SQL Stored Procedures

and Application Modernization

John Valance

Division 1 Systems

johnv@div1sys.com

<div1>

www.div1sys.com

About John Valance

- **30+ years IBM midrange experience (S/38 thru IBM i)**
- **17+ years of web development experience**
- **Independent consultant since early 2000**
- **Community Involvement**
  - COMMON Board of Directors
  - Presenter at IBM i groups nationwide
- **Founder and CTO of Division 1 Systems**
  - Web / Mobile applications for IBMi
  - Full SDLC - design, project management, coding, training, consulting
  - Extended team - Can scale up/down to meet client needs
- **Relationship with Zend / RogueWave**
  - Teacher, Reseller, Zend Certified Engineer

<div1>

Stored Procedures: What, Why, Who?

What We Will Cover

- **Introduction**
  - Who, What, Why
  - Application Modernization
- **Creating Stored Procedures**
- **SQL Procedure Language**
  - Language Syntax and Capabilities
- **Creating UDFs**
- **IBM i Considerations**
- **Examples from the trenches (time permitting)**

<div1>

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'

<div1>

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**

<div1>

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;
```

<div1>

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

```
-- Testing:
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
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	Spurling, Michael

<div1>

What Can I Build with SQL/PL?

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

<div1>

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/rbafzprintthis.htm
  - SQL Programming Guide:
    https://www.ibm.com/support/knowledgecenter/ssw_ibm_i_73/sqlp/rbafykickoff.htm

<div1>

Benefits of SQL Stored Procedures

- **Declarative, standardized programming language**
  - Puts SQL in the driver's seat
- **Centralize business logic**
  - Data-Centric programming
  - Business logic close to the DBMS
- **Simplicity of application code**
  - No embedded SQL
  - No ORM issues
- **Impact of change insulation**
  - No level checks
  - Loose coupling between DB and apps
- **Leverage SQL enhancements**
  - Features and performance
- **Security**
  - No SQL injection
  - Define authority on stored procedures vs. tables/views
- **Performance**
  - Execution plan / Precompiled
- **Modernize DB interface**
  - Similar to a View
  - Renamed fields, Derived fields
  - Provide simple interface to complex legacy DB structure

<div1>

Whom Is This For?

- **IT Managers / Directors**
  - Don't replace – Refactor!
- **RPG Programmers**
  - Modernize your database skills
  - You probably know SQL already
- **Database Administrators**
  - Play an active role in application modernization
- **Project Managers**
  - Focus on database issues
  - Delegate programming
- **Web Application Developers**
  - Browser applications
  - Ajax developers
  - API / Web Service Developers
- **Users of Analysis & Reporting Tools**
  - Excel spreadsheets / VBA apps
  - Crystal Reports and other reporting tools

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;
```

<div1>

JV1

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

<div1>

Slide 15

JV1 John Valance, 4/7/2017

Anatomy of a Stored Procedure

```
create or replace procedure jvalance.sp_cust ( )
language sql
result sets 1
begin
    -- declare the cursor for the selec
    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)

