

SQL Stored Procedures and the SQL Procedure Language

John Valance

Senior Consultant - Modernization and
Digital Transformation Strategist

Profound Logic Software



jvalance@profoundlogic.com

About John Valance



- **30+ years IBM midrange experience (S/38 thru IBM i)**

- ▶ 19 years of web development experience

- **Profound Logic employee since Nov 2018**



- ▶ Services Division: Senior Consultant -
Modernization and Digital Transformation Strategist

- **Independent consultant 2000 - 2018**



- ▶ Founder and CTO of Division 1 Systems
- ▶ Specializing in Modernization and Web Enablement of IBM i

- **Community Involvement**

- ▶ COMMON Board of Directors - since May 2017
- ▶ Frequent presenter at IBM i groups nationwide
- ▶ Taught PHP for Zend Technologies for 4 years



Stored Procedures: What, Why, Who?



What We Will Cover

- Creating Stored Procedures
- SQL Procedure Language
 - ▶ Language Syntax and Capabilities
- Creating User Defined Functions (UDF)
- IBM i Considerations

Not covering:

- External Stored Procedures (RPG, CLP, etc.)
- Triggers

What Are Stored Procedures?

- **Any program object on IBM i**
 - ▶ Known to DB2 via CREATE PROCEDURE statement
- **2 types:**
 - ▶ SQL (written in SQL/PL)
 - ▶ External (RPG, CL, any language)
 - ▶ We will focus on SQL stored procedures
- **Can be called from any environment that supports SQL**
- **Can have parameters for input / output**
- **Can return result sets**
- **Can be selected from the database repository**
 - ▶ `SELECT * FROM QSYS2/SYSPROCS WHERE ROUTINE_SCHEMA = 'MYLIBR'`

What is SQL/PL?

- **SQL Procedure Language**
- **Allows SQL scripts to be built**
 - ▶ Any SQL statements, plus variables, conditions, loops, etc.
 - ▶ Data-centric programming
- **DB2 SQL/PL is proprietary**
 - ▶ but all major DB vendors have proprietary PL
- **Compiled using an SQL client (ACS recommended)**
- **Generates an ILE/C language program, with embedded SQL calls**

Simple example - sp_Cust

```
create or replace procedure jvalance.sp_cust ( )
language sql
result sets 1
begin
    -- declare the cursor for the select statement
    declare c1 cursor with return for
        select trim(STATE) as STATE,
               ZIP,
               CUST_id,
               COMPANY,
               trim(LASTNAME) || ', ' || trim(FIRSTNAME) AS NAME
        from zendsvr6.sp_cust
        where COUNTRY = 'US'
        order by STATE, ZIP;

    -- open the cursor to return results to the caller
    open c1;

end;
```

Running sp_Cust from ACS

- ACS = Access Client Solutions

- ▶ Formerly known as Client Access
- ▶ Has an SQL client which is perfect for developing stored procedures on IBM i

```
755 -- Testing:
756 call jvalance.sp_cust();
757
```

STATE	ZIP	CUST_ID	COMPANY	NAME
AL	30696	3042	Gold Coast Supply	Falls, Elaine
AL	32145	2984	Professional Divers, Ltd.	Mathers, Shirley
AL	32145	3041	Divers of Blue-green	Bean, Nancy
CA	90410	3984	Blue Glass Happiness	Taylor, Christine
CA	90740	3054	Catamaran Dive Club	Dupont, Nicole
CA	91770	3053	American SCUBA Supply	Cinciripini, Lynn
CA	92195	3052	Underwater Sports Co.	Walling, Dave
CA	95443	3051	San Pablo Dive Center	O'Brien, Patricia
FL	30643	6312	Aquatic Drama	Owen, Gillian
FL	32274	1645	Action Club	Stirling, Michael

What Can I Build with SQL/PL?

- **Stored procedures**
- **User Defined Functions (UDF)**
- **Triggers - before & after (add/change/delete)**

Redbooks / References

New Redbook - April 2016!!

- SQL Procedures, Triggers, and Functions on IBM DB2 for i
 - ▶ <http://www.redbooks.ibm.com/abstracts/sg248326.html>
 - ▶ Download the PDF!!
- Also:
 - ▶ DB2 for i SQL reference
https://www.ibm.com/support/knowledgecenter/ssw_ibm_i_72/db2/rbafzpdf.pdf
 - ▶ SQL Programming Guide:
https://www.ibm.com/support/knowledgecenter/ssw_ibm_i_73/sqlp/rbafypdf.pdf

Building Procedures with SQL/PL



Anatomy of a Stored Procedure

```
create or replace procedure jvalance.sp_cust ( )
language sql
result sets 1
begin
    -- declare the cursor for the select statement
    declare c1 cursor with return for
        select trim(STATE) as STATE,
               ZIP,
               CUST_id,
               COMPANY,
               trim(LASTNAME) || ', ' || trim(FIRSTNAME) AS NAME
        from zendsvr6.sp_cust
        where COUNTRY = 'US'
        order by STATE, ZIP;

    -- open the cursor to return results to the caller
    open c1;

end;
```

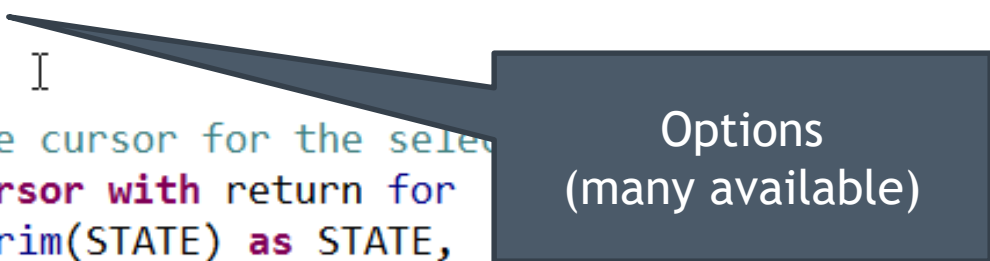
Anatomy of a Stored Procedure

```
create or replace procedure jvalance.sp_cust ( )  
language sql  
result sets 1  
begin  
    -- declare the cursor for the select statement  
    declare c1 cursor with return for  
        select trim(STATE) as STATE,  
            ZIP,  
            CUST_id,  
            COMPANY,  
            trim(LASTNAME) || ', ' || trim(FIRSTNAME) AS NAME  
        from zendsvr6.sp_cust  
        where COUNTRY = 'US'  
        order by STATE, ZIP;  
  
    -- open the cursor to return results to the caller  
    open c1;  
  
end;
```

