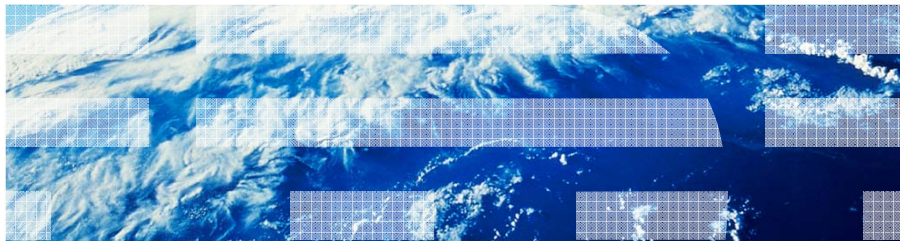




Using Rational Open Access: RPG Edition to Access SQL Functions

Dan Cruikshank dcrank@us.ibm.com




stols@us.ibm.com
ibm.com/systems/services/labservices

© 2011 IBM Corporation




Agenda

- Rational Open Access Overview
- Enhancing Performance with SQL
- Building a Handler
- Accessing data using SQL Blocked FETCH

IBM Systems Lab Services and Training 

Rational Open Access Overview

3 IBM Systems Lab Services and Training – ibm.com/systems/services/labservices © 2011 IBM Corporation

IBM Systems Lab Services and Training 

Rational Open Access: RPG Edition

- Provides a way for RPG programmers to use the simple and well-understood RPG I/O model to access SQL procedures used by SQL based languages.
 - Everyone is now playing by the same rules
- Open Access opens up RPG's file I/O capabilities allowing HLL programmers to write innovative I/O handlers that:
 - Transform traditional record at a time I/O operations to SQL set based operations
 - Take advantage of data centric programming techniques
 - RI
 - Auto-generated values
 - Advanced embedded SQL programming techniques

4 IBM Systems Lab Services and Training – ibm.com/systems/services/labservices © 2011 IBM Corporation

Open Access Structure

- An Open Access application has three parts:
 - An RPG program that uses normal RPG coding to define an Open Access file and use I/O operations against the file.
 - A handler procedure or program that is called by Open Access to handle the I/O operations for the file.
 - The data access service program that the handler is using or communicating with.
- Open Access is the linkage between parts 1 and 2.
- Licensed program 5733-OAR is required to use Open Access at runtime.
 - Fee based
 - Announced for 7.1, PTF'ed to 6.1
 - 6.1 PTFs: SI39480, SI39914

RPG HANDLER Keyword

```
FSOMEFILE IF E      DISK
```

```
F          handler('HANDLER_W1')
```

```
F          ExtDesc('EMPLOYEE')
```

```
F          USROPN
```

```
/Free
```

```
Open(e) Surrogate;
```

```
READ(e) Surrogate;
```

```
Close(e) Surrogate;
```

```
Return;
```

```
/End-Free
```

1 Line
of code per
file

RPG IO operations
coded as usual

IBM Systems Lab Services and Training IBM

The HANDLER Program

```

D/COPY QOAR/QRPGLESRC,QRNOOPENACC
D HANDLER_W1 PR
D Inp_Ds...
D HANDLER_W1 PI LIKEDS(QrnOpenAccess_T
D Inp_Ds...
D LIKEDS(QrnOpenAccess_T)

P Declare_SQL_VIEW_INT1...
P E
P OpenCursor_SQL_VIEW_INT1...
P E
P FetchFirstFrom_SQL_VIEW_INT1...
P B EXPORT
P FetchNextFrom_SQL_VIEW_INT1...
P E
P CloseCursor_SQL_VIEW_INT1...
P E
P Prepare_SQL_VIEW_INT1...
P E
P Return_A_Row_From_An_Array...
P E

/FREE
//Your code here
return;
/END-FREE
    
```

Data structure template provided in QOAR.

