Access Path Stability on Db2 for z/OS

Tony Andrews
tandrews@themisinc.com

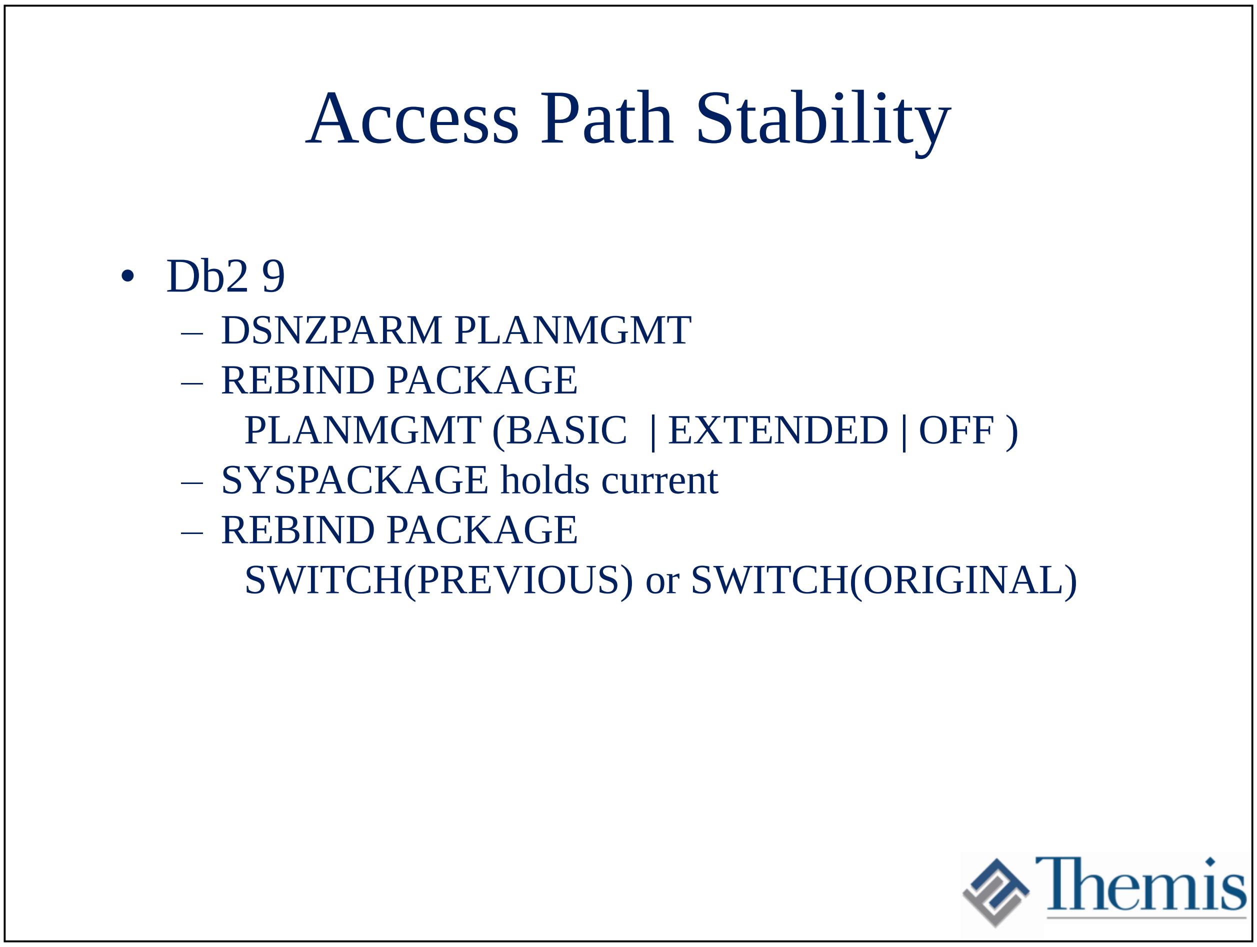
Follow @ThemisTraining

Agenda

- Review of the history
- Plan Management
- Access Path Compare
- Access Path Reuse
- Bind / Explain Output
- Best Practices

Stability History

Themis



Access Path Stability

- Db2 9
  - DSNZPARM PLANMGMT
  - REBIND PACKAGE
    PLANMGMT (BASIC | EXTENDED | OFF)
  - SYSPACKAGE holds current
  - REBIND PACKAGE
    SWITCH(PREVIOUS) or SWITCH(ORIGINAL)

Access path stability features were first introduced in Db2 9 for z/OS in the maintenance stream. More complete support came in Db2 10. The original feature consisted of a new bind option called PLANMGMT (short for plan management). I have chosen to refer to this feature as "Access Path Stability" instead of "Plan Stability" to avoid confusion with the Db2 object called a "plan". The PLANMGMT bind option is valid on REBIND PACKAGE only (not BIND PACKAGE). The idea of this feature is to save off previous "copies" of a package (all internal structures in SPT01 including the access path) when REBINDing a package. A value of BASIC will preserve the "previous" copy of the package and a value of EXTENDED will preserve both the "previous" and "original" copies. In Db2 9, the catalog only contained information about the "current" copy of the package. None of the meta-data about the extra "copies" was available until the copy was activated via a SWITCH. To make an old "copy" of the package active you can REBIND with the SWITCH option.