Create procedure
statement

Anatomy of a Stored Procedure

```
create or replace procedure jvalance.sp_cust ( )  
language sql  
result sets 1  
begin  
    -- declare the cursor for the select  
    declare c1 cursor with return for  
        select trim(STATE) as STATE,  
            ZIP,  
            CUST_id,  
            COMPANY,  
            trim(LASTNAME) || ', ' || trim(FIRSTNAME) AS NAME  
        from zendsvr6.sp_cust  
        where COUNTRY = 'US'  
        order by STATE, ZIP;  
  
    -- open the cursor to return results to the caller  
    open c1;  
  
end;
```



Options
(many available)

Anatomy of a Stored Procedure

```
create or replace procedure jvalance.sp_cust ( )
language sql
result sets 1
begin
    -- declare the cursor for the select statement
    declare c1 cursor with return for
        select trim(STATE) as STATE,
               ZIP,
               CUST_id,
               COMPANY,
               trim(LASTNAME) || ', ' || trim(FIRSTNAME) AS NAME
        from zendsvr6.sp_cust
        where COUNTRY = 'US'
        order by STATE, ZIP;

    -- open the cursor to return results to the caller
    open c1;

end;
```

Procedure Body

Anatomy of a Stored Procedure

```
create or replace procedure jvalance.sp_cust ( )  
language sql  
result sets 1  
begin I  
  -- declare the cursor for the select statement  
  declare c1 cursor with return for  
    select trim(STATE) as STATE,  
           ZIP,  
           CUST_id,  
           COMP  
    trim  
  from zendsvr  
  where COUNTRY  
  order by STATE, ZIP;  
  
  -- open the cursor to return results to the caller  
  open c1;  
  
end;
```

Body is a compound SQL statement enclosed in begin/end block

Anatomy of a Stored Procedure

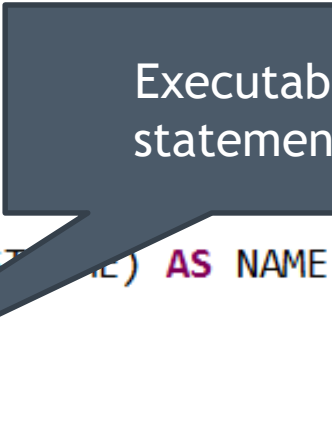
```
create or replace procedure jvalance.sp_cust ( )  
language sql  
result sets 1  
begin  
    -- declare the cursor for the select statement  
    declare c1 cursor with return for  
        select trim(STATE) as STATE,  
               ZIP,  
               CUST_id,  
               COMPANY,  
               trim(LASTNAME) || ', ' || trim(FIRSTNAME) AS NAME  
        from zendsvr6.sp_cust  
        where COUNTRY = 'US'  
        order by STATE, ZIP;  
  
    -- open the cursor to return results to the caller  
    open c1;  
  
end;
```

Declarations

Anatomy of a Stored Procedure

```
create or replace procedure jvalance.sp_cust ( )
language sql
result sets 1
begin
    -- declare the cursor for the select statement
    declare c1 cursor with return for
        select trim(STATE) as STATE,
               ZIP,
               CUST_id,
               COMPANY,
               trim(LASTNAME) || ', ' || trim(FIRSTNAME) AS NAME
        from zendsvr6.sp_cust
        where COUNTRY = 'US'
        order by STATE, ZIP;

    -- open the cursor to return results to the caller
    open c1;
end;
```



Executable statements

SQL = Declarative Programming

- **Procedural programming (RPG)**
 - ▶ Specify **how to get** the data
- **Declarative programming (SQL)**
 - ▶ Specify **what you want** from the database
 - ▶ Database will figure out the most efficient way to execute
- **Always select from PF / Table, not LF or View**
- **Database will create an access plan at run time**
- **Create indices (LFs) to improve performance**
 - ▶ SQL Performance Center in ACS
 - ▶ Help with DB tuning based on runtime analysis

Compiling and Running

Need to use an SQL Client to run the CREATE PROCEDURE

- SQL client choices

- ▶ ACS = Access Client Solutions (** best choice **)



- ▶ Green screen STRSQL (awkward)



\$\$

- ▶ Eclipse Data Tools plug-in for RDi, Zend Studio, etc. (pretty good)

- ▶ Other SQL Clients - JDBC, ODBC (ex.: <http://www.sql-workbench.net/>)

- Biggest issue is handling output parameters

- ▶ ACS or Client Access handles this well

- ▶ ACS / CA also gives best diagnostic messages

- ACS is FREE!!

- ▶ <https://www-03.ibm.com/systems/power/software/i/access/solutions.html>

Where To Put Source Code?

Store your source code in an .sql file

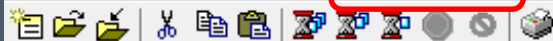
- On your PC
 - ▶ With ACS installed, double click to open and run
- On IBM i IFS
 - ▶ Use an IDE like Eclipse, RDi, Zend Studio
 - Includes SQL syntax highlighting
 - May need to install Data Tools Platform SQL Dev Tools (help menu... Install New Software)
 - ▶ Can open IFS file in ACS
 - Right click... Open With... System Editor
- SRCPF ? (maybe, but not for me)
 - ▶ SEU? (really??)
 - ▶ RUNSQLSTM or STRSQL (hmmm...)