<div1>

Anatomy of a Stored Procedure

```
create or replace procedure jvalance.sp_cust ( )
language sql
result sets 1
begin
    -- declare the cursor for the select stateme
    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

<div1>

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,
                COMP
                trim                                    STNAME) AS NAME
        from zendsvr
        where COUNTR
        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

<div1>

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

<div1>

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(FIRST NAME) 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

<div1>

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 – stored with object**
- **Create indices (LFs) to improve performance**
  - SQL Performance Center in ACS
  - Help with DB tuning based on runtime analysis

<div1>

Compiling and Running

Need to use an SQL Client to run the CREATE PROCEDURE

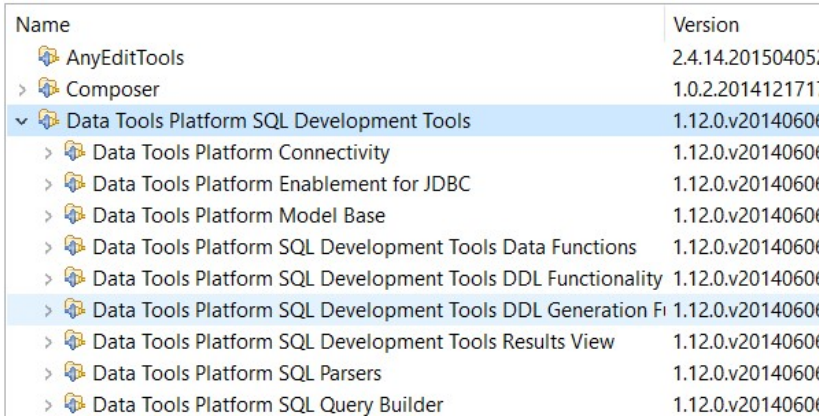
- **SQL client choices**
  - ACS = Access Client Solutions (** best choice **) 
  - Green screen STRSQL (awkward)  $$
  - Eclipse Data Tools plug-in for RDi, Zend Studio, etc. (pretty good)
  - Other SQL Clients – JDBC, ODBC (ex.: http://www.sql-workbench.net/)
- **Biggest issue is handling output parameters**
  - 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

<div1>

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.v20140600
Data Tools Platform Connectivity	1.12.0.v20140600
Data Tools Platform Enablement for JDBC	1.12.0.v20140600
Data Tools Platform Model Base	1.12.0.v20140600
Data Tools Platform SQL Development Tools Data Functions	1.12.0.v20140600
Data Tools Platform SQL Development Tools DDL Functionality	1.12.0.v20140600
Data Tools Platform SQL Development Tools DDL Generation F	1.12.0.v20140600
Data Tools Platform SQL Development Tools Results View	1.12.0.v20140600
Data Tools Platform SQL Parsers	1.12.0.v20140600
Data Tools Platform SQL Query Builder	1.12.0.v20140600

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(COGNO

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(COGNO

File Edit View Run Options Connection

```
739 create or replace procedure jv
740 language sql
741 result sets 1
742 begin
743     -- declare the cursor for the select st
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

Position Cursor on Statement to Run

Then…

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

<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(COGNO

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(COGNO

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 th
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	
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>
```

<div1>

Running from PHP

```
<h1>Customer Listing</h1>
<table>
    <tr>
        <th width="10%">State</th>
        <th width="20%">Zip Code</th>
```

Customer Listing

172.25.0.1:10080/sandbox/sql-prezi/cust_list_DB2_table_style.php

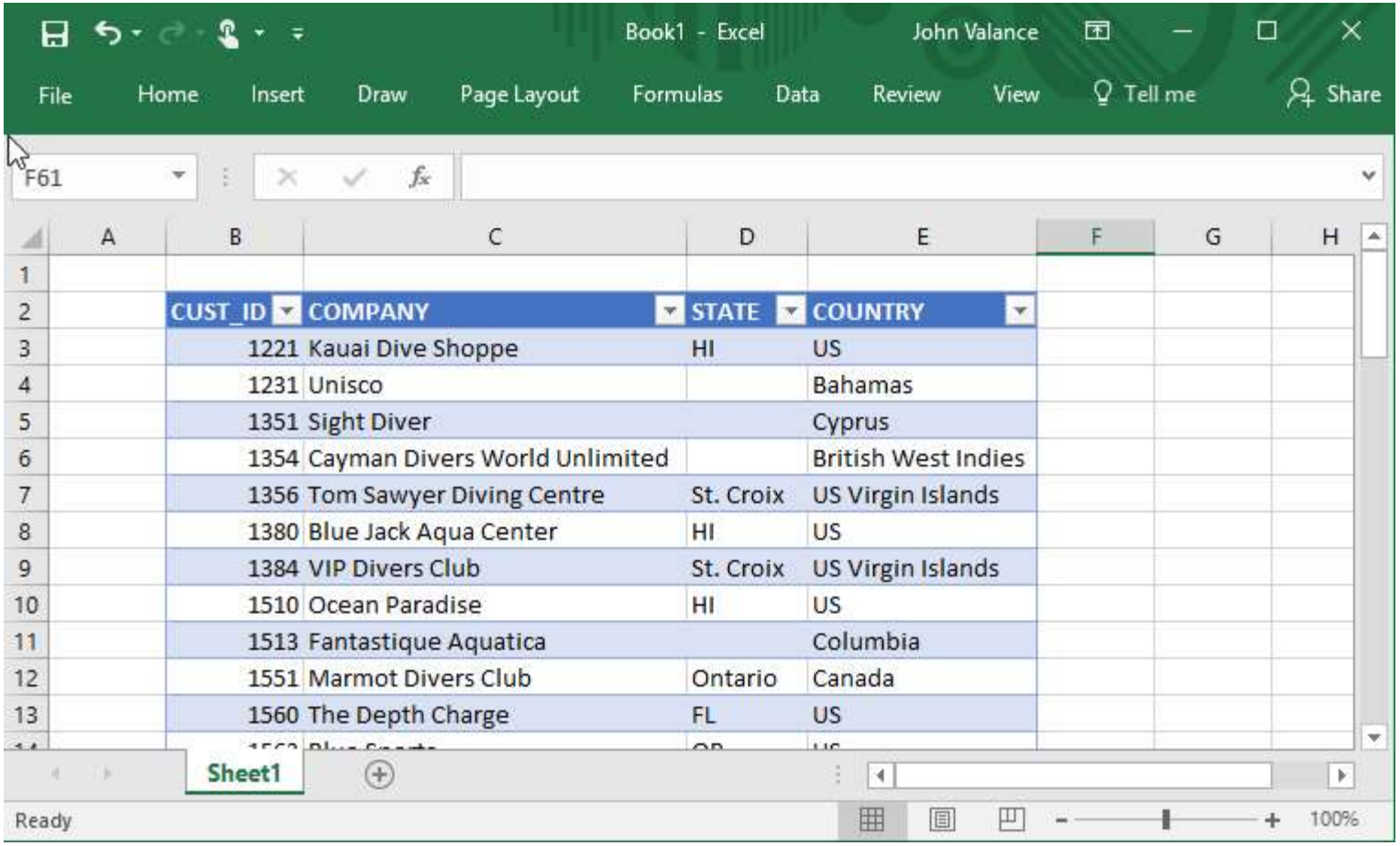
Customer Listing

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.

<div1>

Running from Excel

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



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

<div1>

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

<div1>

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)
)
```

