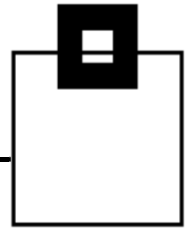
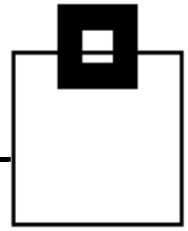


Carefree Migration
–
Setting up a DB2 Precheck Environment

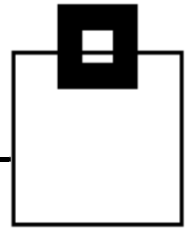
Ulf Heinrich
SEGUS, Inc
u.heinrich@segus.com

Agenda



- Death of plans, have no fear
  - What will happen to plans during the migration
  - How the DB2 automatic rebinds work and how you can avoid them!
- Pinpointing all areas of degradation risk
  - Plans containing DBRMS
  - Plans and packages bound prior to V7
  - Static and dynamic SQL tuning needs
- Predicting access path degradations
  - Setting up a DB2 10 environment
  - Prechecking access paths prior to migration
  - Looking for typical access path change patterns

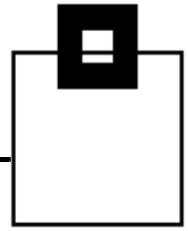
DB2 for z/OS version migration



Each new version of DB2 has advantages and disadvantages regarding performance and resources – because of enriched features and functions …

- Synergy with z platform
- Safe query optimization
- Query/Access Path Performance Enhancements
- Performance improvements from distributed data facility

DB2 for z/OS version migration

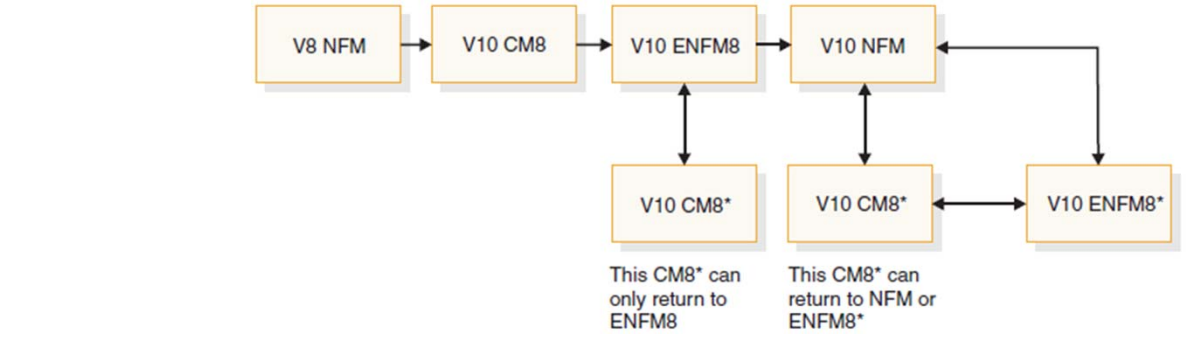


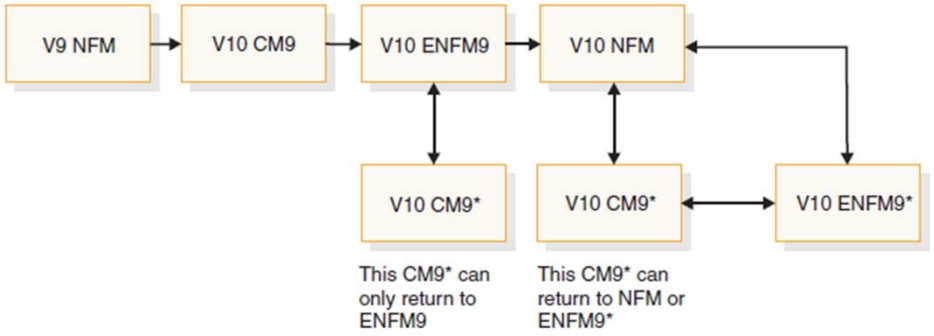
Being well prepared reduces the risk of a bad migration

- Check prereq's (z/OS, DB2 APARs, RACF,...)
  - fallback SPE APAR PK56922
    see also information APARs II14474/II14477
- Verify removed/deprecated features and functions
- Verify catalog statistics
- Pre-migration check DSNTIJPM discovers gotchas to lookout for
- Pre-migration job DSNTIJPA (delivered via APAR PM04968) discovers cleanup needs

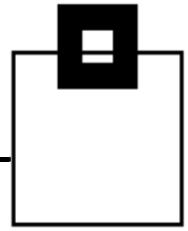
DB2 for z/OS version migration

If you experience a severe application or performance error in DB2 10 you need to fallback to DB2 V8, or DB2 9.





DB2 for z/OS version migration

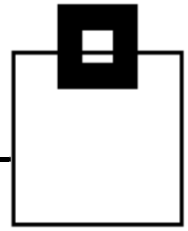


Beside the rare case where real errors force the need of a fallback, more often performance issues occur as a global **REBIND** is requested.

