

# How to Write SQL Stored Procedures

Rob Bestgen  
Db2 for i Consultant  
bestgen@us.ibm.com



## SQL as a development language

SQL is a well established, standardized language for database access

SQL is also a programming language

- **SQL/PSM** (<https://en.wikipedia.org/wiki/SQL/PSM>) is a full procedural programming language
  - Silly quiz: what does PSM stand for?
- PSM enhances portability
  - Supported across Db2 Family
  - Similar to proprietary DBMS procedure languages (PL/SQL, T-SQL, etc...)

Makes it easier for SQL programmers to be productive faster on IBM i

## SQL Procedures

- A **procedure** is SQL's version of a program
- Similar to any other high-level language program
- Invoked with a CALL statement
- Supports parameters, both input and output
  - and Result Sets!
- Allows natural interaction between logic programming and database access in a single (SQL) language

## Procedure

- 1 Create it (one time, like a pgm)

```
CREATE OR REPLACE PROCEDURE total_val (IN Member# CHAR(6),  
                                         OUT total DECIMAL(12,2))
```

```
LANGUAGE SQL
```

```
BEGIN
```

```
CALL refresh_accounts;
```

```
SELECT SUM(curr_balance) INTO total  
FROM accounts
```

```
WHERE account_owner=Member# AND  
       account_type IN ('C','S','M');
```

```
END
```

- 2 CALL it (many times) from an SQL interface

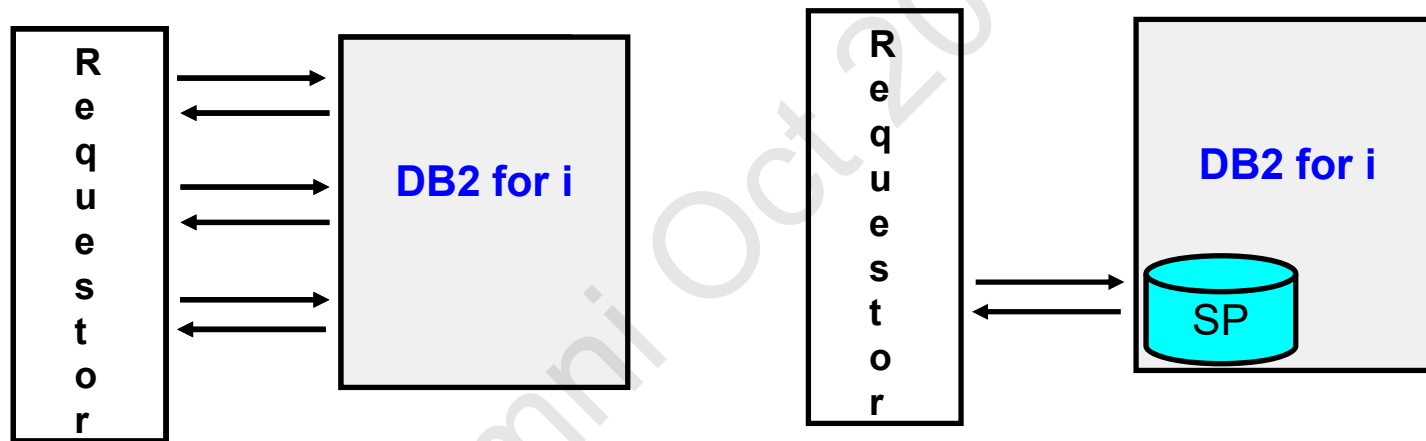
```
CALL total_val('135790', balance_var)
```

## Why use SQL procedures?

- Combines application logic with data access
  - One language, natural interaction
- Encapsulate complex logic
- Can leverage security control using owner adoption
- Provide underpinning behind service interfaces (like REST, microservices)
- Reduce data movement (network traffic)
- Enables (batch) programmatic processing of SQL
- Error handling/recovery for data access

## Stored Procedure to encapsulate

- Encapsulate complex logic into a single service
- Reduce traffic flow
  - One request initiates multiple transactions and processes



- Further enhance data flow and reduce work files by returning result sets

# General SQL Procedure layout

Compound  
Statement

```

CREATE PROCEDURE(parms...)
LANGUAGE SQL
<create options>
BEGIN
  <declare variables>
  <declare conditions>
  <declare cursors>
  <declare handlers>

  <program logic >
END
  
