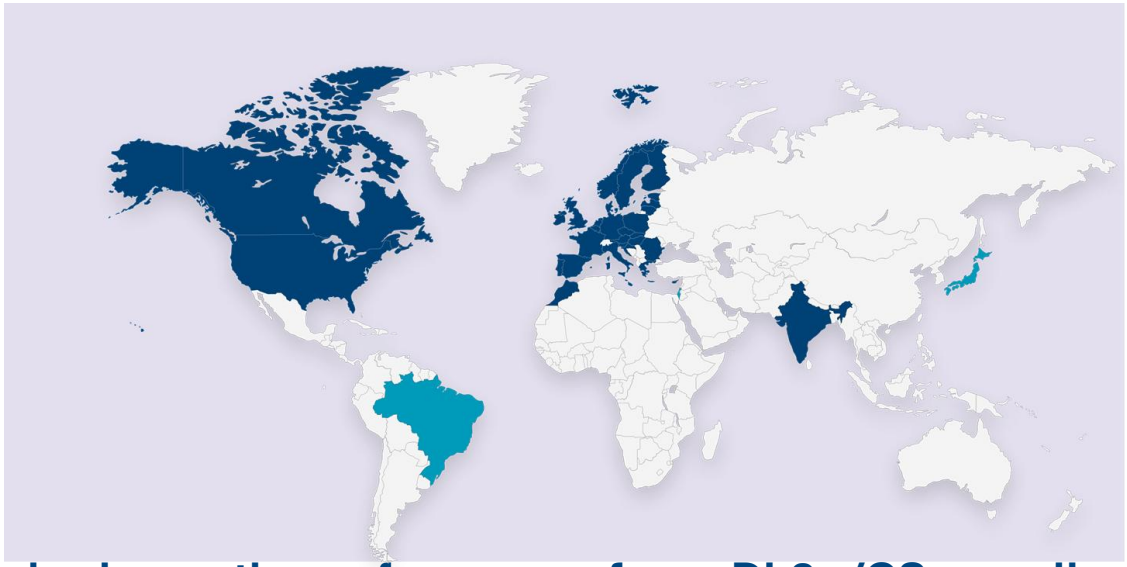
infotel Software

Managing Packages in large z/OS Db2 Environments

Infotel Software Group

- Infotel headquarters in Paris and labs in Toulouse, France
- Subsidiaries in USA, Canada, UK, Germany, Netherlands, India, Marocco and Spain
- World-wide network of distributors



- IBM technology partner
  (developed High Performance Unload)
- Provides utilities and solutions to boost and enhance the performance of your Db2 z/OS as well as utilities for Oracle DB and PostgreSQL.

Agenda

- Packages & Explain Tables
- Functions and Facilities for Successful Package Management
- Deployment and How to Determine “inconsistency”
- Explain Table Management and even more …
- Success Story at one of the world‘s largest Db2 sites

Packages & Explain Tables

- Db2 Packages
- Db2 Explain & Explain Tables
- Packages & Explain Tables
- Package Components

Db2 Packages Introduced

infotel Software

- DB2 Version 2 Release 3 GA October 25, 1991
- One of the significant features was the introduction of packages
- Up till then, DBRMs were bound to a PLAN
  = NOT very flexible and a bottleneck for development
- Packages: simply 1:1 DBRM bound on its own as a PACKAGE and defined in a list of package names belonging to a PLAN.
  = Very flexible, made Db2 application programs easier to support.
  = NO bottlenecks
- Package Versions now enabled multiple versions of a named package

Db2 Packages Introduced

Packages are identified by five different attributes:

- Location
- Collection
- Name
- Version
- Consistency token

Db2 Packages

Packages are simply named application programs and are part of a collection.
Collections are used to group packages for a certain or multiple number of applications.
Versions – enable multiple versions of a single package

More PRO than CONTRA – benefits

- Flexibility to keep multiple versions. Ease of use for application developers
- Reduced BIND times – only a single DBRM is optimized.
  When changed, only this package needs to be (re)bound
- Ability to support remote access with static SQL
- Ability to fall back from access path regression
- Simplified access to mirror applications and objects
- and more….

Db2 Packages

infotel Software

CONTRA

- If not managed properly, the number of package versions will grow out of control.
- Careful with VERSION(AUTO)

This requires a management system to ensure how many package versions should be kept and which ones !

Cleaning-up packages no longer in use is a periodic task affecting both the Catalog and Explain tables.

One major drawback - there is no easy method to determine which “old” packages are obsolete and can be freed.

Db2 Explain & Explain Tables

infotel Software

Historically, Explain tables have been available since early versions of Db2.

Refer to tables used to capture and store SQL query performance information.

Purpose is to store details about the access path Db2 selects for SQL statements, allowing users to analyze query performance.