*Most performance improvements are implemented by migrating to DB2 10 and **rebinding***

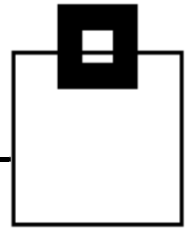
- If you migrate and rebind plans or packages, the CPU savings can be twice as much than without **rebinding**
- Some enhancements are implemented through normal DB2 activities through **rebinding**
- **REBIND** is needed to obtain the best performance and memory improvements
- Early DB2 10 performance benchmarking and customer experience has shown a 5 to 10% CPU reduction in transactions after **rebinding**
- **REBIND is not required for migration to DB2 10, but REBIND is strongly recommended**

DB2 for z/OS version migration



- The enhancements to aggressively merge views and table expressions to avoid materialization to work files is available in conversion mode after **rebind**.
- Getting the best performance improvements and eliminating regression does depend upon **rebind** in most situations
- STMTID support (as well as 401 generation) requires **REBIND** in NFM
- Eliminating performance regression may depend upon **REBIND**
- To use the enhanced monitoring support functions, you must **rebind** or bind any existing pre-DB2 10 package in DB2 10 new function mode
- Storage constraint relief depends upon **REBIND**
- Changing to use RELEASE(DEALLOCATE) requires a **REBIND**
- All plans containing DBRMs **must be rebound**
- The residual predicate enhancements are available in conversion mode after a **rebind**.
- All packages that were last bound on V5 or lower **must be rebound**
- Static SQL statements with DEGREE(ANY) for parallel processing **should be rebound, or it will be serial.**
- Improvements in access paths can be very significant, such as stage 2 predicates that can become stage 1.

DB2 for z/OS version migration



DB2 10 for z/OS Performance Topics

"DB2 10 delivers by improving performance and reducing CPU usage. Most customers can achieve out-of-the-box CPU savings of 5 to 10 percent for traditional workloads and up to 20 percent for specific workloads."

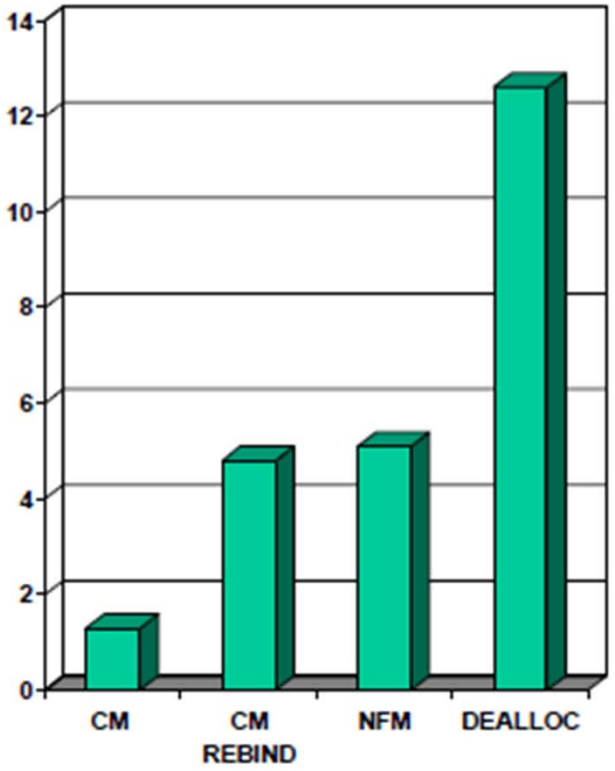
REBIND is needed to obtain the best performance and memory improvements".

DB2 for z/OS version migration

Measurements of IBM Relational Warehouse Workload (IRWW) with data sharing

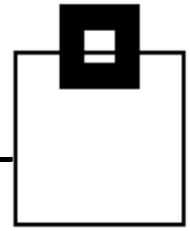
Base: DB2 9 NFM REBIND with PLANMGMT EXTENDED

- DB2 9 NFM → DB2 10 CM without REBIND showed 1.3% CPU reduction
- DB2 10 CM REBIND with same access path showed 4.8% CPU reduction
- DB2 10 NFM brought 5.1% CPU reduction
- DB2 10 CM or NFM with RELEASE DEALLOCATE 12.6% CPU reduction from DB2 9



CPU Percent reduced from DB2 9

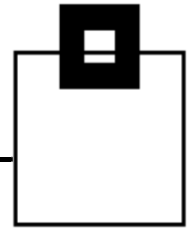
DB2 for z/OS version migration



DB2 9 has already started the process of moving the plan and package static statement storage above-the-bar

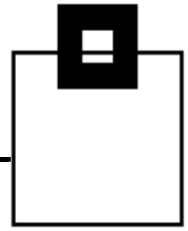
- Depending on the statement 5% - 90% moves
- To achieve the VSCR in the below-the-bar storage for the CT and PT, you need to rebind. The storage below the bar associated with CT and PT pages only occurs for plans and packages that are not rebound on DB2 10. Once they are rebound, this storage is allocated above-the-bar
- In addition, to get the most benefit from virtual storage relief, native SQL procedures created on DB2 9 need to be regenerated on DB2 10

DB2 for z/OS version migration



>> So please be kind. REBIND! <<

DB2 for z/OS version migration



Some gotchas to watch out for:

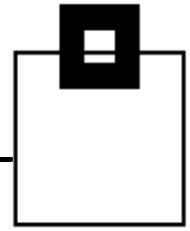
- Binding DBRMs directly into plans is no longer supported

  → For pre-existing plans you can use the COLLID parameter of the REBIND PLAN command to create packages.

  If you execute a plan that is bound from DBRMs, DB2 performs an automatic rebind that creates packages from the DBRMs and binds those packages into a plan. (DSN_DEFAULT_COLLID_*plan-name)*

However, the recommendation is to use REBIND with the COLLID option.

DB2 for z/OS version migration



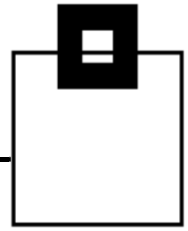
Some gotchas to watch out for:

- The COLLID option of REBIND PLAN binds the DBRMs to packages, and binds the packages to the specified plan. COLLID applies only to plans to which DBRMs are bound directly.

