

IUG 2026 - Sproc Talk - Outline

Introduction

Hi, I'm Dan

I've worked in libraries for 30 years

I've been using Polaris since it was in beta (roughly 1999)

Currently I'm a Polaris Administrator and SQL Hacker at Library Systems & Services

Cyberpunk Librarian

Who is this session for?

Your time at IUG is limited and it is *valuable* so in order to help you spend your time in the best ways

It would be helpful if you know a little bit about T-SQL and Microsoft SQL Server

It's even better if you know how to write SQL queries

I've designed this sesh around the idea that you kinda sorta know what you're looking at when you're looking at SQL

We're gonna talk about sprocs!

What they are

What they do

Why they're important to Polaris

How Polaris uses them

And how to make your own (and why you might want to)

And you know what that means? Right?

That's right! It means I haven't learned a damn thing and we're gonna go live on a real Polaris server and we're going to do *activities!*

What do we talk about when we talk about sprocs?

Namely - what is a sproc?

Sproc is a shortening of stored procedure

Great, what's a stored procedure?

You ever watch a cooking show? No, really, I'm going somewhere with this. So, have you ever watched a cooking show where the host is making something and they say "Here's one I prepared in advance" and they pull an entire Thanksgiving dinner out of the oven?

Yeah, stored procedures are kinda like that. They're SQL queries that you prepare in advance and you can pull them out whenever you need them.

A stored procedure is the way you save SQL queries to the database itself

And this allows for consistency

You can execute upon that query whenever you want

Sprocs can be simple in that they can literally just be a SQL query that runs when you want it to

Or they can be more complex and you can pass values to them when you run them

Stored procedures can be called in a variety of ways

Reports

Within database jobs

Within SQL queries and even within other stored procedures

Okay cool... so what?

Well, I have often joked that Polaris is a friendly front end for SQL Server

Almost every time you take an action in Polaris, whether it's looking up a patron, checking out an item, adding a vendor, updating a MARC record... anytime you do something — you execute a sproc (likely multiple sprocs)

If you really want to know how Polaris works and how it gets things done? It's stored procedures (use the astronaut meme)

There are over 4,500 stored procedures in the Polaris database

And we're gonna look at *all* of them! I hope you used the restroom during the interval!

Polaris Stored Procedures

Like I said, almost everything you do in Polaris executes a stored procedure.

If you hit Enter

If you click a button

If you search for, open, modify, and save a file

Even when you log on... you executed a sproc (*Literally*. It's called Polaris.Polaris.ILS_LoginUser)

Which brings us to our next point

Where in the hell do they keep the sprocs around here?

Well, for the Polaris stored procedures, you'll find them under Databases -> Polaris -> Programmability -> Stored Procedures

And when you expand the Stored Procedures folder, give it a couple of seconds to fully populate. Yeah, it takes a little time for that to happen.

Right then, now that we've found them, how are they organized?

Well, quite well actually. Sure there are some exceptions, but it's not too bad. You can get an idea of what a sproc does by its prefix. The major ones are:

Acq - Acquisitions

Cat - Cataloguing

Circ - Circulation

CNV - Typically you only see these tables if you migrated from another ILS, which means you're likely to see them unless, somehow, y'all built a brand new library that never existed before and never had an ILS. I'm sure those places exist, but... yeah. These are sprocs used to help CoNVert parts and pieces of your old ILS into Polaris.

Community - Community profiles and related functionality

IDX - Indexing related sprocs

ILL - Interlibrary Loan

IRX - Deals with Radix levels and data structures that I am not going to even pretend to understand.

PAC - PowerPAC

PAPI - Polaris API

RPT - If ever you ran a report, chances are, here's where it comes from.

SA - System Administration related tasks.

Z39.50 - Keep in mind, that almost every search you do in Polaris is a clandestine Z39.50 search

Sprocs - What do they do and how do they work?

Well, rather than just *telling* you how these things work, let's go look at one... in the wild!

Here we find the wiley sproc in its natural habitat...

This sproc is called Rpt_NewTitles.

We can see from the prefix that this sproc powers a report! So... guess what it does?

It's also a small one, which is easier to deal with. No joke, some of these things are BIG.

Let's go line by line, shall we?

Polaris.Polaris.Rpt_NewTitles

[CODE]

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

ALTER PROCEDURE [Polaris].[Rpt_NewTitles]
@dtImported datetime,
@OrganizationList as NVARCHAR(MAX)
AS

/
*****
*****
Rpt_NewTitles -- list all new bibliographic records added to the database
after a
given date

Created: 08-Jun-2001 Beth Silliman
Modified: 09-Sep-2002 Jim Mieczkowski

Unicode support.