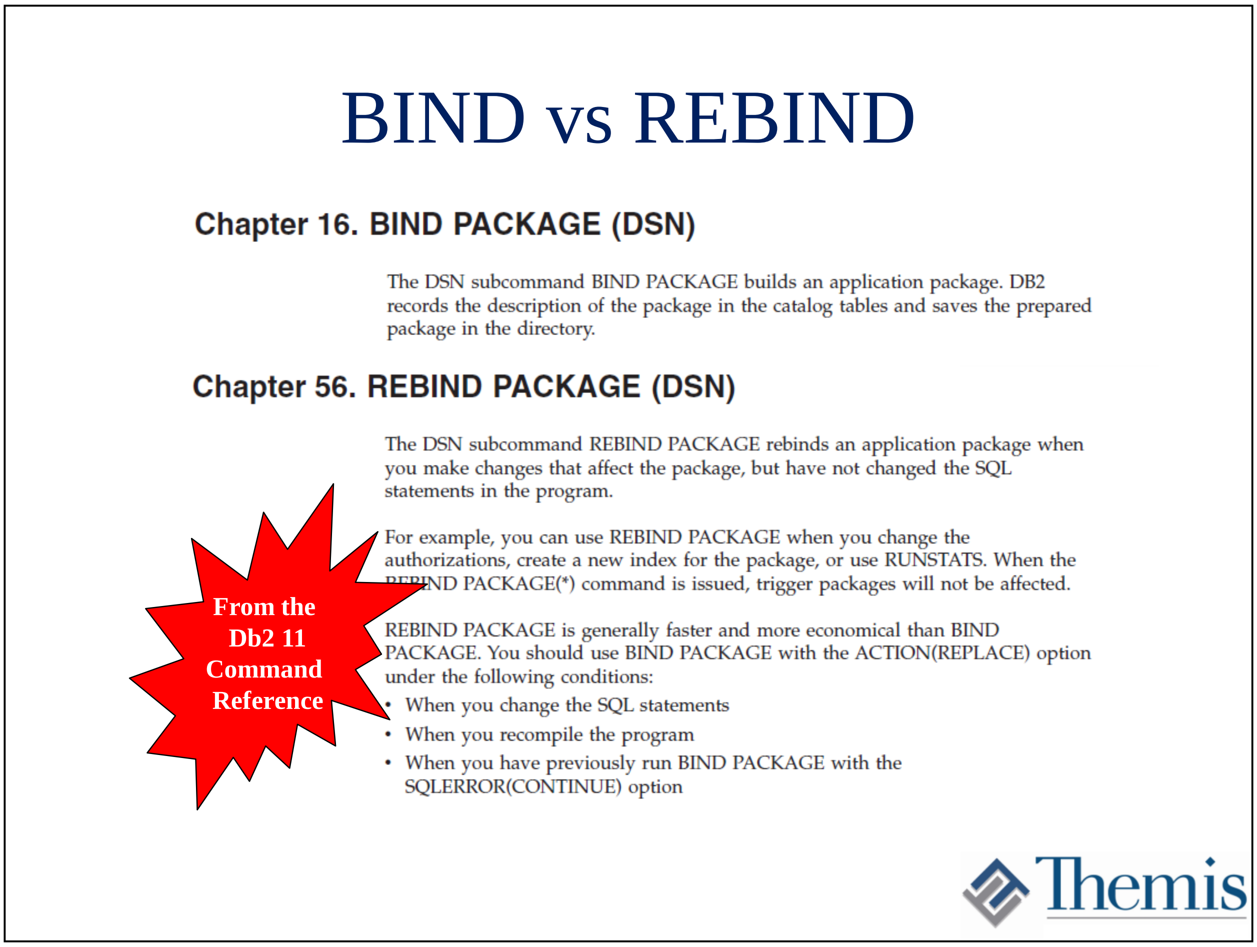
Access Path Stability

- Db2 10
  - SYSPACKCOPY catalog table
    - COPYID 1 = PREVIOUS
    - COPYID 2 = ORIGINAL
  - REBIND APRETAINDUP
  - Native SQL stored procedure packages
  - APCOMPARE added
  - APREUSE added
  - EXPLAIN PACKAGE statement added
  - BIND / REBIND with EXPLAIN(ONLY) added

Db2 10 added catalog support for the inactive “copies” of a package. SYSIBM.SYSPAKCOPY contains all the same information as SYSIBM.SYSPACKAGE for the inactive copies. A column called COPYID will indicate which is the “previous” and which is “original”.

Additionally, the option APRETAINDUP was introduced that will allow us to avoid storing extra copies of a package when the access paths come out the same. Support for native stored procedure packages was also introduced.

Also introduced were several more bind options for stability along with the ability to gather explain data in new situaitons.



BIND vs REBIND

Chapter 16. BIND PACKAGE (DSN)

The DSN subcommand BIND PACKAGE builds an application package. DB2 records the description of the package in the catalog tables and saves the prepared package in the directory.

Chapter 56. REBIND PACKAGE (DSN)

The DSN subcommand REBIND PACKAGE rebinds an application package when you make changes that affect the package, but have not changed the SQL statements in the program.

For example, you can use REBIND PACKAGE when you change the authorizations, create a new index for the package, or use RUNSTATS. When the REBIND PACKAGE(*) command is issued, trigger packages will not be affected.

REBIND PACKAGE is generally faster and more economical than BIND PACKAGE. You should use BIND PACKAGE with the ACTION(REPLACE) option under the following conditions:

- When you change the SQL statements
- When you recompile the program
- When you have previously run BIND PACKAGE with the SQLERROR(CONTINUE) option

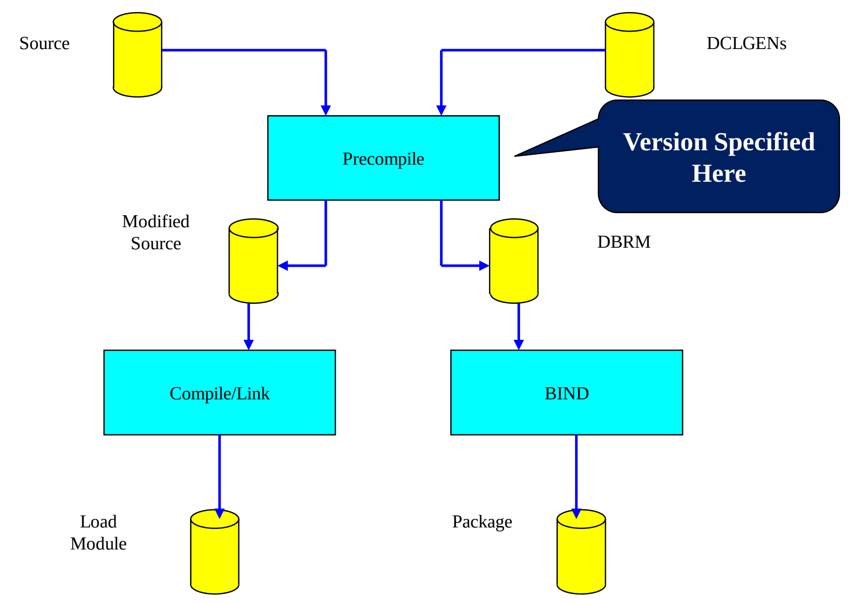
From the Db2 11 Command Reference

It is important to remember the distinction between BIND and REBIND. BIND is typically performed when the source code of a program changes. A new DBRM is produced (and likely a new "version"). BIND then generates access paths and runtime structures for a new package.

REBIND will be run when the application code has not changed, but something else in the environment has changed. The SQL remains the same but access paths selection is re-run and new run time structures are generated. REBIND will typically be run when RUNSTATS have changed, a new index is built, objects have been dropped and recreated, or a new release of Db2 is installed.

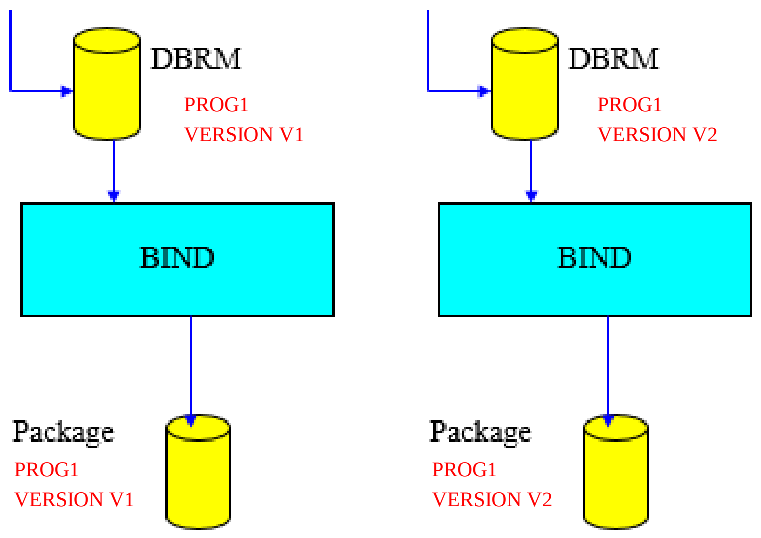
Some of the stability features will work only on REBIND while others are valid for both BIND and REBIND.