- If the installation uses the RACF access control module, owners of plans with DBRMs need to explicitly rebind the plans to convert the DBRMs to packages.

Check APARs PM62876 & PM79925

DB2 for z/OS version migration



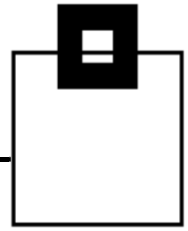
Some gotchas to watch out for:

- Plans and packages created before DB2 10 containing static SQL statements using parallelism
  → DB2 incrementally rebinds those packages and plans after migration

Incremental rebinds can cause performance degradation, so you should manually rebind

You should consider rebinding those packages after migration, as soon as your Version 10 system is stable.

DB2 for z/OS version migration



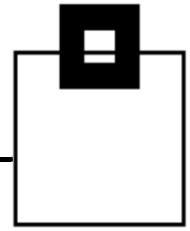
Some gotchas to watch out for:

- Queries with the LIKE predicate often evaluated by DB2

  → The predicate evaluation enhancements are available in CM without rebinding, but rebinding avoids having to generate the machine code dynamically at execution time

You should consider rebinding those packages after migration, as soon as your Version 10 system is stable.

DB2 for z/OS version migration



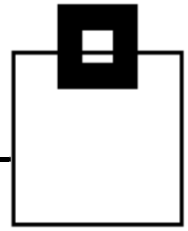
Some gotchas to watch out for:

- DB2 10 allows an access plan to overflow to work files and continue processing RIDs, even when one of the RID thresholds are encountered at run time.

  → The RID pool work file overflow enhancement is available in conversion mode. A rebind of applications is not required

However, you're advised to rebind or bind applications to reset the RID thresholds that are stored within a package.

DB2 for z/OS version migration



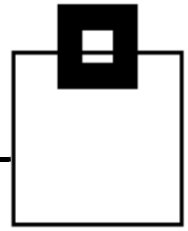
Some gotchas to watch out for:

- Plans and packages from DB2 V5 or earlier need to be rebound

  →DB2 automatically rebinds them if you set DSNZPARM ABIND=YES or COEXIST (AUTO BIND field of panel DSNTIPO)

  You might experience an execution delay the first time that such a plan is loaded. Also, DB2 might change the access path due to the autobind, potentially resulting in a more efficient access path.

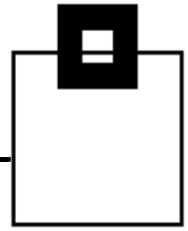
DB2 for z/OS version migration



Some gotchas to watch out for:

The (out-of the-box) improvements do require a REBIND in most situations, and that does mean checking and testing, but DB2 version changes also take testing, so combining the work for a dramatic improvement will work for many customers

DB2 for z/OS version migration



Bottom Line:

The DB2 Optimizer has improved algorithms and a rewritten approach to handle performance information for tuning and for exceptions.

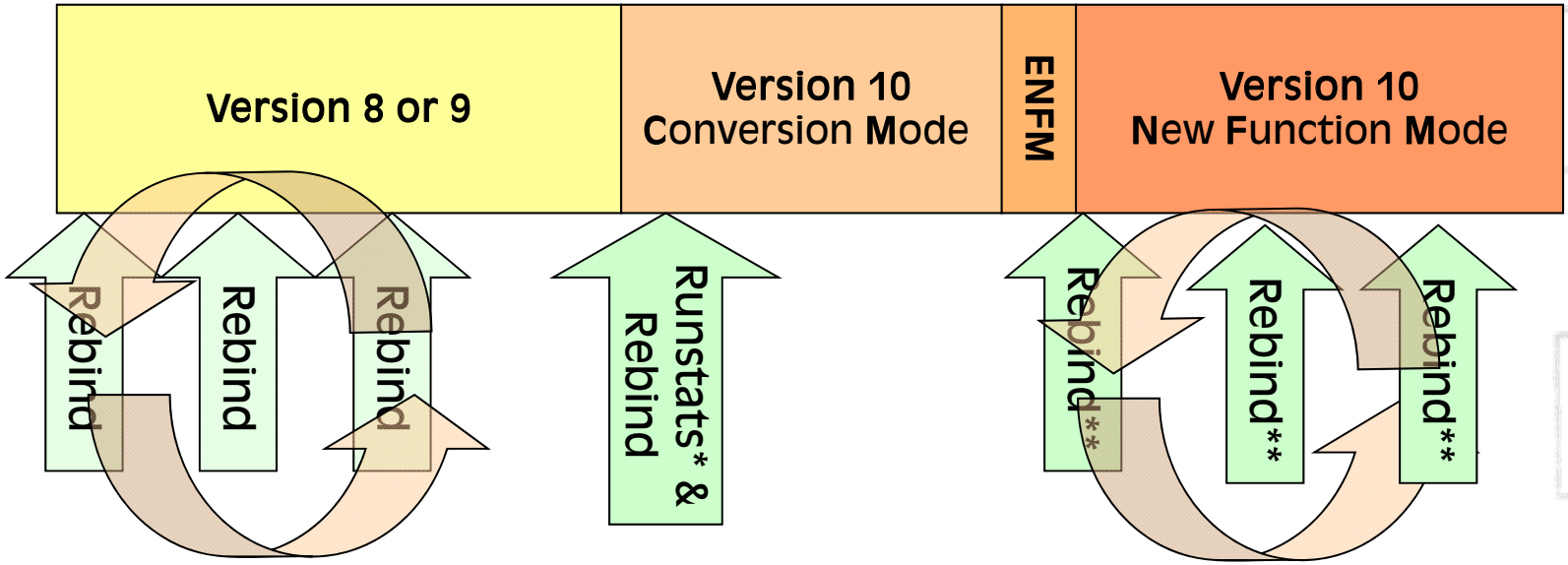
Improved algorithms widen the scope of optimization.