Name	Version
AnyEditTools	2.4.14.20150405
Composer	1.0.2.201412171
▼ Data Tools Platform SQL Development Tools	1.12.0.v2014060
> Data Tools Platform Connectivity	1.12.0.v2014060
> Data Tools Platform Enablement for JDBC	1.12.0.v2014060
> Data Tools Platform Model Base	1.12.0.v2014060
> Data Tools Platform SQL Development Tools Data Functions	1.12.0.v2014060
> Data Tools Platform SQL Development Tools DDL Functionality	1.12.0.v2014060
> Data Tools Platform SQL Development Tools DDL Generation Fi	1.12.0.v2014060
> Data Tools Platform SQL Development Tools Results View	1.12.0.v2014060
> Data Tools Platform SQL Parsers	1.12.0.v2014060
> Data Tools Platform SQL Query Builder	1.12.0.v2014060

Run SQL Scripts in ACS

C:\Users\John Valance\Google Drive\business\PolarBeverage\PolarLink\PM & prog\CreateStoredProcs.sql* - Run SQL Scripts - 172.25.0.1(COGNOSERV)

File Edit View Run Options Connection



Connect to Database

```
739 create or replace procedure jvalance.  
740 language sql  
741 result sets 1  
742 begin  
743     -- declare the cursor for the select statement  
744     declare c1 cursor with return for  
745         select trim(STATE) as STATE, ZIP, CUST_id, COMPANY, trim(LASTNAME)  
746         from zendsvr6.sp_cust  
747         where COUNTRY = 'US'  
748         order by STATE, ZIP;  
749  
750     -- open the cursor to return results to the caller  
751     open c1;  
752  
753 end  
754 ;  
755 -- Testing:  
756 call jvalance.sp_cust();
```

STATE	ZIP	CUST_ID	COMPANY	NAME
AL	30696	3042	Gold Coast Supply	Falls, Elaine
AL	32145	2984	Professional Divers, Ltd.	Mathers, Shirley
AL	32145	3041	Divers of Blue-green	Bean, Nancy
CA	90410	3984	Blue Glass Happiness	Taylor, Christine
CA	90740	3054	Catamaran Dive Club	Dupont, Nicole

<div1>

Run SQL Scripts in ACS

C:\Users\John Valance\Google Drive\business\PolarBeverage\PolarLink\PM & prog\CreateStoredProcs.sql* - Run SQL Scripts - 172.25.0.1(COGNOSERV)

File Edit View **Run** Options Connection



```
739 create or replace procedure jvalance.sp_cust
740 language sql
741 result sets 1
742 begin
743     -- declare the cursor for the select statement
744     declare c1 cursor with return for
745         select trim(STATE) as STATE, ZIP, CUST_id, COMPANY, trim(LASTNAME) as NAME
746         from zendsvr6.sp_cust
747         where COUNTRY = 'US'
748         order by STATE, ZIP;
749
750     -- open the cursor to return results to the caller
751     open c1;
752
753 end
754 ;
755 -- Testing:
756 call jvalance.sp_cust();
```

Position Cursor on Statement to Run
Then...

- Use Run Menu, or...
- Click Run Icons , or...
- Press Ctrl+R

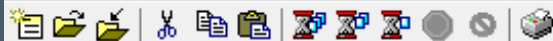
STATE	ZIP	CUST_ID	COMPANY	NAME
AL	30696	3042	Gold Coast Supply	Falls, Elaine
AL	32145	2984	Professional Divers, Ltd.	Mathers, Shirley
AL	32145	3041	Divers of Blue-green	Bean, Nancy
CA	90410	3984	Blue Glass Happiness	Taylor, Christine
CA	90740	3054	Catamaran Dive Club	Dupont, Nicole

<div1>

Run SQL Scripts in ACS

C:\Users\John Valance\Google Drive\business\PolarBeverage\PolarLink\PM & prog\CreateStoredProcs.sql* - Run SQL Scripts - 172.25.0.1(COGNOSERV)

File Edit View Run Options Connection



```
739 create or replace procedure jvalance.sp_cust ( )
740 language sql
741 result sets 1
742 begin
743     -- declare the cursor for the select statement
744     declare c1 cursor with return for
745         select trim(STATE) as STATE, ZIP, CUST_id, COMPANY, trim(LASTNAME)
746         from zendsvr6.sp_cust
747         where COUNTRY = 'US'
748         order by STATE, ZIP;
749
750     -- open the cursor to return results to the caller
751     open c1;
752
753 end
754 ;
755 -- Testing:
756 call jvalance.sp_cust();
```

Run the CREATE

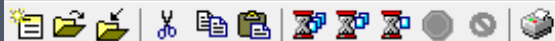
STATE	ZIP	CUST_ID	COMPANY	NAME
AL	30696	3042	Gold Coast Supply	Falls, Elaine
AL	32145	2984	Professional Divers, Ltd.	Mathers, Shirley
AL	32145	3041	Divers of Blue-green	Bean, Nancy
CA	90410	3984	Blue Glass Happiness	Taylor, Christine
CA	90740	3054	Catamaran Dive Club	Dupont, Nicole

<div1>

Run SQL Scripts in ACS

C:\Users\John Valance\Google Drive\business\PolarBeverage\PolarLink\PM & prog\CreateStoredProcs.sql* - Run SQL Scripts - 172.25.0.1(COGNOSERV)

File Edit View Run Options Connection



```
739 create or replace procedure jvalance.sp_cust ( )
740 language sql
741 result sets 1
742 begin
743     -- declare the cursor for the select statement
744     declare c1 cursor with return for
745         select trim(STATE) as STATE, ZIP, CUST_id, COMPANY, trim(LASTNAME) as LASTNAME
746         from zendsvr6.sp_cust
747         where COUNTRY = 'US'
748         order by STATE, ZIP;
749
750     -- open the cursor to return results to the client
751     open c1;
752
753 end
754 ;
755 -- Testing:
756 call jvalance.sp_cust();
```

Run the CALL

Results are displayed

STATE	ZIP	CUST_ID	COMPANY	LASTNAME
AL	30696	3042	Gold Coast Supply	
AL	32145	2984	Professional Divers, Ltd.	Mathers, Shirley
AL	32145	3041	Divers of Blue-green	Bean, Nancy
CA	90410	3984	Blue Glass Happiness	Taylor, Christine
CA	90740	3054	Catamaran Dive Club	Dupont, Nicole