Program Preparation



This picture shows the program preparation process for a typical statically bound program. The precompiler separates the host language code from the SQL. The SQL is placed in a Database Request Module (DBRM). If package versioning is being used then a version is assigned by the precompiler and placed in the DBRM. The DBRM is then run through a BIND which produces a package.

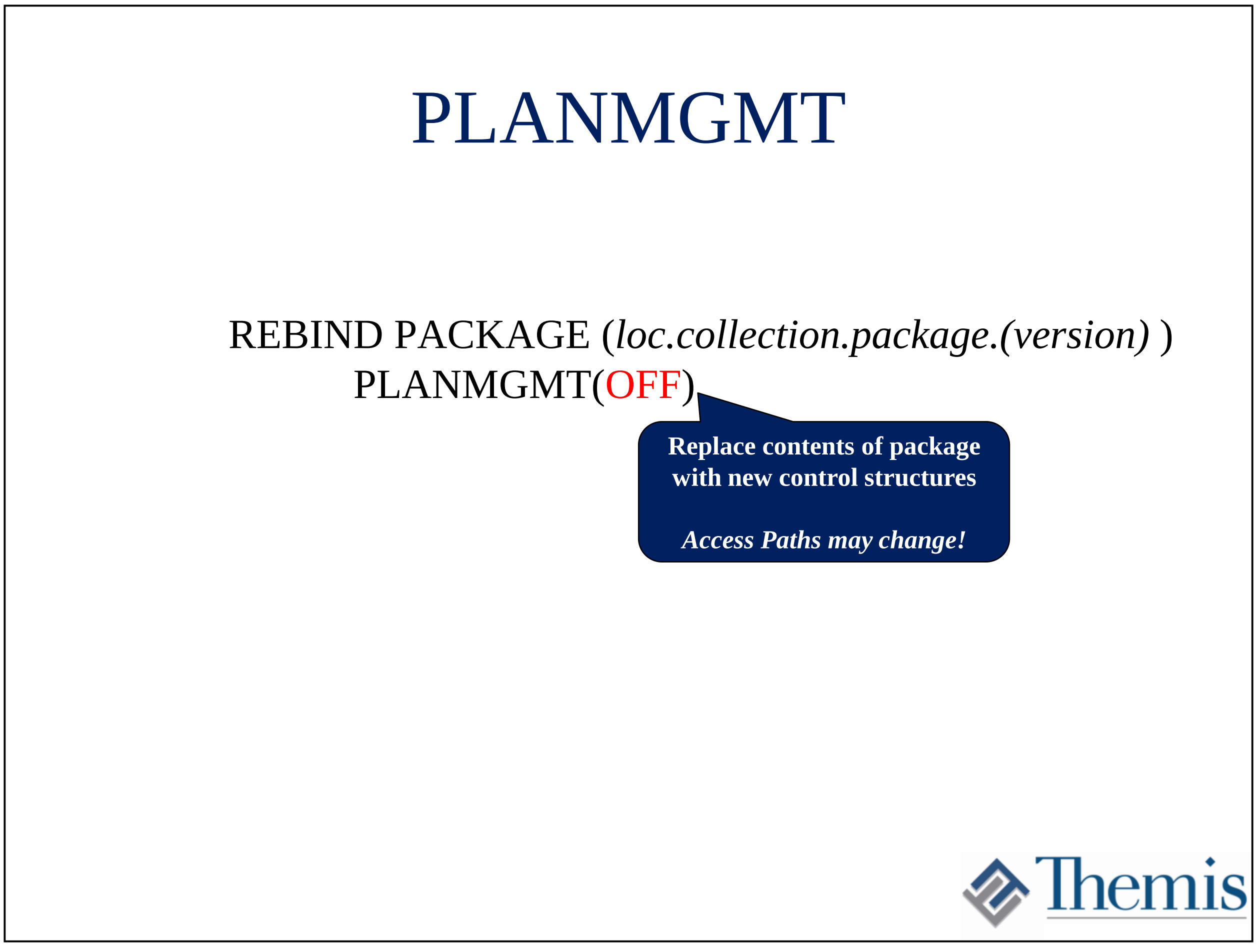
Package Versioning



If package versioning is being employed and the same program is compiled multiple times then the BIND process will not overlay the old "versions" but will instead add new ones. None of this is new, but we review it here to distinguish package "versions" from the "copies" that will be produced by the PLANMGMT bind option. Each "version" of a package may now have up to 3 "copies".

Plan Management

Themis



PLANMGMT

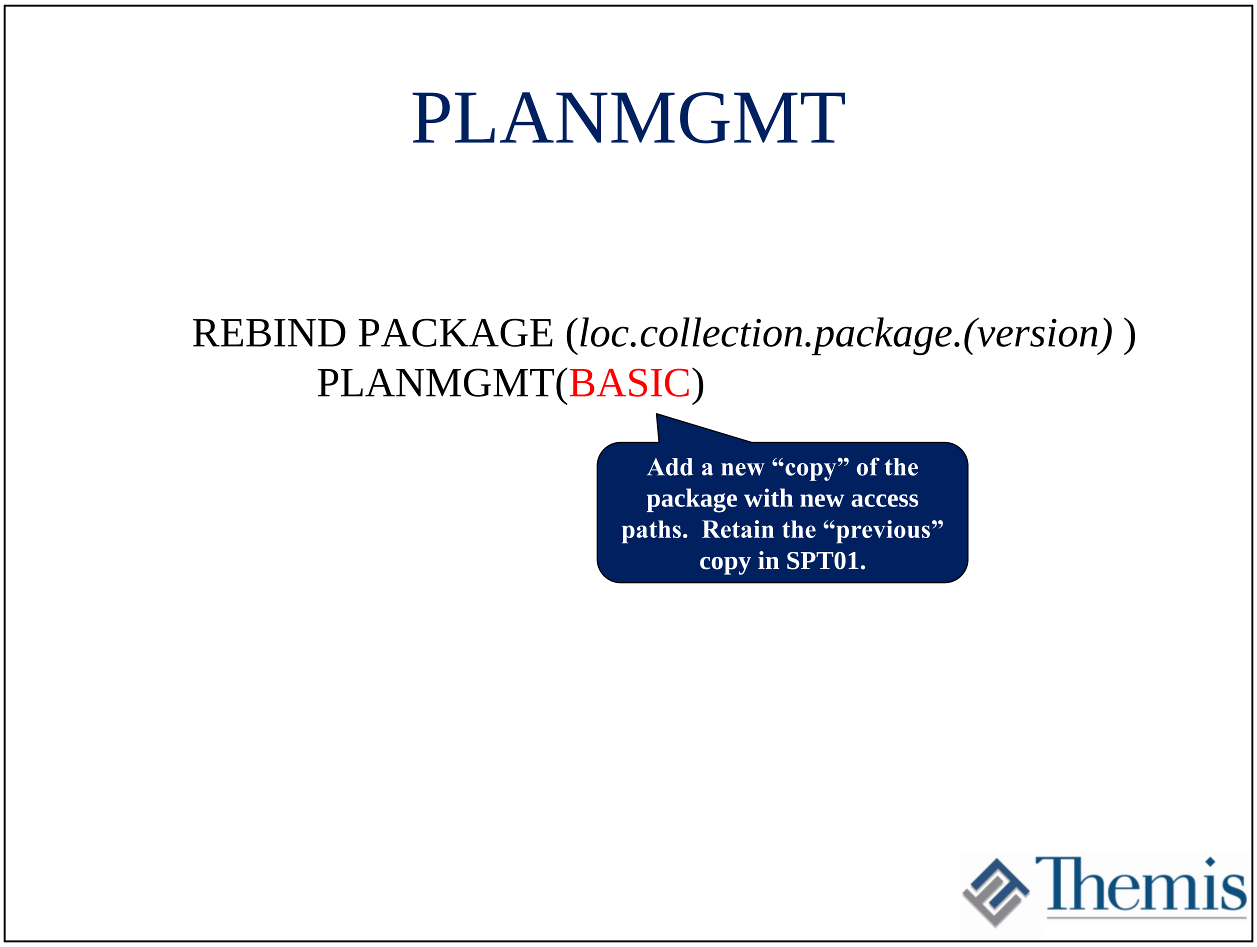
REBIND PACKAGE (*loc.collection.package.(version)*)
PLANMGMT(OFF)