→ Exploit the enhancements you paid for!

DB2 for z/OS version migration

When to REBIND?

- Avoid incremental/auto- (RE-)BINDs



* Consider the enhanced RUNSTATS when doing a skip migration (new clusterratio, DRF)

** like the 3R's in operating your DB2

DB2 for z/OS version migration

How to REBIND?

… so there are strong reasons to exploit the new features and REBIND, but a REBIND can be surprising!

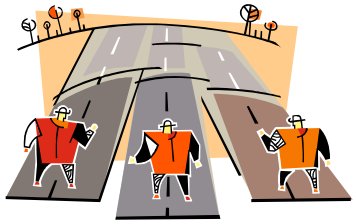
How to get the many improvements in performance of DB2 10, without the risk of regressions?

DB2 for z/OS version migration

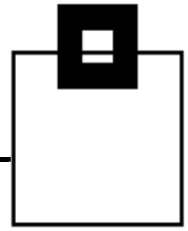
How to secure access paths during version migration?

Options:

1. Trial and error – may be risky, only possible when you have enough time/resources to play around
2. No REBINDs at all – no exploitation of enhancements already paid for
3. Pre-Check the results – most controllable and most efficient



Securing Access Paths



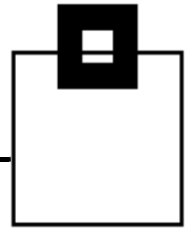
– Option 1 Trial and error

DB2 10 REBIND can compare the new and old access paths at bind or rebind processing and indicate that a warning or error is issued when an access path changes (APREUSE/APCOMPARE)*.

→ REBIND in DB2 10 takes more CPU and elapsed time, but more concurrent REBINDs are possible (NFM only)

*Requires APAR PM33767, PM25678
There is no guarantee to succeed
Requires package to be rebound at least under DB2 9

Securing Access Paths



– Option 1 Trial and error

Can Package Stability help?

Package Stability is an access path backup!

Pro's

- Easy fallback

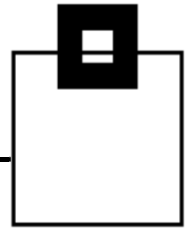
Con's

- 15% - 20% CPU overhead for each REBIND
- 2 – 3 times more storage requirements for DSNDB01.SPT01

DB2 10 changes the default for PLANMGMT from OFF (in DB2 9) to EXTENDED.

→ setting PLANMGMT=OFF in DB2 10 falls back to pre 10 default

Securing Access Paths



– Option 1 Trial and error

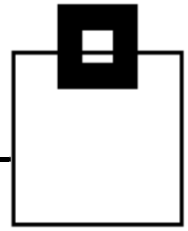
Preparation using Package Stability:

- Make sure you have plan table data for all your access paths REBIND EXPLAIN(YES)

- Make sure you have APAR PK52522 applied to your DB2 V8 (if doing skip migration)

- Use ZPARM PLANMGMT default (EXTENDED)

Securing Access Paths



– Option 1 Trial and error

Migration using Package Stability:

- Get DB2 10 CM stable
- Set ZPARM STATSCLUS to ENHANCED (default)*
  - new clusterratio calculation + DRF
- (Run extended RUNSTATS)*
- Execute global REBINDs

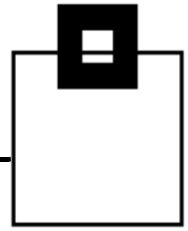
→ Current access path is DB2 10

→ Previous access path is a prior DB2 10 access path or prior version

- Original access path will remain your prior version access path

*if doing skip migration

Securing Access Paths



– Option 1 Trial and error

Fallback using Package Stability:

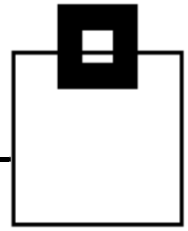
- Do a REBIND PACKAGE … SWITCH(ORIGINAL)

Cleanup:

- Do a FREE PACKAGE … PLANMGMTSCOPE(INACTIVE)

If you are satisfied with the resulting access paths move on to NFM

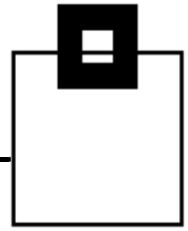
Securing Access Paths



– Option 2 No REBINDs at all

>> ***To rebind or not to rebind, "when?" is the question****: Whether 'tis nobler in the mind to suffer the slings and arrows of outrageous access paths, Or to take arms against a sea of troubles, And by rebinding, end them? To die: to sleep; No more; and by a sleep to say we end the heart-ache and the thousand natural shocks that old plans are their to. 'Tis a consummation devoutly to be wished to die, to sleep. To sleep: perchance to dream: aye, there's the rub; For in that sleep of death what dream may come when plans have shuffled off this mortal coil, must give us pause: There's the respect that makes calamity of so long life for old plans and packages.*<< by Roger Miller

Securing Access Paths



– Option 2 No REBINDs at all

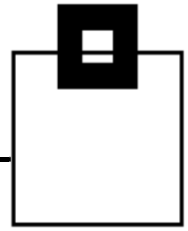
No REBINDs at all is NOT the solution!

You'll degrade performance by not REBINDing! (e.g. fast column processing – SPROCs, VSCR, etc. etc.)