Tables are populated by the Explain statement or by setting the EXPLAIN(YES) option when binding or rebinding a package.

The Explain feature is an elementary part of Db2 used by both DBAs and developers alike.

Db2 supported just 2 or 3 Explain tables back in version 1.

In V13 more than 20 tables may be utilized – tendency is growing each release !

Db2 Explain & Explain Tables

When binding packages with EXPLAIN(YES) new entries will be added to Explain tables.

Potential problems:

- If not managed properly, the volume of data held within the Explain tables will continuously grow (and overgrow)
- Especially “common” or general Explain tables shared by many users
- Bind PACKAGE cost will increase
- Each BIND with EXPLAIN(YES) will always create new rows in the associated Explain tables, even if the Access Path didn’t change
- This requires a management system to ensure which rows within the diverse Explain tables should be kept and which ones deleted.

Db2 Explain & Explain Tables

Cleaning-up Explain tables is also a periodic task affecting all Explain tables.

Package and Explain table management works hand-in-hand.
Both must be rigorously managed, in order to avoid self-made problems.

Packages & Explain Tables

Due to the rapid growth of packages and Explain data within your Db2 subsystem clean-ups within the Catalog and Explain tables, as well as Reorgs are required.

ONE single package version has multiple entries (rows) in the Db2 Catalog – e.g.

- Tables - SYSPACKAGE, SYSPACKDEP, SYSPACKSTMT
  SYSPACKSTMT_STMTB, SYSPACKSTMT_STMT
  SYSPACKSTMTDEP and a few more !
- Indexes - DSNKKX01 (U) & DSNKKX02 (U) on SYSPACKAGE
  DSNKDX01 (N) & DSNKDX03 (N) on SYSPACKDEP
  DSNKSX01 (UC) on SYSPACKSTMT
  DSNKSX02 (A) on SYSPACKSTMT_STMTB
  DSNPKX01 (A) on SYSPACKSTMT_STMT
  DSNKNX01 (N) on SYSPACKSTMTDEP
- Tablespaces DB=DSNDB06.TS=SYSTSPKG, SYSTSPKD, SYSTSPKS, SYSTSPVR, SYSTSPKX, SYSTSPSD
- SKPTs (DSNDB01.SPT01)

Packages & Explain Tables

After clean-ups = FREE PACKAGE :

1. Reduce Explain table volume periodically

2. Reorg the Catalog table spaces and indexes previously named - this should be in your scheduled REORGs

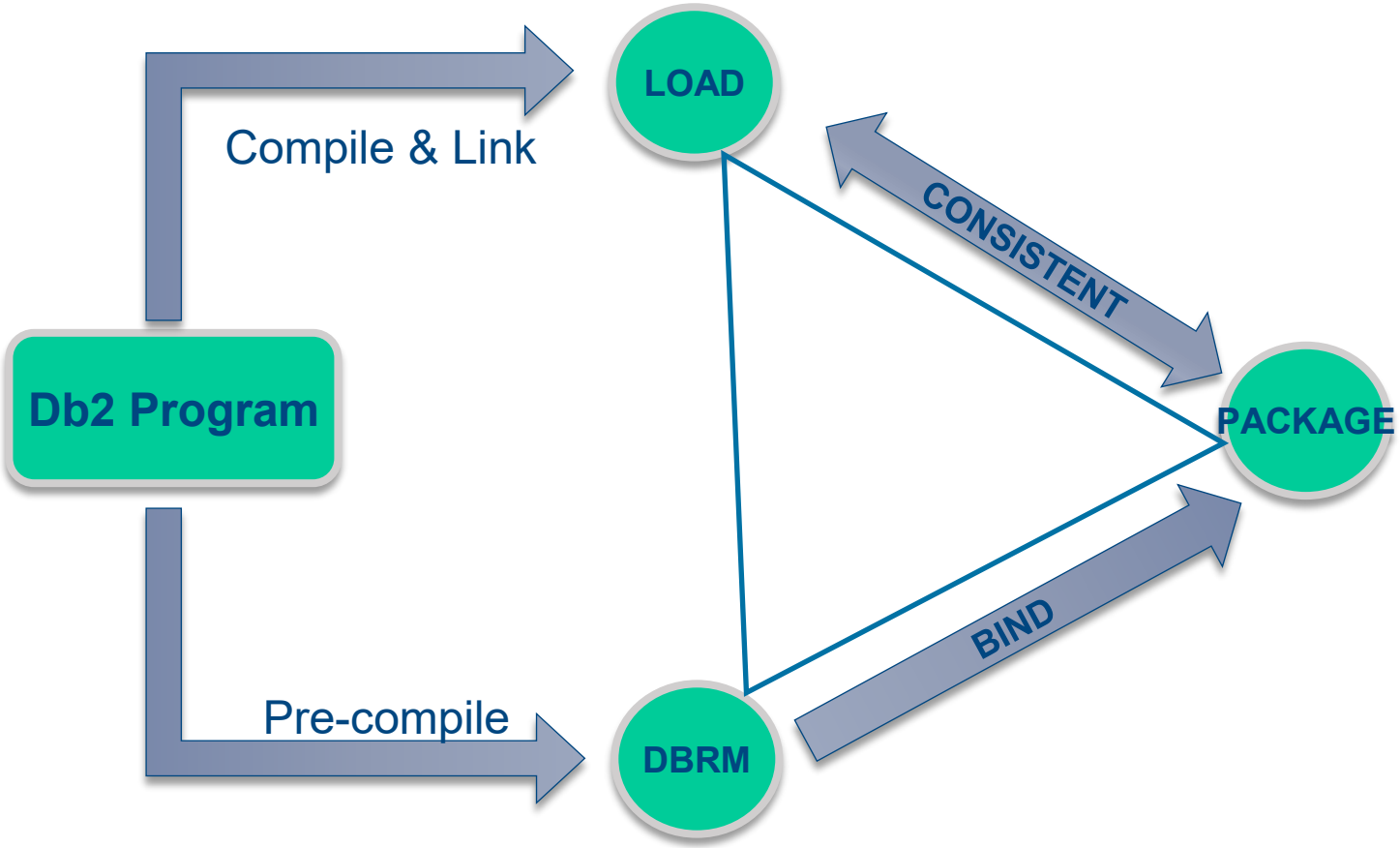
3. Reorg large Explain table spaces / indexes

