A Technical White Paper

By: Brenda Honeycutt, SEGUS Inc

Package Stability (PLANMGMT) DB2 9 and 10

A great feature with some weak points

IBM introduced Plan Stability in DB2 Version 9 with application of APAR PK52523. Better known as Package Stability[1] or Access Path Stability, this feature can create a backup of the existing DB2 Internal Access Plan during the REBIND processing. This provides us an easy fallback in case of a performance problem with one of the new Access Paths.

This white paper explains how Package Stability works and points out some significant differences between DB2 9 and DB2 10.

In order to understand how Package Stability works, we must understand a few technical terms used in this document to reference access paths:

Access Path	General term for the way DB2 accesses data from DB2 tables.
Internal Access Plan	The access path stored in the DB2 directory (SPT01) and used by DB2 internal processing during the runtime of an application program.
Visible Access Path	The access path stored in the PLAN_TABLE for readability and documentation purposes.

During a migration to a new version of DB2, IBM recommends global REBINDs of all critical application packages in order to exploit all performance improvements. Rebinding with Package Stability helps us to exploit new Optimizer enhancements while minimizing the risk. By saving the existing Access Paths, Package Stability provides us an emergency fallback in the event new Access Paths cause performance problems.[2] Using Package Stability, there is no need to fiddle around with Optimization Hints or BINDs into a second collection before the migration as IBM recommended in the past (e.g., with a V7 to V8).

IBM suggests the following strategy in a migration scenario, e.g., V8 to V9 or V8 to V10: Use the system-wide ZPARM PLANMGMT=EXTENDED for the first Rebind after the migration. This saves the V8 Access Paths as an original copy and creates new access paths using the DB2 9 or 10 Optimizer code. Subsequent Rebinds, e.g., in the maintenance process, can then be executed with PLANMGMT(EXTENDED) or PLANMGMT(BASIC), both of which saves a previous copy. This allows fallback to the previous Access Paths whenever needed, and still keeps the original access path, in this case the V8 access path, regardless of how many Rebinds are subsequently executed.

So far, so good. Let’s see what happens if you execute a Rebind with PLANMGMT.

1 The keyword used for the ZPARM and REBIND parameter is PLANMGMT, but it works for packages only, not plans.

2 Package Stability increases space requirements in SPT01. It also increases the REBIND execution time up to 40% in DB2 9 and up to 50% in DB2 10.

Remember, we all fought in our shops for standards requiring every Bind or Rebind in production be done with EXPLAIN(YES) so that we can see the existing Access Paths in the PLAN_TABLE in case of a performance problem caused by static SQL. What do we now have in these tables? The DB2 catalog table SYSIBM.SYSPACKAGE contains some general information about our package. The SYSIBM.SYSPACKSTMT contains the SQL statement text and the dependencies between the package and DB2 objects, e.g., tables and indexes used in SYSIBM.SYSPACKDEP. The PLAN_TABLE contains the Visible Access Paths of our SQL, and maybe some more DSN_* tables, depending on our taste and our Explain tool.

And, finally, SPT01 in the DB2 Directory contains DB2 internal information about each package version, their SQL, and their DB2 Internal Access Plans.

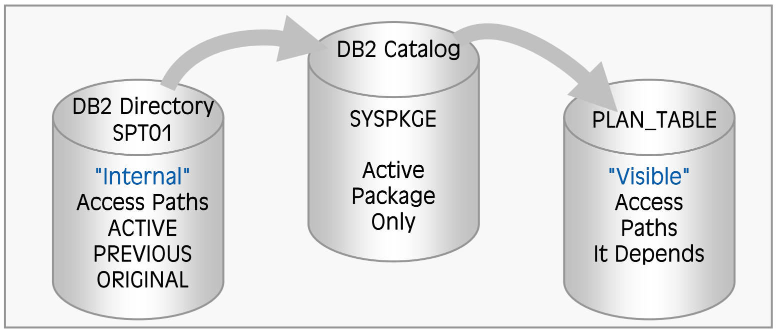

Fig. Directory – Catalog – PLAN_TABLE

Whenever you execute a standard Rebind, DB2 replaces the information in SYSIBM.SYSPACKAGE (especially, the BINDTIME, which is important to find the corresponding information in the Explain tables) and SYSIBM.SYSPACKDEP, which will now reflect the current dependencies. DB2 also replaces the information about our package, its SQL, and the DB2 Internal Access Plan in SPT01.

PLAN_TABLEs are not maintained by DB2 itself because they are user tables. Therefore, DB2 will just add some rows to store the new Visible Access Path. Typically the PLAN_TABLE will grow with every REBIND, so many of us use tools or home-grown procedures to clean up the user PLAN_TABLEs so that they do not keep information about packages that were already replaced in the DB2 catalog.

So what's new with Package Stability? If we Rebind with PLANMGMT, we have two options to choose from: BASIC and EXTENDED. PLANMGMT(EXTENDED) creates kind of a baseline copy (original) that will never be overwritten.

Command	active	previous	original
BIND	01.07.2009		
REBIND (EXTENDED)	02.07.2009		01.07.2009

Once we execute a REBIND with PLANMGMT(EXTENDED) to save an original access path, subsequent usage of PLANMGMT(EXTENDED) is the same as PLANMGMT(BASIC). Both options save the current access path as the previous one in the directory and stores the new one as current.