*****
*****/

SET NOCOUNT ON

CREATE TABLE #Branches (OrganizationID INT)
IF LEN(@OrganizationList) = 1 AND @OrganizationList= N'0' -- ALL SELECTED
BEGIN
    INSERT INTO #Branches
        SELECT OrganizationID FROM Polaris.Organizations(NOLOCK) WHERE
OrganizationCodeID in (1,3)
END
ELSE
```

```

BEGIN
    EXEC (N'INSERT INTO #Branches SELECT OrganizationID FROM Organizations
(NOLOCK) WHERE OrganizationID in(' + @OrganizationList + N')')
END

SELECT
    BR.RecordStatusID,
    BR.BrowseAuthor,
    BR.BrowseTitle,
    BR.BrowseCallNo,
    BR.MARCPubDateOne,
    BR.ImportedDate,
    RS.RecordStatusName,
    MTM.Description,
    O.Name,
    O.OrganizationID,
    LTrim(str(BR.BibliographicRecordID)) AS BibliographicRecordID
FROM
    Polaris.BibliographicRecords BR(NOLOCK)
    INNER JOIN Polaris.PolarisUsers PU(NOLOCK)
    ON (BR.CreatorID = PU.PolarisUserID)
    INNER JOIN Polaris.MARCTypeOfMaterial MTM(NOLOCK)
    ON (BR.PrimaryMARCTOMID = MTM.MARCTypeOfMaterialID)
    INNER JOIN Polaris.RecordStatuses RS(NOLOCK)
    ON (BR.RecordStatusID = RS.RecordStatusID)
    INNER JOIN Polaris.Organizations O(NOLOCK)
    ON (PU.OrganizationID = O.OrganizationID)
    inner join #Branches B
    on (O.OrganizationID = B.OrganizationID)
WHERE
    BR.RecordStatusID IN (1,2) AND
    BR.ImportedDate > @dtImported

ORDER BY
O.Name ASC, BR.ImportedDate DESC

return
GO

```

SET ANSI_NULLS ON - Evaluates both {expression} = NULL and {expression} <> NULL as False if the value of {expression} is NULL.

Fun fact - This is deprecated and since SQL Server 2017, ANSI_NULLS are always ON.

SET QUOTED_IDENTIFIER ON - This gets way into the weeds as to the history of SQL Server, T-SQL, and ANSI, but here's what it does. When you have QUOTED_IDENTIFIER set to ON then:

[CODE]

```
SELECT "Polaris" -- This fails
SELECT 'Polaris' -- This works
```

You'll see GO occasionally, indeed, you see it twice here. If you're wondering why, it's because it's used to send batches of commands one at a time. In this case, ANSI_NULLS must be set on their own and, after they are, QUOTED_IDENTIFIER must be set on their own.

ALTER PROCEDURE *Polaris.Rpt_NewTitles*

Stored procedures can be created, altered, dropped, and executed. In this example, I chose to pull the code for the procedure in such a way that, if I ran the code, it would change the procedure rather than trying to create it again and I *really* don't want to drop it!

@dtImported datetime,

@OrganizationList as NVARCHAR(MAX)

Here we're defining our parameters that will be passed to the stored procedure. As you can see we've got a datetime parameter that looks like it has something to do with when bibliographic records were imported. And this OrganizationList? Well, that's literally going to be holding on to your libraries. We'll get to that in a sec.

AS

SET NOCOUNT ON

We'll come back to those comments, because I love them. Seriously, I do. But if you're wondering what SET NOCOUNT ON does, in true coding fashion, setting NOCOUNT to ON actually turns *off* a feature.

Look, I didn't design this crap.

Normally, you would get another result set after running your sproc and that result set would give you a count of the rows affected. Do you need that? Probably not, so you set NOCOUNT to ON and now you don't get that extra result set.

Now, beloved... the comments. Look at this. This sproc lists "all new bibliographic records added to the database after a given date." Hey, Beth Silliman wrote this back in 2001. Polaris was *brand freakin' new* in 2001. Since I'm pushing 50, this code was written almost literally half my life ago. This code is a quarter of a century old. And it was modified once, just over a year later by Jim Mieczkowski. At one time, he was the CIO of Polaris. Jim added unicode support. And we know that because they *told* us that. This is why I always tell anyone who'll listen, comment your code. You never know how long it'll last. That query you wrote to do that one thing, and you're sure that you'll come up with something better later on?

Don't count on it. People tend to have a way of finding better things to do than fixing something that's not actually all that broken. Who knows? 25 years later, someone might be looking at your code to figure out how it works. And that somebody... might be *you*. I am almost positive that I have reports and sprocs that I wrote some ten years ago that are still running in some libraries today. Do the future a favour, tell them about your code.

So let's take a look at all of this:

[CODE]

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

ALTER PROCEDURE [Polaris].[Rpt_NewTitles]
@dtImported datetime,
@OrganizationList as NVARCHAR(MAX)
AS

/
*****
*****
Rpt_NewTitles -- list all new bibliographic records added to the database
after a
given date
```



```
Created: 08-Jun-2001 Beth Silliman  
Modified: 09-Sep-2002 Jim Mieczkowski
```

```
Unicode support.
```

```
*****  
*****/
```

```
SET NOCOUNT ON
```