Data structure passed by IBM I to handler program

Determine SQL function based on I/O operation

7 IBM Systems Lab Services and Training – ibm.com/systems/services/labservices © 2011 IBM Corporation

IBM Systems Lab Services and Training IBM


Open Access Data Structure (QRNOOPENACC) Layout

Subfield	Type	Set by	Used by
structLen	UBIN4	RPG	Handler
parameterFormat	CHAR(8)	RPG	Handler
userArea	Pointer 2	RPG	Handler and RPG programmer
stateInfo	Pointer 4	Handler	Handler
recordLevels 1	Pointer 2	RPG	Handler
inputBuffer	Pointer 2,3	Handler	RPG
inputNullMap 1	Pointer 2,3	Handler	RPG
outputBuffer	Pointer 2,3	RPG	Handler
outputNullMap 1	Pointer 2,3	RPG	Handler
namesValues 1	Pointer 2,3	RPG and Handler	RPG and Handler

Passed automatically by IBM I
Provided as /COPY member


Subfield	Type	Set by	Used by
keyNullMap 1	Pointer 2,3	RPG	Handler
keyNamesValues 1	Pointer 2,3	RPG	Handler
Indara	Pointer 2,3	RPG and Handler	RPG and Handler
Prtctl	Pointer 2,3	RPG and Handler	RPG and Handler
openFeedback	Pointer 4	Handler	RPG
ioFeedback	Pointer 4	Handler	RPG
deviceFeedback	Pointer 4	Handler	RPG
externalFile	QrnObject_T	RPG	Handler
externalMember	CHAR(10)	RPG	Handler
compileFile 1	QrnObject_T	RPG	Handler
recordName 1	CHAR(10)	RPG and Handler	RPG and Handler
rpgOperation	UINT(4)	RPG	Handler
rpgStatus	INT(4)	Handler	RPG
inputBufferLen	UINT(4)	RPG	Handler
inputNullMapLen 1	UINT(4)	RPG	Handler
outputBufferLen	UINT(4)	RPG	Handler
outputNullMapLen 1	UINT(4)	RPG	Handler
keyLen	UINT(4)	RPG	Handler
keyNullMapLen 1	UINT(4)	RPG	Handler
inputDataLen	UINT(4)	Handler	RPG
openFeedBackLen	UINT(4)	Handler	RPG
ioFeedBackLen	UINT(4)	Handler	RPG
deviceFeedBackLen	UINT(4)	Handler	RPG
numKeys 1	UINT(4)	RPG	Handler
Rrn	UINT(4)	RPG and Handler	RPG and Handler
formLen	UINT(4)	RPG	Handler
formOff	UINT(4)	RPG	Handler
functionKey	UINT(1)	Handler	RPG

8 IBM Systems Lab Services and Training – ibm.com/systems/services/labservices © 2011 IBM Corporation

IBM Systems Lab Services and Training 

Enhancing Performance with SQL

9 IBM Systems Lab Services and Training – ibm.com/systems/services/labservices © 2011 IBM Corporation

IBM Systems Lab Services and Training 

So Why SQL?

- SQL set processing best for reading/writing large amounts of data
 - Automatic input/output blocking, searched updates/deletes, etc.
- Take advantage of SQL functions not available to HLL such as:
 - Index Only Access
 - Advanced indexing support
 - Expression based indexing
 - Fast count and aggregate support via Encoded Vector Index(EVI)
 - Fast summary access via Materialized Query Table (MQT)
- Advanced join technology -Look ahead Predicate Generation (LPG)
- CPU parallelism via Symmetric Multiprocessing (SMP)

10 IBM Systems Lab Services and Training – ibm.com/systems/services/labservices © 2011 IBM Corporation

IBM Systems Lab Services and Training IBM

DB2 enhancements not available to HLL

- The SQL Query Engine (SQE) is a total rewrite of the original or Classic Query Engine (CQE)
 - This includes new data access primitives
- Only HLL programs using embedded SQL will have access to the new technology

