Package Problems And Parameters

By Craig S. Mullins

You may encounter some problems when implementing packages that will prove to be difficult to resolve. This article examines potential problem areas and overviews new plan and package parameters to be used as you implement packages in your environment.

Potential Problem Areas

One of these problems is determining at runtime which package is actually being used for execution. If multiple versions exist at one time, then any of them could be executed if the corresponding load module is run. It is difficult to determine which version is actually being executed.

DB2 chooses which version of a package to run based on the consistency token, not the version. The consistency token is the internally-formatted, hexadecimal timestamp stored with both the DBRM and the modified source at precompile time. If you need to determine which of many package versions is actually executing, you can use three options.

1. Browse the load module for the program in question. Match the date and time placed there by the linkage editor against the PCTIMESTAMP column of SYSIBM.SYSPACKAGE. The times will not match exactly, but they should be very close. For example, issue the following query to obtain a list of all versions in any collection for a specified package name:

```
SELECT      COLLID, CONTOKEN, VERSION, PCTIMESTAMP
FROM        SYSIBM.SYSPACKAGE
WHERE       NAME = "your package name here"
ORDER BY    COLLID
```

2. In a test environment, issue a dynamic EXPLAIN inside the program. EXPLAIN will generate a PLAN_TABLE row containing a VERSION column, identifying the version of the package that is being run. It should be noted that this method is resource-intensive as EXPLAIN determines access paths and incurs I/O to write to the PLAN_TABLE. It also requires that the executor of the package actually has a PLAN_TABLE into which the row can be placed.

3. The final, and most reliable, method is to turn on the DB2 performance trace. Class(3) contains IFCID 177 that records package allocation information. You can then use a performance monitor to view the trace information. However, the performance trace is also very resource-intensive and is not generally recommended for continuous use in your production environment.

Another potential problem revolves around the VALIDATE bind parameter. If you specify VALIDATE(BIND), the owner of the plan must have the execution privileges for all of the packages in that plan. So, if you wanted to include in the package list, e.g. COL1.*, and only three packages are currently in that collection, e.g. PKG1, PKG2 and PKG3, it is not sufficient that the user has execute authority on COL1.PKG1, COL1.PKG2 and COL1.PKG3. The user should have execute authority on COL1.* because, in the future, it is possible that a new package could be added and the user may not be granted the same authority on that package. If that scenario occurs, it would invalidate the bind. The point is, if you need to add packages to collections, grant execute authority on COLL.*. Another way around that is to specify VALIDATE(RUN). But this gets messy. It degrades performance significantly because validation must occur at runtime. The general recommendation is to avoid VALIDATE(RUN).

Package Bind Parameters

The following is a list of pertinent BIND parameters for binding packages.

EXPLAIN (YES/NO)

When binding packages, you can specify either yes or no for the EXPLAIN parameter. The recommendation is to specify YES so that an accurate record of the access paths for the package is recorded in your PLAN_TABLE. *Of course, the default is no.*

QUALIFIER (qualifier)

The qualifier option can be used to qualify the SQL statements in a program. Whatever is specified in the

QUALIFIER parameter will be used to qualify all of the unqualified SQL in your program. In the past, the only way that SQL statements were qualified was by owner. Now, by specifying qualifier, the owner does not necessarily have to be used as the qualifier for all the tables in the program.

The default for qualifier is owner. This is true whether the owner is explicitly declared or implicit. If the owner parameter is not used to specify an owner, the owner will default to the authid of the user performing the bind. So, it is possible to explicitly declare a different owner than the binder and to specify a different qualifier as well.

It is important to note there is a difference between owner and qualifier. The owner is the user who will be recorded in the DB2 Catalog as the owner of this package or the owner of this plan. Qualifier is simply a string that will be used to qualify the unqualified SQL statements in the package or plan. Qualifiers do not necessarily have to be valid authids.

SQLERROR (NOPACKAGE | CONTINUE)

Two options are available for the SQLERROR parameter: NOPACKAGE and CONTINUE. NOPACKAGE is the recommended option when not operating in a distributed environment. By specifying NOPACKAGE, a package will not be created when a SQL error is encountered. The other option is CONTINUE. You may wonder why you would want to continue and create a package if an error is encountered. SQL syntax may vary from environment to environment and, if you are operating in a distributed environment, you may want to continue to create the package anyway with the understanding that the SQL will function properly at the remote location.

