The Progress Electronic Magazine

In this issue:

Publisher's Statement:

The E-Zine is freely available again! I have been so busy lately that I cannot get the E-Zine out in a regular way. Plus it was reaching so fewer people – not a good way to champion the use of Progress!

Tim Kuehn has been good enough to write an introduction article about using dynamic queries. This is one of the newer more powerful ways that Progress has made for programming queries out of the database without resorting to SQL. He has told me he has plans for a more advanced article in the future – if you like what you read – email him to "put the pressure on." ☺ If you have some work for him – email him for that too!

Plus, if you have freely available routines that you would like to speak more about – this publication is a means to do that! If you have some freely available source code or applications – write up an article on it!

To your success,
Scott Auge
Founder, Amduus Information Works, Inc.
sauge@amduus.com

Coding Article: Introduction to Dynamic Queries

Written by Tim Kuehn timk@tdkcs.waterloo.on.ca

```
 /***************************************************************************
    Program:  qry1.p
    Created:  tdk  2002-10-23
Description:  Dynamic query code

Last change: TDK 2002-10-25 12:12:14 PM
***************************************************************************/

/*
An extremely useful development in the Progress 4GL has been dynamic
access to buffers, fields, and queries. With these "objects", it is
possible to manipulate db information from without knowing the name of
the fields, tables, or database when the program's written. This is a
powerful improvement from the older, pre-version 9 days where every
variation of a query had to be specified at compile time, or a "query"
version of the runtime had to be purchased and code written on-the-fly
in order to make arbitrary queries.

It is also a departure from prior P4GL coding techniques which required
the creative use of include files and such to accomplish the task at
hand. This article is a basic introduction into dynamic queries and
compared to static queries. The database used is sports2000, and all
code examples are written with that in mind.

The first step in creating a query is to make the query "object." The
statement
*/

CREATE QUERY cqh.

/*
causes Progress to allocate a query object in memory. The "handle"
variable cqh is then used to access and / or set the object's
attributes and methods to setup and run the query.

A query object by itself is rather useless though - there are a number
of initializations that have to be done to make this work. The is similar
to a static query where table buffers are assigned to a query. Where in
a static query you would do this:
*/

DEFINE QUERY customer-query
   FOR customer.

/*
for a dynamic query the method to call is ADD-BUFFER() or SET-BUFFERS().
To do this, we use the query's SET-BUFFERS() method to set the query's
associated buffers to the customer table buffer.
*/

cqh:SET-BUFFERS(BUFFER customer:HANDLE).

/*
The next step is to define the query conditions. If we wanted to
dump the entire customer table, our static query would look like
```

```
this:
*/

OPEN QUERY customer-query
   FOR EACH customer.

/*
The dynamic equivalent is:
*/

cqh:QUERY-PREPARE("FOR EACH customer").
cqh:QUERY-OPEN().

/*
Having started the query, the next step is to read the
resulting records. Using a familiar static query:
*/

GET FIRST customer-query NO-LOCK.

REPEAT WHILE AVAILABLE customer:

    DISPLAY customer.name WITH DOWN.
    GET NEXT customer-query NO-LOCK.

END.

/*
Then dynamically:
*/

cqh:GET-FIRST(NO-LOCK).

REPEAT WHILE NOT cqh:QUERY-OFF-END:
   DISPLAY customer.name WITH DOWN.
   cqh:GET-NEXT(NO-LOCK).
END.

/*
Having completed using the query, it's then necessary to de-
allocate the memory used by the query object. This isn't a concern
with static queries since memory allocation and release is handled
by the 4GL according to program scoping rules. Dynamic queries and
similar objects are globally scoped and need to be manually de-
allocated.
*/

cqh:QUERY-CLOSE().
DELETE OBJECT cqh.

/*
Note that calling the QUERY-CLOSE() method is entirely optional.
This method only needs to be called if you wanted to re-open the
same query, or change the query specification to something else
and run the new query specification.

This simplistic example has demonstrated how to setup, execute,
and de-allocate a static query and it's dynamic counterpart. While
the results are identical, there is a bit more overhead with the
dynamic query.

Suppose the nature of the query needed to be changed - the
user wants to have an arbitrary sort-order based on any field
```

```
or set of fields in the customer table, and wants to be able to
specify that at run-time. This could theoretically be done with
a pair of static query using an intermediate temp-table, but
that could become quickly unworkable for large data sets.

With a dynamic query all that's required is to change the query
submitted to the QUERY-PREPARE() method.

In this instance:
*/

cqh:QUERY-PREPARE("FOR EACH customer " +
                  "BY custnum "        +
                  "BY country").

/*
changes the sort order accordingly. The same can be done when selecting
subsets of data from a table, join conditions, and anything else that
would be done with a static query.

In such cases where the specification isn't known ahead of time, the
SUBSTITUTE() function is useful. If, for instance, a user wanted to
specify a customer number range and / or state. If we stored this
information is-cust-num-range-enabled, cust-num-start, cust-num-end,
is-state-enabled, and cur-state-code - a static query would look
like so:
*/

OPEN QUERY
   FOR EACH customer
   WHERE (is-cust-num-enabled = NO OR
         (customer.custnum >= cust-num-start AND
          customer.custnum <= cust-num-end)) AND
         ((is-state-enabled = NO) OR
          (customer.state   = cur-state-code)):

/*
Putting "OR" statements in a query is an invitation to performance
problems. With dynamic queries this becomes a non-issue:
*/

IF is-cust-num-enabled THEN
   DO:
   cqs = SUBSTITUTE("customer.cust-num >= &1 AND " +
                    "customer.cust-num <= &2",
                    cust-num-start,
                    cust-num-end) +

   IF is-state-enabled
      THEN " AND "
      ELSE "".

   END.
ELSE

   ASSIGN
      cqs = "".

IF is-state-enabled THEN
   ASSIGN
      cqs = cqs +
            SUBSTITUTE("customer.state = ""&1""",
                       cur-state-code).
```

```
   /* Take a look at the query before we prepare it */

MESSAGE "Query string " cqs
   VIEW-AS ALERT-BOX INFO BUTTONS OK.

cqh:QUERY-PREPARE("FOR EACH customer WHERE " +
                  cqs).

/*
>From this example it can be seen how to build an arbitrary
query using a dynamic query on a table. It would take a
lot of complicated coding (read "expensive developer time
and effort") to do the same thing with a static query.

Be careful when constructing a query specification string
that successive entries in the condition list are separated
by a space.
*/
```