Make sure you do

- Global REBINDs after DB2 version migration
- Global REBINDs after DB2 system maintenance
- Practice the 3Rs – REORG → RUNSTATS → REBIND

Securing Access Paths

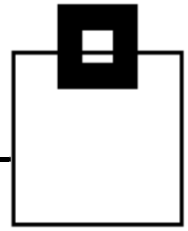


– Option 3 Pre-check

- Goal: Detection of potential worsened access paths before migration of the production system by comparing the access paths of a prior DB2 production system with the access paths on a DB2 10 test system

- based on the original production statistics, CPU, # of CPs, BPs, rid & sort pool,…)*
- without affecting the production system
- at the earliest possible stage
- for static and dynamic SQL

*just became possible by production modeling
V9 APAR PM26475 & V10 APAR PM26973



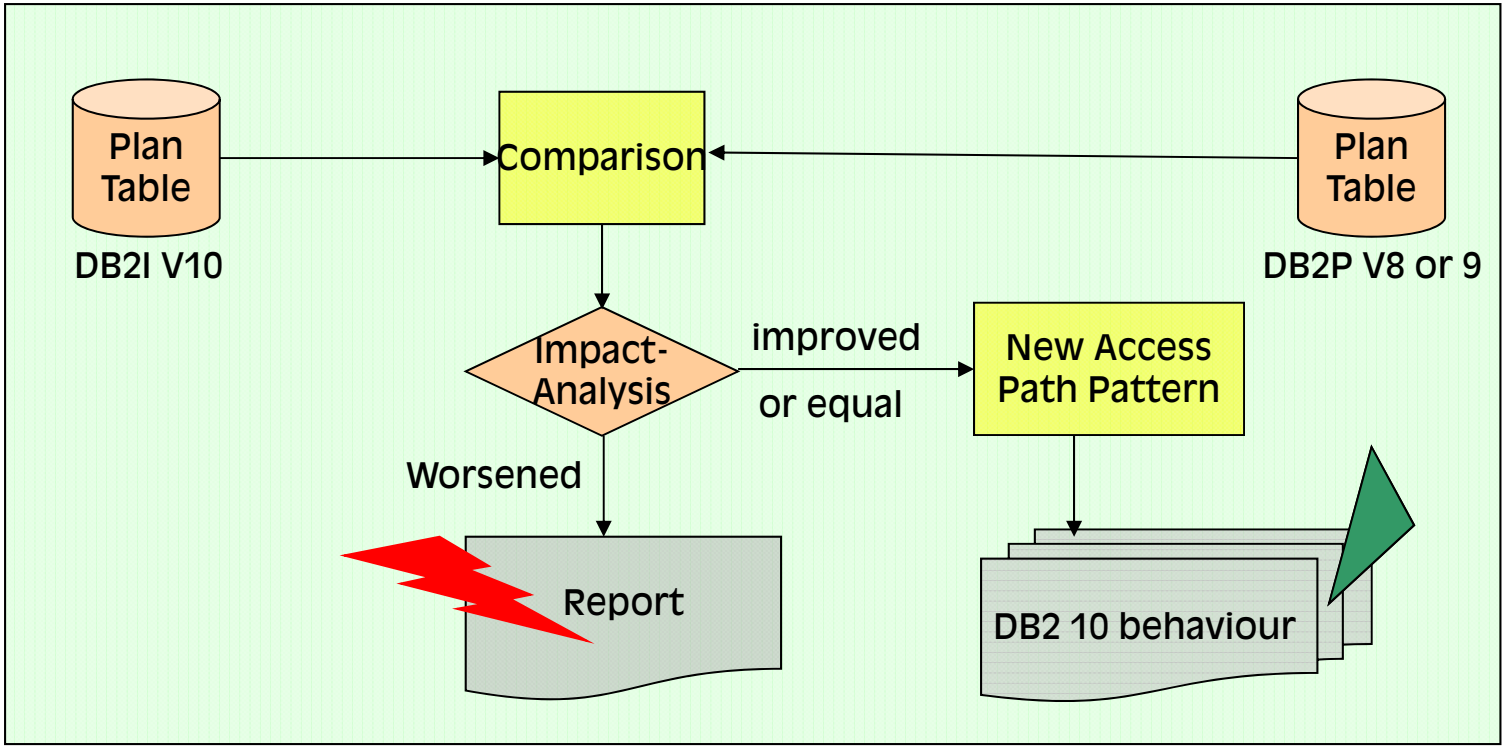