4. DO IT asap

   DO NOT JUST SAY LATER and
   KEEP IT ON THE WHITE-BOARD

   IT‘S REQUIRED TO GAIN RESULTs

Package Components



Functions and Facilities for Successful Package Management

- Pre-requisites for Successful Package Management
- Functions and Facilities Required

Pre-requisites for Successful Package Management

infotel Software

Targets to be accomplished for successful Package Management or **“what we need to do” ?**

- Ease the problems of package management by easily locating all required LOADLIBs / DBRM-Libs / Db2 package consistency tokens
- Minimize the number of package versions in the Catalog
- Clean up EXPLAIN tables
- Avoid superfluous BINDs
- Create ready-to-use BACKUP & FREE PACKAGE jobs
- Re-create lost or damaged DBRM(s) from the Catalog.

Pre-requisites for Successful Package Management

infotel Software

“but how” can these targets be accomplished ?

1) Build cross-reference data containing

- required package versions and matching LOADLIB modules
- required package versions and matching DBRM modules
- packages accessed by LOADLIB modules
- package versions not used

2) Generate Reports

- packages with matching CONTOKEN found LOADLIB / DBRM modules & vice-versa
- package versions not found in any LOAD or DBRM modules (orphans)
- Load libraries and DBRMs processed
- Explain tables affected with maintenance statistics

Functions and Facilities Required

- Administration User Interface with easy maintenance and functions to include / exclude PDS and PDSE LOAD and DBRM libraries
- Assist user to locate relevant LOADLIBs and DBRM LIBs
- Resolves LOAD modules and related CSECT names together with link & pre-compiler information
- Determine inconsistency between DBRMs, packages and load modules.
- Automatically build BACKUP package jobs and FREE jobs.
- Check DBRM and load mod. consistency when deploying to production

Functions and Facilities Required

- BIND AVOID facilities
- BACKUP option before FREE jobs
- FREE “orphan” and/ or “invalid” packages
- KEEP max. “n” newest versions of a single package FREE “older”
- Clean up inconsistent or “old” Explain Tables
- Re-create lost DBRM(s) from Catalog package information

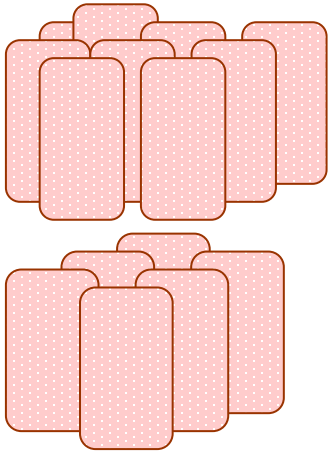
is this a DIY project ?
are there software solutions available ?

Deployment and How to Determine "inconsistency"

- During which process or phase should package management participate?
  - Production deployment
  - Development systems
  - DevOps systems

- What is "inconsistency" and how can it be determined

Production Environment



LOADLIB modules going to production

DBRM modules going to production

PM utilities check consistency between LOADLIB members and DBRM members being deployed

or vice-versa
Package, LOADLIB or DBRM driven in any combination !

