Extract From Embedded SQL in RPG

Beyond the Basics

Paul Tuohy
ComCon
System *i* Developer
5, Oakton Court,
Ballybrack
Co. Dublin
Ireland

System i Developer
The Summit of Education Excellence

Phone: +353 1 282 6230
e-Mail: paul@systemideveloper.com
Web: www.systemideveloper.com
www.ComConAdvisor.com

ComCon

Paul Tuohy

System i Developer
The Summit of Education Excellence
ComCon

Paul Tuohy, author of "Re-engineering RPG Legacy Applications" and "The Programmer's Guide to iSeries Navigator", is one of the most prominent consultants and trainer/educators for application modernization and development technologies on the IBM Midrange. He currently holds positions as CEO of ComCon, a consultancy firm based in Dublin, Ireland, and founding partner of System i Developer, the consortium of top educators who produce the acclaimed RPG & DB2 Summit conference. Previously, he worked as IT Manager for Kodak Ireland Ltd. and Technical Director of Precision Software Ltd.

In addition to hosting and speaking at the RPG & DB2 Summit, Paul is an award-winning speaker at COMMON, COMMON Europe Congress and other conferences throughout the world. His articles frequently appear in System i NEWS, iSeries Experts Journal, The Four Hundred Guru, RPG Developer and other leading publications.

Reminder – Using a Cursor

System i Developer ComCon

Sequential read of a file – Fetch row at a time

```
 H option(*srcStmt : *noDebugIO))
 d data             Ds                   qualified
 d  deptNo                         3a
 d  deptName                      36a    varying
  /include STANDARD
  /free
   exec SQL
      declare C1 cursor for
        select deptNo, deptName from department order by deptNo
          for read only;

   exec SQL
      open C1;

   exec SQL
      fetch next from C1 into :data ;

   doW (SQLCODE >= 0 and SQLCODE <> 100);
      dsply ('Fetch Loop ' + data.deptNo + ' ' + data.deptName);

      exec SQL
         fetch next from C1 into :data ;
   endDo;

   exec SQL
      close C1;
   *inLR = *on;
```

Multi Row Fetch

System i Developer ComCon

A Multi Row Fetch is a much more efficient way of retrieving rows

```
 H option(*srcStmt : *noDebugIO)
 d MAX_ROWS         C                    10
 d i                s             10i 0
 d getRows          s             10i 0 inz(MAX_ROWS)

 d data             Ds                   dim(MAX_ROWS) qualified
 d  deptNo                         3a
 d  deptName                      36a    varying
  /include STANDARD

  /free
   exec SQL declare C1 scroll cursor for
        select deptNo, deptName from department order by deptNo
          for read only;

   exec SQL
        open C1;

   exec SQL
        fetch first from C1 for :getRows rows into :data ;

   for i = 1 to SQLERRD(3);
      dsply ('Normal ' + data(i).deptNo + ' ' + data(i).deptName);
   endFor;

   exec SQL
        close C1;
```

Multi Row Fetch Considerations

System *i* Developer
The Summit of Education Excellence
ComCon

Much faster than a FETCH Loop
- That alone is reason enough to use it

An easy way of generating a result set
- When using embedded SQL for stored procedures

DS Array can be passed as a parameter
- Provides an easy means of using result sets in RPG applications

Data Structure Array or Multiple Occurrence Data Structure (MODS)
- MODS is the older (and more cumbersome) technique
- DS Arrays are much easier

Only a finite number of rows may be retrieved
- Pre-V6R1 – 64K of data
- Post V6R1 – 16M of data

What if the result set exceeds the size of the DS array?
- Does "subfile paging" ring a bell?

Fetch Options

System *i* Developer
The Summit of Education Excellence
ComCon

Alternatives to Next processing
- Fetch ***keyword***

Keyword	Positions Cursor
next	On the next row after the current row
prior	On the row before the current row
first	On the first row
last	On the last row
before	Before the first row - must not use INTO
after	After the last row - must not use INTO
current	On the current row (no change in position)

relative n	n < -1 Positions to nth row before current
	n = -1 Same as Prior keyword
	n = 0 Same as Current keyword
	n = 1 Same as Next keyword
	n > 1 Positions to nth row after current

Sequential Multi Row Fetch

Sequential read of a “page” at a time

```
 H option(*srcStmt : *noDebugIO)
d MAX_ROWS         C                   3
d i                s             10i 0
d getRows          s             10i 0 inz(MAX_ROWS)

d data             Ds                  dim(MAX_ROWS) qualified
d  deptNo                         3a
d  deptName                      36a   varying
 /include STANDARD

 /free
  exec SQL declare C1 scroll cursor for
       select deptNo, deptName from department order by deptNo
         for read only;

  exec SQL open C1;

  doU SQLCODE <> 0;
     exec SQL
          fetch relative 1 from C1 for :getRows rows into :data ;

     for i = 1 to SQLERRD(3);
        dsply ('Sequential ' + data(i).deptNo + ' ' + data(i).deptName);
     endFor;
  endDo;

  exec SQL close C1;
  *inLR = *on;
```

Naughty!!!

