Southeast Michigan
IBM i User Group

Stored Procedures for Hybrid Development

MIKE PAVLAK mikepavlak@gmail.com

Agenda

What are Stored Procedures

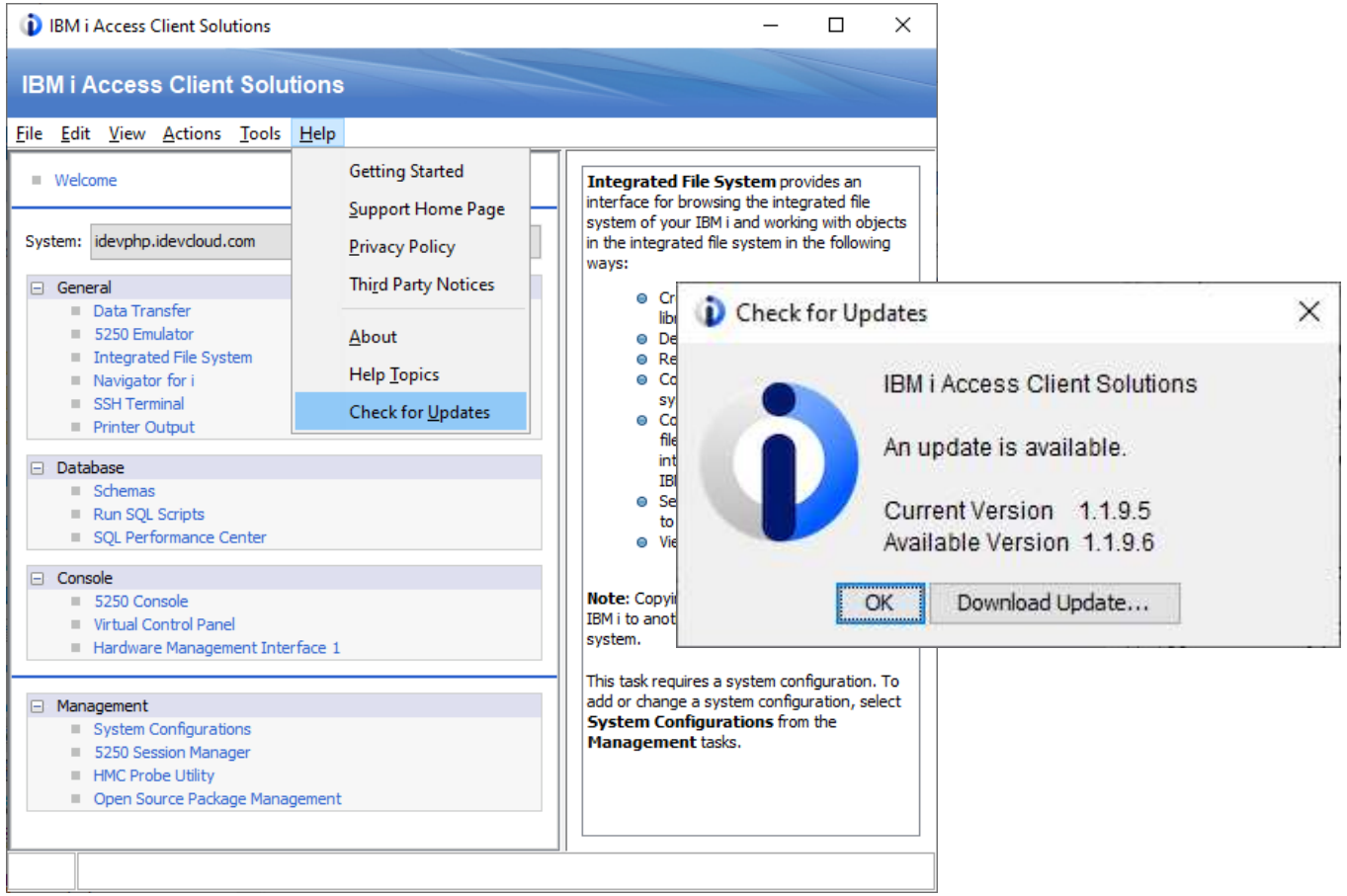
Why use stored procedures

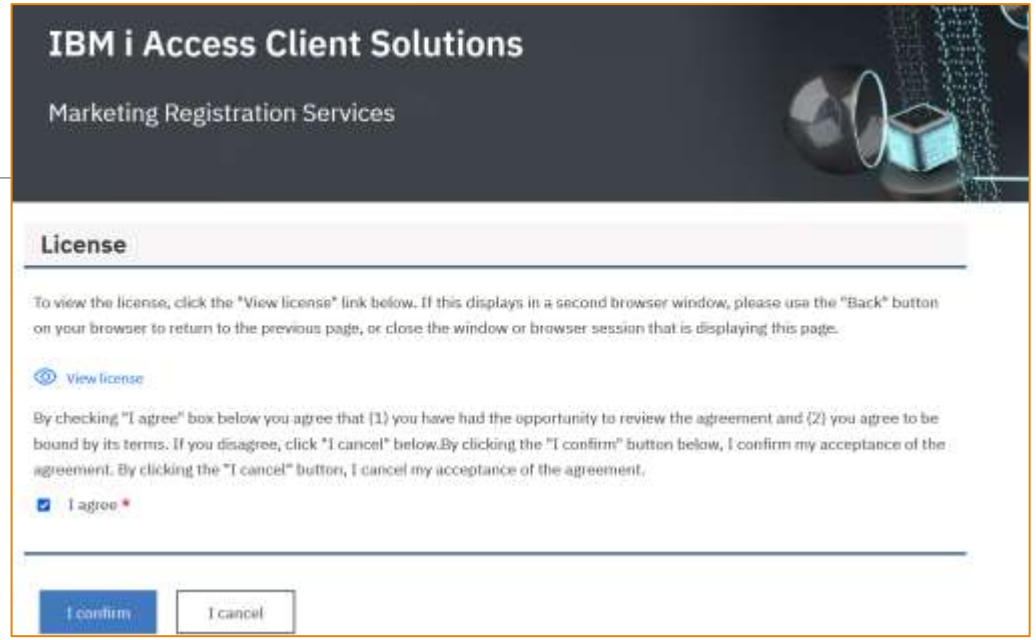
PSM vs Procedures

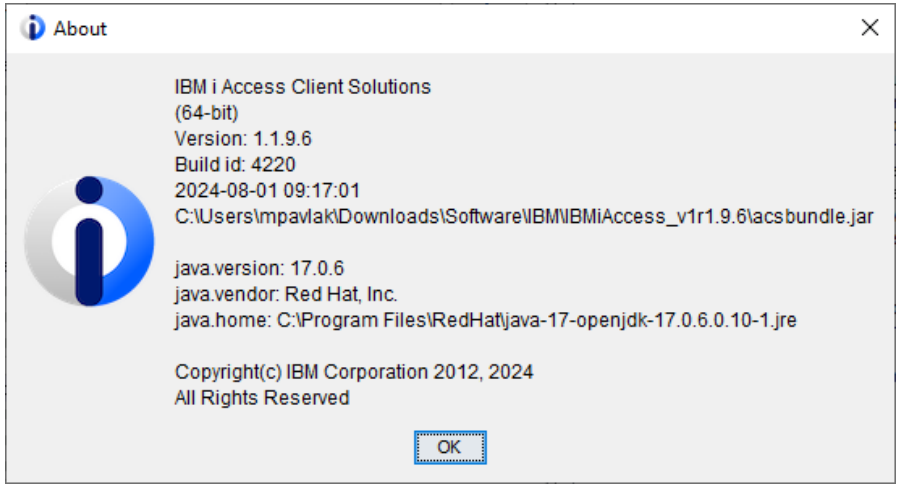
Procedures:

- SQL Stored procedure
- SQL Function
- RPG “Classic”
- RPGLE Program
- RPGLE Subprocedure with parms
- RPGLE Subprocedure with return
- What’s next?