Replace contents of package with new control structures

Access Paths may change!

Themis

The PLANMGMT option is the first of the new stability options and is only valid for REBIND. It was introduced in Db2 9 after GA. The original default was OFF. This default was changed in Db2 10 to EXTENDED and may be controlled with subsystem parameter PLANMGMT. When PLANMGMT(OFF) is used then no previous of this package version will be retained. Access path selection will be invoked and the old control structures will be discarded. This describes the behavior of REBIND prior to Db2 9.

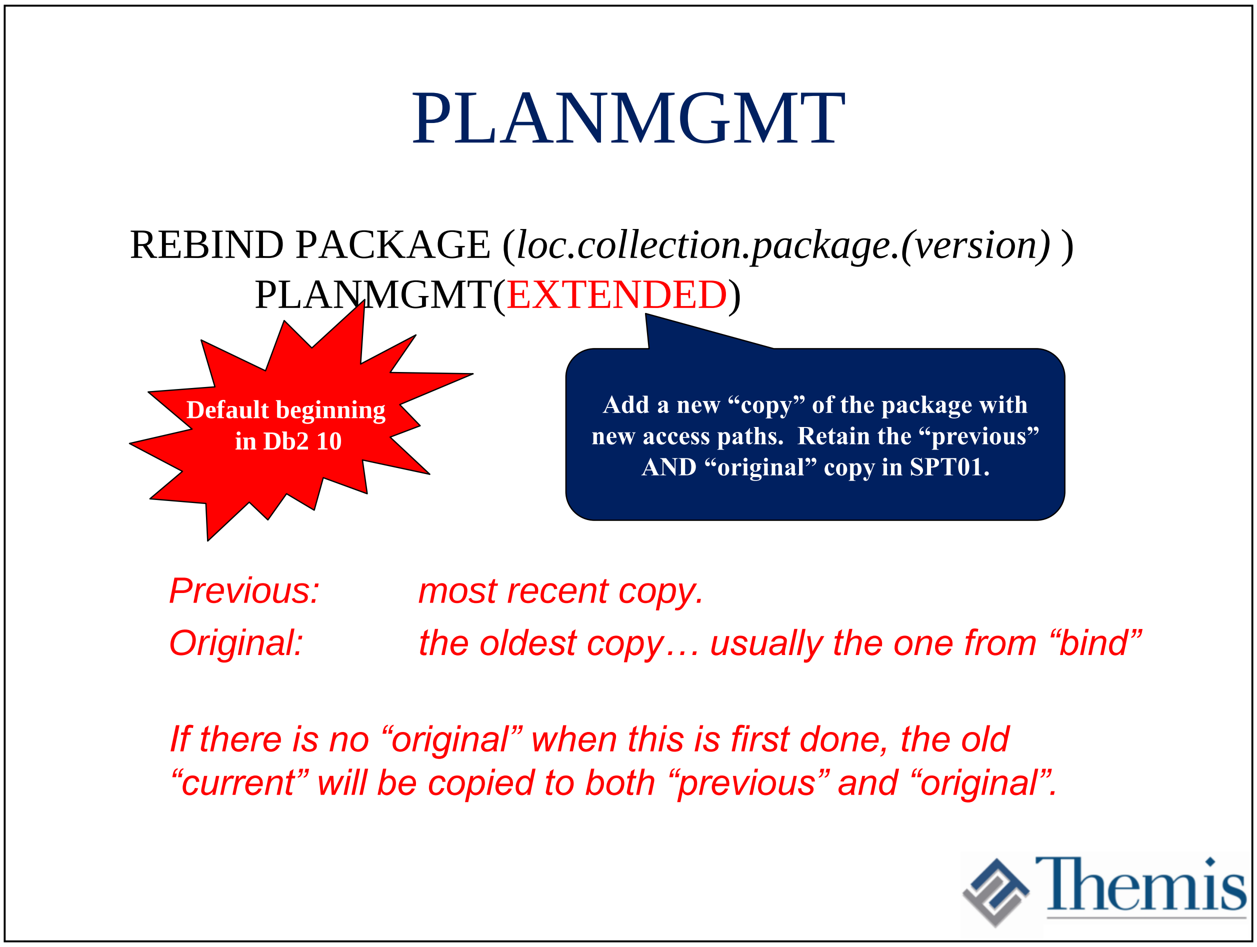


PLANMGMT

REBIND PACKAGE (*loc.collection.package.(version)*)
PLANMGMT(BASIC)

Add a new "copy" of the package with new access paths. Retain the "previous" copy in SPT01.

When a value of BASIC is used the "previous" copy of the package version will be retained in SPT01. A new "copy" of the package will be created and will immediately become the "active" copy of the package version. We can then switch back to the "previous" if we don't like the results of the REBIND. Each subsequent REBIND with this option will discard the "previous" and generate a new "current" copy.



PLANMGMT

REBIND PACKAGE (*loc.collection.package.(version)*)
PLANMGMT(EXTENDED)

Default beginning in Db2 10

Add a new "copy" of the package with new access paths. Retain the "previous" AND "original" copy in SPT01.

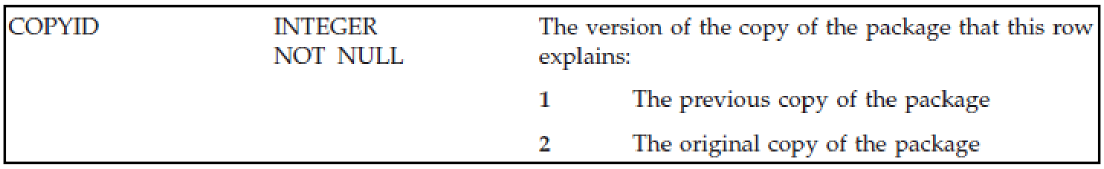
Previous:	*most recent copy.*
Original:	*the oldest copy… usually the one from "bind"*

If there is no "original" when this is first done, the old "current" will be copied to both "previous" and "original".

If PLANMGMT(EXTENDED) is used an additional copy of the package is retained. In addition to the "current" and "previous" copy, an "original" will also be retained. The "original" will be the oldest copy of the package that is available (usually as a result of the BIND). This copy is never overlaid by subsequent BINDs with EXTENDED, while the "previous" copy will be replaced each time.

