 at 21:15

answered Oct 26 '08 at 17:50

dnolan

1,959 16 20

3 What if more than one Object is involved? For example, what if the sproc queries information from both the Customer and the Orders table? – DOK Oct 26 '08 at 18:27

2 Thanks Mitch, let's clarify. That "App" prefix is a placeholder for another abbreviation indicating the actual app's name (or acronym). With 3 apps sharing one database, then, there might be ICA_Product_Add, CRM_Product_Add and BPS_Product_Add. – DOK Oct 27 '08 at 10:54

2 Why would you duplicate every procedure 3 times for 3 apps? The whole point of Store Procedures is to have a single place where a given action happens. "ICA_Product_Add, CRM_Product_Add and BPS_Product_Add" destroys that. – Jason Kester Oct 27 '08 at 11:44

2 Jason, those sprocs may be inserting to different tables. They might have different input parameters or return values. Or they may have different behavior. If the sprocs do the same thing, I agree, there should only be one version. As someone else suggested, shared sprocs might have no prefix.

33

Here's some clarification about the sp_ prefix issue in SQL Server.

Stored procedures named with the prefix sp_ are system sprocs stored in the Master database.

If you give your sproc this prefix, SQL Server looks for them in the Master database first, then the context database, thus unnecessarily wasting resources. And, if the user-created sproc has the same name as a system sproc, the user-created sproc won't be executed.

The sp_ prefix indicates that the sproc is accessible from all databases, but that it should be executed in the context of the current database.

Here's a nice explanation, which includes a demo of the performance hit.

Here's another helpful source provided by Ant in a comment.

edited Oct 26 '08 at 18:31

answered Oct 26 '08 at 18:19

DOK

28.5k 7 53 89

Hmm I don't understand. Why does sp give a performance hit? Is usp or gsp okay? – Erwin Rooijakkers May 15 '14 at 9:48

1 @user2609980 DOK says SQL Server searches for `sp_` prefixed proc in Master DB first, then in current DB if not found – GôTô Dec 6 '14 at 21:26

+1 for clearly stating something that has more convoluted explanations elsewhere. Not news to me, but I think this is simple and concise explanation to someone starting out. – undrline Jan 16 at 13:51

The link to the demo of the performance hit is from an article written in 2001. It's changed since then, here's a more thorough article (from 2012) by Aaron Bertrand: sqlperformance.com/2012/10/t-sql-queries/sp_prefix – Michael J Swart Apr 4 at 14:23

16

Systems Hungarian (like the above "usp" prefix) makes me shudder.

We share many stored procedures across different, similarly-structured databases, so for database-specific ones, we use a prefix of the database name itself; shared procedures have no prefix. I suppose using different schemas might be an alternative to get rid of such

Responding to Ant's comment:

1. The difference between a table and a view is relevant to those who design the database schema, not those who access or modify its contents. In the rare case of needing schema specifics, it's easy enough to find. For the casual SELECT query, it is irrelevant. In fact, I regard being able to treat tables and views the same as a big advantage.
2. Unlike with functions and stored procedures, the name of a table or view is unlikely to start with a verb, or be anything but one or more nouns.
3. A function requires the schema prefix to be called. In fact, the call syntax (that we use, anyway) is very different between a function and a stored procedure. But even if it weren't, the same as 1. would apply: if I can treat functions and stored procedures the same, why shouldn't I?

edited Oct 26 '08 at 18:21

answered Oct 26 '08 at 17:29

Sören Kuklau
15.5k 5 38 72

1 Sooo, how do you know whether you're interacting with a procedure, a function, a view, a table, or anything else? – Ant Oct 26 '08 at 17:47

1 I would imagine that functions might begin with "Get" or be a name that doesn't begin with a verb. Everything else would be a procedure because after all, they are called stored procedures. Procedures hide the specifics like views, tables and anything else. – Mark Stock Oct 26 '08 at 18:09