PSA – Update ACS?







What are we talking about (part 1)

Using SQL to ***create*** and ***persist*** business logic.

Write routines that are database artifacts. Not operating system objects.

Db2 is constantly being enhanced (i.e. *every TR ever made has had Db2 enhancements!*)

Three major methods for achieving this in SQL:

- Procedures
- Functions
- Triggers

What are we talking about (part 2)

Two types of procedures:

Persistent Stored Module or PSM

- This is kinda like Stored procedures, but maybe a superset
  - Cross platform
  - ANSI standard
  - May not support ILE artifacts

IBM i Db2 Stored procedures

- Proprietary to IBM i Db2
- ANSI compliant
- IBM Rochester standard (just as good as ANSI, IMHO)
- Will support all IBM i platform artifacts

What are Db2 Stored Procedures?

Database Artifacts with prebuilt routines and/or business logic.

*When DB2 for i executes the CREATE PROCEDURE, CREATE TRIGGER, or CREATE FUNCTION statement for an SQL PSM procedure, trigger, or function, DB2 for i uses a multiple phase process to create an Integrated Language Environment (ILE) C program or service program object (*PGM or *SRVPGM). During this process, DB2 for i generates intermediary ILE C code with embedded SQL statements. This ILE C code is then precompiled, compiled, and linked automatically.*

SQL Procedures, Triggers, and Functions on IBM DB2 for i Redbook

Why use Stored Procedures

Data is the most valuable resource in the data center

Access to the data is HUGE. Easy access is better.

Contemporary options:

- APIs (typically provisioned using HLL)
- Record Level Access (Highly proprietary)
- NoSQL (non-native – hydrated via other mechanisms)
- Procedures, triggers and functions
  - All you need is a database connection
  - Typically used by trusted resources (rarely public facing as APIs)
  - Highly efficient and optimized

"***Data is the new oil***." Clive Humby Sheffield School of Computer Science 2006

PSM = Persistent Stored Module

Ansi Standard

Advantages:

- Common language ANSI SQL
- Portable
- Three primary categories
  - Procedures
  - Triggers
  - Functions

Chances are, your existing Stored procedures are already PSM compatible!

Comments

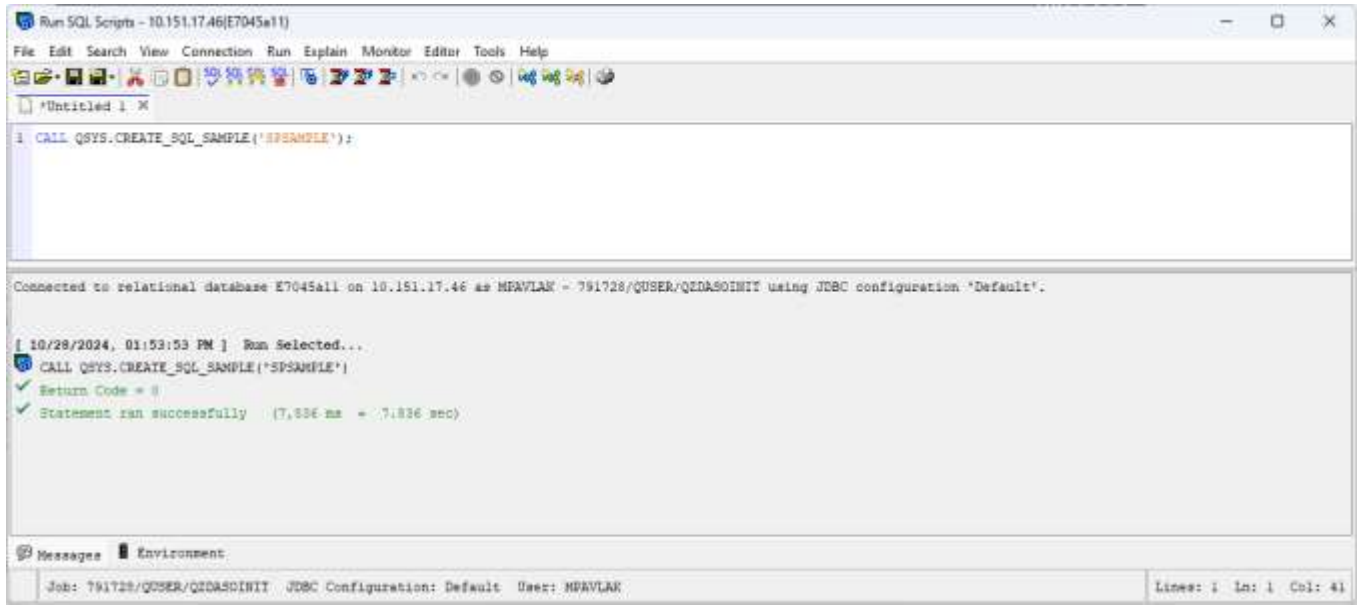
Can be added via

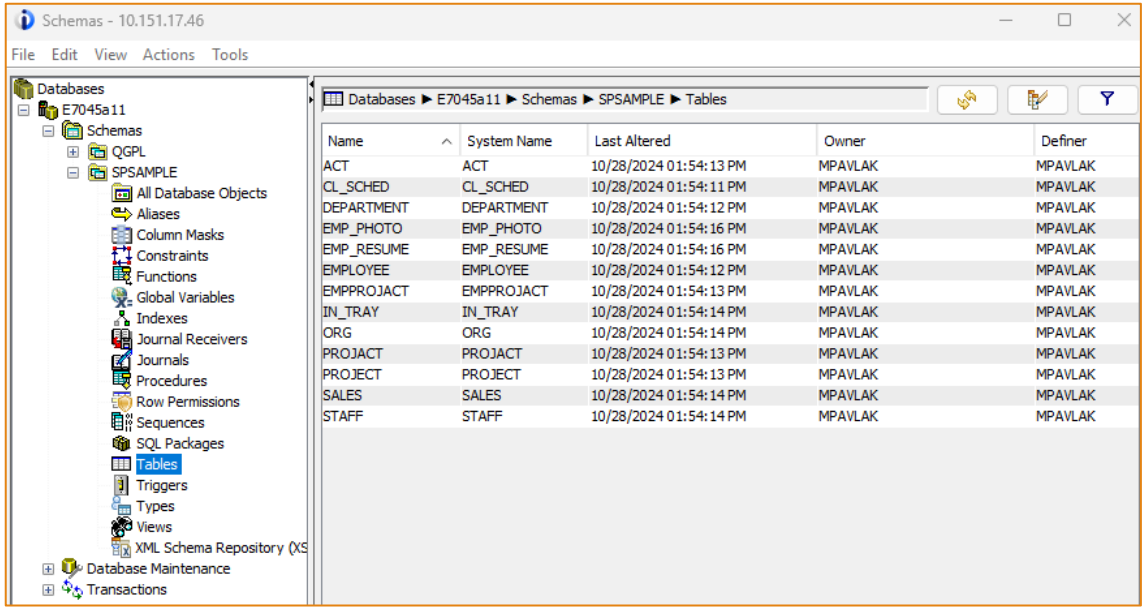
- /* comment goes here*/ Multi-line comment
- -- comment goes here Single line comment

You'll see more of this in a bit

Sample data

Create some sample data for the exhibits





Let’s look at a simple use case

Need routine to show sales quantity and number of orders for department “A00” by employee.