FETCH RELATIVE

FETCH RELATIVE is relative to the current cursor position in the result set

- 0 is the current position of the cursor
- 1 is the next row
  - i.e. ***Fetch relative 1*** is the same as ***Fetch Next***
- -1 is the previous row
  - i.e. ***Fetch relative -1*** is the same as ***Fetch Prior***

As rows are fetched, cursor is placed on last row read

Paging Multi Row Fetch – A Sample Program

System *i* Developer
The Summit of Education Excellence
ComCon

To page forward/back through a result set

- Using a multi row fetch
- A simple example
  - *declareAndOpen()* contains the same Declare Cursor and Open Cursor as previous
  - *closeCursor()* contains the same Close Cursor as previous example
  - Complete listing in notes

```
 H option(*srcStmt : *noDebugIO)
d MAX_ROWS        C                     11
d pageSIze        s               10i 0 inz(MAX_ROWS)

  /include STANDARD
  /free
   dsply 'Number of rows per page: ' ' ' pageSize;
   if (pageSize > (MAX_ROWS-1));
      pageSize = (MAX_ROWS-1);
   endIf;
   declareAndOpen();
   getRows(pageSize);
   closeCursor();
   *inLR = *on;
  /end-Free
```

Paging Considerations

System *i* Developer
The Summit of Education Excellence
ComCon

Paging considerations:-

- SQLCODE not set if rows read < page size
  - Use GET DIAGNOSTICS to determine if EOF reached
  - Or use SQLERRD(5)
- EOF not set if last row of page is last row of result set
  - i.e. EOF not set if 10 rows in result set, 10 rows in page
- Read one more row than page size
  - To detect EOF

Factors

- The size of a page
- The number of rows just read
- EOF

Controlling the relative position

- For first page, set relative position to 1
- If Page Back requested, set relative position to (1 - (rows on this page + page size))
  - i.e. Next Page starts with the first row of the previous page
- Read page size + 1
- If not EOF – set relative position to 0
  - i.e. Next Page starts with the last row read
- If EOF – set relative position to (1 – rows just read)
  - i.e. Next Page starts with the first row of this page

Paging Multi Row Fetch – getRows() (1 of 3)

System *i* Developer
ComCon

These are the D Specs for the getRows() subprocedure

- ***direction*** - F = Forward, B = Back, E = End
- ***getPageSize*** - set to pageSize + 1
- ***relativeRow*** Initialized to 1 for the first page read

```
  p getRows...
  p                  b
  d                  PI
  d pageSize                      10i 0 const           the requested Page Size

  d data             Ds                 dim(MAX_ROWS)  DS array for the fetch
  d                                     qualified
  d  deptNo                        3a
  d  deptName                     36a   varying

  d i                s            10i 0
  d direction        s             1a   inz('F')       paging direction
  d getPageSize      s            10i 0                rows to retrieve on the fetch
  d relativeRow      s            10i 0 inz(1)         relative offset for next read
  d backRows         s            10i 0                number of rows fetched
  d lastRow          s            10i 0                status for EOF
```

Paging Multi Row Fetch – getRows() (2 of 3)

System *i* Developer
ComCon

The basic logic is (continued on next slide)

- Set the no. of rows to retrieve on the fetch
- If page back requested – set relative offset to start of previous page
- Fetch the page
- Store the no of rows retrieved
- Check for EOF
- Assume next relative offset is from last row just read
- If EOF - set relative offset to start of this page

```
 /free
  doU (direction = 'E');
     getPageSize = pageSize + 1;                         no. of rows to retrieve
     if (direction = 'B');                               Page back?
        relativeRow = (1 - (pageSize + backRows));      offset to start of previous page
     endIf;
     exec SQL fetch relative :relativeRow from C1        Fetch page
                  for :getPageSize rows into :data;
     backRows =  SQLERRD(3);                             Store rows retrieved
     exec SQL get diagnostics                            Check for EOF
                    :lastRow = DB2_LAST_ROW;
     relativeRow = 0;                                    Assume next relative offset
     if (lastRow = 100);                                 EOF?
        dsply ('Reached EOF');
        relativeRow = (1 - backRows);                    offset to start of this page
     endIf;
```

Paging Multi Row Fetch – getRows() (3 of 3)