[Editor: Some additional code sent in by Tim]

```
/***************************************************************************
    Program:  qry1-code.p
    Created:  tdk  2002-10-25
Description:   Code for the query1 article

Last change: TDK 2002-10-25 12:08:38 PM
***************************************************************************/
DEFINE VARIABLE cqh        AS HANDLE     NO-UNDO.  /* Current query handle */
DEFINE VARIABLE cqs        AS CHARACTER  NO-UNDO.  /* Current query string */

   /* Create / Define a query          */

CREATE QUERY cqh.

   /* Assign a buffer to a query       */

DEFINE QUERY customer-query
   FOR customer.

cqh:SET-BUFFERS(BUFFER customer:HANDLE).

   /* Open the query statically and    */
   /* dynamically                      */

OPEN QUERY customer-query
   FOR EACH customer.

cqh:QUERY-PREPARE("FOR EACH customer").
cqh:QUERY-OPEN().

   /* Read the static query's records  */

GET FIRST customer-query NO-LOCK.

REPEAT WHILE AVAILABLE customer:
    DISPLAY customer.name WITH DOWN.
    GET NEXT customer-query NO-LOCK.
END.

   /* read the dynamic query's records */

cqh:GET-FIRST(NO-LOCK).

REPEAT WHILE NOT cqh:QUERY-OFF-END:
```

```
   DISPLAY customer.name WITH DOWN.
   cqh:GET-NEXT(NO-LOCK).
END.

   /* Release memory allocated to the  */
   /* the dynamic query                */

cqh:QUERY-CLOSE().
DELETE OBJECT cqh.

/*************************************************************************/
/* An example on building a more efficient dynamic query */

   /* The variables we need for this   */

DEFINE VARIABLE is-cust-num-enabled AS LOGICAL    NO-UNDO.
DEFINE VARIABLE cust-num-start      AS INTEGER    NO-UNDO.
DEFINE VARIABLE cust-num-end        AS INTEGER    NO-UNDO.

DEFINE VARIABLE is-state-enabled    AS LOGICAL    NO-UNDO.
DEFINE VARIABLE cur-state-code      AS CHARACTER  NO-UNDO.

   /* User interface to get the        */
   /* specifications we need           */
UPDATE
       is-cust-num-enabled
       cust-num-start
       cust-num-end           SKIP(2)

       is-state-enabled
       cur-state-code
    WITH 1 COLUMNS.

   /* Specifying a static query        */

OPEN QUERY customer-query
   FOR EACH customer
   WHERE (is-cust-num-enabled = NO OR
         (customer.custnum >= cust-num-start AND
          customer.custnum <= cust-num-end)) AND
         ((is-state-enabled = NO) OR
          (customer.state   = cur-state-code)).

   /* Building a dynamic query that    */
   /* does the same thing              */

IF is-cust-num-enabled THEN
   DO:
   ASSIGN
      cqs = SUBSTITUTE("customer.cust-num >= &1 AND " +
                       "customer.cust-num <= &2",
                       cust-num-start,
                       cust-num-end)
            +

            IF is-state-enabled
               THEN " AND "
               ELSE "".

   END.

ELSE

   ASSIGN
      cqs = "".

IF is-state-enabled THEN
   ASSIGN
      cqs = cqs +
            SUBSTITUTE("customer.state = ""&1""",
                       cur-state-code).
```

```
CREATE QUERY cqh.    /* Having deleted a prior instance of this,  */
                     /* a new one needs to be created             */

cqh:SET-BUFFERS(BUFFER customer:HANDLE).

cqh:QUERY-PREPARE("FOR EACH customer " +
                  "WHERE " + cqs).

/* Reading the customer information is left as  */
/* an exercise for the reader                   */

   /* reader-inserted code goes here */

/* Cleanup the query object                     */

DELETE OBJECT cqh.
```