Sales table has SALES_PERSON and employee table has department

```
SELECT SALES_PERSON,
   SUM(sales) AS SalesAmount,
   COUNT(*) AS SalesOrders
FROM spsample.employee
     LEFT OUTER JOIN spsample.sales
         ON SALES_PERSON = LASTNAME
WHERE WORKDEPT = 'A00'
GROUP BY SALES_PERSON;
```

SALES_PERSON	SALESAMOUNT	SALESORDERS
-	-	4
LUCCHESSI	14	9

What’s wrong with this? Why would YOU never push code like this to production?

Big issue?

Hardcoding should be softcoded…

What are the alternatives?

- Write RPG with record level access and parameter
- Write RPG with embedded SQL and parameter
- Write Query Manager Query with parameter
- Write a stored procedure

Consider time investment…

```
SELECT SALES_PERSON,
   SUM(sales) AS SalesAmount,
   COUNT(*) AS SalesOrders
FROM spsample.employee
     LEFT OUTER JOIN spsample.sales
         ON SALES_PERSON = LASTNAME
WHERE WORKDEPT = 'A00'
GROUP BY SALES_PERSON;
```

Why Stored Procedures?

RPG is not the only language in the house anymore

- PHP
- Python
- Node
- C#
- Java

On platform or off?

All can connect to Db2…

Agenda

SQL Stored procedure

SQL Function

RPG “Classic”

RPGLE Program

RPGLE Subprocedure with parms

RPGLE Subprocedure with return

What’s next?

Let’s create a Stored Procedure

We already have the SQL, which is a HUGE start.

Start slow / small and iterate: First, create an SP with no parms.

Leverage the Agile model of development, start small and iterate…

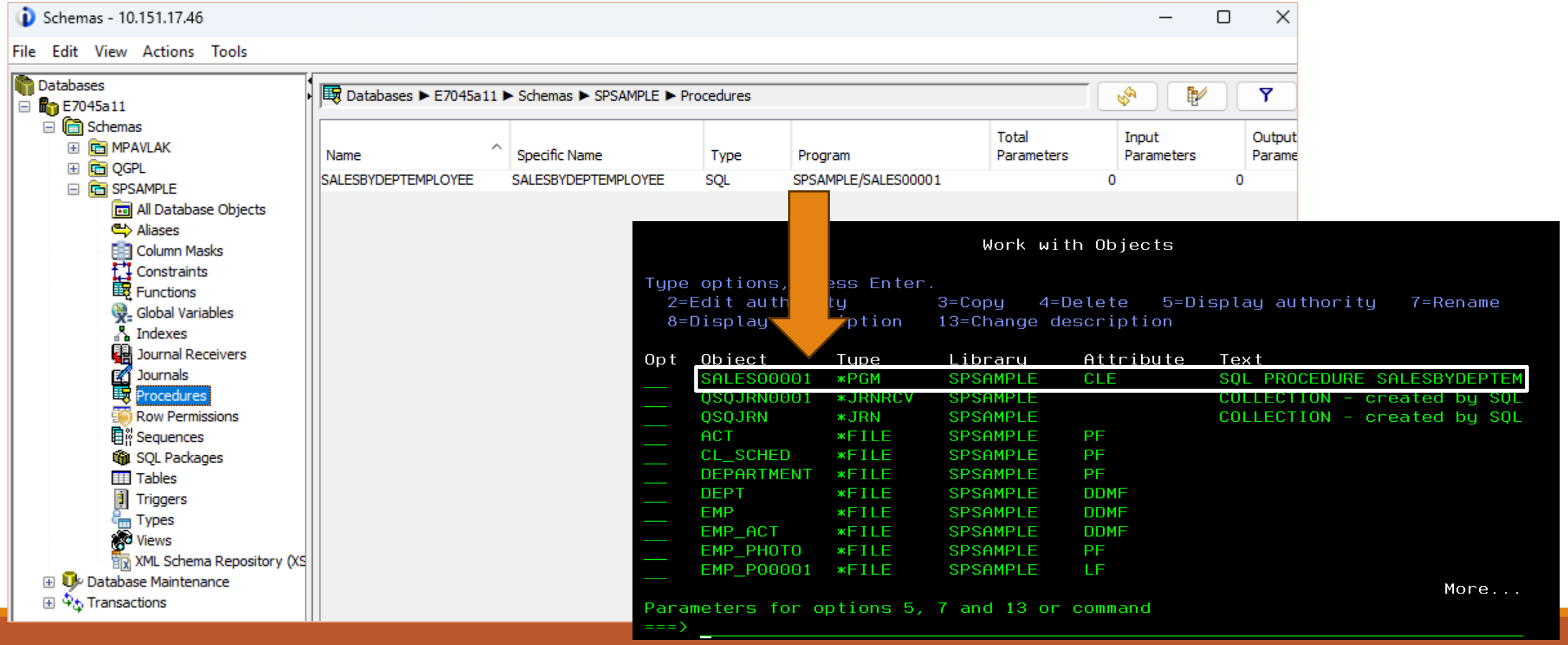
Example – Let's decompose this

```
 1 CREATE OR REPLACE PROCEDURE spsample.salesByDeptEmployee ()
 2         RESULT SETS 1
 3         LANGUAGE sql
 4 P1: BEGIN
 5     DECLARE cur1 CURSOR WITH RETURN FOR
 6         SELECT SUM(sales) AS SalesAmount,
 7                COUNT(*) AS SalesOrders
 8             FROM spsample.employee
 9                  LEFT OUTER JOIN spsample.sales
10                      ON SALES_PERSON = LASTNAME
11             WHERE WORKDEPT = 'A00';
12     OPEN cur1;
13 END;
```

```
14
15 call spsample.salesByDeptEmployee();
16
```