Setting up a Precheck system

for Static SQL

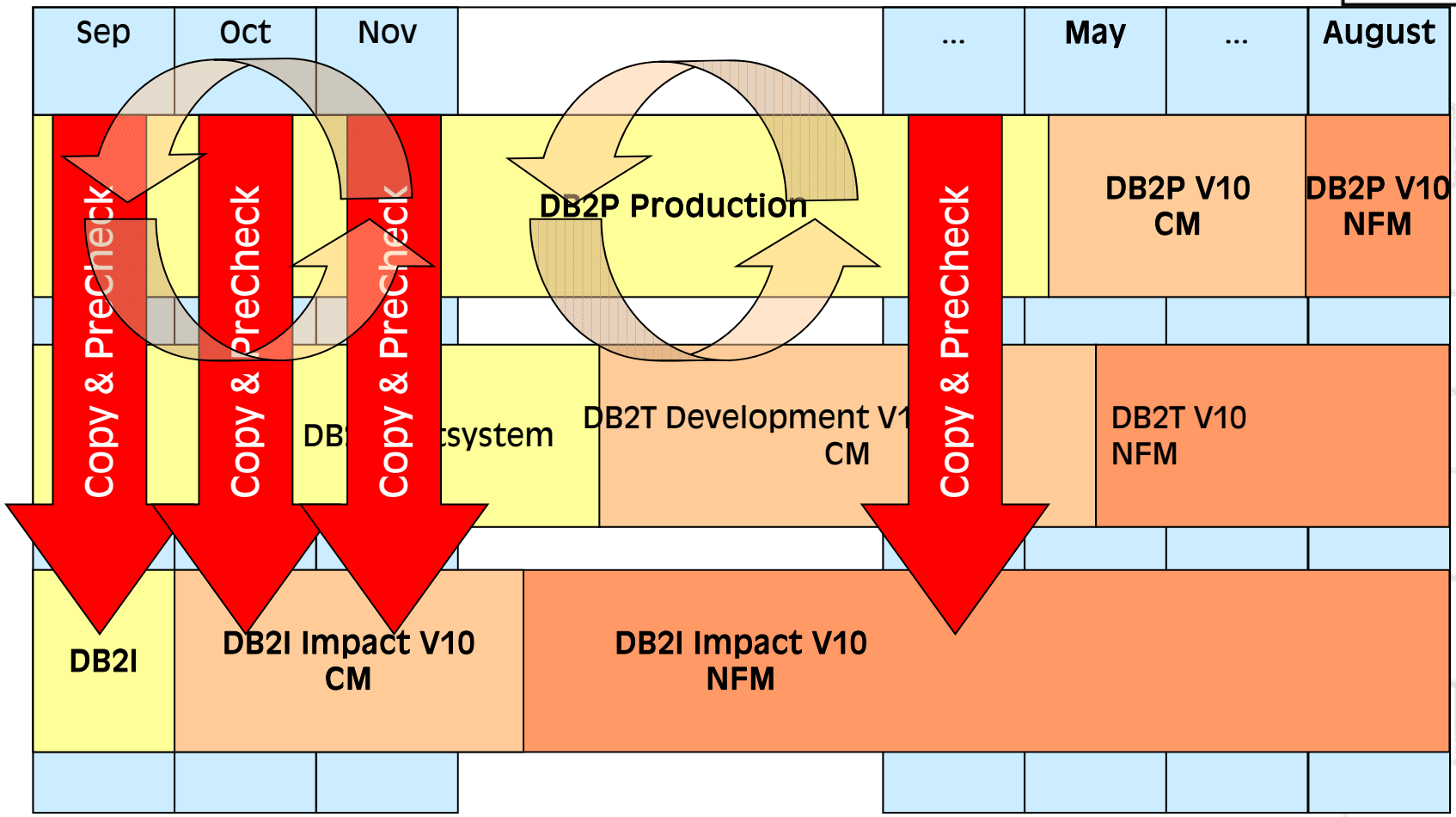
&

for Dynamic SQL

Method of SQL Performance Precheck



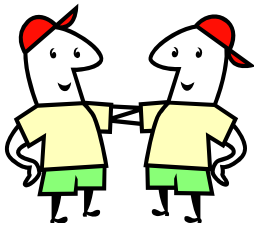
Repeatable SQL Performance Precheck



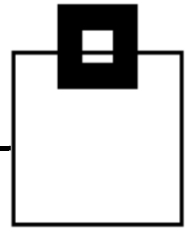
Setting up a Precheck System

Gathering data for the simulation environment

- DB2P: Production
  - Clone if you'd like to be able to run your apps
  - Extract all/subset DDL
  - Extract the relevant catalog statistics
  - Extract CP speed, # of CPs, BPs, RID & Sort pool
  - Static SQL:
    - Extract Packages and PLAN_TABLE data
  - Dynamic SQL:
    - Take a snapshot of the Dynamic Statement Cache
    - Explain all captured statements to a temporary PLAN_TABLE