<div1>

Running from PHP

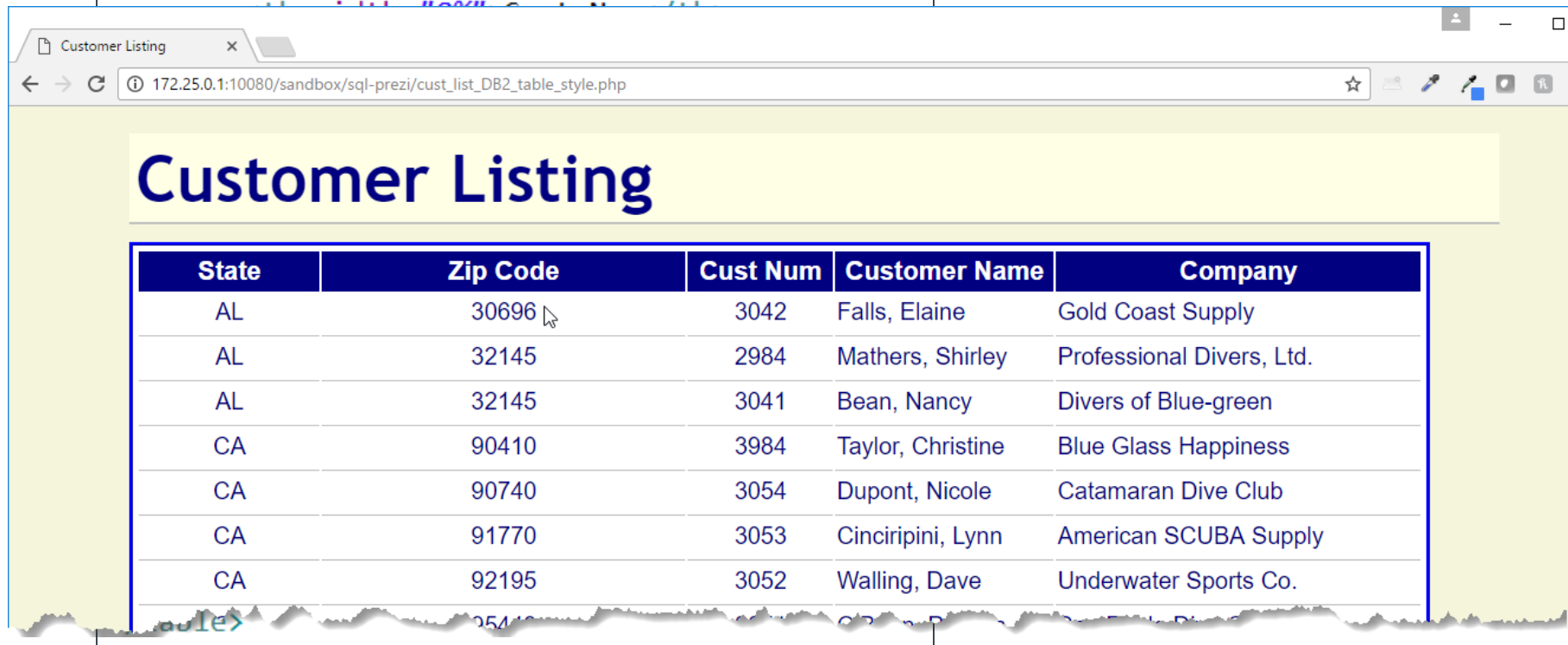
```
<h1>Customer Listing</h1>
<table>
  <tr>
    <th width="10%">State</th>
    <th width="20%">Zip Code</th>
    <th width="8%">Cust Num</th>
    <th width="12%">Customer Name</th>
    <th width="20%">Company</th>
  </tr>
  <?php
    $conn = db2_connect(DBHOST,DBUSER,DBPSWD);
    $sql = "call jvalance.sp_cust()";
    $stmt = db2_prepare($conn, $sql);
    db2_execute($stmt);

    while ($row = db2_fetch_assoc($stmt)) {
      display_row( $row );
    }

    db2_close ( $conn );
  ?>
</table>
```

Running from PHP

```
<h1>Customer Listing</h1>
<table>
  <tr>
    <th width="10%">State</th>
    <th width="20%">Zip Code</th>
    <th width="10%">Cust Num</th>
    <th width="30%">Customer Name</th>
    <th width="30%">Company</th>
  </tr>
  <tr>
    <td>AL</td>
    <td>30696</td>
    <td>3042</td>
    <td>Falls, Elaine</td>
    <td>Gold Coast Supply</td>
  </tr>
  <tr>
    <td>AL</td>
    <td>32145</td>
    <td>2984</td>
    <td>Mathers, Shirley</td>
    <td>Professional Divers, Ltd.</td>
  </tr>
  <tr>
    <td>AL</td>
    <td>32145</td>
    <td>3041</td>
    <td>Bean, Nancy</td>
    <td>Divers of Blue-green</td>
  </tr>
  <tr>
    <td>CA</td>
    <td>90410</td>
    <td>3984</td>
    <td>Taylor, Christine</td>
    <td>Blue Glass Happiness</td>
  </tr>
  <tr>
    <td>CA</td>
    <td>90740</td>
    <td>3054</td>
    <td>Dupont, Nicole</td>
    <td>Catamaran Dive Club</td>
  </tr>
  <tr>
    <td>CA</td>
    <td>91770</td>
    <td>3053</td>
    <td>Cinciripini, Lynn</td>
    <td>American SCUBA Supply</td>
  </tr>
  <tr>
    <td>CA</td>
    <td>92195</td>
    <td>3052</td>
    <td>Walling, Dave</td>
    <td>Underwater Sports Co.</td>
  </tr>
</table>
```