About the author: Tim Kuehn founded TDK Consulting Services in 1986, and has been developing in Progress since v8.1 He has also worked in xBase (dbase, Foxbase, Foxpro and Clipper), Unix administration, and embedded real-time systems using C and assembler. If you have a thorny programming question you'd like addressed in a future article or contract work he can be reached at timk@tdkcs.waterloo.on.ca.

Publishing Information:

Scott Auge publishes this document. I can be reached at sauge@amduus.com.

Amduus Information Works, Inc. assists in the publication of this document by providing an internet connection and web site for redistribution:

Amduus Information Works, Inc.
1818 Briarwood
Flint, MI 48507
http://www.amduus.com

Other Progress Publications Available:

This document focuses on the programming of Progress applications. If you wish to read more business oriented articles about Progress, be sure to see the Profile's magazine put out by Progress software:

http://www.progress.com/profiles/

Products Available From Amduus:

Amduus Information Works, Inc. is a Progress reseller and ASPen partner. We primarily develop UNIX/Linux based applications for the web. We also perform integration of Progress applications through such languages and tools as MQ Series, C, and C++.

Amduus provides support for the following applications: Blue Diamond, Denkh, Denkh HTML Reporter, Red Arrow Portal (CMS), Survey Express and other software.

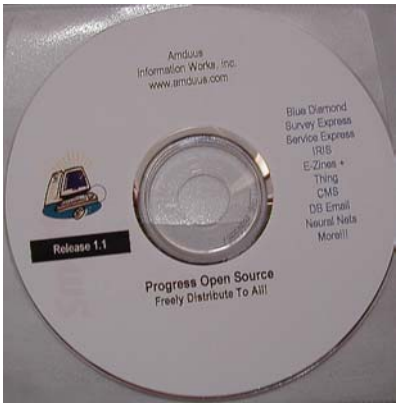
Article Submission Information:

Please submit your article in Microsoft Word format or as text. Please include a little bit about yourself for the About the Author paragraph.

Looking for technical articles, *marketing Progress* articles, articles about books relevant to programming/software industry, white papers, etc.

Order Form for Progress Open Source CD-ROM

COUPON 001A



This is an offer for the CD-ROM at lower than list savings!
This is a great way to support the E-Zine too!

Mail this form to:
Amduus Information Works, Inc.
1818 Briarwood
Flint, MI 48507

Please send _____ copies of the Open Source CD-ROM at $25.00 per disk to:

Name _____
Company _____
Address _____
City _____
State _____ Country _____
Zip _____

Please make your checks/money orders out to: Amduus Information Works, Inc. Cash works too!
This offer only valid in the United States of America.

The CD-ROM includes (all source code included):

- Blue Diamond/IRIS – Webspeed alternatives
- Survey Express – easily create text templates of surveys and then have the program generate the web pages automatically
- Service Express – Web based Help Desk.
- The Progress E-Zines, books on learning to program in Webspeed (PDF/Word/HTML)
- Denkh HTML Reporter – web based report writer
- CMS – a web content management system
- DB Email – Use pop3 to download emails into a Progress database
- Neural Networks – experiments in spam recognition and text message classification
- Denkh – create PDF file reports for Webspeed/UNIX CHUI!
- More!