Setting up a Precheck System

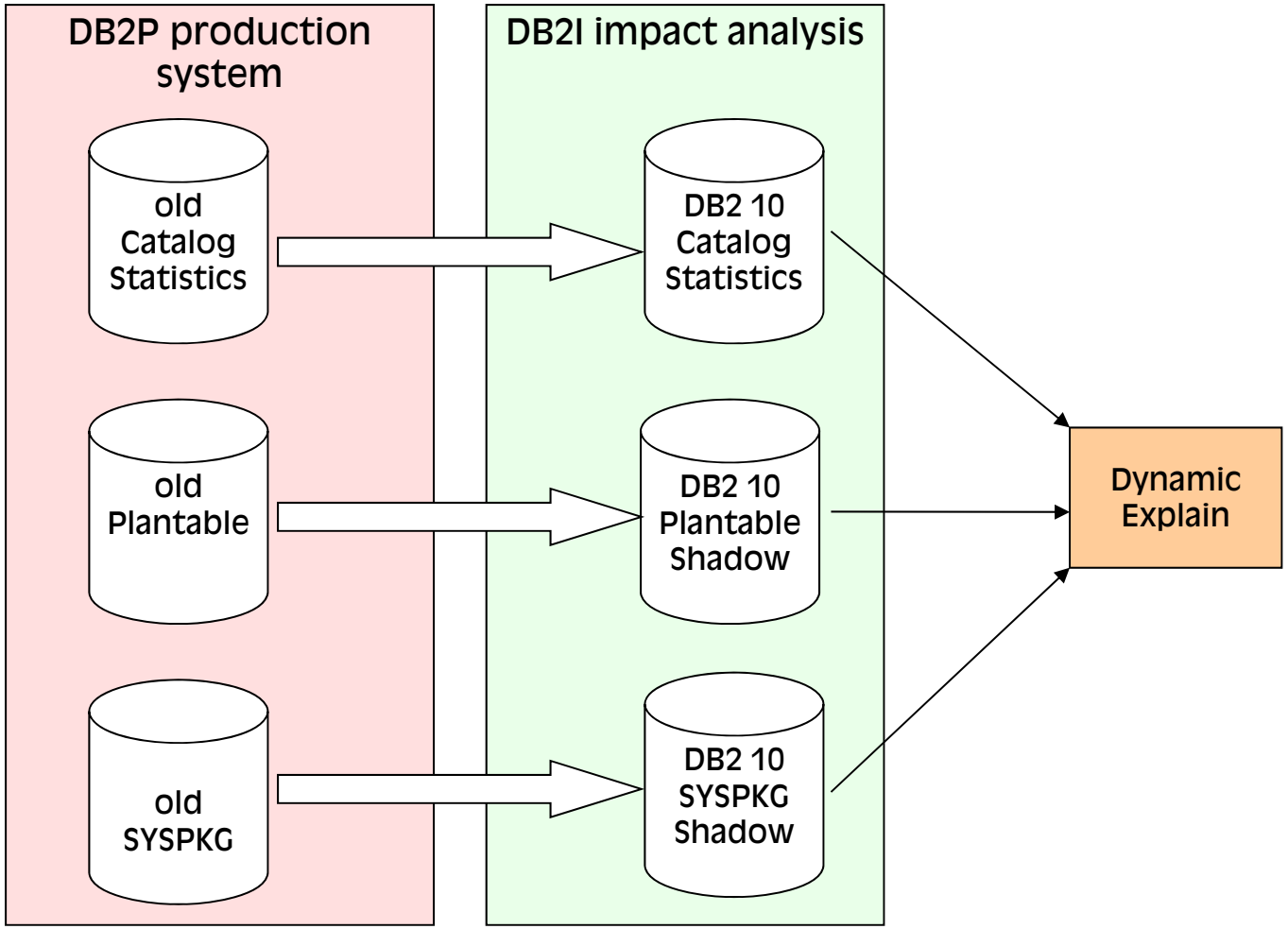


Applying data to the precheck environment

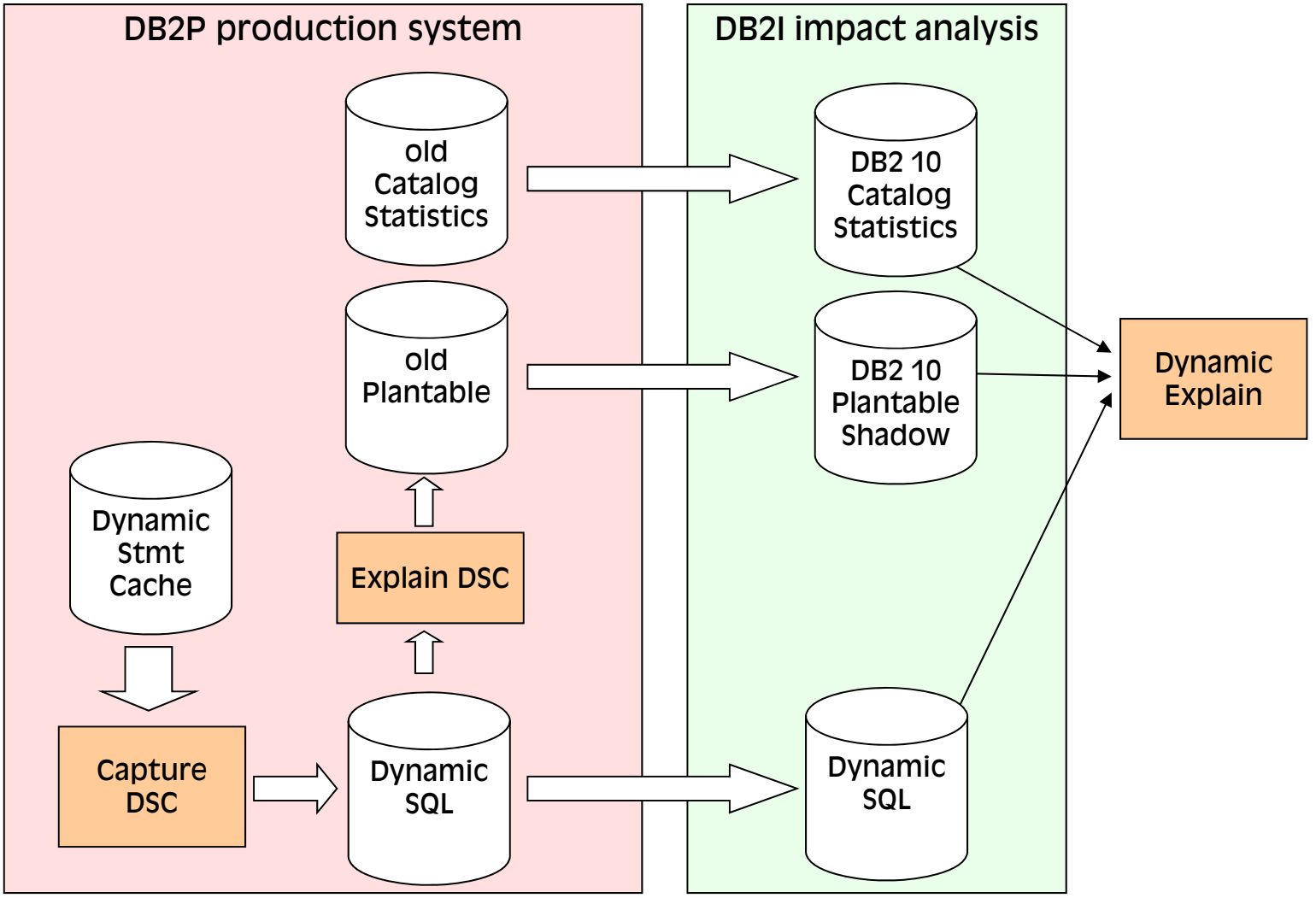
- DB2I: Impact analysis
  - Have a DB2 10 NFM sandbox
  - Use your cloned source
  - Apply naming changes if desired and
  - Create DDL DEFINE NO
  - Update the relevant catalog statistics
  - Apply environment modeling
  - Static SQL:
    - Import Packages and PLAN_TABLE
  - Dynamic SQL:
    - Import captured SQL and PLAN_TABLE