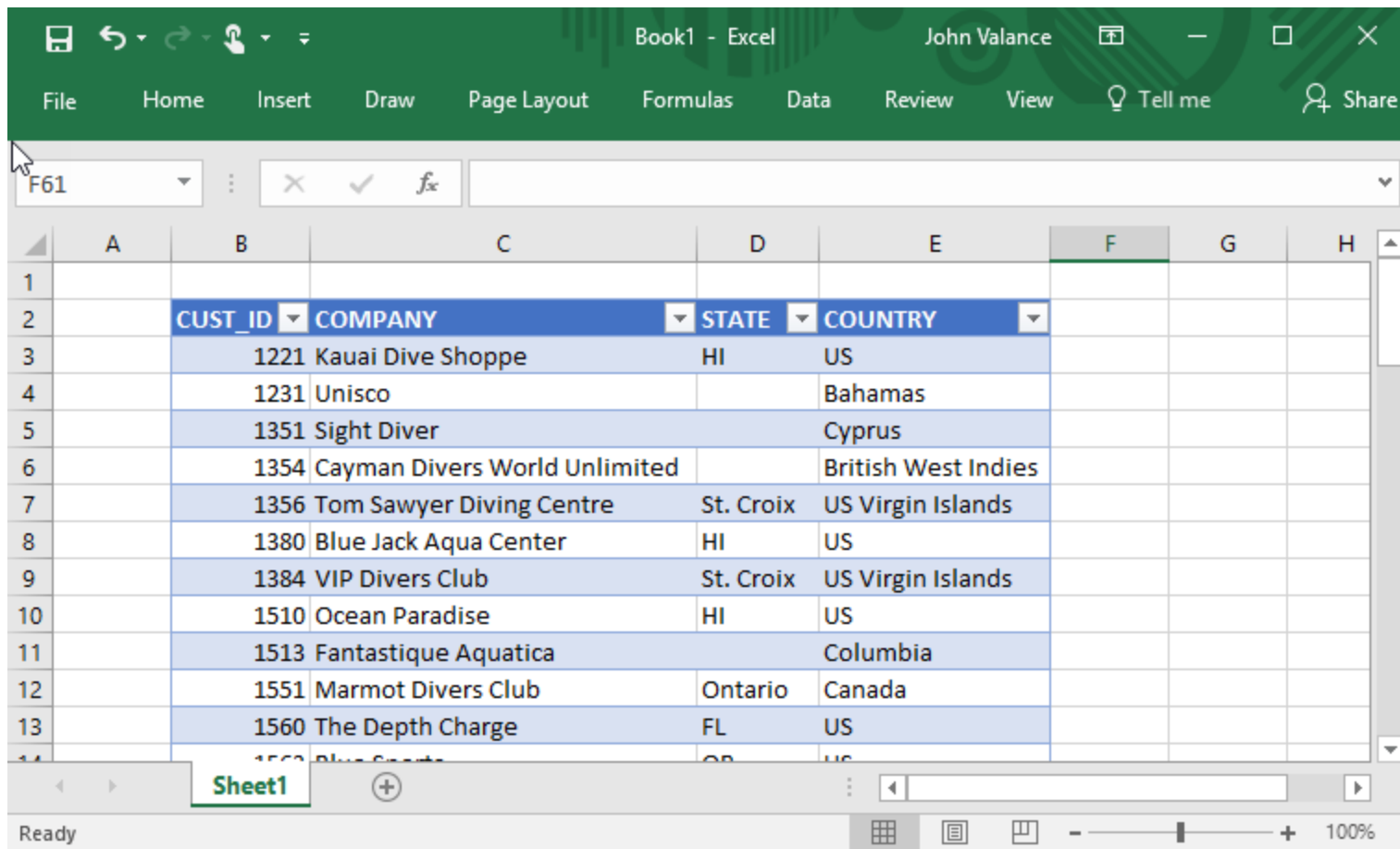


Customer Listing

State	Zip Code	Cust Num	Customer Name	Company
AL	30696	3042	Falls, Elaine	Gold Coast Supply
AL	32145	2984	Mathers, Shirley	Professional Divers, Ltd.
AL	32145	3041	Bean, Nancy	Divers of Blue-green
CA	90410	3984	Taylor, Christine	Blue Glass Happiness
CA	90740	3054	Dupont, Nicole	Catamaran Dive Club
CA	91770	3053	Cinciripini, Lynn	American SCUBA Supply
CA	92195	3052	Walling, Dave	Underwater Sports Co.

Running from Excel

Use MS Query, and edit SQL directly: call `sp_cust()`



The screenshot shows the Microsoft Excel interface with a table of customer data. The table is located in the range B2:E14. The columns are labeled CUST_ID, COMPANY, STATE, and COUNTRY. The data is as follows:

CUST_ID	COMPANY	STATE	COUNTRY
1221	Kauai Dive Shoppe	HI	US
1231	Unisco		Bahamas
1351	Sight Diver		Cyprus
1354	Cayman Divers World Unlimited		British West Indies
1356	Tom Sawyer Diving Centre	St. Croix	US Virgin Islands
1380	Blue Jack Aqua Center	HI	US
1384	VIP Divers Club	St. Croix	US Virgin Islands
1510	Ocean Paradise	HI	US
1513	Fantastique Aquatica		Columbia
1551	Marmot Divers Club	Ontario	Canada
1560	The Depth Charge	FL	US

Adding Parameters

- Add 2 input parameters: State and Country filters

- ▶ Add to parameter list
- ▶ Add to WHERE clause to serve as result filters

```
create or replace procedure jvalance.sp_cust_parm (  
    IN in_State char(2),  
    IN in_Country varchar(20)  
)  
language sql  
result sets 1  
begin  
    declare c1 cursor with return for  
    select * from zendsvr6.sp_cust  
    where STATE = in_State  
    and COUNTRY = in_Country;  
  
    open c1;  
  
end
```

- ▶ Separate multiple parameters with commas
- ▶ Can all go on one line, but easier to read on separate lines

Types of Parameters

- IN = Input
- OUT = Output
- INOUT = Input and Output

```
create procedure sp_SaveOrderHeader (  
    IN in_CustNum dec(8,0),  
    IN in_ShipTo dec(4,0),  
    INOUT io_OrderNum dec(12,0),  
    OUT out_message varchar(150)  
)
```