System *i* Developer
The Summit of Education Excellence
ComCon

The basic logic is (continued from previous slide)

- If no rows retrieved, load first page
  - Usually caused by paging beyond start of result set
- Display page
  - This example display all rows retrieved
  - Usually display backRows or pageSize
    - Whichever is less
- Prompt for next paging option

```
           if (backRows = 0);
              exec SQL fetch first from C1 for :getPageSize rows into :data;
              backRows =  SQLERRD(3);
           endIf;

           for i = 1 to backRows;
              dsply ('Paging ' + data(i).deptNo + ' ' + data(i).deptName);
           endFor;
           dsply 'Direction (F/B/E) ' ' ' direction ;
        endDo;
       /end-Free
      p                  e
```

```
H option(*srcStmt : *noDebugIO)
d MAX_ROWS        C                    11
d pageSIze        s             10i 0 inz(MAX_ROWS)

 /include STANDARD
 /free
  dsply 'Number of rows per page: ' ' ' pageSize;
  if (pageSize > (MAX_ROWS-1));
     pageSize = (MAX_ROWS-1);
  endIf;
  declareAndOpen();
  getRows(pageSize);
  closeCursor();
  *inLR = *on;
 /end-Free

p declareAndOpen...
p                 b
d                 PI

 /free
  exec SQL declare C1 scroll cursor for
       select deptNo, deptName from department order by deptNo
         for read only;
  exec SQL open C1;
 /end-Free
p                 e
```

```
p getRows...
p                 b
d                 PI
d pageSIze                      10i 0 const

d data            Ds                  dim(MAX_ROWS) qualified
d  deptNo                        3a
d  deptName                     36a   varying

d i               s             10i 0
d direction       s              1a   inz('F')
d getPageSize     s             10i 0
d relativeRow     s             10i 0 inz(1)
d backRows        s             10i 0
d lastRow         s             10i 0

 /free
  doU (direction = 'E');
     getPageSize = pageSize + 1;
     if (direction = 'B');
        relativeRow = (1 - (pageSize + backRows)) ;
     endIf;
     exec SQL fetch relative :relativeRow from C1
                 for :getPageSize rows into :data;
     backRows =  SQLERRD(3);
     exec SQL get diagnostics :lastRow = DB2_LAST_ROW;









     relativeRow = 0;
     if (lastRow = 100);
        dsply ('Reached EOF');
        relativeRow = (1 - backRows);
     endIf;

     if (backRows = 0);
        exec SQL fetch first from C1 for :getPageSize rows into :data;
        backRows =  SQLERRD(3);
     endIf;

     for i = 1 to backRows;
        dsply ('Paging ' + data(i).deptNo + ' ' + data(i).deptName);
     endFor;
     dsply 'Direction (F/B/E) ' ' ' direction ;
  endDo;
 /end-Free
p                 e

p closeCursor...
p                 b
d                 PI

 /free
  exec SQL close C1;
 /end-Free
p                 e
```

A Multi Row Insert

System *i* Developer
The Summit of Education Excellence
ComCon

Insert multiple rows using a DS Array

- Specify the number of rows on the INSERT statement
- Should really be using commitment control

```
     d MAX_ROWS         C                    100
     d numOrderDetails...
     d                  s              10i 0

     d orderHeader    e ds                   extName(ORDHEAD) qualified

     d orderDetail    e ds                   extName(ORDDETL) qualified
     d                                       dim(MAX_ROWS)
      /free
       exec SQL set option naming = *SYS, datFmt = *ISO, datSep = '-';
       exec SQL insert into ORDHEAD values( :orderHeader);
       if (SQLCODE = 0);
          exec SQL insert into ORDDETL :numOrderDetails rows
                          values (:orderDetail);
       endIf;
       if (SQLCODE = 0);
          exec SQL commit;
       else;
          exec SQL rollBack;
       endIf;
```

By the Speaker

System *i* Developer
The Summit of Education Excellence
ComCon

"Re-Engineering RPG Legacy Applications"

- ISBN 1-58347-006-9

"The Programmers Guide to iSeries Navigator"

- ISBN 1-58347-047-6
- www.mcpressonline.com
- www.midrange.com
- www.amazon.com
- etc.

iSeries Navigator for Programmers

- A self teach course
- www.lab400.com

Article links at

- www.comconadvisor.com

RE-ENGINEERING
RPG LEGACY
APPLICATIONS
A Step-by-Step Tutorial to Modernize Business Applications
PAUL TUOHY
THE PROGRAMMER'S GUIDE TO iSERIES NAVIGATOR
iSeries NAVIGATOR for Programmers
V5R3 Edition
A powerful tool for programmers