Catalog Support

- SYSIBM.SYSPACKCOPY contains rows for the “previous” and “original” copies
- A copy of SYSPACKAGE for package copies not currently in use.
- Timestamps of when they were bound also available



COPYID	INTEGER NOT NULL	The version of the copy of the package that this row explains:
		1 The previous copy of the package
		2 The original copy of the package

The meta-data for “copies” of a package that are not currently is stored in SYSIBM.SYSPACKCOPY. This catalog table was added in Db2 10.

Switching between copies

REBIND PACKAGE (*loc.collection.package.(version)*)
SWITCH(PREVIOUS)

"Previous" becomes current and current becomes "previous".

REBIND PACKAGE (*loc.collection.package.(version)*)
SWITCH(ORIGINAL)

"Original" becomes current and "current" becomes "previous". "Original" remains unchanged.

Themis

We can switch back to other "copies" of the package with the REBIND commands shown here. Pay attention to what happens in the inactive copies.

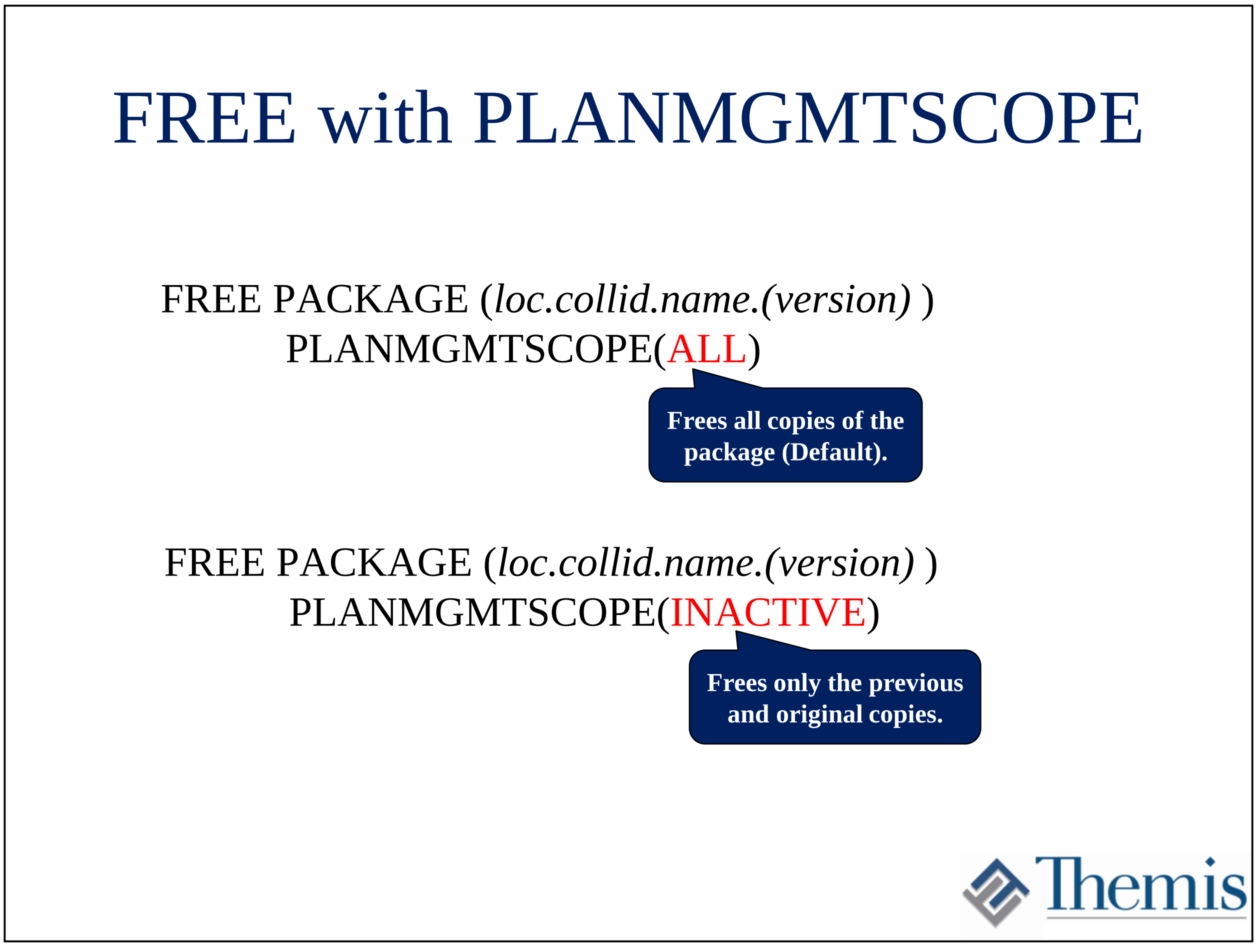
Considerations

- When a package goes invalid… Think about which copies of the package may be affected.
  - If a table is dropped, all copies of the dependent packages will be marked invalid.
  - If an index is dropped, only copies that included that index in an access path will be affected.

- Switching to ORIGNIAL eliminates the “previous” copy.

- Please distinguish between package versions and copies. Each version may have the three copies.

Themis



FREE with PLANMGMTSCOPE

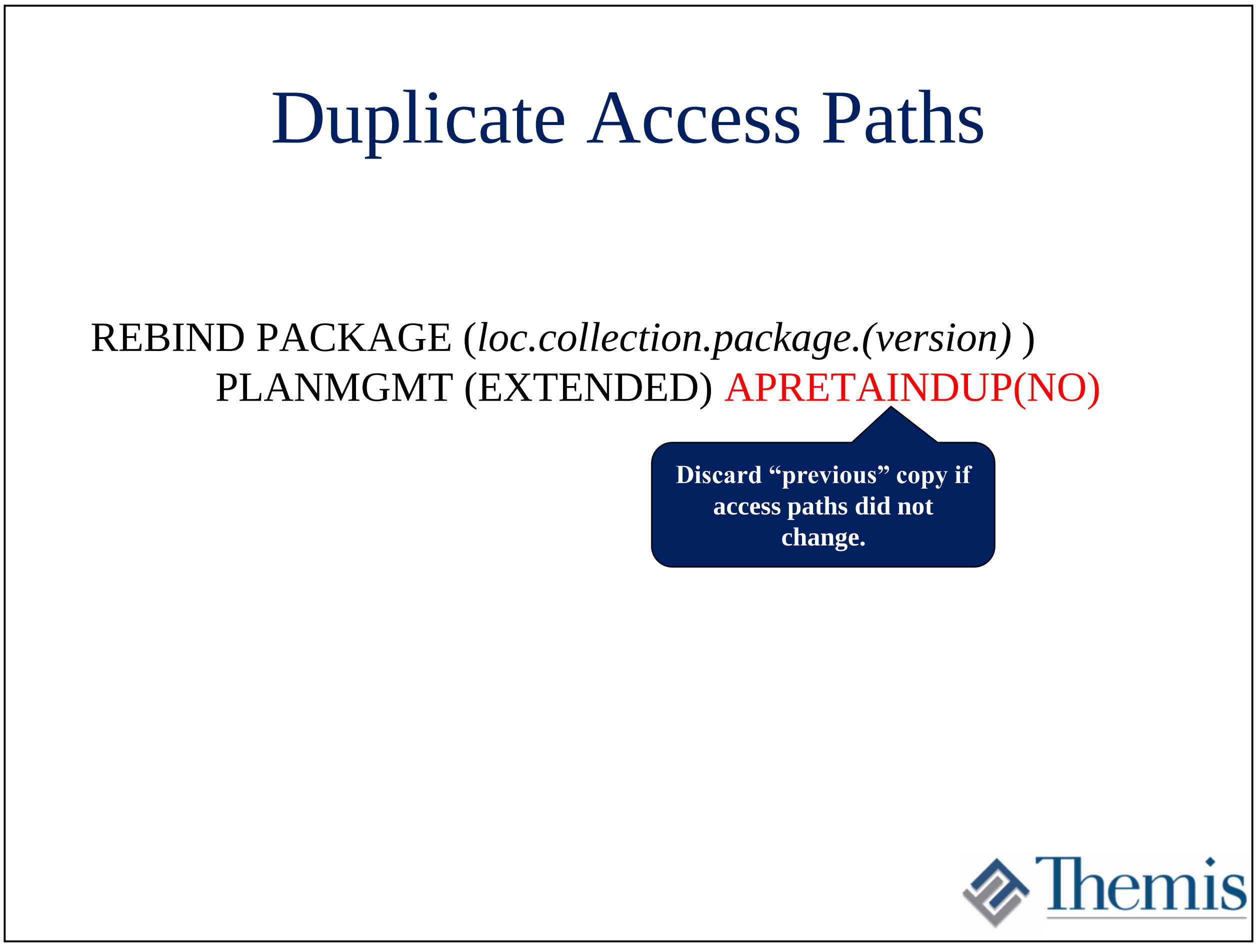
FREE PACKAGE (*loc.collid.name.(version)*)
PLANMGMTSCOPE(ALL)