3 But it's not Hungarian. The "usp" isn't a Hungarian variable declaration. The "u" doesn't stand for "update", it stands for "user", as in "user defined stored procedure", and it's merely protecting from SQL Server looking in the Master DB every time it's searching for your stored procedure. Naturally, there are other ways, but "usp" is generally widely considered a standard in many corps, and from what I have seen it works well. It's also taught by Microsoft, and a Microsoft recommended naming convention: msdn.microsoft.com/en-us/library/ms124456(v=SQL.100).aspx – asus3000 Jun 8 '17 at 13:28

10

Starting a stored procedure name with `sp_` is bad in SQL Server because the system sprocs all start with sp_. Consistent naming (even to the extent of hobgoblin-dom) is useful because it facilititates automated tasks based on the data dictionary. Prefixes are slightly less useful in SQL Server 2005 as it supports schemas, which can be used for various types of namespaces in the way that prefixes on names used to. For example, on a star schema, one could have *dim* and *fact* schemas and refer to tables by this convention.

For stored procedures, prefixing is useful for the purpose of indentifying application sprocs from system sprocs. `up_` vs. `sp_` makes it relatively easy to identify non-system stored procedures from the data dictionary.

2 Naming sprocs "sp_" is a really bad idea for speed, too, because SQL Server tries to optimise its lookups for those based on the assumption that they are system procedures. Have a look here, 5th point down: rakph.wordpress.com/2008/04/19/tips-store-procedure – Ant Oct 26 '08 at 17:46

9

TableName_WhatItDoes

- Comment_GetByID
- Customer_List
- UserPreference_DeleteByUserID

No prefixes or silly hungarian nonsense. Just the name of the table it's most closely associated with, and a quick description of what it does.

One caveat to the above: I personally always prefix all my autogenerated CRUD with zCRUD_ so that it sorts to the end of the list where I don't have to look at it.

answered Oct 26 '08 at 23:48

Jason Kester
3,788 9 30 38

Segregating the "z" items from the rest sounds like a great idea. – DOK Oct 27 '08 at 11:00

I like this method. They need to be easy to find. When I'm looking thru a list of verb first sprocs and see 200 Gets, 200 Inserts, 200 updates, it's hard to find all the ones for a specific table or grouping. I've used the verb method first, and it gets to be a mess quick. Table name first solves this issue. So for example above in the answer, all your Comment or Customer ones would be grouped together, easy to find. – astrosteve Aug 29 '16 at 22:41

1 And what if you have a query joining several tables? – leandronn Nov 17 '16 at 20:39

8

I have used pretty much all of the different systems over the years. I finally developed this one, which I continue to use today:

Prefix :

Action Specifier:

```
Ins - INSERT
Sel - SELECT
Upd - UPDATE
Del - DELETE
```

(In cases where the procedure does many things, the overall goal is used to choose the action specifier. For instance, a customer INSERT may require a good deal of prep work, but the overall goal is INSERT, so "Ins" is chosen.

Object:

For gen (CRUD), this is the table or view name being affected. For rpt (Report), this is the short description of the report. For tsk (Task) this is the short description of the task.

Optional Clarifiers:

These are optional bits of information used to enhance the understanding of the procedure. Examples include "By", "For", etc.

Format:

[Prefix][Action Specifier][Entity][Optional Clarifiers]

Examples of procedure names:

```
genInsOrderHeader

genSelCustomerByCustomerID
genSelCustomersBySaleDate

genUpdCommentText

genDelOrderDetailLine

rptSelCustomersByState
rptSelPaymentsByYear

tskQueueAccountsForCollection
```

4 Now, there's an interesting take on the prefix. That looks like a good way to segregate sprocs by their usage. – DOK Oct 26 '08 at 19:01

4

I always encapsulate the stored procedures in **packages** (I'm using Oracle, at work). That will reduce the number of separate objects and help code reuse.