```

Declarations

Logic -

Can contain nested  
compound statements

# Basic Procedure (Language) Constructs

- DECLARE – define variables. Variables automatically initialized when procedure is called
- SET – assign a value to a variable or parameter
- SET OPTION – ‘compile’ options
- Comments - either `/* */` or `--`
- Logic statements
  - IF THEN ELSE END IF
  - CASE
- Looping constructs
  - FOR
  - LOOP
  - REPEAT
  - WHILE
- Error handling and feedback
  - CONDITIONS and HANDLERS
  - GET DIAGNOSTICS
  - SIGNAL and RESIGNAL
  - RETURN
- Result Sets



# SQL Procedure example



```
create or replace procedure exitcheck(out totalcount int)
language sql modifies sql data
begin
  declare pgmcount int default 0;
  declare exc_occurred int default 0;
  declare continue handler for sqlexception set exc_occurred = 1;

  set totalcount = 0; /* assume the best */
  Loop1: for Loop1 as c1 cursor for
  with exitlist (entry) as (
    values('QIBM_QDB_OPEN'),('QIBM_QSQ_CLI_CONNECT'))
  select entry from exitlist
  do
    if exc_occurred = 1 then leave Loop1; end if;
    set pgmcount = 0;
    call regcount(Loop1.entry,pgmcount);
    if pgmcount > 0 then
      set totalcount = totalcount + pgmcount;
    end if;
  end for;
  if exc_occurred = 1 then
    set totalcount = -1;
  end if;
end; /* exitcheck */
```

# Techniques to Consider

Omni Oct 2019

## Parameter passing options

Leverage default parameters to lessen the number of procedures needed

Example:

```
create or replace procedure trimfile
  (thelib char(10),
   trimdate date default (current date - 1 year))
language sql
begin
  declare d_sql varchar(3000);
  ...
  set d_sql = 'delete from ' concat thelib concat '.orders where order_date < ?';
  prepare stmt1 from d_sql;
  execute stmt1 using trimdate;
end;
```

Invoke as:

```
call trimfile('mylib', '10/01/2019')
```

or

```
call trimfile('mylib');
```

## Controlling procedure name

Procedures can have long, meaningful SQL names

- There can also be multiple procedures with the same (long) name in the same library, with different number of parameters
- Control corresponding pgm/srvpgm object name to help organize objects
  - Use **SPECIFIC**

Example (assuming the procedures do different things):

```
create or replace procedure my_meaningful_proc_name()  
language sql  
SPECIFIC MYPROC0 /* control name of object in IBM i library */  
...
```

```
create or replace procedure my_meaningful_proc_name(parm1 int)  
language sql  
SPECIFIC MYPROC1 /* control name of object in IBM i library */  
...
```

## Mixing static and dynamic

Using dynamic SQL within SQL procedures is very powerful

- Build statements based on input and environment situations
- Take advantage of dynamic's 'late binding'

**Static** – Things you know about during the procedure creation

**Dynamic** – to handle things that can vary

Use **prepare/execute** and **execute immediate** to drive dynamic

```
create or replace procedure trimfile(thelib char(10), trimdate date) language sql
begin
  declare d_sql varchar(3000);
  ...
  Set d_sql = 'delete from ' concat thelib concat '.orders where order_date < ?';
  prepare stmt1 from d_sql;
  execute stmt1 using trimdate;
end;
```

## Control compile - SET OPTION

The SET OPTION controls how the procedure is created. Common options:

- Compile for debugging

```
CREATE OR REPLACE PROCEDURE MYPROC(...)  
LANGUAGE SQL SET OPTION DBGVIEW = *STMT
```

**Note:** alternative is to use the **ALLOW DEBUG MODE** clause

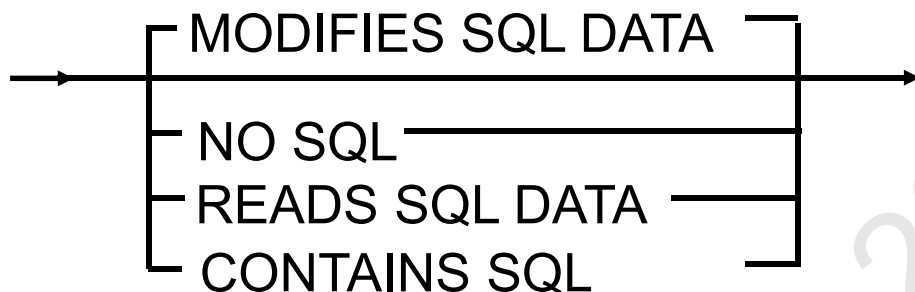
- Commitment control level
  - Specify lowest commit level to accomplish what you need

```
SET OPTION COMMIT = *NC
```