Frees all copies of the package (Default).

FREE PACKAGE (*loc.collid.name.(version)*)
PLANMGMTSCOPE(INACTIVE)

Frees only the previous and original copies.

The FREE command will free all copies of the package. Using the FREE command with option PLANMGMTSCOPE(INACTIVE) will only remove the “previous” and “original” copies of the package.



Duplicate Access Paths

REBIND PACKAGE (*loc.collection.package.(version)*)
PLANMGMT (EXTENDED) APRETAINDUP(NO)

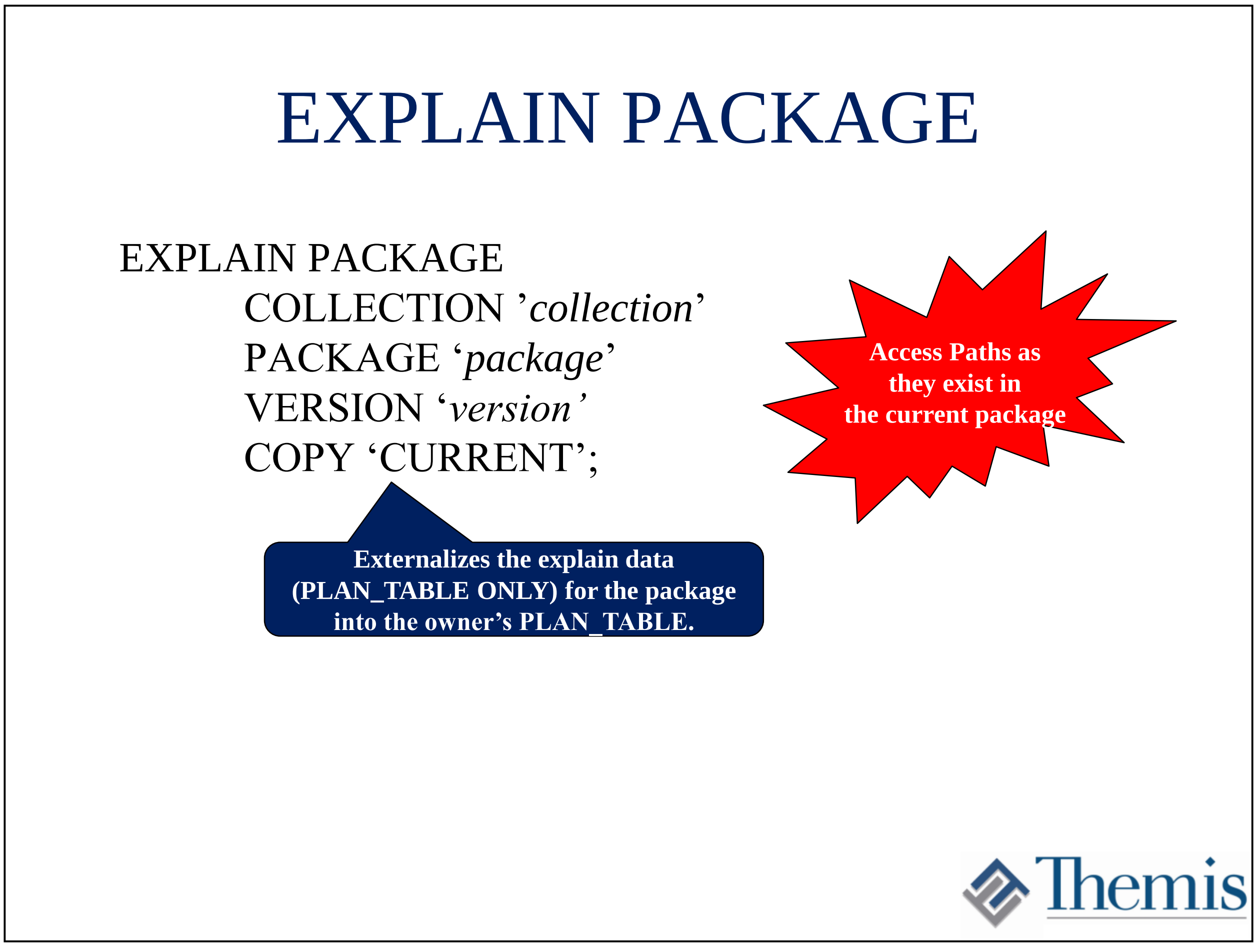
Discard "previous" copy if access paths did not change.

Keeping multiple copies of packages in Db2 might have a significant impact on the size of SPT01 in the Db2 directory. One way to control this is the APRETAINDUP option on REBIND. Setting this option to "NO" will discard the previous copy if the access paths do not change as a result of the REBIND.

How does it work?

- An copy of the PLAN_TABLE entries for the package is stored internally (not readable by humans) at bind time in SPT01 beginning in Db2 9.
- This allows the BIND / REBIND process to compare access paths across versions and copies.
- So many possibilities…

The APRETAINDUP option is enabled by the fact that Db2 now keeps an internal copy of the plan table entries inside the package itself. This data is not directly readable by humans, but enables Db2 to compare previous access paths to the new ones…. and so much more!



EXPLAIN PACKAGE

```
EXPLAIN PACKAGE
        COLLECTION 'collection'
        PACKAGE 'package'
        VERSION 'version'
        COPY 'CURRENT';
```

Access Paths as they exist in the current package

Externalizes the explain data (PLAN_TABLE ONLY) for the package into the owner's PLAN_TABLE.