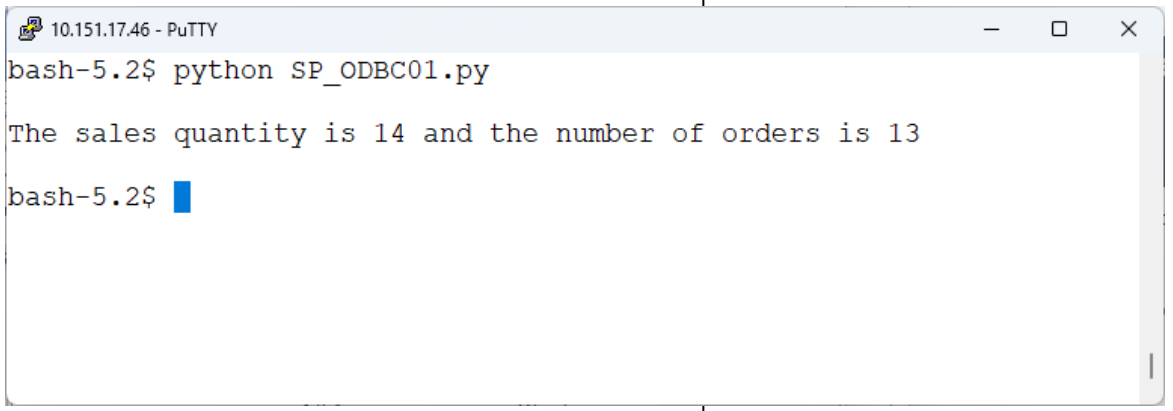
SALESAMOUNT	SALESORDERS
14	13

Under the covers



Python based?

```
# ODBC Stored Procedure Example

import pyodbc

conn = pyodbc.connect('DSN=*LOCAL')

cursor = conn.cursor()

SQL = 'call spsample.salesByDeptEmployee()'

cursor.execute(SQL)

row = cursor.fetchone()

print(f'The sales quantity is {row.SALESAMOUNT} and the number of orders is {row.SALESORDERS}')
```



Agile - Iterate

Now let's add a parameter

```
1  CREATE OR REPLACE PROCEDURE spsample.salesByDeptEmployee2 (
2                  IN DEPT_IN VARCHAR(3)
3              )
4          RESULT SETS 1
5          LANGUAGE sql
6  P1: BEGIN
7      DECLARE cur1 CURSOR WITH RETURN FOR
8          SELECT SUM(sales) AS SalesAmount,
9                 COUNT(*) AS SalesOrders
10             FROM spsample.employee
11                 LEFT OUTER JOIN spsample.sales
12                     ON SALES_PERSON = LASTNAME
13                         --WHERE WORKDEPT = 'A00';
14             WHERE WORKDEPT = DEPT_IN;
15     OPEN cur1;
16 END;
```

```
18 call spsample.salesByDeptEmployee2('A00');
```

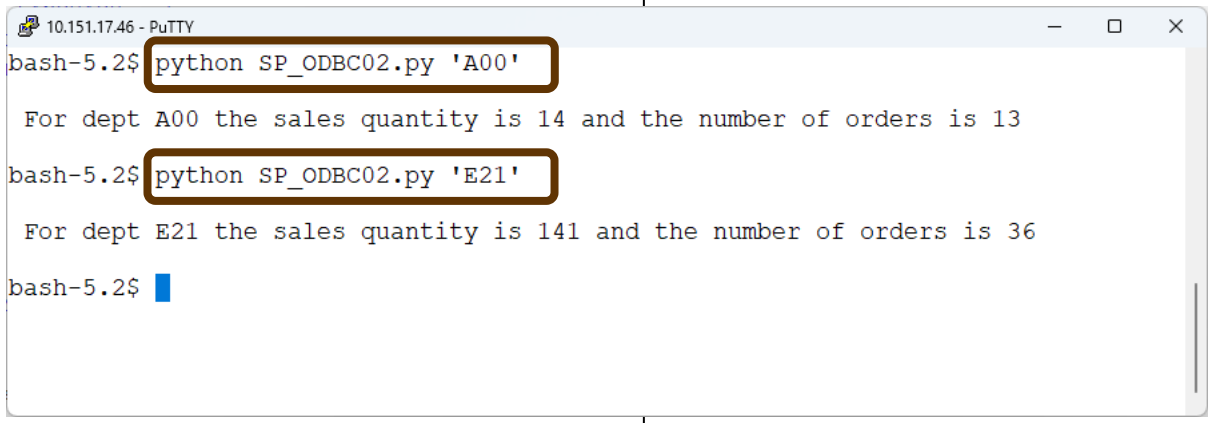
SALESAMOUNT	SALESORDERS
14	13

```
20 call spsample.salesByDeptEmployee2('E21');
```

SALESAMOUNT	SALESORDERS
141	36

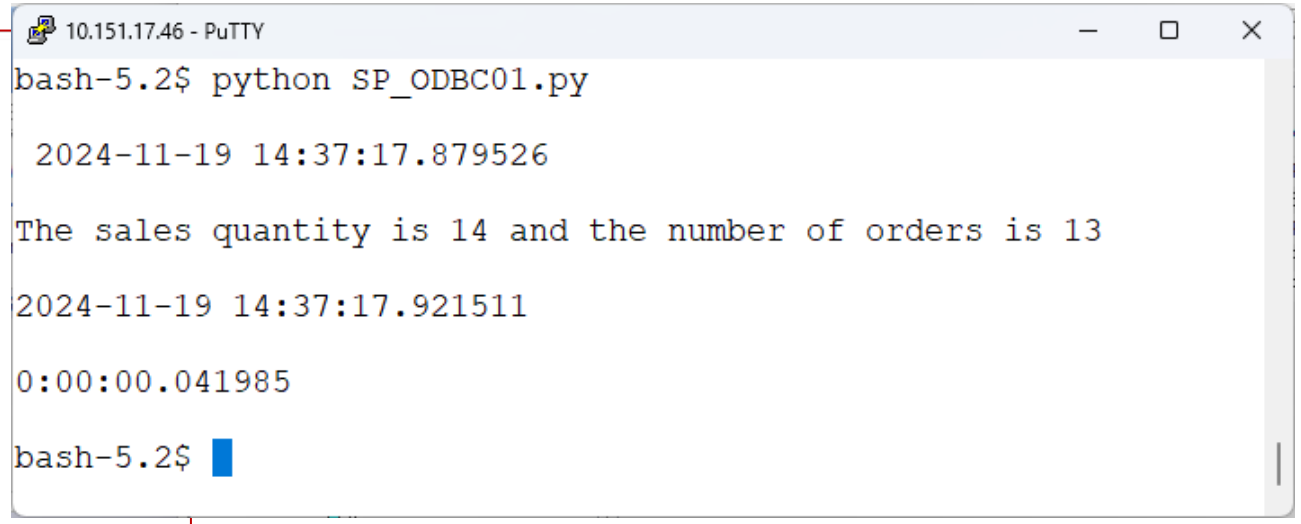
Python again?

```
# ODBC Stored Procedure Example

import pyodbc
import sys


conn = pyodbc.connect('DSN=*LOCAL')
cursor = conn.cursor()

DEPT = sys.argv[1]

SQL = 'call spsample.salesByDeptEmployee2(?)'
cursor.execute(SQL, DEPT)

row = cursor.fetchone()

print(f'\n For dept {DEPT} the sales quantity is {row.SALESAMOUNT} and the number of orders is {row.SALESORDERS}\n')
```



Performance? Add a little code…

```python
# ODBC Stored Procedure Example

import pyodbc, datetime as dt

ts1 = dt.datetime.now()

print('\n', ts1)
conn = pyodbc.connect('DSN=*LOCAL')

cursor = conn.cursor()

SQL = 'call spsample.salesByDeptEmployee()'

cursor.execute(SQL)

row = cursor.fetchone()

print(f'\nThe sales quantity is {row.SALESAMOUNT} and the number of orders is {row.SALESORDERS}\n')

ts2 = dt.datetime.now()
print(ts2, '\n')
diff = ts2-ts1
print(diff, '\n')
```



Agenda

- SQL Stored procedure
- SQL Function
- RPG “Classic”
- RPGLE Program
- RPGLE Subprocedure with parms
- RPGLE Subprocedure with return
- What’s next?