- Target release
  - Specify target release to help ensure program can run at the earliest necessary release.
  - Note this does not ensure dynamic SQL will run at that release!

```
SET OPTION TGTRLS = V7R1M0
```

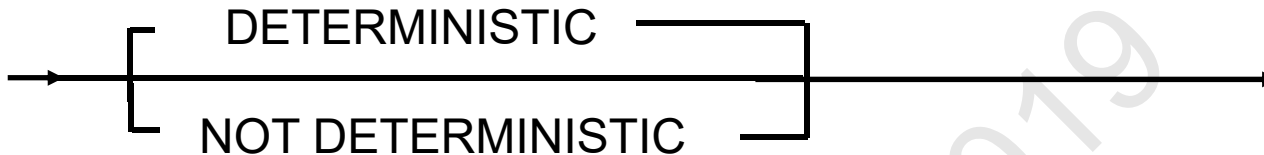
## Create Option – Data access allowed



- MODIFIES SQL DATA – Most any SQL statement allowed
- READS SQL DATA – Read Only statements
- CONTAINS SQL – Simple local statements (SET, DECLARE)
- NO SQL – No SQL allowed (external procedures only)

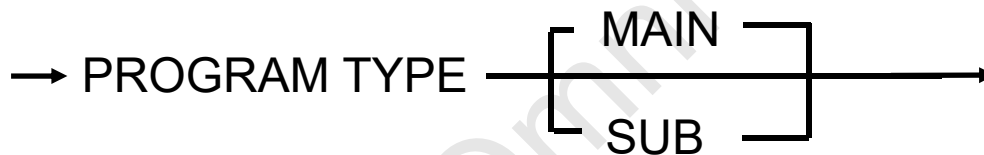
Note: Create routine at the 'lowest' option possible for your situation  
– Lowest to highest: NO, CONTAINS, READS, MODIFIES

# Create Options



- **DETERMINISTIC**

- procedure/function will always return the same result from successive calls with identical input arguments
- Best performing option, but make sure it is true!



- **PROGRAM TYPE**

- SUB: creates a service program object (better performance)
- MAIN: creates a program object



## Looping Constructs

For the most part, which looping construct to use is a developer's choice

- LOOP
- WHILE
- REPEAT

The FOR is a unique solution when processing the rows of a cursor result

- Saves steps of defining a separate cursor and variables
- Allows columns in SELECT statement to be accessed directly!
- Cursor can be used in WHERE CURRENT OF... operation
- Main disadvantage is that the table(s) must be known at create procedure time

Ex:

```
FOR loopvar AS loopcursor CURSOR FOR  
SELECT firstname, middinit, lastname FROM emptbl  
DO  
    SET fullname=lastname||', ' || firstname||' ' || middinit;  
    INSERT INTO namestbl VALUES( fullname );  
END FOR;
```

## Leverage DB2 for IBM i services

IBM i services are a great SQL way to get system related information

- Natural integration into an SQL procedure

```
create or replace procedure myproc(...)
language sql
begin
  declare exc_occurred int default 0;
  declare host_name varchar(257);
  declare server_ipaddr varchar(45);
  declare vrm varchar(10);
  declare continue handler for sqlexception set exc_occurred = 1;
  ...
  set (host_name,vrm, server_ipaddr) =
    (select local_host_name, host_version, server_ip_address
     from qsys2.tcpip_info);

  if vrm = 'V7R1M0' then /* special case to handle i7.1 */
    ...
  end if;
end;
```

<http://ibm.biz/DB2foriServices>

# Db2 for IBM i Services

- Complete listing found on IBM i developerWorks: <https://ibm.biz/Db2Services>
- Service objects found in QSYS2, unless otherwise noted