ISOLATION (CS | RR) And RELEASE (COMMIT | DEALLOCATE)

Two other bind package parameters that were available for plans can now be specified at a lower granularity — at the package level. These are ISOLATION and RELEASE. This enables you to set up a plan containing multiple packages where each package utilizes its own isolation and release strategy. For example, one package could use an isolation strategy of cursor stability and a release strategy of commit. Another package could specify repeatable read isolation and release deallocate. Both could then be included in the same plan. DB2 would switch strategies for each package during execution.

Prior to DB2 V2.3 and packages, only one isolation strategy and one release strategy could be specified per plan regardless of the number of DBRMs in the plan.

MEMBER (DBRM-name)

The MEMBER parameter is used to specify the DBRM that will be bound into the specified package. There is no default for this parameter.

COPY (collection.package)

COPY is used to indicate that a package that currently exists in DB2 is to be copied into a new collection. The collection-id specified for the new package must be different from the collection-id of the package being copied.

The MEMBER and COPY parameters are mutually exclusive. You specify either MEMBER or COPY, but not both. If you specify MEMBER, the package will be created from a DBRM member. If you specify COPY, the package will be copied from an existing package.

COPYVER (version)

If you specify the COPY parameter, you can also specify the COPYVER parameter. COPYVER will indicate the explicit version you want to copy. The default is the empty string.

ACTION (ADD | REPLACE)

It is also possible to specify an ACTION parameter of add or replace. ADD is to be used if the package already exists and REPLACE should be used to replace a current package with a new one. Of course, if you specify REPLACE for a new package, DB2 will add the package — much like it adds plans in the current environment.

REPLVER (version)

If you are replacing a package that already exists, the REPLVER parameter can be used to indicate that a specific version is to be replaced. The default for REPLVER is the version that is in the DBRM.

New Plan Bind Parameters

In addition to the availability of parameters for binding packages, DB2 V2.3 provides new parameters for binding plans. Following is a list of pertinent new BIND parameters for plans.

PKLIST (list of packages)

The PKLIST parameter is used to list the packages that are to be available to this plan at execution time. The packages are listed one after the other, separated by commas. Wild cards can be used for any of the three parts of the package name that can be specified here (LOCATION, COLLECTION and PACKAGE).

MEMBER (DBRM-name)

The MEMBER parameter works as it did previously. It is used to specify the DBRMs that will be bound into the specified plan. However, both the MEMBER parameter and the PKLIST parameter can be used to include both DBRMs and packages in the plan.

There is no default for this parameter.

CACHESIZE

The CACHESIZE parameter is used to specify the size of the authorization cache for this plan. The authorization cache is a portion of memory set aside for the plan to store valid authids that can execute the plan. By storing the authids in memory, the cost of I/O can be reduced.

The cache can vary in size from 0 to 4096 bytes in 256-byte increments. For a plan with a small number of users, specify the minimum size of 256. If the plan will have a large number of users, then calculate the appropriate size as follows:

```
CACHESIZE = ( [number of concurrent
users] * 8 ) + 32
```

Take the number returned by the formula and round up to the next 256-byte increment, making sure not to exceed 4096. Note 32 is added because there are always 32 control bytes used by the authid cache.

One final suggestion — if the plan is executed only infrequently, or has been granted to the public, do not cache authids. Specify a CACHESIZE of zero.

CURRENTSERVER

The CURRENTSERVER parameter is used to indicate that the plan is to connect to a remote location immediately when it begins to execute the first SQL statement.

ENABLE/DISABLE

Control over where a package or plan can execute is provided by DB2 V2.3 by means of the BIND parameters: ENABLE and DISABLE. These two parameters can limit execution of a package or plan to a specific operational platform (e.g., batch, CICS, IMS, etc.) and, possibly, even to a specific connection name.

For example, to enable a plan to be run only in the CICSX1 region, the following BIND could be performed:

```
BIND PLAN (PLNAME) . . . ENABLE (CICS)
CICS (CICSX1)
```

ENABLE can be used to indicate that only the listed platforms should be utilized when executing the package or plan being bound. By contrast, DISABLE is used to specifically list the platforms that cannot be used to execute the package or plan; all other available platforms are permitted.

