SQL PL

in a Nutshell

Ted Holt

Senior Software Developer, Profound Logic Software

Senior Technical Editor, The Four Hundred

fourhundredguru@gmail.com

We will answer these questions:

1. What is SQL PL?
2. What can I do with it?
3. What are the features?
4. How do I run it?
5. Should I quit writing RPG?

What is SQL PL?

- A programming language
- An interface to relational database management systems
- Based on SQL/Persistent Stored Modules (SQL/PSM), an ISO standard
- Supported on all members of the DB2 family
  - DB2 for z/OS
  - DB2 for Linux, Unix and Windows
  - DB2 for I
- Well-suited for data-centric programming
- Easy
  - Simple syntax
  - Limited in scope
  - Database statements are freely mixed with control statements

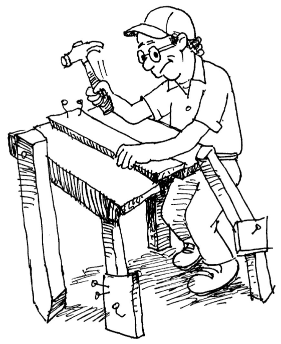
What can I do with SQL PL?

You can use SQL PL to . . .

- implement data-centric programming;
- define stored procedures (*PGM objects)
- define triggers
- define user-defined functions
- write SQL scripts (interpreted)

You *cannot* use SQL PL to . . .

- write to device files



What are the features of SQL PL?

The two types of statements

A SQL PL routine (stored procedure, trigger, function, script) is built of statements.

- Simple statements
- Compound statements (“block”)

Simple statements

- SQL statements (CREATE TABLE, INSERT, etc.)
- Assignment (SET, VALUES . . . INTO)
- Conditional (IF, CASE)
- Iteration (FOR, WHILE, REPEAT, LOOP)
- Transfer of control (RETURN, LEAVE, ITERATE, GOTO)
- CALL

Compound statements

- BEGIN {ATOMIC | <u>NOT ATOMIC</u>}
- declarations
- one or more statements
- END

Compound statements

- It is most common to use a compound statement for the body.
- A compound statement may contain both simple and compound statements.
- Terminate declarations and statements with semicolons.
- No semicolons after THEN and DO.
- An atomic statement is treated as a whole.
  The statements in a non-atomic statement stand on their own.

Declarations

- Must be defined in this order:
  - variables
  - conditions
  - cursors
  - condition handlers
- Must be referenced only within the compound statement in which it is declared. (That includes nested compound statements.)

Declarations — Variables

```
declare  v_Item   char(12);
declare  v_Sep    varchar (1)    default ',';
```

- The DEFAULT keyword gives the initial value. Default is null.
- A variable may be referenced within the compound statement in which it is declared and in nested compound statements.
- Use SET to modify a variable.
- Do not prefix colons to variable references, as RPG requires.

Declarations — Conditions

```
declare CreateFailed condition for sqlstate value '42710';
declare CreateFailed condition for sqlstate '42710';
declare CreateFailed condition for '42710';
```

All three forms are identical

- Gives a name to a SQL state.
- Allows you to make your code more descriptive

Declarations — Cursors

```
create or replace procedure Allocate
  (in  p_Item  char (6))

begin atomic

   declare   SqlState          char (5);
   declare   v_Qty_Available   dec  (5);
   declare   v_Order           dec  (9);
   declare   v_Line            dec  (3);
   declare   v_Qty             dec  (6);

   declare c1 cursor for
      select oo.Order, oo.Line, oo.QtyOpen
        from OpenSalesOrders as oo
       where oo.Item = p_Item
       order by oo.Order, oo.Line;

   open c1;
```

```
   repeat

      fetch c1 into v_Order, v_Line, v_Qty;

      if SqlState < '02000' then

         . . .

      end if;

   until SqlState >= '02000'
     or v_Qty_Available <= 0

   end repeat;

   close c1;

end
```

Declarations — Condition Handlers

```
declare condition-type handler for exception-ID(s)
   statement;
```

- Three types – continue, exit, undo
- All three execute one statement
- An exception ID may be SQL state or a condition name

```
      declare CreateFailed  condition for sqlstate '42710';
      declare continue handler for CreateFailed     statement;
```

or `declare continue handler for sqlstate ‘42710’ statement;`

- Separate multiple exception ID’s with commas

Declarations — Condition Handlers

```
declare CreateFailed  condition for sqlstate '42710';

declare continue handler for CreateFailed
   begin
      . . . more code . . .
   end;

create table plants
   (ID dec(3),  Location  varchar(16),
    primary key (ID));
```