DB2 for i Service	Type of	IBM i 7.2	IBM i 7.1
<b>Work Management Services</b>			
<b>PTF Services</b>		<b>View</b>	<b>Storage Services</b>
QSYS2.PTF_INFO	QSYS2.GET_JOB_INFO()	UDTF	QSYS2.USER_STORAGE <b>View</b> Base SF99701 Level 26
QSYS2.GROUP_PTF_INFO			QSYS2.SYSTMPSTG <b>View</b> Base
SYSTOOLS.GROUP_PTF_CU			QSYS2.SYSDISKSTAT <b>View</b> Base SF99701 Level 12
SYSTOOLS.GROUP_PTF_DE	QSYS2.ACTIVE_JOB_INFO()	UDTF	QSYS2.MEDIA_LIBRARY_INFO <b>View</b> SF99702 Level 9 SF99701 Level 38
<b>Security Services</b>		<b>View</b>	<b>Product Services</b>
QSYS2.USER_INFO	QSYS2.SCHEDULED_JOB_INFO	UDTF	QSYS2.LICENSE_INFO <b>View</b> SF99702 Level 9 SF99701 Level 38
	QSYS2.MEMORY_POOL()	<b>View</b>	<b>Spool Services</b>
	QSYS2.MEMORY_POOL_INFO	UDTF	QSYS2.OUTPUT_QUEUE_ENTRIES() <b>UDTF</b> SF99702 Level 9 SF99701 Level 38
QSYS2.FUNCTION_INFO	QSYS2.SYSTEM_STATUS()	<b>View</b>	QSYS2.OUTPUT_QUEUE_ENTRIES <b>View</b> SF99702 Level 9 SF99701 Level 38
QSYS2.FUNCTION_USAGE	QSYS2.SYSTEM_STATUS_INFO	<b>View</b>	<b>System Health Services</b>
QSYS2.GROUP_PROFILE_EN	QSYS2.OBJECT_LOCK_INFO	<b>View</b>	QSYS2.SYSLIMTBL <b>Table</b> Introduced: Base Enhanced: SF99702 Level 3 Enhanced: SF99701 Level 26 Enhanced: SF99701 Level 34
QSYS2.SQL_CHECK_AUTHO	QSYS2.RECORD_LOCK_INFO	<b>View</b>	QSYS2.SYSLIMITS <b>View</b> Introduced: Base Enhanced: SF99702 Level 3 Enhanced: SF99701 Level 26 Enhanced: SF99701 Level 34
QSYS2.SET_COLUMN_ATTR	<b>Communication Services</b>		
QSYS2.DRDA_AUTHENTICAT	SYSIBMADM.ENV_SYS_INFO	<b>View</b>	
<b>Message Handling Services</b>	QSYS2.TCPIP_INFO	<b>View</b>	<b>Journal Services</b>
QSYS2.REPLY_LIST_INFO	QSYS2.SET_SERVER_SBS_ROUTING()	Procedure	QSYS2.JOURNAL_INFO <b>View</b> SF99702 Level 3 SF99701 Level 32
QSYS2.JOBLOG_INFO	QSYS2.SERVER_SBS_ROUTING	<b>View</b>	QSYS2.DISPLAY_JOURNAL() <b>UDTF</b> Base Introduced: Base Enhanced: SF99701 Level 26
<b>Librarian Services</b>		<b>View</b>	<b>Java Services</b>
QSYS2.LIBRARY_LIST_INFO	QSYS2.NETSTAT_INFO	<b>View</b>	QSYS2.SET_JVM() <b>Procedure</b> SF99702 Level 5 SF99701 Level 34
QSYS2.OBJECT_STATISTICS	QSYS2.NETSTAT_INTERFACE_INFO	<b>View</b>	QSYS2.JVM_INFO <b>View</b> SF99702 Level 5 SF99701 Level 34
	QSYS2.NETSTAT_JOB_INFO	<b>View</b>	<b>Application Services</b>
	QSYS2.NETSTAT_ROUTE_INFO	<b>View</b>	QSYS2.QCMDEXC() <b>Procedure</b> Base Introduced: Base Enhanced: SF99701 Level 26

## Use global variables

Global variables provide several advantages

- communicate across procedures
- define 'configuration' or default values
- 'catch' OUTPUT procedure parameters for testing

**Example.** Use global variable `mylib.verbose` to control whether to dump out trace information

```
create or replace procedure myproc(...)  
language sql  
begin  
  declare exc_occurred int default 0;  
  ... handlers here for expected errors...  
  ...  
  if mylib.verbose = 1 then  
    ... dump out trace type information here to aid debugging  
  end if;  
end;
```