Brand New This Century

The Optimizer Exposed

11 IBM Systems Lab Services and Training – ibm.com/systems/services/labservices © 2011 IBM Corporation

IBM Systems Lab Services and Training IBM

Areas providing best return using Open Access

- SQL blocked FETCH versus RPG READE loop
- SQL blocked INSERT versus RPG WRITE loop
- SQL searched UPDATE/DELETE versus RPG UPDATE/DELETE loop
- SQL JOIN versus RPG READE then RPG CHAIN
- SQL MERGE versus HLL Archive/Purge


12 IBM Systems Lab Services and Training – ibm.com/systems/services/labservices © 2011 IBM Corporation

READE and RPG BLOCK(*YES *NO *IT DEPENDS)

- BLOCK(*YES) Allows compiler generated blocking for input only files
 - Positioned by SETGT, SETLL or CHAIN
 - Processed via RPG **READ** operations
 - OVRDBF SEQONLY(*YES n) required to increase blocking factor
- If any READE or READPE operations are used, **NO** record blocking will occur for the input file (as stated in the RPG Manual)
 - If BLOCK(*YES) is specified the RPG compiler will issue a severity 10 error indicating blocking is not allowed
- The compiler will generate blocking code for output only files, however:
 - IBM i OS may turn off blocking at file OPEN
 - A message will be logged stating the output only ODP has been changed to SEQONLY(***NO**)
- SQL Read/Write only cursors are always blocked at most efficient blocking factor per release


Reasons for Using Embedded SQL

- So why not just use SQL Stored Procedures?
- Take advantage of HLL programming constructs not available to DB2 SQL Procedure Language (DB2 SQL PL)
 - Arrays
 - Indicator variables
 - Externally described data structures
 - Blocked FETCH and/or INSERT support
 - Built-in functions not available in DB2

IBM Systems Lab Services and Training 

Building a Handler

15 IBM Systems Lab Services and Training – ibm.com/systems/services/labservices © 2011 IBM Corporation

IBM Systems Lab Services and Training 

Methods for Passing Data To and From a Handler

- Two methods available
 - Structure-based
 - Column-based
- Structure-based
 - Data and Key values passed using buffers
 - Field names not provided
 - Buffers can be externally described
 - Good for getting feet wet
- Column-based
 - Data and key values passed as individual items
 - Each item contains column name, data and key attributes
 - More advanced capabilities
- Dynamic SQL required in either case to minimize number of handlers

16 IBM Systems Lab Services and Training – ibm.com/systems/services/labservices © 2011 IBM Corporation

Dynamic SQL Concepts

- Statement is defined and executed at program run time
 - PREPARE then DECLARE dynamic cursor for SELECT
 - PREPARE then EXECUTE for UPDATE, DELETE
 - EXECUTE IMMEDIATE for DDL statements (ALIAS, CREATE INDEX, etc)
- Two types of dynamic SELECT statements
 - Fixed list – result columns are predictable and can be pre-defined
 - Varying list – result columns can vary, SQLDA required to define host variables
- Dynamic SQL required for generic handlers
 - Format, statement or procedure based

Generic Database Handlers

- Format-based
 - Best for reengineering startup and structure-based method
 - Handler program uses template based on physical file format
 - Data access module utilizes fixed-list dynamic SQL
- Statement-based
 - Best for:
 - Column-based method
 - reengineering multiple distinct format LFs to use a single handler
 - Data access service module utilizes varying-list dynamic SQL
- Procedure-based
 - Intended for 7.1 and beyond
 - Best for result set consumption and “one and done” type functions
 - E.g. SQL MERGE, ROLLUP, CUBE, etc.