I call this the preamble. Almost every stored procedure you'll see in SQL Server will look something like this. You'll have your ANSI_NULLS set, your QUOTED_IDENTIFIER, you'll have the operator and the name of the procedure, parameters, and the NOCOUNT. Please, put some comments in there somewhere too. The majority of Polaris sprocs have these comments. They've helped me a lot over the years.

Now, I'm going to do something I don't normally recommend, and I'm going to hand wave *most* of the rest of this code. And the reason I'm going to do that is I'm talking about structure and not substance. So instead of going through what all this does, which I'm sure you'll be thrilled to learn that it pulls a list of bibliographic records added to the system since a given date, I'm just going to say that "your query goes here." However, there are a couple things here worth pointing out:

[CODE]

```
inner join #Branches B  
on (O.OrganizationID = B.OrganizationID)
```

And...

[CODE]

```
BR.ImportedDate > @dtImported
```

See these guys? They're the part of the code that uses those parameters we talked about earlier. We'll get into that in just a second, but I didn't want you to miss these because yeah, they're pretty important!

Okay, cool cool. The rest of this stuff? It's the query. This is where your query goes, right? So like the year

2026 in general, we're gonna go straight to the bottom and look at these last two lines.

[CODE]

```
return  
GO
```

So, you don't need to use RETURN, but it's good form to do so. And now that I've told you that, I can't tell you the last time I did that myself. I mean, hell, I'm the guy up here wearing a punk rock t-shirt... what do you expect? The RETURN statement brings your stored procedure to an end. It returns to your regularly scheduled SQL Server, already in progress. Now, with that, you might find yourself thinking, well... if RETURN takes you out of the sproc, why is there a GO after it? Wouldn't it exit before that?

Yes, and here's the little secret: The SQL Server execution system? It never sees GO. And you know what that means?

That means it never collects \$200.

No, going back to what I said before, the GO statement tells SQL Server to run code in batches, on their own. So what you're seeing here, this code simply tells SQL Server, run this batch. SQL Server gets the batch, but not the word GO.

Okay, great, let's run this and see what happens!

Running down a ~~dream~~ sproc

So this is the easy part, especially for this sproc. We use the EXEC command, and if we're proper thinking individuals we use the full name of the sproc. Now, keep in mind, we need to give that sproc a date and an OrganizationID. And I dunno how y'all do bibliographic records at your library, so I'm going to play it safe and use a 0, because, if you go back and look at the code, it specifically says:

[CODE]

```
IF LEN(@OrganizationList) = 1 AND @OrganizationList= N'0' -- ALL SELECTED
```

In other words, if I pass a 0, that means we get all the bib records!

So, here's the line:

[CODE]

```
EXEC Polaris.Polaris.Rpt_NewTitles '2026-03-01',0
```

That should pull all the bibs imported into this system since March 1, 2026 and, yes, it does.

Now, I'm not going to tell you that all of the Polaris stored procedures are that simple, but this is the foundational level of how things happen in the backend databases.

So, yeah... cool. Go check out some sprocs and see what their code looks like! As a colleague of mine often says "Always be noobin'." Go find yourself something new!

Speaking of something new... let's write our own sproc!

Nothin's says lovin' like sprocs from the oven

NOTE TO SELF: For the example sproc, let's use something that provides a list of MaxItems and MaxReuqestItems for a given patron code at a given branch.

Procedure

1. Log into the server
2. Open SSMS and login
3. Open up the databases → Polaris → expand Programmability then Stored Procedures
4. Right-click Stored Procedures and select New Stored Procedure
5. Take a few minutes to go over the template
6. Delete the comments at the top
7. ANSI NULLS and QUOTED IDENTIFIER are already set to go
8. Update the comments in the middle
9. Explain the CREATE PROCEDURE line and edit for the example
10. Open up the saved sproc and explain
11. Create it

12. Run it with parameters 52,3