## Calling out

It is sometimes necessary to do a call outside of SQL

- Leverage an OS command
- Get information from outside SQL

**QSYS2.QCMDXEC** is a good way to do these 'outside' calls

```
create or replace procedure myproc(...)
language sql
begin
...
CALL QSYS2.QCMDXEC(
  'QSYS/SBMJOB CMD(QSYS/RUNSQL SQL("CALL QGPL.DOSOMETHING()") '
  CONCAT ' COMMIT(*NONE) NAMING(*SQL))');    /* submit job to do work */

CALL QSYS2.QCMDXEC('QSYS/DLYJOB DLY(5)'); /* wait for job to get going... */
...
end;
```

## Speaking of CL interaction

- Interaction between native (CL) and SQL is limited when SQL is driving things
  - Errors tend to be generic SQL0443 message
  - often not an issue if simple error handling is sufficient
- If detailed handling is important, you could write your own wrapper to invoke CL
  - Or use an SQL Service against the joblog messages

Example:

```
create or replace procedure myproc() language sql
begin
  declare underlying_error char(7);
  declare exc_occurred int default 0;
  declare continue handler for sqlexception set exc_occurred = 1;
  CALL QSYS2.QCMDEXC('QSYS/SBMJOB ...') /* submit job to do work */
  if exc_occurred = 1 then
    set underlying_error = (select message_id from
                           table(QSYS2.JOBLOG_INFO('*')) x
                           where ordinal_position = (
                               select max(ordinal_position) - 2
                               from table(QSYS2.JOBLOG_INFO('*')) x
                               where message_id = 'SQL0443')
                           -- do some acknowledgement of the error....
    end if;
end;
```

## Running with elevated authority

In some cases it is advantageous to run a procedure with elevated authority

- when invoked, procedure runs under a higher (pgm owner's) authority

Authority to call procedure is usually restricted

Note: adopted authority does not work for IFS files

Use SET statement with USRPRF option:

```
CREATE OR REPLACE PROCEDURE myproc(...)  
LANGUAGE SQL  
SET OPTION DYNUSRPRF = *OWNER, USRPRF = *OWNER
```

## Feedback and Error Handling

Procedures can leverage a rich set of error and message handling capabilities

- GET DIAGNOSTICS
- SQLSTATE and SQLCODE variables
- CONDITIONS and HANDLERS
- SIGNAL and RESIGNAL
- RETURN statement

Omni Oct 2019



## Feedback & Error Handling

- GET DIAGNOSTICS
  - Retrieve information about last statement executed
    - Row\_count, return\_status, error status....
  - CURRENT or STACKED
    - CURRENT – statement that was just executed
    - STACKED – statement before error handler was entered
      - Only allowed within error handler

Example:

```
DECLARE update_counter INTEGER;  
...  
UPDATE orders SET status='LATE'  
WHERE ship_date < CURRENT DATE;  
GET DIAGNOSTICS update_counter = ROW_COUNT;  
...
```

## Error handling

Every non-trivial procedure, practically speaking, needs error handling

- **DECLARE ... HANDLER FOR...**

Get use to having a handler in your 'template' procedure to encourage usage

- Use **CONDITIONs** to convey meaning for cryptic **SQLSTATE**

```
create or replace procedure myproc(...)
language sql
begin
  declare already_exists int default 0;
  declare alreadyexists condition for '42710';
  declare dupekey condition for '23505';
  declare continue handler for alreadyexists set already_exists=1;
  declare continue handler for dupekey set already_exists=1;
  ...
end;
```

## General exception handler

It is often useful to have a general exception handler to catch any unexpected errors so as to allow the procedure to finish gracefully

SQL provides a built-in condition called SQLEXCEPTION

```
create or replace procedure myproc(...)
language sql
begin
  declare exc_occurred int default 0;
  ... other handlers here for expected errors...
  DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET exc_occurred = 1;
  ...
  IF exc_occurred = 1 then
    -- do some acknowledgement of the error....
  END IF;
end;
```

## Returning result sets

Result sets are a unique capability for procedures

- allow an answer set(s) to be returned from the CALL
- Consolidate complex processing for determining an answer set under one CALL
- Communicate much more information back than just a reason code

Example:

```
create or replace stop_processing()
  dynamic result sets 1 language sql modifies sql data
begin
  declare status varchar(50);
  declare total_job_count, job_count int default 0;

  declare cursor1 cursor with return for
    with cte("status", "jobs found", "jobs ended") as
      (values(status, total_job_count, job_count))
    select * from cte;

  call killjobs(total_job_count, job_count); /* find and kill jobs */
  set status = case when total_job_count = 0 then 'no jobs found'
    when total_job_count > job_count then 'not all jobs ended'
    else 'success. All jobs ended' end;

  open cursor1 ; -- cursor left open for client application
end;
```