- Continue handler – return to the statement following the one that caused the exception

Declarations — Condition Handlers

```
declare exit handler for sqlexception
   begin
      set ErrorMsg = 'SQLSTATE=' concat sqlstate;
      signal sqlstate '99001'
         set message_text = ErrorMsg;
   end;
```

- Exit handler – Leave the compound statement
- Undo handler – Roll back the changes and leave the compound statement; only permitted in ATOMIC statements

Assignment statements

- SET
- May assign to two or more variables in one statement.
- May use a query to retrieve a value.

```
declare   v_Option   varchar(8);
declare   v_Post     char(6);
declare   v_Counter  integer   default 0;

set v_Option = ‘RETRY';
set v_Option = ‘RETRY', v_Post = ‘NOPOST';
set v_Counter = v_Counter + 1;
set v_Counter = (select count(*) from Sales);
```

Conditional statements — IF

- `IF … THEN … END IF`
- `IF … THEN … ELSE … END IF`
- `IF … THEN … ELSEIF … THEN … ELSE … END IF`

- No semicolons after THEN and ELSE
- One statement after THEN and ELSE
- Notice the space in END IF

Conditional statements — IF

```
create trigger ItemInsert
   no cascade
   before insert on items
   referencing new row as n
   for each row
   mode db2row
if n.Stocking_UOM = 'KG' and n.weight <= 0 then
   signal sqlstate '86100'
      set Message_text = 'Weight must be positive or null';
end if
```

Conditional statements — CASE

- CASE value WHEN … ELSE … END CASE
- CASE WHEN … ELSE … END CASE

- If there is no ELSE and no case is true, CASE fails with SQLSTATE 20000.
- Ends with END CASE, not END

Conditional statements — CASE

```
case
   when Stocking_UOM = 'KG' then
      if n.weight <= 0 then
         signal sqlstate '86100'
            set Message_text = 'Weight must be positive or null';
      end if;
   when Stocking_UOM in ('CM', 'M') then
      if n.length <= 0 then
         signal sqlstate '86100'
            set Message_text = 'Length must be positive or null';
      end if;
   else
      signal sqlstate '86199'
         set Message_text = 'Invalid unit of measure';
end case
```

Iteration statements

• FOR	iterates over a read-only result set
• LOOP	must be broken from within the loop body
• WHILE	top-tested loop
• REPEAT	bottom-tested loop

Iteration statements — FOR

FOR condition DO body END WHILE

```
create or replace Procedure Billing (in p_BillingCycle dec(3))
for One_Customer as
   select c.AccountNumber
     from customers as c
    where c.BillingCycle = p_BillingCycle
 do
   call Bill200R (One_Customer.AccountNumber);
end for
```

Iteration statements — LOOP

```
Loop1:
   Loop
      fetch c_Bill into v_Company, v_Order, v_Line, v_Item, v_Qty;
      if SqlState = '02000'
         then leave Loop1;
      end if;

      . . . more code . . .

   end loop;
```

Iteration statements — WHILE

```
while v_List <> ' ' do

   set v_Pos = Locate (v_Sep, v_List);
   set v_Department = dec (substr ( v_List, 1, v_Pos - 1 ) );
   insert into session.DeptList values(v_Department);
   set v_List = substr (v_List, v_Pos + 1);

end while;
```

Iteration statements — REPEAT

```
repeat
   fetch c1 into v_Order, v_Line, v_Qty;
   if SqlState < '02000' then
      . . . more code . . .
   end if;
   until SqlState >= '02000'
      or v_Qty_Available <= 0
end repeat;
```

Labels

```
Main_routine: begin

. . .
if pOption = 'X' then
          leave Main_routine;
   end if;
. . .

end Main_routine;
```

- Indicated by a trailing colon.
- You may label any executable statement, but the only practical places for labels are loop structures and compound statements.
- You may include a label after END. That label must match the label for the corresponding BEGIN.