SQL function

We’l save that for a little bit…Stay tuned, it’s coming! I promise…

Agenda

- [x] SQL Stored procedure
- [x] SQL Function
- [ ] RPG “Classic”
- [ ] RPGLE Program
- [ ] RPGLE Subprocedure with parms
- [ ] RPGLE Subprocedure with return
- [ ] What’s next?

Where to from here? RPG?

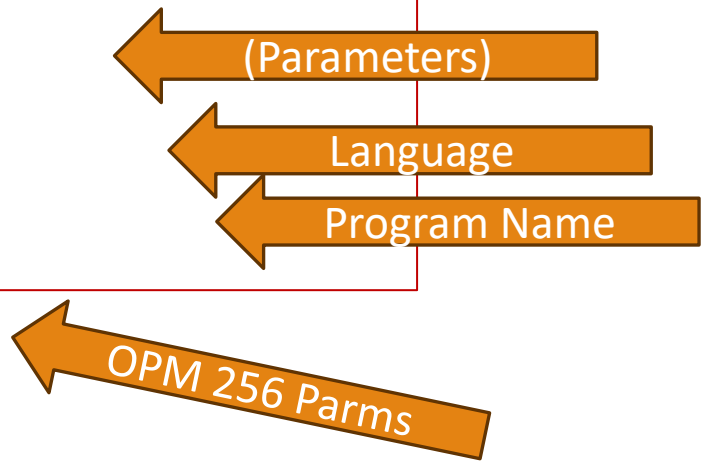
Let’s start with a classic… RPG/400-ish

MPTEST1 > QRPGLESRC > COMMONPGM.RPGLE

```
1     ******************************************************************************
2     C     *ENTRY        PLIST
3     C                   PARM                    CODE           10
4     C                   PARM                    NAME           10
5     ******************************************************************************
6     C     CODE          IFEQ      '1'
7     C                   movel     'IBM'         name
8     C                   ELSE
9     C     CODE          IFEQ      '2'
10    C                   movel     'Zend'        name
11    C                   ELSE
12    C                   movel     'wrong code'  name
13    C                   ENDIF
14    C                   ENDIF
15    C*
16    C                   SETON                                        LR
17    C                   RETURN
18
```

Stored Procedure Call

```
1 /* Let's call a classic RPG program */
2
3 CREATE OR REPLACE PROCEDURE spsample.callRPGzend (
4      INOUT DIGIT1 CHAR(10),
5      INOUT NAME1  CHAR(10))
6      LANGUAGE RPG
7      EXTERNAL NAME ZENDPHP74.COMMONPGM
8      PARAMETER STYLE GENERAL;
```



Stored Procedure Call

```
1 /* Let's call a classic RPG program */
2
3 CREATE OR REPLACE PROCEDURE spsample.callRPGzend (
4     INOUT DIGIT1 CHAR(10),
5     INOUT NAME1  CHAR(10))
6     LANGUAGE RPG
7     EXTERNAL NAME ZENDPHP74.COMMONPGM
8     PARAMETER STYLE GENERAL;
```

```
call spsample.callRPGzend('1         ', '          ');
```

```
call spsample.callRPGzend('1         ', '          ')
✓ Return Code = 0
Output Parameter #1 (DIGIT1) = 1
Output Parameter #2 (NAME1) = IBM
✓ Statement ran successfully   (66 ms)


[ 11/18/2024, 07:44:57 PM ]  Run Selected...
call spsample.callRPGzend('2         ', '          ')
✓ Return Code = 0
Output Parameter #1 (DIGIT1) = 2
Output Parameter #2 (NAME1) = Zend
✓ Statement ran successfully   (72 ms)


[ 11/18/2024, 07:45:00 PM ]  Run Selected...
call spsample.callRPGzend('3         ', '          ')
✓ Return Code = 0
Output Parameter #1 (DIGIT1) = 3
Output Parameter #2 (NAME1) = wrong code
✓ Statement ran successfully   (66 ms)
```

Python again?

PyODBC has not implemented the .callproc method! D'oh!

Other languages (Java, PHP) or connectors like Db2 have…

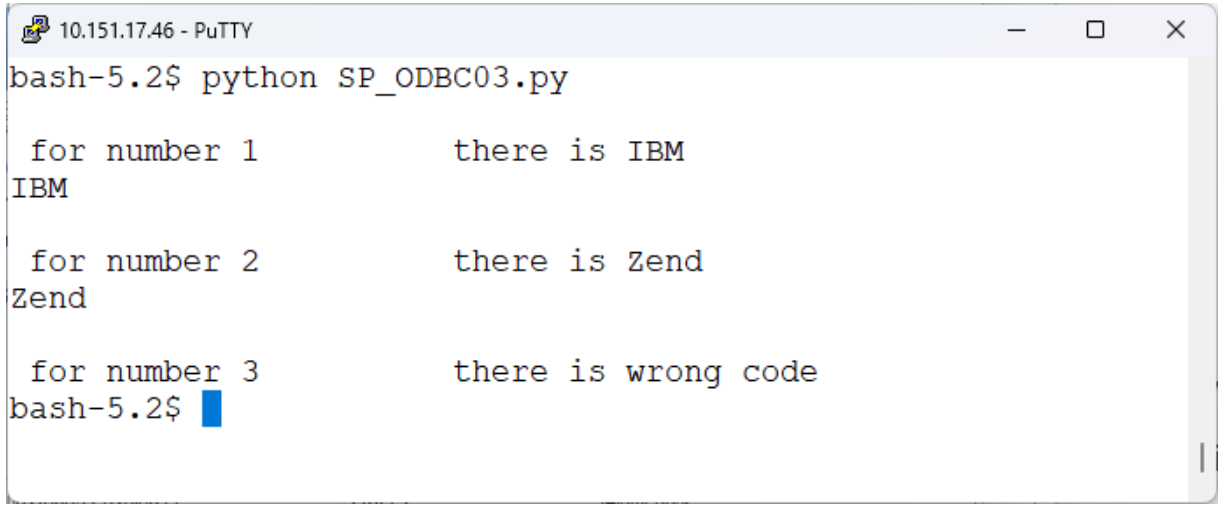
OR we can return the values as an array or result set…

More on that in another presentation…

Generic Python Connector

The file has been modified. Upload?

```python
# ODBC Stored Procedure Example

import ibm_db as dbi

conn = dbi.connect("*LOCAL", "PHPUSER", "phpuser1")

digit = '1         '
name1 = '          '
stmt, digit, name1 = dbi.callproc(conn, 'spsample.callRPGZend', (digit, name1))
print(f"\n for number {digit} there is {name1}")
print(name1)

digit = '2         '
name1 = '          '
stmt, digit, name1 = dbi.callproc(conn, 'spsample.callRPGZend', (digit, name1))
print(f"\n for number {digit} there is {name1}")
print(name1)

digit = '3         '
name1 = '          '
stmt, digit, name1 = dbi.callproc(conn, 'spsample.callRPGZend', (digit, name1))
print(f"\n for number {digit} there is {name1}")
```