# Constructing a result set

Result sets are a good way to return (conditional) data as a service

Example: return a list of vehicles from optional input filters

```
create or replace procedure get_car_list
(in g_make varchar(20) default ""
, in g_model varchar(20) default "", in g_year int default 0)
dynamic result sets 1 language sql modifies sql data
begin
declare d_sql varchar(1000);
declare w_clause varchar(500) default ' where';
declare have_where int default 0;
declare cursor1 cursor with return for statement1;

if g_make <> "" then
set w_clause = w_clause concat ' make = ?';
set have_where = 1;
else
set w_clause = w_clause concat ' "" = ?';
set g_make = ""; /* make sure not null */
end if;
if g_model <> "" then
set w_clause = w_clause concat ' and model = ?';
set have_where = 1;
else
set w_clause = w_clause concat ' and "" = ?';
set g_model = ""; /* make sure not null */
end if;
```

```
if g_year > 0 then
set w_clause = w_clause concat ' and year >= ?';
set have_where = 1;
else
set w_clause = w_clause concat ' and 0 = ?';
set g_year = 0; /* make sure not null */
end if;

set d_sql =
'SELECT make, model, year, color, style FROM vehicles'
concat case when have_where <> 0 then w_clause
else "" end;

prepare statement1 from d_sql;
/* open cursor can have 'extra' variables */
open cursor1 using g_make, g_model, g_year;
-- cursor left open for client application
end;
```

# Result Set Considerations

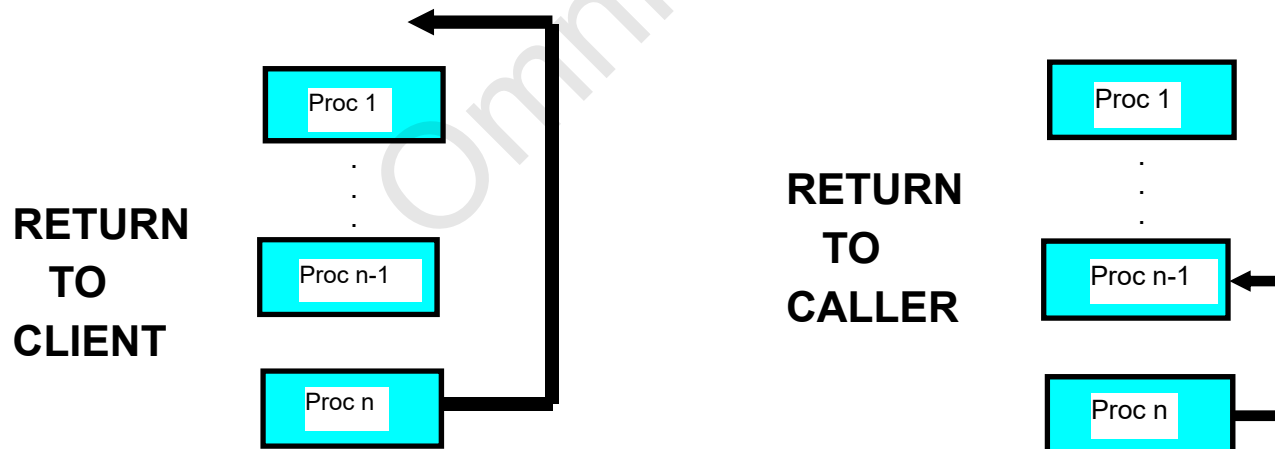
## Result Set Consumer Control

- RETURN TO CLIENT

Ex: DECLARE c1 CURSOR WITH RETURN TO CLIENT FOR SELECT \* FROM t1

- RETURN TO CALLER

Ex: DECLARE c1 CURSOR WITH RETURN TO CALLER FOR SELECT \* FROM t1



## Result Set Consumption

- Consume result sets from one procedure inside another
  - ASSOCIATE LOCATOR & ALLOCATE CURSOR statements

```

...
DECLARE sprs1 RESULT_SET_LOCATOR VARYING;
CALL GetProjs(projdept);
ASSOCIATE LOCATOR (sprs1) WITH PROCEDURE GetProjs;
ALLOCATE mycur CURSOR FOR RESULT SET sprs1;
SET totstaff=0;
myloop: LOOP
  FETCH mycur INTO pname, prstaff;

  IF row_not_found=1 THEN
    LEAVE fetch_loop;
  END IF;
  SET totstaff= totstaff + prstaff;
  IF prstaff > moststaff THEN
    SET bigproj = pname;
    SET moststaff= prstaff;
  END IF;
END LOOP;
CLOSE mycur;
...