The valid connection types that can be specified to either the ENABLE or DISABLE parameter are:

- BATCH — Controls execution in the TSO environment
- DLIBATCH — Controls execution using the DL/I Batch Support Facility
- DB2CALL — Controls execution in the Call Attach Facility environment
- CICS — Controls execution in the CICS environment
- IMS — Controls execution for IMS environments (DLIBATCH,IMSBMP, & IMSMPP)
- IMSBMP — Controls execution as an IMS Batch Message Processor (BMP)
- IMSMPP — Controls execution as an IMS Message Processing Program (MPP)
- REMOTE — Controls remote execution (valid for packages only).

The DLIBATCH, CICS, IMSBMP and IMSMPP options can be combined with a parameter of the same name to control connection name. For example:

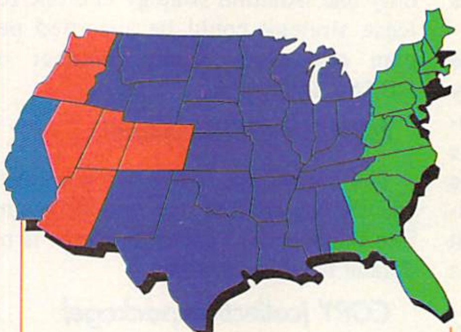
```
DISABLE (DLIBATCH) DLIBATCH (ZMKT1019)
```

ENABLE and DISABLE are mutually exclusive parameters. That is, for any single BIND, you can specify either ENABLE or DISABLE, but not both. Failure to specify ENABLE or DISABLE

See Package Problems *on page 31*

RDJ

Cardinal
BUSINESS MEDIA INC.



Karin Altonaga
Regional Sales Manager
AZ, CO, NV, OR, UT, WA
446 S. Anaheim Hills Rd., Suite 161
Anaheim Hills, CA 92807
818/577-5970
FAX 818/577-0073

Lori Hidrogo
Regional Sales Manager
AL, AR, ID, IL, IN, IA, KS, KY, LA, MI, MN, MS, MO, MT, ND, NE, NM, OH, OK, SD, TN, TX, WI, WY and Western Canada
12225 Greenville Avenue, Suite 700
Dallas, Texas 75243
214/669-9000
FAX 214/669-9909

Edward P. Marecki
Regional Sales Manager
CT, DC, DE, FL, GA, MA, ME, MD, NH, NJ, NY, NC, PA, RI, SC, VA, VT, WV, Quebec, Ontario
1 Richmond Square
Providence, Rhode Island 02906
401/351-0274
FAX 401/351-0276

Vickie Miyaoka
Regional Sales Manager
CA
112 Falkirk Court
Sunnyvale, California 94087
408/481-0707
FAX 408/481-0706

SALES STAFF

CIRCLE	ADVERTISER	PAGE
15	BMC SOFTWARE, INC.	BACK COVER
41	CANDLE CORPORATION	7
-	DIGITAL CONSULTING, INC.	25
-	ESJ NOVEMBER CONFERENCE	I.B. COVER
61	INFORMATION BUILDERS	3
57	INFOTEL CORPORATION	9
55	MICRO FOCUS INC.	5
59	NEON SYSTEMS, INC.	27
25, 27	PLATINUM TECHNOLOGY, INC.	I.F. COVER, 1, 16, 17
36	RELATIONAL ARCHITECTS INTERNATIONAL	19
42, 44	SYSDATA INTERNATIONAL	22, 23
	VM/ESA SUPPLEMENT	
-	IBM/SQL/DS	7
-	IBM/ADSTAR	11
-	IBM/TCPIP	18,19
-	IBM/SEARCH MANAGER 370	28
-	IBM/C/370	I.B.COVER
-	IBM/VM	BACK COVER
369	STERLING SOFTWARE	I.F.COVER

ADVERTISER INDEX

Oracle7's new so-called cost-based optimizer). Other chapters are titled Tuning A New Database, Monitoring And Tuning An Existing Database and Tuning The Data Dictionary.

System Administrators

A final section on Tuning For System Administrators fills 82 pages and covers tuning long-running jobs and tuning for specific system requirements. A chapter of considerable interest to all Oracle7 users and potential users titled Tuning In The Client/Server Environment is hidden here and talks about such things as the vagaries of working with the new SQL Net component.