Condition code = 0

NO

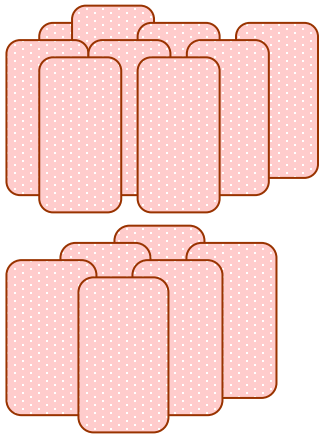
REJECT
-805
POTENTIAL !

YES

MIGRATE to PROD

COPY LOAD modules
BIND packages

Production Environment



LOADLIB modules
in production

DBRM modules
in production

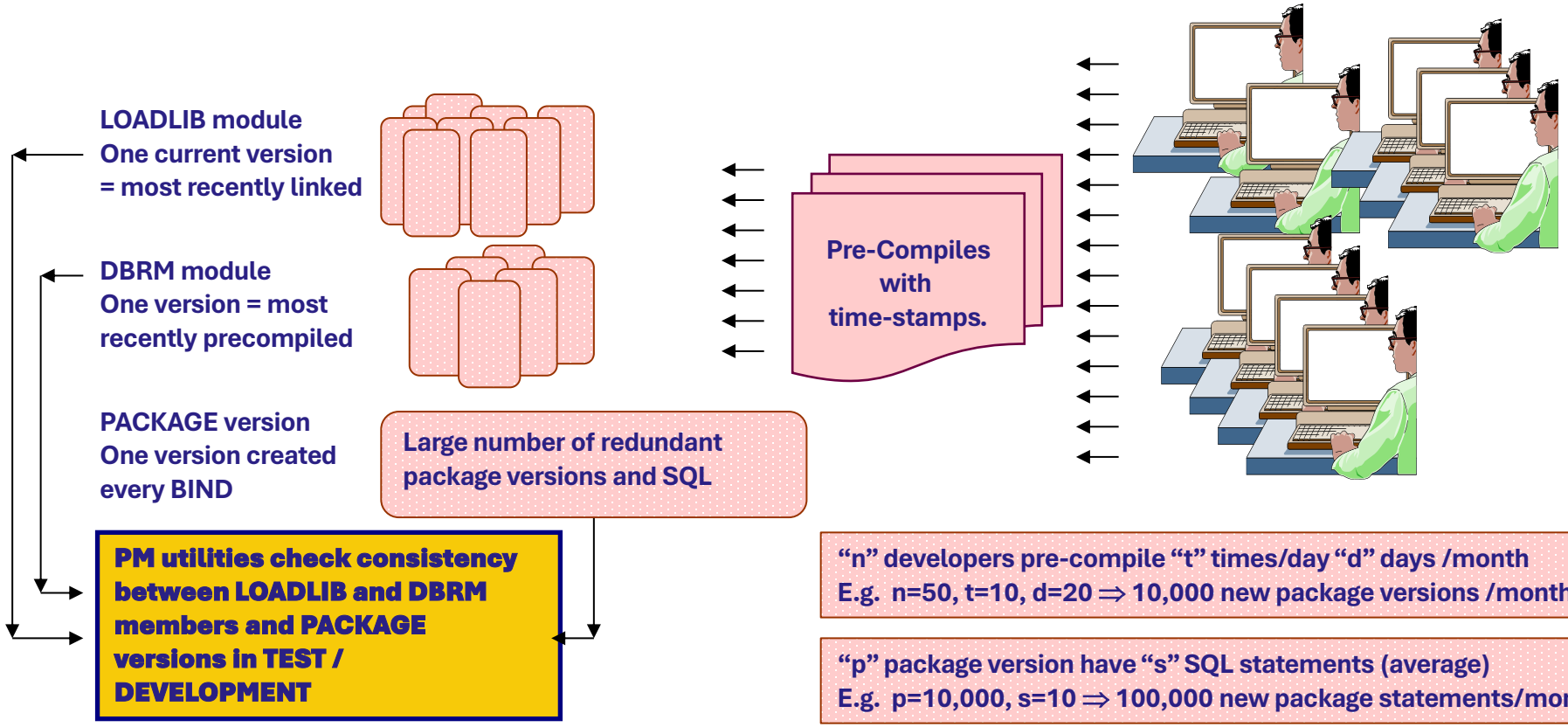
PM utilities should periodically check consistency between LOADLIB and DBRM members and PACKAGEs in PRODUCTION