```

**Consume  
Result Set!**

PROJNAME	PRSTAFF
GENERAL ADMIN ...	6.00
PAYROLL PROGR...	2.00
PERSONNEL PRO...	1.00
ACCOUNT PROGR...	2.00

\*\* DESCRIBE PROCEDURE &  
DESCRIBE CURSOR statements  
can be used to dynamically  
determine the number and  
contents of a result set

## More Information

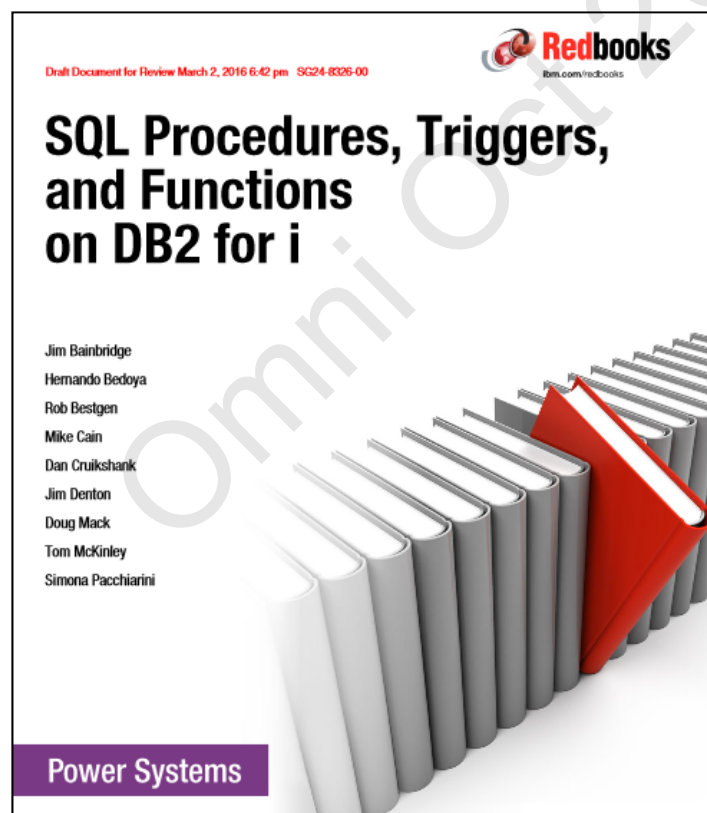
Omni Oct 2019



# The Full Story

RedBook available

<http://www.redbooks.ibm.com/redpieces/abstracts/sg248326.html>



## Additional Information

- **DB2 for i Websites**

- Homepage: [www.ibm.com/systems/power/software/i](http://www.ibm.com/systems/power/software/i)
- Technology Updates  
[www.ibm.com/developerworks/ibmi/techupdates/db2](http://www.ibm.com/developerworks/ibmi/techupdates/db2)
- developerWorks Zone: [www.ibm.com/developerworks/data/products.html](http://www.ibm.com/developerworks/data/products.html)

- **Forums**

- developerWorks:  
<https://ibm.com/developerworks/forums/forum.jspa?forumID=292>

- Articles on procedure resolution related to default parameters

- <http://www.ibm.com/developerworks/ibmi/library/i-sqlnaming/index.html>
- [http://www.ibm.com/developerworks/ibmi/library/i-system\\_sql2/index.html](http://www.ibm.com/developerworks/ibmi/library/i-system_sql2/index.html)

- **Customized consulting, education, architecture and design reviews**
  - **Advanced SQL and Datacentric Programming**
  - **SQL Performance Best Practices, Monitoring and Tuning**
  - **RCAC**
  - **...**
- **Consulting on any Db2 for i topic!**

For more information, contact [mackd@us.ibm.com](mailto:mackd@us.ibm.com)





## Special notices

This document was developed for IBM offerings in the United States as of the date of publication. IBM may not make these offerings available in other countries, and the information is subject to change without notice. Consult your local IBM business contact for information on the IBM offerings available in your area.