Appendices

And if all this were not enough, several nice appendices are included. One, Planning For Oracle7, is an absolute must for anyone who is doing so. A second, Hot Performance Tips, should probably be entitled Subject XREF For Those Too Harried To Read The Rest Of The Book. It contains excellent point solution advice (and slyly references earlier chapters) structured around answers to direct, real-world questions from programmers, DBAs, planners, etc.

This book, then, is a solid answer to the question "What sort of information do people covet?" when it's asked among database professionals. In fact, after you buy your own copy, it might be well to keep it under lock and key around the office. It's almost guaranteed to, in a classic Jekyll and Hyde sort of phenomenon, turn your info-phobe colleagues into covetous info-philes who are just liable to spirit off this hefty number while you're away from your desk. To be forewarned is to be forearmed. RDJ

ABOUT THE AUTHOR

Richard Brooks is a member of the Group Technical Staff at Texas Instruments in the Distributed Information Systems Department, 6550 Chase Oaks Blvd., Building 2, M/S 8467, Plano, TX 75023, CompuServe: 76516,3465.

Package Problems

Continued from page 12

will enable the package or plan to all environments.

Be careful when using ENABLE and DISABLE because they may operate a little differently than you might expect. ENABLE says that an environment is explicitly enabled for execution. For example, enabling CICS means that CICS is the only place that can execute the plan or package. So, ENABLE is limiting the environments in which this package or plan can execute. By contrast, specifying DISABLE CICS, the environment is actually more open because only one specific area is disabled, implicitly enabling everything else. The bottom line: ENABLE is more limiting than DISABLE.

When a package or a plan must be run in a specific environment or environments, use the ENABLE parameter to ensure that only proper access is permitted. When a package or plan must never be executed in a specific environment, use DISABLE to close off that environment.

Conclusion

Hopefully this article has made you aware of some of the implementation considerations surrounding packages. If you have some different ideas from those you had when you first began reading this article, then it has done its job. One final reiteration of the general theme of this article is when you develop your package standards, naming conventions and implementation plan, remember to be firm but flexible. If you are, your packages will be properly wrapped and ready to be opened when needed! RDJ

ABOUT THE AUTHOR

Craig S. Mullins, a senior education specialist for Platinum Technology, Inc., has 10 years experience in database development. He is the author of "DB2 Developer's Guide," *second edition* (ISBN 0-672-30512), *Prentice Hall, Carmel, IN. Platinum Technology, Inc., 1815 S. Meyers Rd., Oakbrook Terrace, IL 60181, (800) 442-6861. CompuServe 70410, 237 or Prodigy WHNX44A.*

C On The Mainframe

Continued from page 24

the country are connected for order entry, SNA (using 3270 terminal sessions) is an ideal choice. It emphasizes efficient, reliable and secure data transfer, and enables a high degree of centralized management.

TCP/IP is a better choice if the goals are more dynamic and informal. Instead of communicating with a set of predefined sites within an organization, TCP/IP enables communication to occur with as many other systems as possible. Thus, data resources from all over the world can be accessed when TCP/IP "internetworking" is used. Connecting two different organizations' SNA networks is no easy task. But, connecting a mainframe to thousands of other networks, most of which are non-IBM, is impossible with SNA. TCP/IP is both easier to program and use than SNA. It is the native protocol of most UNIX systems and its applications are oriented toward the needs of modern workstations.

Conclusion

Mainframes can internetwork by using the open, vendor-independent TCP/IP protocols, BSD sockets and the C programming language. Software developers in the mainframe community must move their products to a C/S model where the mainframe is one of many machines that are internetworked. If this takes place, the mainframe is likely to remain a viable computing resource for years to come. RDJ

ABOUT THE AUTHOR

Gregory Scott Hester is a C analyst at SAS Institute, Inc. (Cary, NC), specializing in network communications for mainframes and UNIX-based systems. He has been working with mainframes and UNIX systems for nine years. His experience has been in network application development using TCP/IP, the C programming language/library, and development of low-level systems programming tools in C. SAS Institute, Inc., Cary, NC 27513, (919) 677-8001 Ext. 7548. Internet: sasgsh@unx.sas.com, IBMMAIL: USSASP4S.