Agenda

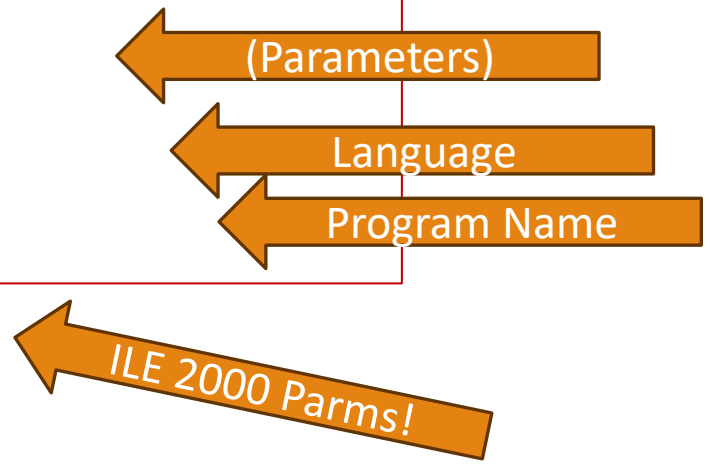
- [x] SQL Stored procedure
- [x] SQL Function
- [x] RPG “Classic”
- [ ] RPGLE Program
- [ ] RPGLE Subprocedure with parms
- [ ] RPGLE Subprocedure with return
- [ ] What’s next?

RPG Free – Super simple

```
1   **free
2   // Program: CONCAT
3   // Description: Concatenate two input character parameters into a third parameter
4
5   DCL-PI *n;
6       Parm1  CHAR(5);
7       Parm2  CHAR(5);
8       Parm3  CHAR(11);
9   END-PI;
10
11  // Initialize the result to blank to avoid trailing garbage characters
12  Parm3 = *BLANKS;
13
14  // Concatenate char1 and char2 into result
15  Parm3 = %TRIM(Parm1) + '-' + %TRIM(Parm2);
16
17  // Program ends, result is returned
18  *INLR = *ON;
```

RPG Free – FTW!

```
1 /* Let's call a free from RPG program */
2
3 CREATE OR REPLACE PROCEDURE spsample.callRPGzend2 (
4     INOUT PARAMIN1 CHAR(5),
5     INOUT PARAMIN2 CHAR(5),
6     INOUT PARAMOUT3  CHAR(11))
7     LANGUAGE RPGLE
8     EXTERNAL NAME MPTEST1.CONCAT
9     PARAMETER STYLE GENERAL;
```



RPG Free Stored Procedure Call

```
1 /* Let's call a free from RPG program */
2
3 CREATE OR REPLACE PROCEDURE spsample.callRPGzend2 (
4     INOUT PARAMIN1 CHAR(5),
5     INOUT PARAMIN2 CHAR(5),
6     INOUT PARAMOUT3  CHAR(11))
7     LANGUAGE RPGLE
8     EXTERNAL NAME MPTEST1.CONCAT
9     PARAMETER STYLE GENERAL;
```

```
call spsample.callRPGzend2('Hi   ', 'Ho   ', '          ');

call spsample.callRPGzend2('Yo   ', 'Yo   ', '          ');

call spsample.callRPGzend2('NoNo ', 'GoGo ', '          ');
```

```
[ 11/19/2024, 09:49:50 AM ]  Run Selected...
call spsample.callRPGzend2('Hi   ', 'Ho   ', '          ')
✓ Return Code = 0
Output Parameter #1 (PARAMIN1) = Hi
Output Parameter #2 (PARAMIN2) = Ho
Output Parameter #3 (PARAMOUT3) = Hi-Ho
✓ Statement ran successfully   (86 ms)


[ 11/19/2024, 09:53:16 AM ]  Run Selected...
call spsample.callRPGzend2('Yo   ', 'Yo   ', '          ')
✓ Return Code = 0
Output Parameter #1 (PARAMIN1) = Yo
Output Parameter #2 (PARAMIN2) = Yo
Output Parameter #3 (PARAMOUT3) = Yo-Yo
✓ Statement ran successfully   (79 ms)


[ 11/19/2024, 09:53:19 AM ]  Run Selected...
call spsample.callRPGzend2('NoNo ', 'GoGo ', '          ')
✓ Return Code = 0
Output Parameter #1 (PARAMIN1) = NoNo
Output Parameter #2 (PARAMIN2) = GoGo
Output Parameter #3 (PARAMOUT3) = NoNo-GoGo
✓ Statement ran successfully   (78 ms)
```

Agenda

- [x] SQL Stored procedure
- [x] SQL Function
- [x] RPG “Classic”
- [x] RPGLE Program
- [ ] RPGLE Subprocedure with parms
- [ ] RPGLE Subprocedure with return
- [ ] What’s next?

But I got subprocedures!

Special thank you to the great ***Charlie Guarino*** for the sample subprocedure programs…

```
ctl-opt NOMAIN;
Dcl-Pr RemoveBlanks;
  Charin          Char(24);
  Charout         Char(24);
End-Pr;


Dcl-Pr ReverseStringFunction  Char(24);
  Charin          Char(24)  const;
End-Pr;
```

```
 dcl-proc ReverseStringFunction export;

  Dcl-Pi ReverseStringFunction  char(24);
    charin          Char(24)    const;
  End-Pi;

dcl-s backctr  zoned(2);
dcl-s frontctr zoned(2);
dcl-s charout  char(24);

backctr = %len(%trim(charin));

if backctr = *zero;
  return charout;
endif;

charout = *blanks;
frontctr = 1;

dou  backctr = *zero;
  %subst(charout:frontctr:1) = %subst(charin:backctr:1);
  frontctr +=1;
  backctr -=1;
enddo;

 // When here field charout is the reverse of charin

  return charout;

end-proc ReverseStringFunction;
```

```
 dcl-proc RemoveBlanks  export;

//  Procedure interface
 Dcl-Pi RemoveBlanks;
   charin          Char(24);
   charout         Char(24);
 End-Pi;

dcl-s counter   zoned(2);
dcl-s index     zoned(2);
dcl-s stringlen zoned(2);

 stringlen = %len(%trim(charin));

if stringlen = *zero;
  return;
endif;

charout = *blanks;
index = *zero;

for counter = 1 to stringlen;

// Strip off blank characters
    if %subst(charin:counter:1) = *blank;
      iter;
    endif;

// When here a non-blank character has been encountered, add to charout
  index +=1;
  %subst(charout:index:1) = %subst(charin:counter:1);

endfor;


 // When here field charout contains all non-blank characters of charin

 end-proc RemoveBlanks;
```

Stored Procedure to sub procedure

Not too different…

```
1  /* Let's call a subprocedure...   */
2
3  CREATE OR REPLACE PROCEDURE spsample.callRPGsubp1 (
4      IN STRINGIN     CHAR(24),
5      OUT STRINGOUT   CHAR(24))
6      LANGUAGE RPGLE
7      DETERMINISTIC
8      READS SQL DATA
9      RETURNS NULL ON NULL INPUT
10     NOT FENCED
11     EXTERNAL NAME MPTEST1.STRINGSVC(RemoveBlanks)
12     PARAMETER STYLE GENERAL;
```