Run / Test SP with Parameters

```
create or replace procedure jvalance.sp_cust_parm (  
    IN in_State char(2),  
    IN in_Country varchar(20)  
)  
language sql  
result sets 1  
begin  
    declare c1 cursor with return for  
    select * from zendsvr6.sp_cust  
    where STATE = in_State  
    and COUNTRY = in_Country;  
  
    open c1;  
  
end
```

May yield
unexpected results!!



```
call jvalance.sp_cust_parm('HI', 'US');  
call jvalance.sp_cust_parm('OR', '');  
call jvalance.sp_cust_parm('', 'Canada');
```

These both
return 0 rows

Ignoring Blank Filter Parameters

```
create or replace procedure jvalance.sp_cust_parm (  
    IN in_State char(2),  
    IN in_Country varchar(20)  
)  
language sql  
result sets 1  
begin  
    declare c1 cursor with return for  
    select cu.*,  
    from zendsvr6.sp_cust cu  
    where (trim(in_State) = '' or trim(STATE) = trim(in_State))  
    and (trim(in_Country) = '' or trim(COUNTRY) = trim(in_Country))  
    ;  
  
    open c1;  
  
end
```


Ignoring Blank Filter Parameters

```
create or replace procedure jvalance.sp_cust_parm (  
    IN in_State char(2),  
    IN in_Country varchar(20)  
)  
language sql  
result sets 1  
begin  
    declare c1 cursor with return for  
    select cu.*,  
    from zendsvr6.sp cust cu  
    where (trim(in_State) = '' or trim(STATE) = trim(in_State))  
    and (trim(in_Country) = '' or trim(COUNTRY) = trim(in_Country))  
    ;  
    open c1;  
end
```

```
817  
818 call jvalance.sp_cust_parm('HI', 'US');  
819 call jvalance.sp_cust_parm('OR', '');  
820 call jvalance.sp_cust_parm('', 'Canada');
```

CUST_ID	COMPANY	STATE	COUNTRY
1221	Kauai Dive Shoppe	HI	US
1380	Blue Jack Aqua Center	HI	US
1510	Ocean Paradise	HI	US
1624	Makai SCUBA Club	HI	US
5412	Vashon Ventures	HI	US
5515	Ocean Adventures	HI	US

Ignoring Blank Filter Parameters

```
create or replace procedure jvalance.sp_cust_parm (  
    IN in_State char(2),  
    IN in_Country varchar(20)  
)  
language sql  
result sets 1  
begin  
    declare c1 cursor with return for  
    select cu.*,  
    from zendsvr6.sp cust cu  
    where (trim(in_State) = '' or trim(STATE) = trim(in_State))  
    and (trim(in_Country) = '' or trim(COUNTRY) = trim(in_Country))  
    ;  
  
    open c1;  
end
```

```
818 call jvalance.sp_cust_parm('HI', 'US');  
819 call jvalance.sp_cust_parm('OR', '');  
820 call jvalance.sp_cust_parm('', 'Canada');  
821
```

CUST_ID	COMPANY	STATE	COUNTRY
1563	Blue Sports	OR	US
2135	Frank's Divers Supply	OR	US
5165	Larry's Diving School	OR	US
6812	Waterspout SCUBA Center	OR	US

Ignoring Blank Filter Parameters

```
create or replace procedure jvalance.sp_cust_parm (  
    IN in_State char(2),  
    IN in_Country varchar(20)  
)  
language sql  
result sets 1  
begin  
    declare c1 cursor with return for  
    select cu.*,  
    from zendsvr6.sp_cust cu  
    where (trim(in_State) = '' or trim(STATE) = trim(in_State))  
    and (trim(in_Country) = '' or trim(COUNTRY) = trim(in_Country))  
    ;  
  
    open c1;  
  
end
```

```
818 call jvalance.sp_cust_parm('HI', 'US');  
819 call jvalance.sp_cust_parm('OR', '');  
820 call jvalance.sp_cust_parm('', 'Canada');
```

CUST_ID	COMPANY	STATE	COUNTRY
1551	Marmot Divers Club	Ontario	Canada
2156	Davy Jones' Locker	BC	Canada
4531	On-Target SCUBA	Manitoba	Canada

Beware Procedure Signatures!!

- **Procedure Overloading**
 - ▶ Two or more procedures with same name, but different signatures
- **Procedure Signature**
 - ▶ Name + Number of Parameters (data type irrelevant)
 - Ex.: these have different signatures
 - `MyProc(char(5), int)`
 - `MyProc(int)`
 - these have same signature:
 - `MyProc(char(5))`
 - `MyProc(int)`
- **Can cause a lot of confusion**
- **DROP PROCEDURE explicitly using Navigator... Databases**
 - ▶ Drill down to procedures

SQL/PL Basics



Trimming Input Search Params

- All the trim() function calls make the code harder to read

```
create or replace procedure jvalance.sp_cust_parm (  
    IN in_State char(2),  
    IN in_Country varchar(20)  
)  
language sql  
result sets 1  
begin  
    declare c1 cursor with return for  
    select cu.*,  
    from zendsvr6.sp_cust cu  
    where (trim(in_State) = '' or trim(STATE) = trim(in_State))  
    and (trim(in_Country) = '' or trim(COUNTRY) = trim(in_Country))  
    ;  
  
    open c1;  
  
end
```

- Let's create some variables to hold the trimmed values for reuse

Declaring/Using Variables

- Create variables to hold `trimmed()` input values

```
begin
```

```
declare wk_state char(2);  
declare wk_country varchar(20);
```

```
declare c1 cursor with return for  
select *
```

```
from zendsvr6.sp cust
```

```
where (wk_state = '' or trim(STATE) = wk_state)  
and (wk_country = '' or trim(COUNTRY) = wk_country));
```

```
set wk_state = trim(in_State);  
set wk_country = trim(in_Country);
```

```
open c1;
```

```
end
```