Command	active	previous	original
BIND	01.07.2009		
REBIND (EXTENDED)	02.07.2009		01.07.2009
REBIND (BASIC)	03.07.2009	02.07.2009	01.07.2009
REBIND (BASIC)	04.07.2009	03.07.2009	01.07.2009

In DB2 9, the DB2 Catalog contains no information about the existing copies (except for some entries about dependencies in SYSIBM.SYSPACKDEP). SYSIBM.SYSPACKAGE contains the active package only. That means that there is no information about the BINDTIME of the copies, and so there is no relationship between the copies of the DB2 Internal Access Plan, the Visible Access Path in the PLAN_TABLE, and the DBA. Only a SWITCH(PREVIOUS) or SWITCH(ORIGINAL) reveals the package data for that version.

In DB2 10, a new table called SYSPACKCOPY in the DB2 Catalog saves the previous and original package information in column COPYID, where 1 is the previous and 2 is the original.

If you cleanup your PLAN_TABLE on a regular basis, existing copies in the directory cannot be considered because there is no information about these copies in the catalog. This means that after you Rebind with PLANMGMT and run your regular cleanup, your older Visible Access Paths are gone.

The first bad news is you don't even know if there is something to switch to or what your target looks like.

Switching back to one of the two possible backups of a package is done with PLANMGMT SWITCH(PREVIOUS) or SWITCH(ORIGINAL).

Command	active	previous	original
BIND	01.07.2009		
REBIND (EXTENDED)	02.07.2009		01.07.2009
REBIND (BASIC)	03.07.2009	02.07.2009	01.07.2009
REBIND (BASIC)	04.07.2009	03.07.2009	01.07.2009
SWITCH (PREVIOUS)	03.07.2009	04.07.2009	01.07.2009

SWITCH(PREVIOUS) exchanges the active with the previous generation of the package in SPT01 and restores the information in SYSIBM.SYSPACKAGE to the previous generation, incl. the BINDTIME. The bad news is it does not restore the Visible Access Path in the PLAN_TABLE. Therefore, if you have already cleaned up the PLAN_TABLE, you would find no information about the current Visible Access Path. Assuming you run your regular cleanup job after the switch, you would delete all information about this package in the PLAN_TABLE and you are back in the good old EXPLAIN(NO) days.

SWITCH(ORIGINAL) is worse. This stores the original as the active. Yes, this means you now have two identical copies of the package and their DB2 Internal Access Plans in your SPT01.

Command	active	previous	original
BIND	01.07.2009		
REBIND (EXTENDED)	02.07.2009		01.07.2009
REBIND (BASIC)	03.07.2009	02.07.2009	01.07.2009
REBIND (BASIC)	04.07.2009	03.07.2009	01.07.2009
SWITCH (PREVIOUS)	03.07.2009	04.07.2009	01.07.2009
SWITCH (ORIGINAL)	01.07.2009	03.07.2009	01.07.2009

Like SWITCH(PREVIOUS), SWITCH(ORIGINAL) does not restore the Visible Access Path in the PLAN_TABLE.

Another snag is how to secure a new original copy since PLANMGMT doesn't provide a way to overwrite the original. Many of us might find it advantageous to replace the original, for example, during the next round of global Rebinds when applying a new APAR or migrating again. Before you can save a new original, you must first free up the existing copies. PLANMGMT supports two options to delete copies in SPT01; PLANMGMTSCOPE(ALL) and PLANMGMTSCOPE(INACTIVE):

```
FREE PACKAGE (collection. package) [ PLANMGMTSCOPE (ALL | INACTIVE) ]
```

As implied, (ALL) frees all versions of the package, the active version plus all copies. (INACTIVE) leaves the active package and frees the previous and original. A bit cumbersome perhaps, but the delete options combined with the subsequent Rebind with PLANMGMT(EXTENDED) get the job done. Yet, you still don't know what you've deleted before you deleted it.

IBM offers both a subsystem PLANMGMT parameter and a Rebind-specific PLANMGMT option. **Note**: The system-wide default in DB2 9 is NONE, whereas the system-wide default in DB2 10 is EXTENDED. Unless you are free of any space constraints in SPT01, think carefully before using the ZPARM. Therefore, before migrating to DB2 10, make sure you check the size of SPT01 and set the ZPARM to NONE if you find space limitations. In this case, use the Rebind PLANMGMT option explicitly for critical packages.

The conclusion is we've got a nice little feature that is much better than what we had with Opthints and dummy Binds, but it leaves a number of question marks in practical usage.

- How do we know what to switch to, previous or original?
- What happens if a backup is for an inoperable or invalidated package? Neither backup is valid.
- If we switch, how do we know which access path we switched to?
- What can we cleanup in the PLAN_TABLE so that we don't loose our Visible Access Paths whenever we switch back?

SOFTWARE ENGINEERING (SEGUS) has developed a tool called SPT01 Transparency, which reads the information about existing copies of a package directly from SPT01. Within its online interface, you can see the information about the active, previous, and original generations of a package and drill down to the access paths on the SQL statement level. This provides the access path visibility you need to decide which access path you might want to switch to.

SPT01 Transparency automation generates the necessary REBIND SWITCH command based on the access path copy you select. If you need to delete package copies, e.g., to save a new original, SPT01 Transparency also generates the desired PLANMGMT SCOPE command as well. An intelligent cleanup procedure for PLAN_TABLEs and other DSN_* tables keeps the Visible Access Paths for all copies of a package that exists in SPT01. All information is externalized into a DB2 table that can be input for existing procedures.

With SPT01 Transparency, you have the visibility you need to exploit Package Stability and make "performance-wise" decisions.

About the author

Brenda Honeycutt is Technical Publications Manager and Senior Software Consultant for SEGUS Inc, a DB2 solutions provider specializing in DB2 tools and consulting for over 20 years. To get more information about SEGUS Inc, go to www.segus.com.