<div1>

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

<div1>

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
```

<div1>

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
```

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

<div1>

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
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

<div1>

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
```

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

<div1>

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

<div1>

SQL/PL Basics

Trimming Input Search Parms

- 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

<div1>

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
```

<div1>

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
```

Declare variables before cursors

<div1>

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
```

Declaration statements

Executable statements

<div1>

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)

<div1>

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 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   (1[illegible] ms)

Messages
```

Use ?'s as placeholders for output parameters

Output parm values are shown in Messages panel

<div1>

Running sp_Get_Cust_Name

```
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
```

<div1>

Running sp_Get_Cust_Name - ERROR

```
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

<div1>

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`

<div1>

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;
```

<div1>

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 rec
    if wk_first is not null then
        set wk_custname = trim(wk_first) || ' ' || trim(wk_last);
    end if;

    return wk_custname;
end;
```

CREATE FUNCTION

Input parameters only! (Don't specify IN)

Declare returned data type

Declare a variable to hold the return value

Set the return variable value

Use return statement to return the value

<div1>

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

<div1>

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()
)
```

<div1>

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;
```

<div1>

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;
```

<div1>

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;
```

<div1>

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;
```

<div1>

Running procedure sp_Build_JVCUST

```
cl: addlible jvalance;

call jvalance.sp_Build_JVCUST();

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

<div1>

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

<div1>

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

<div1>

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 optio
  2=Change                                          isplay        7=Rename
  8=Display des                 9=Save        10=Restore       11=Move ...

Opt  Object        Typ
__   BLDJVCUST     *PGM                                    D_JVCUST
__   SP_CU00001    *PGM          CLE
__   SP_GE00001    *PGM          CLE
```

Use SPECIFIC option to set short object name

Automatically generated pgm object names
First 5 + seq number

```
create procedure sp_build_jvcust()
specific BLDJVCUST
```

<div1>

Real-world Examples

- **Time Permitting**

Thanks for Attending!

Contact Information

John Valance

johnv@div1sys.com

802-355-4024

Division 1 Systems

www.div1sys.com

<div1>