IBM Systems Lab Services and Training IBM

Using a Fixed-List Handler

- Used with externally described data structures or templates
 - Record format in program is fixed
- Row selection and ordering are dynamic
 - Handler implicitly determines WHERE and ORDER BY from external file attributes
 - DDS Select/Omit criteria is implicitly converted to WHERE
 - System API required to retrieve key columns
 - E.g. file uses K8, K1, K2. K8 is for SETLL and K1 and K2 for ordering
- User area provided for passing explicit WHERE or ORDER BY clauses
 - Must code logic in handler to close existing cursor then prepare, declare and open new cursor
- An SQL Descriptor is not required for fixed list dynamic SQL
 - Data is written to in one operation

19 IBM Systems Lab Services and Training – ibm.com/systems/services/labservices © 2011 IBM Corporation

IBM Systems Lab Services and Training IBM

Handling RPG Using Fixed-List Handler

```

FEMPADDRSL1UF E K DISK
F handler('FMTHANDLER')
    
```

RPG Implicit OPEN

```

FMTHANDLER
    HANDLE_OPEN → PREPARE_SQL_STATEMENT
    HANDLE_CHAIN → FETCH_FIRST_FROM_OPEN_SQL_CURSOR
    HANDLE_CLOSE
    
```

```

FrcdFile_t IF E K DISK TEMPLATE
F EXTDESC('EMPADDRESS')
F RENAME(empAddr:rcdFormat)

D rcdFormat_t...
D DS LIKERECD(rcdFormat)
D TEMPLATE

D keys_t DS
D LIKERECD(rcdFormat:*KEY)
D TEMPLATE

D Ind_Array_t...
D S 5i 0 DIM(7)
D TEMPLATE
    
```

20 IBM Systems Lab Services and Training – ibm.com/systems/services/labservices © 2011 IBM Corporation

Handler Considerations

- Record and Key Data
 - 2 Modes: Name values or Data structures
 - Name values only available for externally described data
 - Data structures map directly to IO buffers
- Establish IO Feedback and stateInfo sizes
 - IO Feedback for returning error
 - stateInfo for tracking where you are
- Not all data is available for SQL statement
 - May need to wait until specific IO operation occurs before constructing and executing statement
 - Utilize userArea and stateInfo parameters to build as you go
- Handler can be program or service program

Record and Key Data Handling

- Name values
 - Each data item is passed with attribute information
 - Field names, size, actual data
- Data structures
 - Data is passed as a record
 - Information about the data must be provided by handler
- Keep it simple
 - useNameValues = '1' can be very complex
 - Ideal for varying-list dynamic SQL programs
 - useNameValues = '0'
 - Ideal for mere mortals
 - Data structures can be mapped to external file definitions
 - Will require multiple handlers – 1 per file format

Handler Program vs Handler Service Program

▪ Handler Program

- Advantages
 - Single program can be generic
 - Record format specific modules can be bound to generic handler
 - *INZSR can be utilized for one time only operations
- Disadvantages
 - Requires extra coding for different open scenarios

▪ Handler Service Program

- Advantages
 - Multiple entry points can be used for different open scenarios
 - Record format specific modules can be bound to generic handler
- Disadvantages
 - *INZSR cannot be used

Database Operation Type Constants

- The RPG IO operation is passed as an unsigned integer
- The RPG OA provided copy book (QRNOPENACC) contains constants mapped to the possible values

	Constant	Example
D	QrnOperation_OPEN...	
D	C	1
D	QrnOperation_READ...	
D	C	4
D	QrnOperation_READE...	
D	C	6
D	QrnOperation_CHAIN...	
D	C	9
D	QrnOperation_SETLL...	
D	C	12
D	QrnOperation_UPDATE...	
D	C	14
D	QrnOperation_WRITE...	
D	C	15
D	QrnOperation_DELETE...	
D	C	16
D	QrnOperation_CLOSE...	
D	C	18

Processing RPG IO Operations

- An rpgStatus value of zero indicates success
 - Any valid IO error status code can be used to signal an error
- Coding tip:
 - Create local handler procedures for each RPG IO Operation
 - Use rpgStatus as the return value
 - For handler programs use return with *LR off or the MAIN keyword (6.1)
 - Not an issue if using service programs