"old"
ORPHANED
PACKAGEs
found

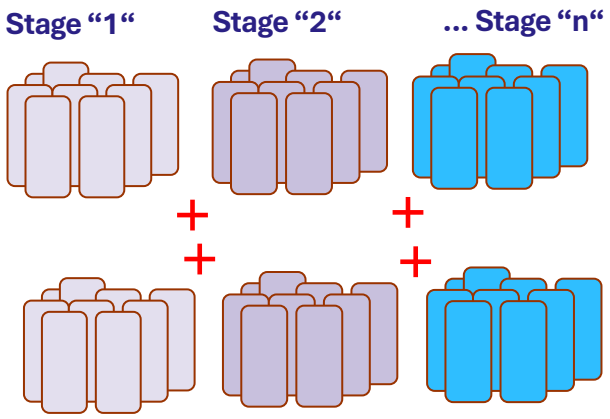
YES

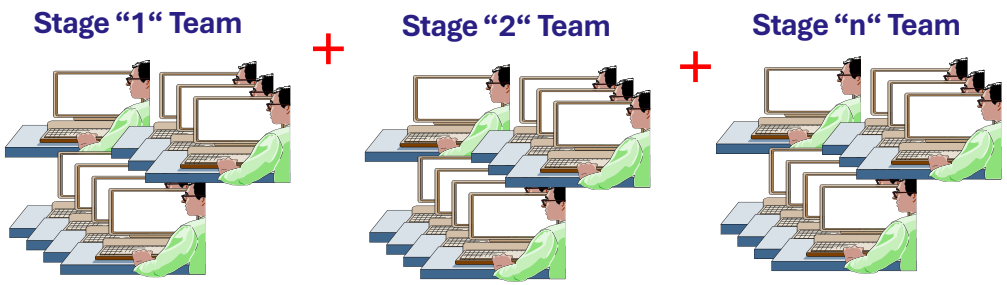
Possible remediation:
FREE "orphaned" PACKAGEs
REORG SYSTSPKx table spaces
"Clean-Up" Explain tables +...

Development Environment



DevOps Environment





Stage x LOAD modules

Stage x DBRM modules

Stage x PACKAGE versions

Stage x Explain Table entries

"n" developers pre-compile "t" times/day "d" days /month
E.g. n=50, t=10, d=20 ⇒ 10,000 new package versions /month

Explain tables contain 1000's of duplicates and superfluous entries

"p" package version have "s" SQL statements (average)
E.g. p=10,000, s=10 ⇒ 100,000 new package statements/month

Which can easily generate 1.000.000 Explain table rows or more

PM utilities check consistency between LOADLIB and DBRM members and PACKAGE versions

FREE "orphaned" PACKAGES & Explain table entries

Relieve Catalog tables: SYSPACKAGE, SYSPACKSTMT, SYSPACKAUTH, SYSPACKDEP + ...
REORG SYSTSPKx table spaces and improve BIND times !

“inconsistency” and how can it be determined in DBRMs, Packages and load modules?

DBRMs are built with a CONTOKEN as an 8-byte char. string (STCK) found in a header-record created by the pre-compiler or compilers with a Db2 coprocessor option (C, C++, COBOL and PL/I)

E.g. `X'1B939B180331E7D2' = 2022-12-26-16.02.31.632115`

- This same CONTOKEN is found in the Db2 package after BIND
- and in the load module when LINKed.
  Again, the pre-compiler does the work of enhancing the application’s source before compiling and includes the CONTOKEN in the object code.

So what is “inconsistency” ?

“inconsistency” means :

The CONTOKEN of a module(*) / DBRM member / PACKAGE

is not the same

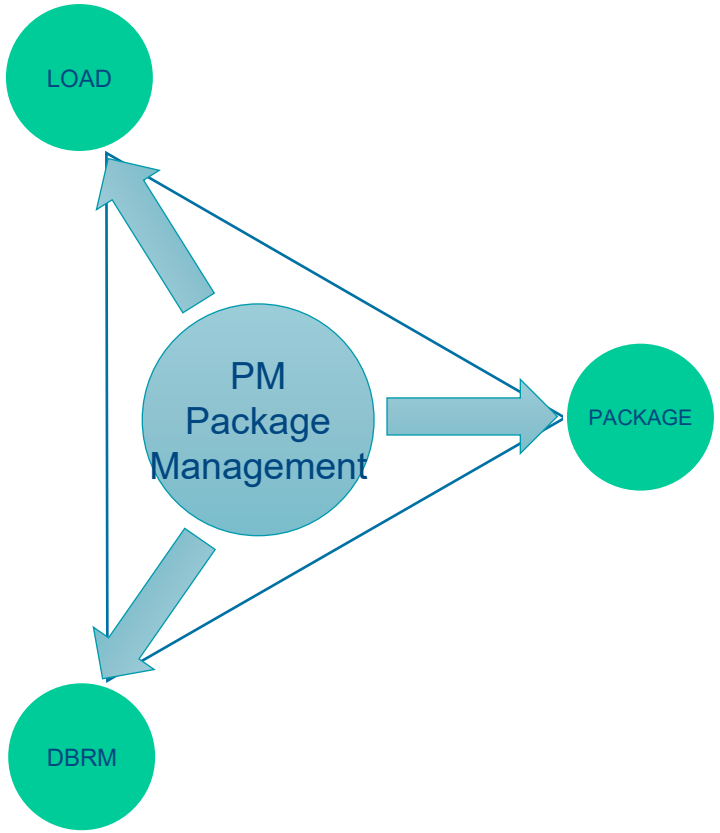
(*) Can be a load module or CSECT within a load module