The naming convention is a matter of taste and something you should agree with all the other developers at project start.

answered Oct 26 '08 at 17:09

Gabriele D'Antona
5,460 3 37 77

Packages are good. Starting with SQL Server 2005, Management Studio enables creating "solutions" to store related sprocs and other SQL statements. – DOK Oct 26 '08 at 17:16

2 @DOK - note that these packages have no footprint in the database itself, though. They are purely artifacts of the front-end tool. You can't query by package in the data dictionary. Oracle packages are first class objects in the system data dictionary and have their own scope. – ConcernedOfTunbridgeWells Apr 12 '11 at 11:47

4

for small databases, i use uspTableNameOperationName, e.g. uspCustomerCreate, uspCustomerDelete, etc. This facilitates grouping by 'main' entity.

for larger databases, add a schema or subsystem name, e.g. Receiving, Purchasing, etc. to keep them grouped together (since sql server likes to display them alphabetically)

i try to avoid abbreviations in the names, for clarity (and new people on the project don't have to wonder what 'UNAICFE' stands for because the sproc is named uspUsingNoAbbreviationsIncreasesClarityForEveryone)

answered Oct 26 '08 at 17:16

Steven A. Lowe
54.4k 16 122 198

at 19:07

@[DOK]: thanks; the 'best' answer is probably the combination that makes sense for your situation. – Steven A. Lowe Oct 27 '08 at 16:42

4

I currently use a format which is like the following

Notation:

[PREFIX]*[APPLICATION]*[MODULE]_[NAME]

Example:

P_CMS_USER_UserInfoGet

I like this notation for a few reasons:

- starting with very simple Prefix allows code to be written to only execute objects beggining with the prefix (to reduce SQL injection, for example)
- in our larger environment, multiple teams are working on different apps which run of the same database architecture. The Application notation designates which group owns the SP.
- The Module and Name sections simply complete the heirarchy. All names should be able to be matched to Group/App, Module, Function from the heirarchy.

answered Oct 26 '08 at 21:02

pearcewg
5,633 17 72 122

2

I always use:

usp[Table Name][Action][Extra Detail]

Given a table called "tblUser", that gives me:

The procedures are alphabetically sorted by table name and by functionality, so it's easy to see what I can do to any given table. Using the prefix "usp" lets me know what I'm calling if I'm (for example) writing a 1000-line procedure that interacts with other procedures, multiple tables, functions, views and servers.

Until the editor in the SQL Server IDE is as good as Visual Studio I'm keeping the prefixes.

answered Oct 26 '08 at 17:54

Ant
4,175 1 24 42

2

application prefix_ operation prefix_ description of database objects involved *(minus the spaces between underscores - had to put spaces in for them to appear)*.

operation prefixes we use -

- "*get*" – returns a recordset
- "*ins*" – inserts data
- "*upd*" – updates data
- "*del*" – deletes data

e.g

wmt_ ins _ customer _details

"workforce management tool, insert details into customer table"

advantages

All stored procedures relating to the same application are grouped together by name. Within the group, stored procedures that carry out the same kind of operation (e.g. inserts, updates, etc.) are grouped together.

This system works well for us, having approx. 1000 stored procedures in one database off the top of my head.

I generally abhor the use of underscores, but the way you use it -- not just to segregate the prefix, but also to segregate the operation -- would make it easier to find when scanning a list of hundreds of sprocs. Pretty_neat_idea. – DOK Oct 26 '08 at 18:49

2

GetXXX - Gets XXX based on @ID

GetAllXXX - Gets all XXX

PutXXX - Inserts XXX if passed @ID is -1; else updates

DelXXX - Deletes XXX based on @ID

answered Oct 26 '08 at 23:11

tsilb
5,447 13 59 93

1

I think the usp_ naming convention does nobody any good.

