SQL Stored Procedures
and the SQL Procedure Language

John Valance
Division 1 Systems
johnv@div1sys.com

© 2017 Division 1 Systems
About John Valance

- 30+ years IBM midrange experience (S/38 thru IBM i)
- 17+ years of web development experience
- Software Developer
  - RPG / Java / PHP / JavaScript / SQL
- Independent consultant - 2000 to 2018
- Founder and CTO of Division 1 Systems (www.div1sys.com)
  - Web / Mobile applications for IBM i
- Community Involvement
  - COMMON Board of Directors
  - Presenter at IBM i groups nationwide
- Profound Logic Employee - as of Nov 2018 (www.profoundlogic.com)
  - Senior Consultant / Modernization and Digital Transformation Strategist
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)
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**
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
```

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

- Stored procedures
- User Defined Functions (UDF)
- Triggers - before & after (add/change/delete)
New Redbook - April 2016!!

- SQL Procedures, Triggers, and Functions on IBM DB2 for i
  - Download the PDF!!

- Also:
  - DB2 for i SQL reference
  - SQL Programming Guide:
Building Procedures with SQL/PL
Anatomy of a Stored Procedure

```sql
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

```sql
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

```sql
create or replace procedure jvalance.sp_cust (
begin
  language sql
  result sets 1
  -- 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)
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
  from zendsvnm
  where COUNTRY = 'USA'
  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
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

```sql
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;
```
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
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/](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](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...)
Run SQL Scripts in ACS

```
create or replace procedure jvalance.
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);
    from zendsvr6.sp_cust
    where COUNTRY = 'US'
    order by STATE, ZIP;

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

end;

-- Testing:
call jvalance.sp_cust();
```

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

Connect to Database
Run SQL Scripts in ACS

- Position Cursor on Statement to Run
- Then...
  - Use Run Menu, or...
  - Click Run Icons, or...
  - Press Ctrl+R

```sql
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)
    from zendsvr6.sp_cust
    where COUNTRY = 'US'
    order by STATE, ZIP;

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

end ;
-- Testing:
call jvalance.sp_cust();
```

<table>
<thead>
<tr>
<th>STATE</th>
<th>ZIP</th>
<th>CUST_ID</th>
<th>COMPANY</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>AL</td>
<td>30696</td>
<td>3042</td>
<td>Gold Coast Supply</td>
<td>Falls, Elaine</td>
</tr>
<tr>
<td>AL</td>
<td>32145</td>
<td>2984</td>
<td>Professional Divers, Ltd.</td>
<td>Mathers, Shirley</td>
</tr>
<tr>
<td>AL</td>
<td>32145</td>
<td>3041</td>
<td>Divers of Blue-green</td>
<td>Bean, Nancy</td>
</tr>
<tr>
<td>CA</td>
<td>90410</td>
<td>3984</td>
<td>Blue Glass Happiness</td>
<td>Taylor, Christine</td>
</tr>
<tr>
<td>CA</td>
<td>90740</td>
<td>3054</td>
<td>Catamaran Dive Club</td>
<td>Dupont, Nicole</td>
</tr>
</tbody>
</table>
Run SQL Scripts in ACS

```sql
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):
    from zendsvr6.sp_cust
    where COUNTRY = 'US'
    order by STATE, ZIP;

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

end;

-- Testing:
call jvalance.sp_cust();
```

Run the CREATE
Run SQL Scripts in ACS

```
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)
    from zendsvr6.sp_cust
    where COUNTRY = 'US'
    order by STATE, ZIP;

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

end;
```

```
-- Testing:
call jvalance.sp_cust();
```

<table>
<thead>
<tr>
<th>STATE</th>
<th>ZIP</th>
<th>CUST_ID</th>
<th>COMPANY</th>
</tr>
</thead>
<tbody>
<tr>
<td>AL</td>
<td>30696</td>
<td>3042</td>
<td>Gold Coast Supply</td>
</tr>
<tr>
<td>AL</td>
<td>32145</td>
<td>2984</td>
<td>Professional Divers, Ltd.</td>
</tr>
<tr>
<td>AL</td>
<td>32145</td>
<td>3041</td>
<td>Divers of Blue-green</td>
</tr>
<tr>
<td>CA</td>
<td>90410</td>
<td>3984</td>
<td>Blue Glass Happiness</td>
</tr>
<tr>
<td>CA</td>
<td>90740</td>
<td>3054</td>
<td>Catamaran Dive Club</td>
</tr>
</tbody>
</table>

Run the CALL
Results are displayed
Customer Listing

<table>
<thead>
<tr>
<th>State</th>
<th>Zip Code</th>
<th>Cust Num</th>
<th>Customer Name</th>
<th>Company</th>
</tr>
</thead>
</table>

```php
<?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);
?>
```
Running from PHP

```
<h1>Customer Listing</h1>
<table>
  <tr>
    <th width="10%">State</th>
    <th width="20%">Zip Code</th>
    <th> Cust Num </th>
    <th> Customer Name </th>
    <th> 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>
```
Use MS Query, and edit SQL directly: `call sp_cust()`
Adding Parameters