Calls to the subprocedure

```
1  /* Let's call a subprocedure...    */
2
3  CREATE OR REPLACE PROCEDURE spsample.callRPGsubp1 (
4      IN STRINGIN      CHAR(24),
5      OUT STRINGOUT    CHAR(24))
6      LANGUAGE RPGLE
7      DETERMINISTIC
8      READS SQL DATA
9      RETURNS NULL ON NULL INPUT
10     NOT FENCED
11     EXTERNAL NAME MPTEST1.STRINGSVC(RemoveBlanks)
12     PARAMETER STYLE GENERAL;
```

```
call spsample.callRPGsubp1('This is a simple phrase ', '                    ');

call spsample.callRPGsubp1('The quick brown fox     ', '                    ');

call spsample.callRPGsubp1('H O T T O G O           ', '                    ');
```

```
[ 11/19/2024, 11:21:11 AM ]  Run Selected...
call spsample.callRPGsubp1('This is a simple phrase ', '                    ')
✓ Return Code = 0
Output Parameter #2 (STRINGOUT) = Thisisasimplephrase
✓ Statement ran successfully   (82 ms)


[ 11/19/2024, 11:21:15 AM ]  Run Selected...
call spsample.callRPGsubp1('The quick brown fox     ', '                    ')
✓ Return Code = 0
Output Parameter #2 (STRINGOUT) = Thequickbrownfox
✓ Statement ran successfully   (71 ms)


[ 11/19/2024, 11:21:16 AM ]  Run Selected...
call spsample.callRPGsubp1('H O T T O G O           ', '                    ')
✓ Return Code = 0
Output Parameter #2 (STRINGOUT) = HOTTOGO
✓ Statement ran successfully   (90 ms)
```

Agenda

- [x] SQL Stored procedure
- [x] SQL Function
- [x] RPG “Classic”
- [x] RPGLE Program
- [x] RPGLE Subprocedure with parms
- [ ] RPGLE Subprocedure with return
- [ ] What’s next?

But I got subprocedures with returns???

Special thank you to the great ***Charlie Guarino*** for the sample subprocedure programs…

```
1   ctl-opt NOMAIN;
2   Dcl-Pr RemoveBlanks;
3     Charin          Char(24);
4     Charout         Char(24);
5   End-Pr;
6
7
8   Dcl-Pr ReverseStringFunction  Char(24);
9     Charin          Char(24)  const;
10  End-Pr;
```

```
69      dcl-proc ReverseStringFunction export;
70
71      Dcl-Pi ReverseStringFunction  char(24);
72        charin          Char(24)    const;
73      End-Pi;
74
75    dcl-s backctr  zoned(2);
76    dcl-s frontctr zoned(2);
77    dcl-s charout  char(24);
78
79    backctr = %len(%trim(charin));
80
81    if backctr = *zero;
82      return charout;
83    endif;
84
85    charout = *blanks;
86    frontctr = 1;
87
88    dou  backctr = *zero;
89      %subst(charout:frontctr:1) = %subst(charin:backctr:1);
90      frontctr +=1;
91      backctr -=1;
92    enddo;
93
94     // When here field charout is the reverse of charin
95
96      return charout;
97
98    end-proc ReverseStringFunction;
```

```
28      dcl-proc RemoveBlanks  export;
29
30    //  Procedure interface
31      Dcl-Pi RemoveBlanks;
32        charin          Char(24);
33        charout         Char(24);
34      End-Pi;
35
36    dcl-s counter   zoned(2);
37    dcl-s index     zoned(2);
38    dcl-s stringlen zoned(2);
39
40     stringlen = %len(%trim(charin));
41
42    if stringlen = *zero;
43      return;
44    endif;
45
46    charout = *blanks;
47    index = *zero;
48
49    for counter = 1 to stringlen;
50
51    // Strip off blank characters
52        if %subst(charin:counter:1) = *blank;
53          iter;
54        endif;
55
56    // When here a non-blank character has been encountered, add to charout
57      index +=1;
58      %subst(charout:index:1) = %subst(charin:counter:1);
59
60    endfor;
61
62
63     // When here field charout contains all non-blank characters of charin
64
65     end-proc RemoveBlanks;
```

Not really a SP…*function maybe*?

```
1  /* Let's call a subprocedure...    */
2
3  CREATE OR REPLACE FUNCTION spsample.callRPGsubp2 (
4      STRINGIN     CHAR(24),
5      STRINGOUT    CHAR(24))
6      RETURNS CHAR(24)
7      LANGUAGE RPGLE
8      DETERMINISTIC
9      RETURNS NULL ON NULL INPUT
10     NOT FENCED
11     EXTERNAL NAME MPTEST1.STRINGSVC(ReverseStringFunction)
12     PARAMETER STYLE GENERAL;
```

And the results…

```
1  /* Let's call a subprocedure...   */
2
3  CREATE OR REPLACE FUNCTION spsample.callRPGsubp2 (
4      STRINGIN    CHAR(24),
5      STRINGOUT   CHAR(24))
6      RETURNS CHAR(24)
7      LANGUAGE RPGLE
8      DETERMINISTIC
9      RETURNS NULL ON NULL INPUT
10     NOT FENCED
11     EXTERNAL NAME MPTEST1.STRINGSVC(ReverseStringFunction)
12     PARAMETER STYLE GENERAL;
```