Example Code

```
QQA_Ds.rpgStatus = *Zero;

select;
when QQA_Ds.rpgOperation = QrnOperation_OPEN;
  QQA_Ds.rpgStatus = Handle_Open();
when QQA_Ds.rpgOperation = QrnOperation_CHAIN;
when QQA_Ds.rpgOperation = QrnOperation_READ;
when QQA_Ds.rpgOperation = QrnOperation_READE;
when QQA_Ds.rpgOperation = QrnOperation_UPDATE;
when QQA_Ds.rpgOperation = QrnOperation_DELETE;
when QQA_Ds.rpgOperation = QrnOperation_WRITE;
when QQA_Ds.rpgOperation = QrnOperation_CLOSE;
  DeAlloc QQA_Ds.stateInfo;
Other;
ENDSL;
return;
```

User Defined Parameters

- userArea and stateInfo
 - Pointers to user defined variables
 - Typically data structures
- userArea
 - Use to pass additional information to handler from RPG program
 - For example, dynamic SQL statement
- stateInfo
 - Use to track program state information
 - For example, last operation successfully executed



Sample User Defined Structures

- Use this data structure to pass additional parameters to further enhance RPG ops.
- For example
 - Open type, blocking factor, indicator array, etc.
- Use this data structure to pass the state information about a given file or operation
- For example
 - File name, last successful op, etc

```

userArea
//userArea Data structure
template
D Optional_Parms...
D           Ds
D Open_attr...
D Read_attr...
D Update_attr...
D Delete_attr...
    
```

```

stateInfo
//stateInfo data structure
template
D stateInfo_parms...
D           Ds
D Last_Op...
D                               25 Varying
    
```



Accessing data using SQL Blocked FETCH

Using an SQL Cursor

- SQL Statements used with a Dynamic SQL Cursor
 1. PREPARE the SQL statement from a variable containing the SQL SELECT statement
 2. DECLARE an SQL cursor from the prepared statement
 3. OPEN the declared SQL cursor passing parameters (if any)
 4. FETCH from the open SQL cursor into a host structure
 5. Process the fetched row/rows
 6. CLOSE the open SQL cursor
- SQL FETCH can return 1 to 32767 rows

Multiple Row FETCH (a.k.a Blocked FETCH)

- Based on host structure
 - RPG - Multiple Occurrence Data Structure or Array
 - COBOL - Occurs clause on declaration of the group item
- Clause on FETCH
 - FOR n ROWS (n = number of rows to be returned)
- Specify number of rows to be retrieved to fill the structure

▪ Example

```
D Result_set      DS              OCCURS(10)
D empnbr          LIKE(empno)
D firstname      LIKE(firstname)
D dpt            LIKE(workdept)
```

```
D v_row_count     S              31P 0
```

```
/FREE
```

```
EXEC SQL
  FETCH FROM Employee_Cursor
  FOR 10 ROWS INTO Result_set;
```

```
EXEC SQL
  GET DIAGNOSTICS :v_Row_Count = ROW_COUNT;
```

```
/END-FREE
```

Number of rows can be 1 to maximum occurrence of data structure

GET DIAGNOSTIC returns actual number of rows fetched

IBM Systems Lab Services and Training IBM

Handling RPG Open

FEMPADDRSL1UF E K DISK
F handler('UPDHANDLER')

RPG Implicit OPEN

'UPDHANDLER'