Executing an “inconsistent” module will cause a -805 SQL error !

Finding “inconsistency” between DBRMs, Packages and load modules

infotel Software

- Determine “inconsistency” between DBRMs, Packages and load modules
- Identify Db2 collection.package versions no longer required
  - with no matching LOADLIB module / CONTOKEN
  - with no current matching DBRM / CONTOKEN.
- Optionally build FREE PACKAGE job(s) including BACKUP package jobs before FREE
- Locate all -805 potential



Finding “inconsistency” between DBRMs, Packages and load modules

infotel Software

HOWEVER – this is where it starts to become complex !

Why so complex?

- The same DBRM may have been bound as a package in multiple collections
- The loadlib module may have been linked with other modules under a different name in a different loadlib. Meaning the name of the DBRM will only be found as a CSECT and not the module name itself.

How can we correlate the DBRM or PACKAGE name to a LOAD module ?

- Locate the required load libraries via the z/OS Catalog Search Interface ✓
- Resolve all modules within all required load libraries, including their CSECT names ✓
- CSECT name = Package name ✓ = gotcha !

A CONTOKEN is a “consistency token” – so why so complex?

infotel Software

- The CONTOKEN built into the loadlib module depends on the programming language and compiler being used. Programming languages supported include :
  C, C++, COBOL (diverse), PLI, ASM & Fortran
- The enhanced source, created by the pre-compiler or coprocessor includes the generated CONTOKEN, but depending upon the compiler, stores the CONTOKEN differently :
  - 8-bytes as in found in the original DBRM - ABCDEFGH - native STCK
  - 8-bytes swapping bytes 1-4 & 5-8 - EFGHABCD - some COBOL compilers
  - 2 x 4 bytes swap and re-string - GHEF, CDAB - some C and C++ compilers
  - 4 x 2-byte pairs (each on HWD boundary) - AB, CD, EF, GH more C and C++ compilers
  - some exceptions and more … obviously due to different compiler development teams!

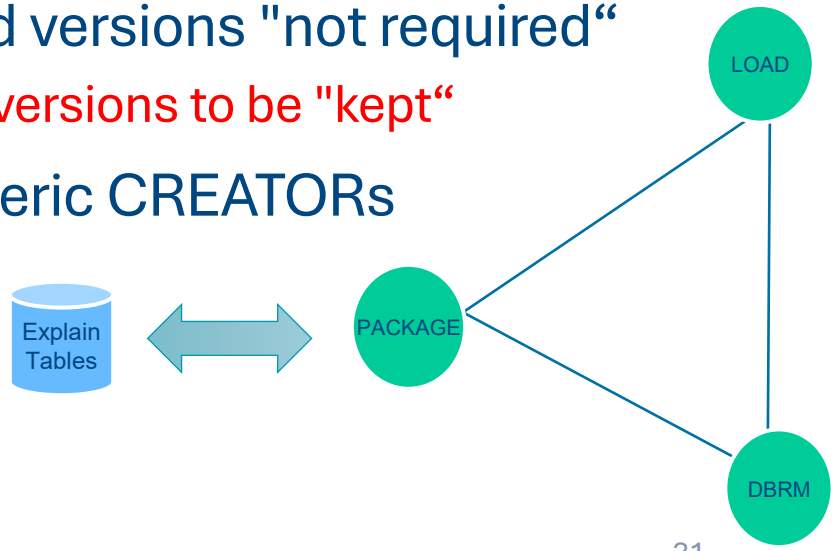
Explain Table Management and even more …

- How to Manage Explain Tables
- Re-create DBRMs from Catalog information
- Avoid superfluous and non-required BINDs
- Revision of what we've talked about

How to Manage Explain Tables

There are several options – whether executed in combination or individually

- Ensure consistency between all Explain tables and the package information in Catalog
- Remove all entries within the Explain tables, which are older than “nnn“ days
- Remove all Explain table entries belonging to explained versions "not required“
  “Not required" is indirectly defined by stating the number of versions to be "kept“
- Find and consider “all“ Explain tables - different or generic CREATORs
- Avoid huge Explain tables holding useless entries