• Add 2 input parameters: State and Country filters
  ▸ Add to parameter list
  ▸ Add to WHERE clause to serve as result filters

```sql
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
```
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

```sql
create or replace procedure jvalance.sp_cust_parm (  
    IN in_State char(2),
    IN in_Country varchar(20)
)  
language sql  
result sets 1  
begi
    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 examples:

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

```sql
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

```sql
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
```

<table>
<thead>
<tr>
<th>CUST_ID</th>
<th>COMPANY</th>
<th>STATE</th>
<th>COUNTRY</th>
</tr>
</thead>
<tbody>
<tr>
<td>1221</td>
<td>Kauai Dive Shoppe</td>
<td>HI</td>
<td>US</td>
</tr>
<tr>
<td>1380</td>
<td>Blue Jack Aqua Center</td>
<td>HI</td>
<td>US</td>
</tr>
<tr>
<td>1510</td>
<td>Ocean Paradise</td>
<td>HI</td>
<td>US</td>
</tr>
<tr>
<td>1624</td>
<td>Makai SCUBA Club</td>
<td>HI</td>
<td>US</td>
</tr>
<tr>
<td>5412</td>
<td>Vashon Ventures</td>
<td>HI</td>
<td>US</td>
</tr>
<tr>
<td>5515</td>
<td>Ocean Adventures</td>
<td>HI</td>
<td>US</td>
</tr>
</tbody>
</table>
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;
call jvalance.sp_cust_parm('HI', 'US');
call jvalance.sp_cust_parm('OR', '');
call jvalance.sp_cust_parm('', 'Canada');
end
```
Ignoring Blank Filter Parameters

```sql
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
```

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

<table>
<thead>
<tr>
<th>CUST_ID</th>
<th>COMPANY</th>
<th>STATE</th>
<th>COUNTRY</th>
</tr>
</thead>
<tbody>
<tr>
<td>1551</td>
<td>Marmot Divers Club</td>
<td>Ontario</td>
<td>Canada</td>
</tr>
<tr>
<td>2156</td>
<td>Davy Jones' Locker</td>
<td>BC</td>
<td>Canada</td>
</tr>
<tr>
<td>4531</td>
<td>On-Target SCUBA</td>
<td>Manitoba</td>
<td>Canada</td>
</tr>
</tbody>
</table>
```
• 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 SearchParms

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

```sql
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

```sql
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
```
• Create variables to hold trimmed() input values

```sql
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
• Create variables to hold trimmed() input values

```sql
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
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
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;
Running sp_Get_Cust_Name

Use ?’s as placeholders for output parameters

Output parm values are shown in Messages panel
Running sp_Get_Cust_Name

call jvalance.sp_Get_Cust_Name(1231, ?, ?);
call jvalance.sp_Get_Cust_Name(1384, ?, ?);
call jvalance.sp_Get_Cust_Name(34876, ?, ?);

[ 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

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

```
[ 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)
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
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)

```sql
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
    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 a variable to hold the return value
Declare returned data type
Set the return variable value
Use return statement to return the value
You can test your UDFs using the SYSDUMMY1 table

- IBM-supplied, single record table, for testing function calls
- In library SYSIBM

```sql
select 1384 as cust_id, fn_Get_Cust_Name(1384) as cust_name
from sysibm.sysdummym1
```
Another Example - `fn_CurrDate8()`

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

```sql
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;
```

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

```
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

```plaintext
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

```plaintext
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, custrowPHONE);
    END FOR;
end;
Running procedure sp_Build_JVCUST

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

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

<table>
<thead>
<tr>
<th>RPG / Native IBMi</th>
<th>SQL / RDBMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Library</td>
<td>Schema</td>
</tr>
<tr>
<td>File</td>
<td>Table</td>
</tr>
<tr>
<td>Record</td>
<td>Row</td>
</tr>
<tr>
<td>Field</td>
<td>Column</td>
</tr>
</tbody>
</table>

### Naming Convention

<table>
<thead>
<tr>
<th>System</th>
<th>SQL</th>
</tr>
</thead>
<tbody>
<tr>
<td>library/file</td>
<td>schema.table</td>
</tr>
<tr>
<td><strong>Can use library lists</strong></td>
<td>Cannot use library lists</td>
</tr>
</tbody>
</table>
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)

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

Work with Objects Using PDM

<table>
<thead>
<tr>
<th>Opt</th>
<th>Object</th>
<th>Type</th>
<th>Description</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BLDJVCUST</td>
<td>*PGM</td>
<td>SQL PROCEDURE sp_build_jvcust</td>
<td>display</td>
</tr>
<tr>
<td></td>
<td>SP_CU000001</td>
<td>*PGM</td>
<td>CLE</td>
<td>Save</td>
</tr>
<tr>
<td></td>
<td>SP_GE000001</td>
<td>*PGM</td>
<td>CLE</td>
<td>Restore</td>
</tr>
</tbody>
</table>

Use SPECIFIC option to set short object name

Automatically generated pgm object names
First 5 + seq number

create procedure sp_build_jvcust()
  specific BLDJVCUST
Thanks for Attending!
Contact Information

John Valance

johnv@div1sys.com
802-355-4024

Division 1 Systems

www.div1sys.com