Declaring/Using Variables

- Create variables to hold `trimmed()` input values

begin

```
declare wk_state char(2);  
declare wk_country varchar(20);
```

Declare variables
before cursors

```
declare c1 cursor with return for  
select *  
from zendsvr6.sp_cust  
where (wk_state = '' or trim(STATE) = wk_state)  
and (wk_country = '' or trim(COUNTRY) = wk_country));
```

```
set wk_state = trim(in_State);  
set wk_country = trim(in_Country);  
open c1;
```

end

Declaring/Using Variables

- Create variables to hold `trimmed()` input values

begin

```
declare wk_state char(2);  
declare wk_country varchar(20);  
  
declare c1 cursor with return for  
select *  
from zendsvr6.sp_cust  
where (wk_state = '' or trim(STATE) = wk_state)  
and (wk_country = '' or trim(COUNTRY) = wk_country|));
```

Declaration
statements

```
set wk_state = trim(in_State);  
set wk_country = trim(in_Country);  
open c1;
```

Executable
statements

end

sp_Get_Cust_Name

More Features of SQL/PL

```
create or replace procedure jvalance.sp_Get_Cust_Name(
```

```
  IN in_cust_id dec(8,0),
```

```
  OUT out_cust_name varchar(42),
```

```
  OUT out_message varchar(100)
```

```
)
```

```
language sql
```

```
result sets 0
```

```
begin
```

```
  declare wk_first char(20);
```

```
  declare wk_last char(20);
```

```
  select FIRSTNAME, LASTNAME
```

```
  into wk_first, wk_last
```

```
  from sp_cust where cust_id = in_cust_id;
```

```
-- Check for null values - which means record not found
```

```
  if wk_first is null then
```

```
    set out_message = 'Customer ID ' || in_cust_id || ' is not valid.';
```

```
    return;
```

```
  else
```

```
    set out_cust_name = trim(wk_first) || ' ' || trim(wk_last);
```

```
    return;
```

```
  end if;
```

```
end;
```

Output Parameters

No Result Sets

SELECT INTO

Conditional Logic
(IF / THEN / ELSE)

Running sp_Get_Cust_Name

```
849  
850 call jvalance.sp_Get_Cust_Name(1231, ?, ?);  
851 call jvalance.sp_Get_Cust_Name(1384, ?, ?);  
852 call jvalance.sp_Get_Cust_Name(34876, ?, ?);  
853
```

Use ?'s as
placeholders for
output
parameters

[Sat Apr 08 20:49:07 EDT 2017] Run Selected...

```
> call jvalance.sp_Get_Cust_Name(1231, ?, ?)
```

Return Code = 0

Output Parameter #2('OUT_CUST_NAME') = George Weathers

Output Parameter #3('OUT_MESSAGE') = <NULL>

Statement ran successfully (15 ms)

Messages

Output parm
values are
shown in
Messages panel

Running sp_Get_Cust_Name

```
849  
850 call jvalance.sp_Get_Cust_Name(1231, ?, ?);  
851 call jvalance.sp_Get_Cust_Name(1384, ?, ?);  
852 call jvalance.sp_Get_Cust_Name(34876, ?, ?);  
853
```

[Sat Apr 08 21:01:10 EDT 2017] Run Selected...

> call jvalance.sp_Get_Cust_Name(1384, ?, ?)

Return Code = 0

Output Parameter #2('OUT_CUST_NAME') = Russell Christopher

Output Parameter #3('OUT_MESSAGE') = <NULL>

Statement ran successfully (99 ms)

Messages

Running sp_Get_Cust_Name - ERROR

```
849  
850 call jvalance.sp_Get_Cust_Name(1231, ?, ?);  
851 call jvalance.sp_Get_Cust_Name(1384, ?, ?);  
852 call jvalance.sp_Get_Cust_Name(34876, ?, ?);  
853
```

[Sat Apr 08 20:50:49 EDT 2017] Run Selected...

> call jvalance.sp_Get_Cust_Name(34876, ?, ?)

Return Code = 0

Output Parameter #2('OUT_CUST_NAME') = <NULL>

Output Parameter #3('OUT_MESSAGE') = Customer ID 34876 is not valid.

Statement ran successfully (127 ms)

Messages

User Defined Functions (UDFs)



User Defined Functions

- **Two types:**
 - ▶ User Defined Scalar Functions
 - ▶ User Defined Table Functions
- **We will focus on scalar functions**
 - ▶ Like BIFs or SQL scalar functions
 - `SUBSTRING('hello world', 7) => returns 'world'`
 - `STRIP(' blank on both ends ') => returns 'blank on both ends'`
 - `UPPER('hello') => returns 'HELLO'`
 - ▶ Can be used in
 - Expressions
 - Ex: `if upper(in_ShowPricing) = 'Y' then...`
 - Select lists as computed columns
 - Ex: `select substring(COMPANY, 1, 5) as COMP_SHORT from SP_CUST`

fn_Get_Cust_Name(in_cust_id)

```
create or replace function
jvalance.fn_Get_Cust_Name(
    in_cust_id dec(8,0)
)
returns varchar(42)
language sql
begin
    declare wk_first char(20);
    declare wk_last char(20);
    declare wk_custname varchar(42) default '';

    select FIRSTNAME, LASTNAME
    into wk_first, wk_last
    from sp_cust where cust_id = in_cust_id;

    -- Check for null values - which means record not found
    if wk_first is not null then
        set wk_custname = trim(wk_first) || ' ' || trim(wk_last);
    end if;

    return wk_custname;
end;
```


fn_Get_Cust_Name(in_cust_id)

CREATE FUNCTION

create or replace function

jvalance.fn Get Cust Name(
in_cust_id **dec**(8,0)

Input parameters only!
(Don't specify IN)

)
returns **varchar**(42)

Declare a
variable to hold
the return
value

language sql

begin

declare wk_first **char**(20);

declare wk_last **char**(20);

Declare returned
data type

declare wk_custname **varchar**(42) **default** '';

select FIRSTNAME, LASTNAME

into wk_first, wk_last

from sp_cust **where** cust_id = in_cust_id;