How to Manage Explain Tables

Standard Explain tables considered

- PLAN_TABLE
- DSN_STATEMNT_TABLE
- DSN_FUNCTION_TABLE

- DSN_STATEMENT_CACHE_TABLE
- DSN_STAT_FEEDBACK

and even more Explain tables

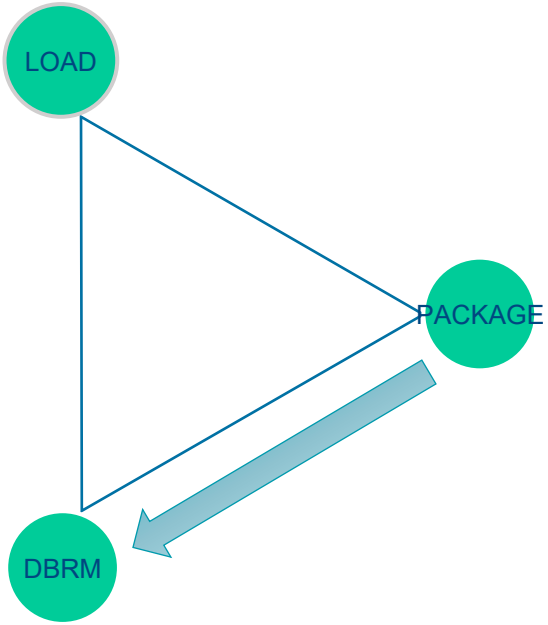
- DSN_COLDIST_TABLE
- DSN_DETCOST_TABLE
- DSN_FILTER_TABLE
- DSN_KEYTGTDIST_TABLE
- DSN_PGRANGE_TABLE
- DSN_PGROUP_TABLE
- DSN_PREDICAT_TABLE
- DSN_PREDICATE_SELECTIVITY table.
- DSN_PTASK_TABLE
- DSN_QUERYINFO_TABLE
- DSN_QUERY_TABLE
- DSN_SORTKEY_TABLE
- DSN_SORT_TABLE
- DSN_STRUCT_TABLE
- DSN_VIEWREF_TABLE

- And continuously being enhanced by Db2

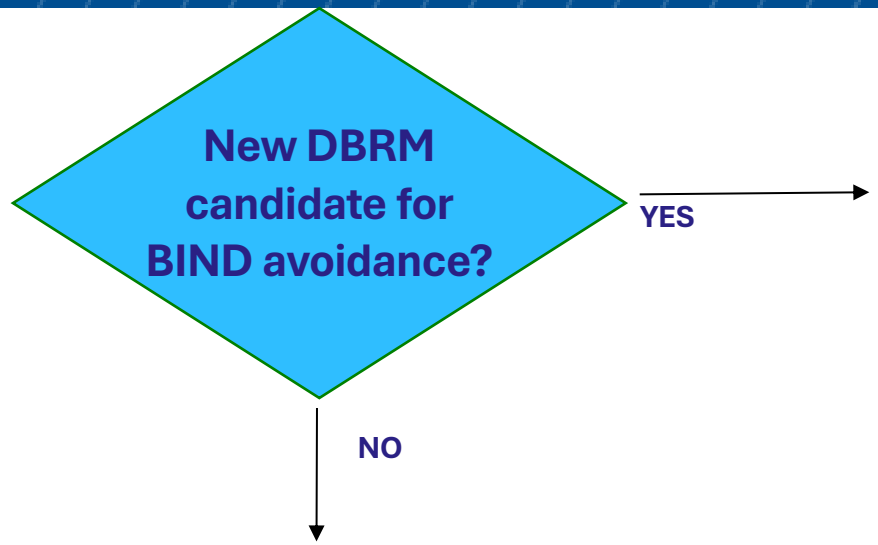
Re-create DBRMs from Catalog information

- Need to BIND a package and the DBRM has gone “lost“
- DBRM library required for migrational purposes
- Source is the Db2 Catalog
- One of more packages from one or multiple collections as base.

  All DBRM information required can be located from within the package & package statement data.



Avoid superfluous and non-required BINDs



If the number of SQLs or the SQL text elements have changed, then the BIND is required

Candidates are nominated when the OLD and NEW DBRMs have the same number of SQL statements and none of these have changed.

Note: changes in STMTNO are irrelevant.