Comments

- Double dash (--) — comment the remainder of the line

```
-- check the customer for credit hold
```

- Block comments (like CL) —/*   . . .   */

```
/* =======================================
   Post accounts payable to general ledger
   2017-02-31 Dexter Fillmore
   ==================================== */
```

Handling exceptions

- If there is no applicable handler for an exception, the system sends the exception to the caller.
- Exception handling is based on SQL state, not SQL code.
- Condition handlers for specific conditions take precedence over condition handlers for general conditions.
- A condition handler executes only one statement, which may be a compound statement.

Handling exceptions

To ignore an exception, write a continue handler that does nothing.

```
begin
   declare CreateFailed condition for sqlstate '42710';
   declare continue handler for CreateFailed begin end;
   create table plants
   ( ID dec(3),  Location  varchar(16), primary key (ID));
   . . . more . . .
end
```

Handling exceptions

To take action for an exception and keep going, write a continue handler.

```
begin
   declare v_Status     integer       default 0;
   declare CreateFailed condition for sqlstate '42710';
   declare continue handler for CreateFailed
      set v_Status = 1;
   create table plants
   ( ID dec(3),  Location  varchar(16), primary key (ID));
```

Handling exceptions

To cancel after a fatal error, write an exit or undo handler.

```
begin
   declare v_Status     integer       default 0;

   declare CreateFailed condition for sqlstate '42710';

   declare exit handler for CreateFailed
      signal sqlstate '88001'
               set message_text = 'Plants table exists';

   create table plants
   ( ID dec(3),  Location  varchar(16), primary key (ID));
```

Forcing a condition

To force a condition, use SIGNAL.

```
if v_Count > 20 then
   signal sqlstate '88001'
          set message_text = ‘Table size exceeded.';
end if;
```

- If a condition handler is defined, the condition handler receives control.
- If not, the condition is sent to the caller.

Forwarding a condition

To forward a condition to the caller, use RESIGNAL.

```
declare exit handler for CreateFailed
   resignal;
```

- RESIGNAL is only permitted within a condition handler.
- You can use RESIGNAL to send the error that caused the handler to take control, or you can send some other SQL state instead.

How do I run it?

Object creation

1. Key source into a source physical file member or stream file.
2. Use RUNSQLSTM to execute the SQL code.

```
RUNSQLSTM SRCFILE(SCRIPTS) SRCMBR(LOADPLANTS) COMMIT(*NONE)
```

   or

```
RUNSQLSTM SRCSTMF('LoadPlants.SQL') COMMIT(*NONE)
```

3. Do not assume that the script succeeded. Check the report!

Debugging Options

- System debugger in ACS
- IBM Data Studio
- Green-screen STRDBG

For information on the graphical debugging options, see the resources slide.

Green-screen debugging

Set the debug view option.

```
create or replace procedure temp1
   set option dbgview = *source
begin
   declare    v_Option    varchar(8);
   declare    v_Post      char(6);
   declare    v_Counter   integer    default 0;
   . . .
end
```

- Create the object.
- STRDBG
- Use EVAL %LOCALVARS to determine the variable names in the generated C code.

```
eval %localvars
```

Green-screen debugging

- To view a numeric variable:

```
eval SQLP_L2.V_COUNTER
```

Remember, C is case-sensitive!

- To view a character variable:

```
eval *SQLP_L2.V_POST :s 6
```

Use * to dereference the pointer.

Specify the length of the string.

Note: VARCHAR variables have two parts: xxx.LEN (length of the value) and xxx.DAT (pointer to the value).

Does it replace RPG?

A Matter of Opinion

- Are you ready to embrace data-centric programming?
- How important is portability?
- Do you want to support another language in your shop?
- Do you want to make yourself more marketable?

Resources

- *SQL Procedures, Triggers, and Functions on IBM DB2 for i*
  Bainbridge et al
  http://www.redbooks.ibm.com/abstracts/sg248326.html?Open
- *IBM Data Studio debugger and IBM DB2 for i*
  Kent Milligan
  http://www.ibm.com/developerworks/ibmi/library/i-debugger-db2-i/
- *DB2 SQL Procedural Language for Linux, Unix & Windows*
  Yip et al
  out of print, but available on the web
  PDF at http://confonet.nic.in/tsp/db2_sql_book.pdf

Resources

- *Toadworld.com DB2 wiki*
  http://www.toadworld.com/platforms/ibmdb2/w/wiki
- *SQL-PL Guide*
  http://www.sqlpl-guide.com/

END PRESENTATION;