HANDLE_OPEN

PREPARE_SQL_STATEMENT

```
select;
when
rpgIO.rpgOperation =
QrnOperation_OPEN;
  rpgIO.rpgStatus =
  Handle_Open(rpgIO);
```

```
If rpgIO.externalFile.library = '*LIBL';
  TableReference = rpgIO.externalFile.name;
Else;
  TableReference =
  %TRIMR(rpgIO.externalFile.library) + '/' +
  %TRIMR(rpgIO.externalFile.name);
ENDIF;
SQLString =
'SELECT * FROM ' +
%TRIMR(TableReference) + '
WHERE WORKDEPT = ? FOR FETCH
ONLY';
IF Prepare_SQL_Statement(SQLString) =
*On; retField = 1299;
ENDIF;
```

```
EXEC SQL Prepare S1 FROM
:v_SQL_String;
If SqlState = '00000';
EXEC SQL
  Declare C1
  SCROLLCURSOR FOR S1;
  If SqlState <> '00000';
    RetField = *ON;
  EndIf;
Else;
  RetField = *ON;
EndIf;
```

31 IBM Systems Lab Services and Training – ibm.com/systems/services/labservices © 2011 IBM Corporation

IBM Systems Lab Services and Training IBM

SQL FETCH Processing

- The SQL format based service program defines the host array based on externalFile
- The host array is populated by the SQL FETCH function on first call
 - The first array occurrence is moved to the inputBuffer
 - Subsequent calls result in the occurrence being incremented and the next row is returned from the array
 - This continues until all occurrences have been returned.
 - The next call results in another FETCH

RPG Read Loop

```
For Times = 1 to rows_requested;
  READ(e) INPRECORD;
  If %EOF;
    Leave;
  Else;
    //Do some work
  EndIf;
EndFor;
```

Handler

SQL Blocked FETCH

```
* D ViewInputArray
D DS LIKERECS(SQLVIEW)
D OCCURS(32767)
D BASED(Pointer)

EXEC SQL
  FETCH NEXT FROM SQLVIEW_C1 FOR 32767 ROWS
  INTO :ViewInputArray;
```

Handler performs 1 I/O and populates host array
Rows are returned 1 at a time per RPG READ
Data is CPU bound, not I/O bound

32 IBM Systems Lab Services and Training – ibm.com/systems/services/labservices © 2011 IBM Corporation

IBM Systems Lab Services and Training IBM

Handling RPG CLOSE

```

*INLR = *On;
Return;
  
```

RPG Implicit CLOSE

'UPDHANDLER'

HANDLE_CLOSE

CLOSE_SQL_CURSOR

```

when
  rpgIO.rpgOperation
  =
  QrnOperation_CLOSE;

  rpgIO.rpgStatus =
  Handle_Close
  (rpgIO);
  
```

```

Dealloc
  rpgIO.stateInfo;

  Close_SQL_Cursor (
  );
  
```

```

EXEC SQL CLOSE
  rpgIOInp_C1;
  
```


33 IBM Systems Lab Services and Training – ibm.com/systems/services/labservices © 2011 IBM Corporation

IBM Systems Lab Services and Training IBM

Rational Open Access: RPG Edition Recap


- Simplifies bridging RPG programs to new SQL services
- Minimal code changes to existing RPG programs
- Provides a “fresh start” for RPG application developers
 - Use latest HLL and ILE capabilities when creating handler and SQL programs
- Allows continued use of RPG ease of use functions without excessive overhead
 - *INLR to reset variables
 - READE for processing groups

34 IBM Systems Lab Services and Training – ibm.com/systems/services/labservices © 2011 IBM Corporation

IBM Systems Lab Services and Training 

Questions?

35 IBM Systems Lab Services and Training – ibm.com/systems/services/labservices © 2011 IBM Corporation

IBM Systems Lab Services and Training 

DB2 for IBM i Consulting and Services

- ✓ Database modernization
- ✓ DB2 Web Query
- ✓ Database design, features and functions
- ✓ DB2 SQL performance analysis and tuning
- ✓ Data warehousing review and assessment
- ✓ DB2 for IBM i education and training

Contact: Dan Cruikshank dcrank@us.ibm.com
IBM Systems and Technology Group
Rochester, MN USA

36 IBM Systems Lab Services and Training – ibm.com/systems/services/labservices © 2011 IBM Corporation