Because Db2 keeps internal copies of the PLAN_TABLE at BIND time it is now possible to see the access paths even if the package was bound with EXPLAIN(NO). The SQL statement shown here will externalize the PLAN_TABLE data from the package to an external PLAN_TABLE. The access paths shown will be the ones chosen at BIND time regardless of when the EXPLAIN statement is run.

EXPLAIN(ONLY)

BIND (or REBIND)
PACKAGE…

…

EXPLAIN(ONLY)

Explain the statements against the current environment without producing a package

The EXPLAIN(ONLY) option on a BIND or REBIND will only produce access paths in the explain tables without producing a package.

Compare Access Paths

Themis

Access Path Compare

BIND (or REBIND) PACKAGE…

…

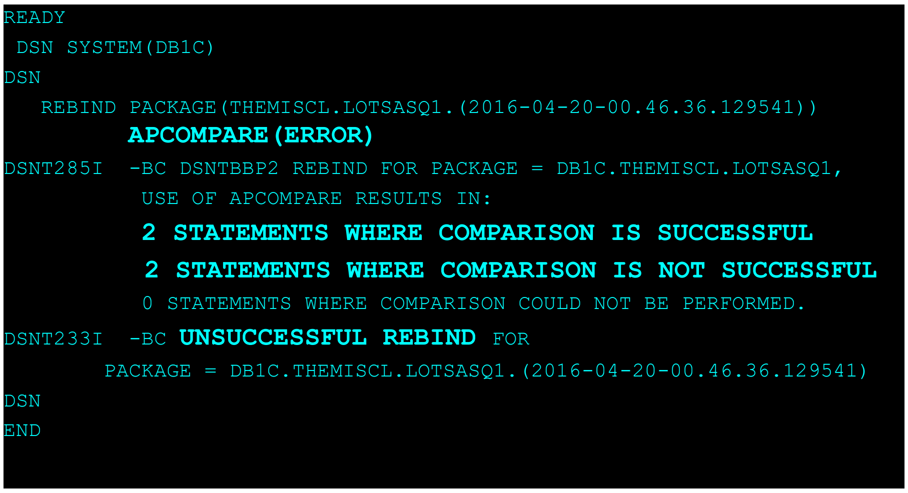
APCOMPARE(NONE | WARN | ERROR)

Warning issued if path changes

Bind fails if *any* paths change

For BIND, the comparison will be with the version being bound (if it exists) or with the most recent version available.

APCOMPARE (ERROR)



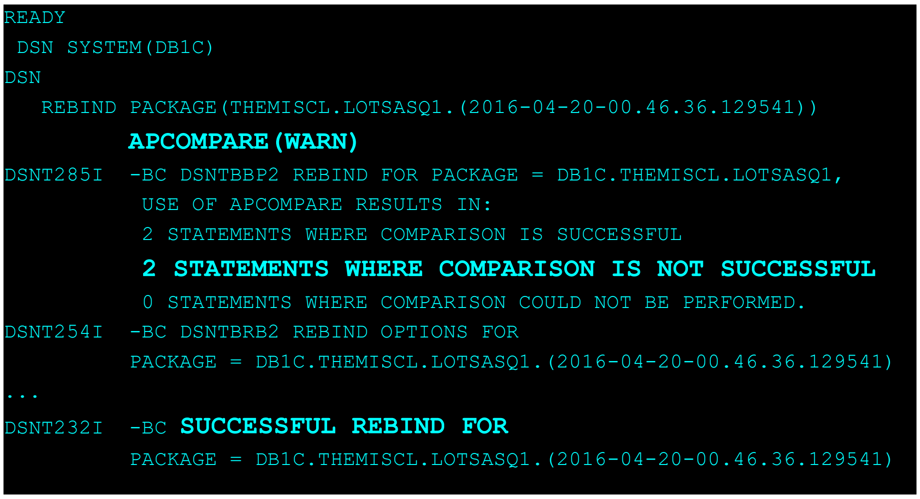
```
READY
 DSN SYSTEM(DB1C)
DSN
   REBIND PACKAGE(THEMISCL.LOTSASQ1.(2016-04-20-00.46.36.129541))
          APCOMPARE(ERROR)
DSNT285I  -BC DSNTBBP2 REBIND FOR PACKAGE = DB1C.THEMISCL.LOTSASQ1,
           USE OF APCOMPARE RESULTS IN:
           2 STATEMENTS WHERE COMPARISON IS SUCCESSFUL
           2 STATEMENTS WHERE COMPARISON IS NOT SUCCESSFUL
           0 STATEMENTS WHERE COMPARISON COULD NOT BE PERFORMED.
DSNT233I  -BC UNSUCCESSFUL REBIND FOR
        PACKAGE = DB1C.THEMISCL.LOTSASQ1.(2016-04-20-00.46.36.129541)
DSN
END
```

Evaluating the Failures

- REBIND the package with EXPLAIN(ONLY)
  - Generates the PLAN_TABLE entries for the “new” access paths
- Run the EXPLAIN PACKAGE statement
  - Generates the PLAN_TABLE entries for the “old” access paths
- Compare and evaluate