In the past, I've used Get/Update/Insert/Delete prefixes for CRUD operations, but now since I use Linq to SQL or the EF to do most of my CRUD work, these are entirely gone. Since I have so few stored procs in my new applications, the naming conventions no longer matter like they used to ;-)

answered Oct 26 '08 at 17:17

Dave Markle
79k 16 134 158

1 Prefixing every sproc with _usp doesn't help distinguish among them. I think some DBA's like that prefix because it indicates the database object type. Maybe we'll hear from one of them who likes it. – DOK Oct 26 '08 at 17:30

After the prefix we usually start the SP name with a verb that describes what the procedure does, and then the name of the entity that we operate on. Pluralization of the entity name is allowed - We try to emphasize readability, so that it is obvious what the procedure does from the name alone.

Typical stored procedure names on our team would be:

```
shopGetCategories
shopUpdateItem
```

answered Oct 26 '08 at 17:20

driis
126k 41 243 324

Well, you never know, when you're working on a database dedicated to one app, whether there will be another app later using the same database. In your situation, it sure does help segregate the sprocs. – DOK Oct 26 '08 at 17:33

1

I don't think it really matters precisely what your prefix is so long as you're logical and consistent. Personally I use

spu_[action description][process description]

where action description is one of a small range of typical actions such as get, set, archive, insert, delete etc. The process description is something short but descriptive, for example

```
spu_archiveCollectionData
```

or

```
spu_setAwardStatus
```

I name my functions similarly, but prefix with udf_

Cruachan
14.1k 4 54 105

spu_, interesting. Dodges the SQL Server sp_ issue. – DOK Oct 26 '08 at 17:45

1

Avoid sp_* in SQl server coz all system stored prcedures begins with sp_ and therefore it becomes more harder for the system to find the object corresponding to the name.

So if you begin with something other than sp_ things become easier.

So we use a common naming of Proc_ to begin with. That makes it easier to identify the procedures if presented with one big schema file.

Apart from that we assign a prefix that identify the function. Like

`Proc_Poll_Interface, Proc_Inv_Interface` etc.

This allows us to find all stored procs which does the job of POLL vs that does Inventory etc.

Anyhow the prefix system depends on your problem domain. But al said and done something similar ought to be present even if it be just to allow people to quicly locate the stored procedure in the explorere drop down for editing.

other eg's of function.

```
Proc_Order_Place
Proc_order_Delete
Proc_Order_Retrieve
Proc_Order_History
```

We followed the function based naming coz Procs are akin to code / function rather than static objects like tables. It doesnt help that Procs might work with more than one table.

answered Oct 26 '08 at 17:58

computinglife

3,833 1 15 16

1

I joined late the thread but I want to enter my reply here:

In my last two projects there are different trends like, in one we used:

> To get Data : s<tablename>_G
> To delete Data : s<tablename>_D
> To insert Data : s<tablename>_I
> To update Data : s<tablename>_U

This naming conventions is also followed in front-end by prefixing the word **dt**.

Example:

```
exec sMedicationInfo_G
exec sMedicationInfo_D
exec sMedicationInfo_I
exec sMedicationInfo_U
```

With the help of above naming conventions in our application we have a good and easy to remember names.

While in second project we used the same naming conventions with lill difference:

> To get Data : sp_<tablename>G
> To delete Data : sp_<tablename>D
> To insert Data : sp_<tablename>I
> To update Data : sp_<tablename>U

Example:

answered Jun 13 '09 at 22:32

community wiki
Gaurav Aroraa

	Interesting. I've never seen it done quite this way, but it's easy to remember, or guess, the correct names. – DOK Jun 14 '09 at 11:36
1	Thanks DOK, Yes, its easy to remember and we developer feel free from any complexity in names – Gaurav Aroraa Jun 15 '09 at 5:29
9	Why not _C _R _U _D? – onedaywhen Jun 15 '09 at 8:24
	@onedaywhen - its a good idea, I will suggest to our DBA so, we can maintain the naming conversions accordingly. But, main motive to this naming convention to present all the object correctly, unless I missed anything ... – Gaurav Aroraa Jul 18 '12 at 5:57
1	"sp_" prefix is not recommended. – Piyey Jul 27 '17 at 21:54