If the number of SQLs and the SQL text elements are the same there is no need to BIND, but the CONTOKEN needs to be reset to the previous value in the source and object code:

- Source code & object code must be modified to match the existing CONTOKEN / VERSION information
- The DBRM must be reset to the previous version

The linked module now contains the old previous CONTOKEN

Revision of what we've talked about so far

- Administration and locating all required LOADLIB / DBRM- and Db2 package information
- Determine "inconsistency" between DBRMs, packages and load modules.
- Determine non-required PACKAGEs and automatically build BACKUP and FREE jobs.
- FREE "orphan" and/ or "invalid" packages
- KEEP max. "n" newest versions of a single package and FREE "older"
- Say "goodbye" to -805 in production systems by checking inconsistency of LOADLIB and DBRMs during deployment

Clean up Db2 Catalog's largest objects – a pre-requisite for REORGs

Ease DBA work

Clean up development and test environments

Keep your production free of inconsistency and -805s

Revision of what we've talked about so far

- Clean up inconsistent or "old" Explain tables
- Re-create "lost" DBRM(s) from Catalog package information
- Avoid Binds when not-required, saving CPU and Catalog processing

Reduce space consumption, save CPU

Ensure Explain table maintenance

Save BINDs – save CPU

Success Story at one of the world‘s largest Db2 sites

infotel Software

Client requirements :

- Easy maintenance with an Administration User Interface
- Assist to locate LOAD and DBRM libraries
- Generate jobs for mass processing
- Search and scan through very large PDSE load libraries
- Determine “inconsistency” between DBRMs, packages and load modules
- Automatically build BACKUP and FREE jobs
- Clean up inconsistent or “old” Explain tables
- Re-create “lost” DBRM(s) from Catalog package information
- Avoid Binds when not-required, saving CPU and Catalog processing

Success Story at one of the world‘s largest Db2 sites

German Banking IT service provider and digitalization partner

- 450 savings banks, 9 state banks, 10 state building societies and regional insurance companies
- Turnover € 2.58 Billion
- 5000 employees
- One of the world’s largest data centres– ca. 400K MIPS. 205 billion tech. transactions p.a.
- Standard solution developed for regional savings banks in COBOL, C and Java
- 80+ production Db2 subsystems. Multiple level test and development systems
- 2 or more new releases of their standard banking solution deployed each year.
- Each release involves more than 25000 packages in each production system.

Success Story at one of the world‘s largest Db2 sites

infotel Software

German Banking IT service provider and digitalization partner

Situation and Challenge

- Home-grown solution was out-dated and could not support C and SQLJ DBRMs
- DevOps development with multiple application stages. Each stage with multiple versions
- Each development stage with own *stage*.LOADLIB, *stage*.DBRMs, *stage*.COLLECTION etc.
- All *stage*.load.libraries are then concatenated at run-time, in order to find the matching module for an executed package - but which package versions are no longer required?
- All packages are bound with EXPLAIN(YES) causing more rows than needed in their Explain tables -> huge Explain tables.
- Out-dated non-required packages had to be automatically deleted to avoid consiberable amounts of DASD being consumed

Success Story at one of the world‘s largest Db2 sites

German Banking IT service provider and digitalization partner

Results – PRODUCTION SYTEMS

- The bi-annual release resulted in 25000 out-dated packages being freed after building a backup, which was kept for a defined period - for 80+ Db2 subsystems
- Tens of thousands of Explain table rows were deleted
- All non-consistencies found for the Cobol, C and Java SQLJ applications were now found Several PDSE load libraries to be searched exceeded 100 K modules each.

Success Story at one of the world‘s largest Db2 sites

infotel Software

German Banking IT service provider and digitalization partner

Results – TEST & DEVELOPMENT SYSTEMS

- Due to the naming conventions used for the package versioning, thousands of packages could be freed every day from the development subsystems
- The problems involved with *stage*.load.libraries and their package versions were finally solved
- Up to 25000 “old” Explain table rows deleted daily

Conclusion

Package Management is a “safety-first” and “must have” for all z/OS Db2 sites
There is software on the market to:

- Maintain all requirements with an integrated Administration User Interface
- Determine “inconsistencies” between DBRMs, packages and load modules
- Automatically build BACKUP and FREE jobs
- Clean up inconsistent or “old” Explain tables
- Re-create “lost” DBRM(s) from Catalog package information
- Avoid Binds when not-required, saving CPU and Catalog processing

Come and see us at our booth for more information, discussions, presentation or demo

Thank you !

Colin Oakhill
colin.oakhill@insoft-infotel.com

Infotel
Le Valmy, 6/8
18 Av. Léon Gaumont
75020 Paris
France

Infotel Corporation
424 22nd Ave N
St Petersburg, FL 33704
USA

Insoft Infotel Software
Sternstraße 9-11
40479 Düsseldorf
Germany

Infotel UK
Balliol Business Park,
Newcastle upon Tyne NE12 8EW,
United Kingdom

Infotel IT Consulting
Ramanujan IT City, Neville Tower,
Chennai, Tamil Nadu, 600113
India