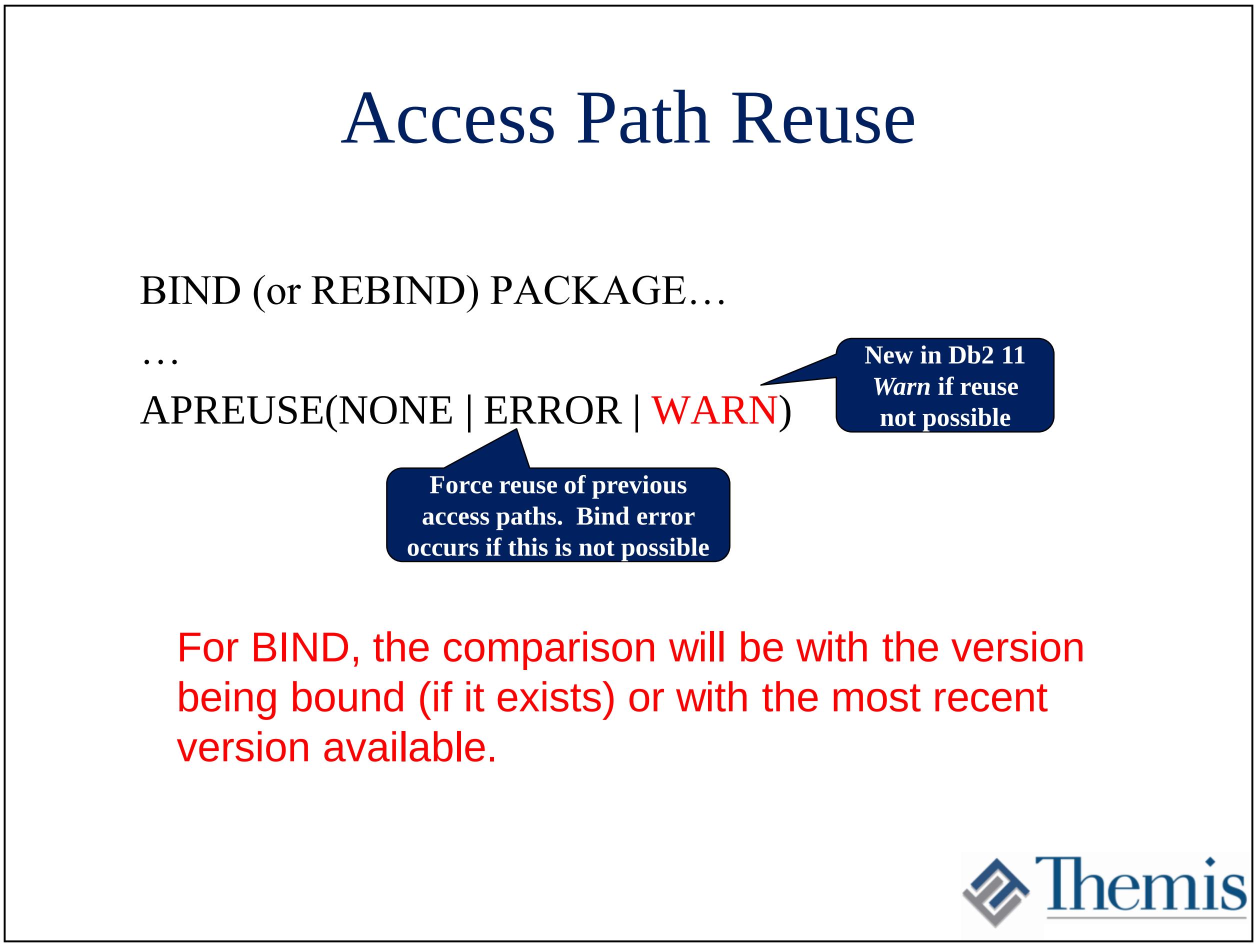
Themis

APCOMPARE(WARN)



```
READY
 DSN SYSTEM(DB1C)
DSN
   REBIND PACKAGE(THEMISCL.LOTSASQ1.(2016-04-20-00.46.36.129541))
          APCOMPARE(WARN)
DSNT285I  -BC DSNTBBP2 REBIND FOR PACKAGE = DB1C.THEMISCL.LOTSASQ1,
           USE OF APCOMPARE RESULTS IN:
           2 STATEMENTS WHERE COMPARISON IS SUCCESSFUL
           2 STATEMENTS WHERE COMPARISON IS NOT SUCCESSFUL
           0 STATEMENTS WHERE COMPARISON COULD NOT BE PERFORMED.
DSNT254I  -BC DSNTBRB2 REBIND OPTIONS FOR
          PACKAGE = DB1C.THEMISCL.LOTSASQ1.(2016-04-20-00.46.36.129541)
...
DSNT232I  -BC SUCCESSFUL REBIND FOR
          PACKAGE = DB1C.THEMISCL.LOTSASQ1.(2016-04-20-00.46.36.129541)
```

Access Path Reuse



Access Path Reuse

BIND (or REBIND) PACKAGE…

…

APREUSE(NONE | ERROR | WARN)

New in Db2 11 *Warn* if reuse not possible

Force reuse of previous access paths. Bind error occurs if this is not possible

For BIND, the comparison will be with the version being bound (if it exists) or with the most recent version available.

Themis

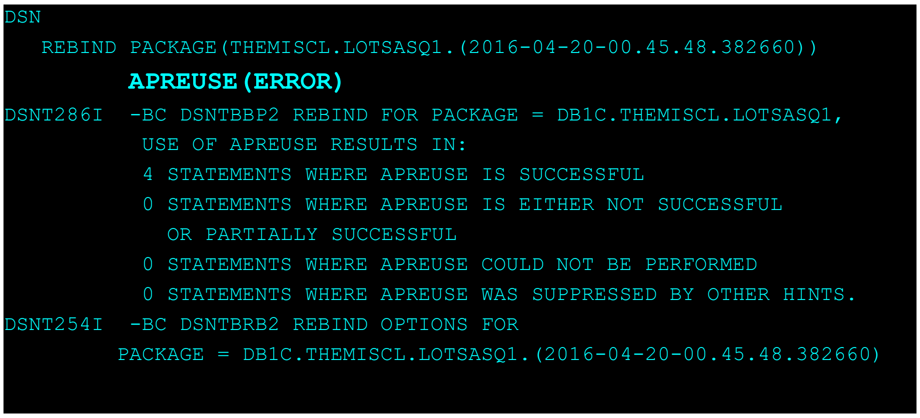
The APREUSE(ERROR) bind or rebind option allows us to request that previous access paths be used for the package. This is actually accomplished by passing the PLAN_TABLE entries from the previous package as a hint to the current BIND or REBIND. The BIND fails if the hint is not accepted by optimizer. The same rules for APCOMPARE apply when determining which package is used for comparison. Db2 11 introduced APREUSE(WARN) which will allow the package to be produced even when the hint fails. New paths are only produced for statements where the hint fails.

Access Path Reuse

- For BIND, the comparison will be with the version being bound (if it exists)

- or with the most recent version available
  - DSNT292I message is issued
  - Because the package versions differ, it is possible that not all statements have a match.
  - APREUSE only applies to statements that are identical between the two versions. Statement numbers need not be the same.

Access Path Reuse



```
DSN
   REBIND PACKAGE(THEMISCL.LOTSASQ1.(2016-04-20-00.45.48.382660))
          APREUSE(ERROR)
DSNT286I  -BC DSNTBBP2 REBIND FOR PACKAGE = DB1C.THEMISCL.LOTSASQ1,
           USE OF APREUSE RESULTS IN:
           4 STATEMENTS WHERE APREUSE IS SUCCESSFUL
           0 STATEMENTS WHERE APREUSE IS EITHER NOT SUCCESSFUL
             OR PARTIALLY SUCCESSFUL
           0 STATEMENTS WHERE APREUSE COULD NOT BE PERFORMED
           0 STATEMENTS WHERE APREUSE WAS SUPPRESSED BY OTHER HINTS.
DSNT254I  -BC DSNTBRB2 REBIND OPTIONS FOR
         PACKAGE = DB1C.THEMISCL.LOTSASQ1.(2016-04-20-00.45.48.382660)
```

Themis

Access Path Reuse

- “Force” is such a harsh word…
- APREUSE will pass previous plan table into the bind as an *optimization hint*
- Sometimes this doesn’t work…
  - Indexes no longer available
  - Query re-write different across different Db2 versions

Themis

Db2 12 Enhancements

- FREE only “original” or “previous” copies of a package or only invalid copies
- SWITCH option will no longer allow you to switch to an invalid copy
- APREUSESOURCE option will allow the REUSE hint to be directed to the “previous” or “original”
- DSN_STATEMNT_TABLE columns will now tell you if APREUSE was effective and which package was used

Themis

Best Practices

Themis

Support for Db2 Version Migration

- Many will simply REBIND everything using APREUSE(ERROR)

- A more thorough approach…
  - REBIND everything using APCOMPARE(ERROR)
  - REBIND packages failing above using EXPLAIN(ONLY)
  - Analyze differences. For Db2 11 you will frequently see more matching columns.

- Possible role for APREUSE(WARN)

Themis

Regular Change Control

- PLANMGMT(EXTENDED) everywhere with possible use of APRETAINDUP(NO).
- APCOMPARE(WARN) *or maybe (ERROR)* in production binds.

Themis

Tony Andrews

Themis Training

tandrews@*themisinc.com*

Twitter: @tonyandrews12

Themis