00001
dlroW olleH

00001
R A C E C A R

00001
desserts stink

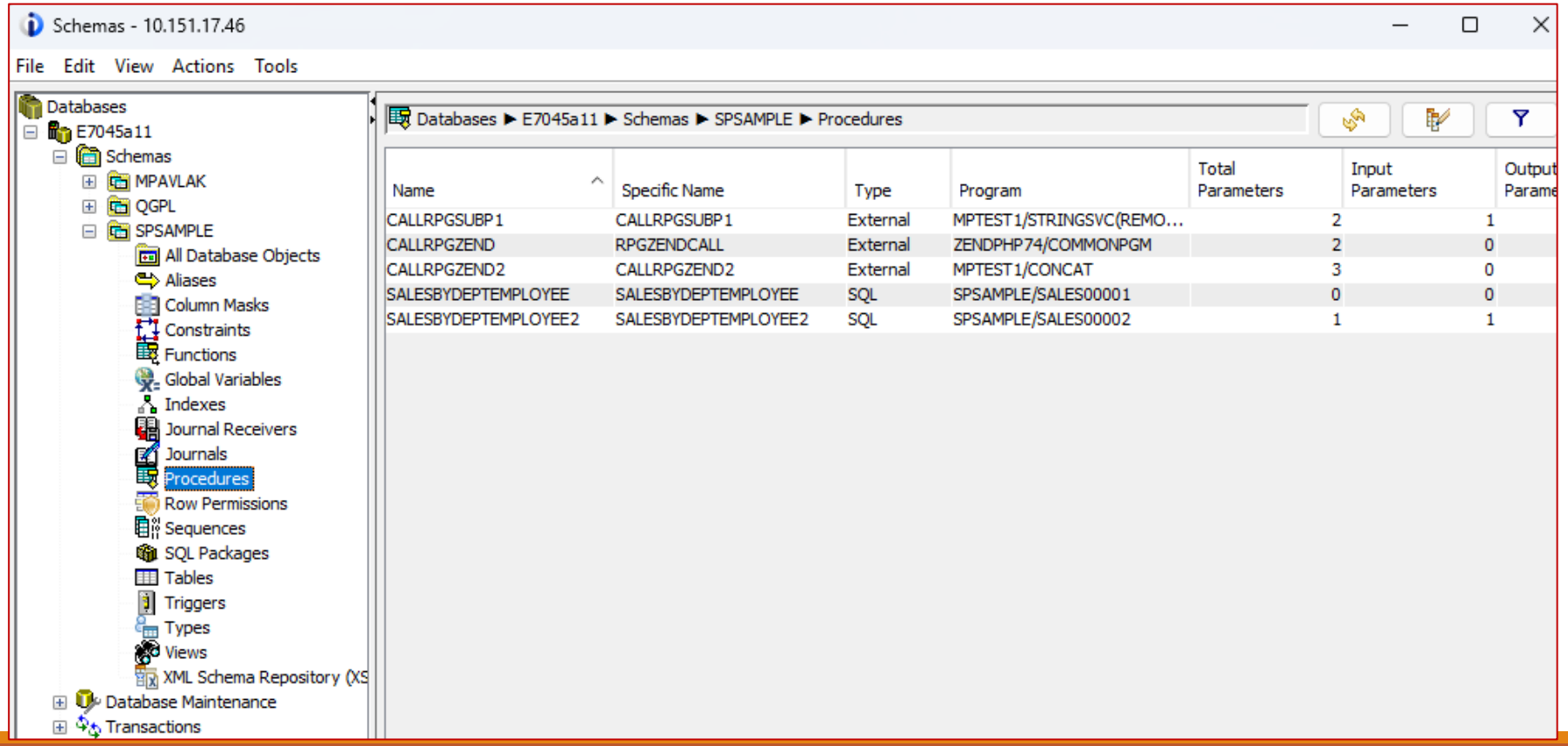
```
Select spsample.callRPGsubp2('Hello World          ', '                    ') from sysibm.sysdummy1;

Select spsample.callRPGsubp2('R A C E C A R        ', '                    ') from sysibm.sysdummy1;

Select spsample.callRPGsubp2('knits stressed       ', '                    ') from sysibm.sysdummy1;
```

Agenda

- [x] SQL Stored procedure
- [x] SQL Function
- [x] RPG “Classic”
- [x] RPGLE Program
- [x] RPGLE Subprocedure with parms
- [x] RPGLE Subprocedure with return
- [ ] What’s next?

Where do we go from here?

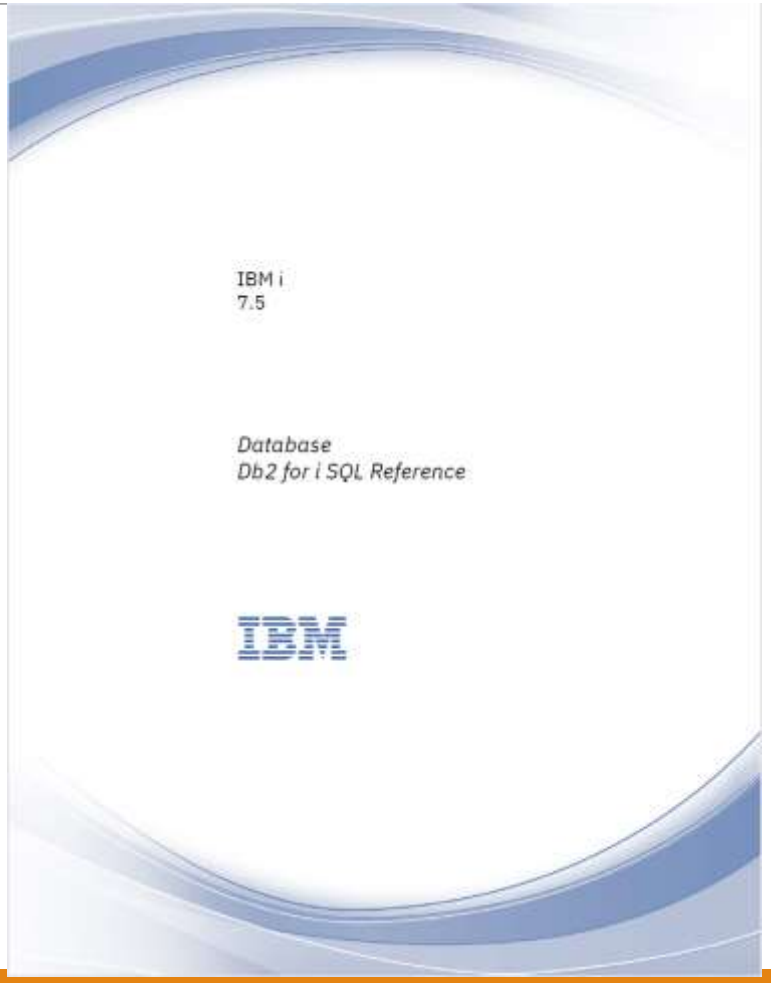
Inventory of SPs?



Db2 manual

2,154 pages of pure Db2 for i goodness!

Too much to pack into a single session, or even a day!

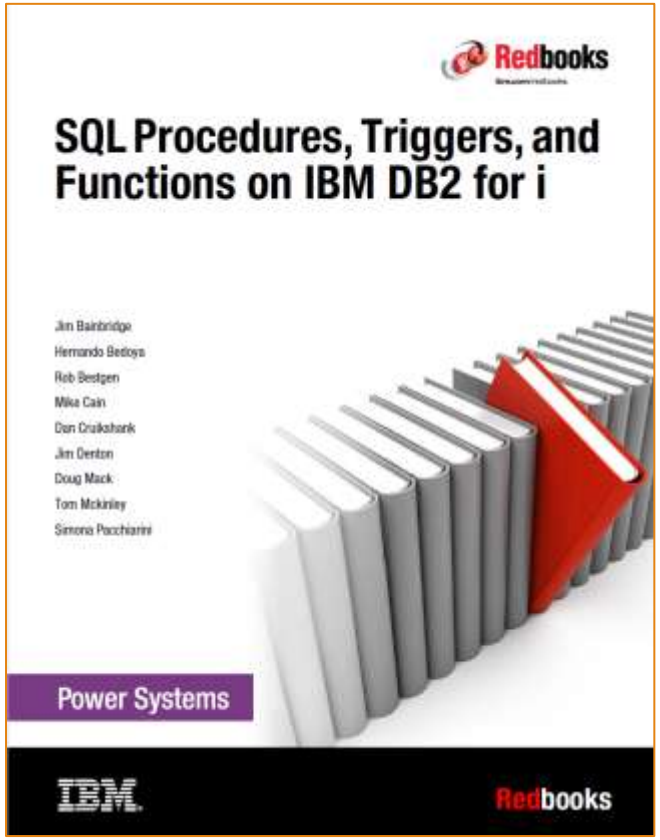


Do you remember Redbooks?

I borrowed some from here SG24-8326-00

Look at the authors…all retired ☹

But the knowledge was captured and is still quite relevant



More cool things you can do

Stored procedure future discussions

- Multiple result sets.
- Threadsafe applications
- More integration types

Stored Procedures for Hybrid Development

Questions?

MikePavlak@gmail.com

@MikeyPEI