Set the return
variable value

-- Check for null values - which means record not found

if wk_first **is not null** **then**

set wk_custname = trim(wk_first) || ' ' || trim(wk_last);

end if;

return wk_custname;

Use return
statement to return
the value

end;

Testing fn_Get_Cust_Name(in_cust_id)

- You can test your UDFs using the SYSDUMMY1 table
 - ▶ IBM-supplied, single record table, for testing function calls
 - ▶ In library SYSIBM

```
select 1384 as cust_id, fn_Get_Cust_Name(1384) as cust_name  
from sysibm.sysdummy1
```

```
878 select 1384 as cust_id, fn_Get_Cust_Name(1384) as cust_name  
879 from sysibm.sysdummy1;  
880 |
```

CUST_ID	CUST_NAME
1384	Russell Christopher

Another Example - fn_CurrDate8()

Format current date as dec(8,0) in YYYYMMDD format

```
create or replace function jvalance.fn_CurrDate8()  
returns dec(8,0)  
language SQL  
set option datfmt = *ISO  
  
BEGIN  
    return dec(replace(char(current date, ISO), '-', ''), 8, 0) ;  
END;
```

```
881 select fn_CurrDate8() as date8  
882 from sysibm.sysdummy1
```

DATE8
20170408

```
insert into prodlib.ORDHDR (  
    OHORD, OHENTDAT  
) values (  
    in_OrderNo, fn_CurrDate8()  
)
```

Looping



The LOOP loop

- Loop infinitely
- Requires a conditional LEAVE or RETURN statement to exit the loop
- Like RPG DO

```
LOOP
    call work_to_do( all_done );
    IF all_done = 1 THEN
        LEAVE;
    END IF;
END LOOP;
```

The WHILE loop

- Loop until condition is false
- Condition tested at beginning of loop
 - ▶ Like RPG DOW
 - ▶ Set condition before loop, or it may never enter loop body
- Can also use **LEAVE** or **RETURN** statements to exit the loop

```
SET all_done = 0;
WHILE all_done = 0 DO
    call work_to_do( all_done, hit_error );
    IF hit_error = 1 THEN
        LEAVE;
    END IF;
END WHILE;
```

The REPEAT UNTIL loop

- Loop until condition is true
- Test condition at end of loop
 - ▶ Like RPG DOU
 - ▶ Always iterate at least once
- Can also use LEAVE or RETURN statements to exit the loop

```
REPEAT
    CALL work_to_do( all_done, hit_error );
    IF hit_error = 1 THEN
        LEAVE;
    END IF;
UNTIL all_done=1
END REPEAT;
```

The FOR loop

```
create or replace procedure jvalance.sp_Build_JVCUST()  
language sql  
result sets 0  
begin  
    /* Populates table JVCUST with selected data from table SP_CUST */  
    declare wk_full_name char(20); -- to hold first + last  
  
    delete from jvalance.JVCUST; -- clear previous data  
  
    FOR custrow AS csr_custs cursor for SELECT * from SP_CUST  
    DO  
        set wk_full_name = trim(custrow.FIRSTNAME) || ' ' || trim(custrow.LASTNAME);  
  
        insert into JVCUST (  
            CUST_ID, COMPANY, CUSTNAME, PHONE  
        ) values (  
            custrow.CUST_ID, custrow.COMPANY, wk_full_name, custrow.PHONE  
        );  
    END FOR;  
end;
```


Running procedure sp_Build_JVCUST

```
92  
93 cl: addlible jvalance;  
94  
95 call jvalance.sp_Build_JVCUST();  
96  
97 select * from jvalance.jvcust;
```

CUST_ID	COMPANY	CUSTNAME	PHONE
1221	Kauai Dive Shoppe	LINA Norman	808-555-0269
1231	Unisco	George Weathers	809-555-3915
1351	Sight Diver	Phyllis Spooner	357-6-876708
1354	Cayman Divers World Unlimited	Joe Bailey	011-5-697044
1356	Tom Sawyer Diving Centre	Chris Thomas	504-798-3022
1380	Blue Jack Aqua Center	Ernest Barratt	401-609-7623
1384	VIP Divers Club	Russell Christopher	809-453-5976
1510	Ocean Paradise	Paul Gardner	808-555-8231
1513	Fantastique Aquatica	Susan Wong	057-1-773434

IBM i Considerations



Nomenclature

RPG / Native IBMi	SQL / RDBMS
Library	Schema
File	Table
Record	Row
Field	Column

Naming Convention

System	SQL
library/file	schema.table
** Can use library lists **	Cannot use library lists

Library List Considerations

- **To use library lists, connect to DB with “System Naming”**
 - ▶ vs. “SQL Naming”, which only allows one library (aka schema)
 - ▶ In CA/ACS: menu Connection... JDBC settings... Format tab
 - ▶ In JDBC-based connections, set property: `naming = system`
- **Run “SET PATH *LIBL ;” in ACS SQL client, before creating procedures**
 - ▶ Path is stored in DB2 repository with the procedure object
- **Do NOT hard-code library names in your stored procedure source code** (use un-qualified object names)
- **Connect to DB with a USRPRF that has the proper library list**
 - ▶ via the USRPRF’s JOBD libl
 - ▶ When compiling and running

Viewing SQL Stored Procedure objects

- Library view (green-screen)

```
WRKOBJPDM LIB(JVALANCE)
          OBJTYPE(*PGM)
          OBJATR(CLE)
```

Work with Objects Using PDM S100388D

Library JVALANCE Position to _____
Position to type _____

Type option 2=Change 8=Display description 9=Save 10=Restore 11=Move ... display 7=Rename

Use SPECIFIC option to set short object name

Automatically generated pgm object names
First 5 + seq number

Opt	Object	Type	Attributes
—	BLDJVCUST	*PGM	
—	SP_CU000001	*PGM	CLE
—	SP_GE000001	*PGM	CLE

create procedure sp_build_jvcust()
specific BLDJVCUST

Thanks for Attending!



Contact Information

John Valance

jvalance@profoundlogic.com

802-355-4024

Profound Logic Software

www.profoundlogic.com