Information in this document concerning non-IBM products was obtained from the suppliers of these products or other public sources. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. Send license inquires, in writing, to IBM Director of Licensing, IBM Corporation, New Castle Drive, Armonk, NY 10504-1785 USA.

All statements regarding IBM future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

The information contained in this document has not been submitted to any formal IBM test and is provided "AS IS" with no warranties or guarantees either expressed or implied.

All examples cited or described in this document are presented as illustrations of the manner in which some IBM products can be used and the results that may be achieved. Actual environmental costs and performance characteristics will vary depending on individual client configurations and conditions.

IBM Global Financing offerings are provided through IBM Credit Corporation in the United States and other IBM subsidiaries and divisions worldwide to qualified commercial and government clients. Rates are based on a client's credit rating, financing terms, offering type, equipment type and options, and may vary by country. Other restrictions may apply. Rates and offerings are subject to change, extension or withdrawal without notice.

IBM is not responsible for printing errors in this document that result in pricing or information inaccuracies.

All prices shown are IBM's United States suggested list prices and are subject to change without notice; reseller prices may vary.

IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply.

Any performance data contained in this document was determined in a controlled environment. Actual results may vary significantly and are dependent on many factors including system hardware configuration and software design and configuration. Some measurements quoted in this document may have been made on development-level systems. There is no guarantee these measurements will be the same on generally-available systems. Some measurements quoted in this document may have been estimated through extrapolation. Users of this document should verify the applicable data for their specific environment.

Revised September 26, 2006



## Special notices (cont.)

IBM, the IBM logo, ibm.com AIX, AIX (logo), AIX 5L, AIX 6 (logo), AS/400, BladeCenter, Blue Gene, ClusterProven, DB2, ESCON, i5/OS, i5/OS (logo), IBM Business Partner (logo), IntelliStation, LoadLeveler, Lotus, Lotus Notes, Notes, Operating System/400, OS/400, PartnerLink, PartnerWorld, PowerPC, pSeries, Rational, RISC System/6000, RS/6000, THINK, Tivoli, Tivoli (logo), Tivoli Management Environment, WebSphere, xSeries, z/OS, zSeries, Active Memory, Balanced Warehouse, CacheFlow, Cool Blue, IBM Systems Director VMControl, pureScale, TurboCore, Chiphopper, Cloudscape, DB2 Universal Database, DS4000, DS6000, DS8000, EnergyScale, Enterprise Workload Manager, General Parallel File System, , GPFS, HACMP, HACMP/6000, HASM, IBM Systems Director Active Energy Manager, iSeries, Micro-Partitioning, POWER, PowerExecutive, PowerVM, PowerVM (logo), PowerHA, Power Architecture, Power Everywhere, Power Family, POWER Hypervisor, Power Systems, Power Systems (logo), Power Systems Software, Power Systems Software (logo), POWER2, POWER3, POWER4, POWER4+, POWER5, POWER5+, POWER6, POWER6+, POWER7, System i, System p, System p5, System Storage, System z, TME 10, Workload Partitions Manager and X-Architecture are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries.

A full list of U.S. trademarks owned by IBM may be found at: <http://www.ibm.com/legal/copytrade.shtml>.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

AltiVec is a trademark of Freescale Semiconductor, Inc.

AMD Opteron is a trademark of Advanced Micro Devices, Inc.

InfiniBand, InfiniBand Trade Association and the InfiniBand design marks are trademarks and/or service marks of the InfiniBand Trade Association.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency which is now part of the Office of Government Commerce.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Linear Tape-Open, LTO, the LTO Logo, Ultrium, and the Ultrium logo are trademarks of HP, IBM Corp. and Quantum in the U.S. and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries or both.

Microsoft, Windows and the Windows logo are registered trademarks of Microsoft Corporation in the United States, other countries or both.

NetBench is a registered trademark of Ziff Davis Media in the United States, other countries or both.

SPECint, SPECfp, SPECjbb, SPECweb, SPECjAppServer, SPEC OMP, SPECviewperf, SPECcapc, SPECchpc, SPECjvm, SPECmail, SPECimap and SPECsfs are trademarks of the Standard Performance Evaluation Corp (SPEC).

The Power Architecture and Power.org wordmarks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org. TPC-C and TPC-H are trademarks of the Transaction Performance Processing Council (TPPC).

UNIX is a registered trademark of The Open Group in the United States, other countries or both.

Revised December 2, 2010

Other company, product and service names may be trademarks or service marks of others.

© 2019 IBM Corporation