→ A true precheck environment, ready for analysis

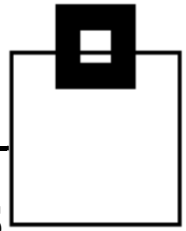
Precheck Scenario for Static SQL - setup



Precheck Scenario for Dynamic SQL



Setting up a Precheck System - statistics



SYSCOLDIST / SYSKEYTGTDIST
CARDF
COLGROUPCOLNO / KEYGROUPKEYNO
COLVALUE / KEYVALUE
FREQUENCYF
HIGHVALUE
LOWVALUE
NUMCOLUMNS / NUMKEYS
QUANTILENO
STATSTIME

SYSCOLUMNS / SYSKEYTARGETS
COLCARDF / CARDF
HIGH2KEY
LOW2KEY
n/a / STATS_FORMAT

SYSCOLSTATS
COLCARD
HIGHKEY
LOWKEY

SYSINDEXES
CLUSTERING*
CLUSTERRATIO
CLUSTERRATIOF
DATAREPEATFACTORF
FIRSTKEYCARDF
FULLKEYCARDF
NLEAF
NLEVELS

SYSINDEXPART
LIMITKEY*

* Columns are not updated by RUNSTATS
_ Columns are not updatable

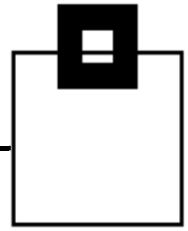
SYSROUTINES
CARDINALITY*
INITIAL_INSTS*
INITIAL_IOS*
INSTS_PER_INVOC*
IOS_PER_INVOC*

SYSTABLES
CARDF
EDPROC*
NPAGES
NPAGESF
PCTROWCOMP

SYSTABLESPACE
NACTIVE
NACTIVEF

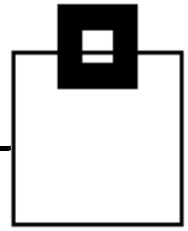
SYSTABSTATS
CARD
CARDF
NPAGES

Setting up a Precheck System - environment



- CPU simulation
  - Copy production to test
  - Check a faster newer machine (Upsize)
  - Check a slower older machine (Downsize)
- ZPARM simulation
  - Change size of SRTPOOL
  - Change size of RID Pool
  - Change size of data cache or Star Join Pool
- BUFFERPOOL
  - Change size of any BUFFERPOOL

Setting up a Precheck System - environment

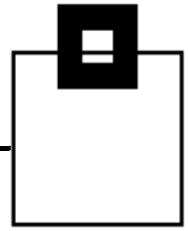


Production Modelling

- Supports optimizer overrides for optimizer relevant system settings
  - New zparms
    - SIMULATED_CPU_SPEED
    - SIMULATED_COUNT
  - New SYSIBM.DSN_PROFILE_ATTRIBUTES*
    - SORT_POOL_SIZE
    - MAX_RIDBLOCKS
    - For bufferpools

*Find DDL in member DSNTIJOS of your SDSNSAMP

Setting up a Precheck System - environment



Production Modelling

→ ZPARM SIMULATED_CPU_SPEED

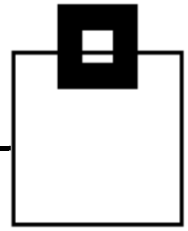
- µs of task or SRB execution time per SU
- OFF or an integer from 1 – 2147483647
- Gather from production by

```
EXPLAIN ALL SET QUERYNO=12345°
FOR SELECT * FROM SYSIBM.SYSDUMMY1;

SELECT
HEX(SUBSTR(IBM_SERVICE_DATA,69,4))
          AS CPU_SPEED,
  FROM PLAN_TABLE
  WHERE QUERYNO=12345°;
```

- Apply the value to DSN6SPRM on test

° must be unique

Setting up a Precheck System - environment



Production Modelling

→ ZPARM SIMULATED_CPU_COUNT*

- OFF or an integer from 1 – 127
- Gather from production by

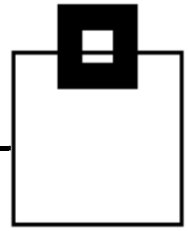
```
SET CURRENT DEGREE='ANY';
EXPLAIN ALL SET QUERYNO=12345°
FOR SELECT * FROM SYSIBM.SYSDUMMY1;

SELECT
HEX(SUBSTR(IBM_SERVICE_DATA,25,2))
          AS CPU_COUNT,
  FROM PLAN_TABLE
  WHERE QUERYNO=12345°;
```

- Apply the value to DSN6SPRM on test

*only if DEGREE='ANY'

° must be unique

Setting up a Precheck System - environment



Production Modelling

- → DSN_PROFILE_TABLE
- → DSN_PROFILE_ATTRIBUTES SORT_POOL_SIZE
  - Gather from production by

```
EXPLAIN ALL SET QUERYNO=12345°
FOR SELECT * FROM SYSIBM.SYSDUMMY1;


SELECT
HEX(SUBSTR(IBM_SERVICE_DATA,9,4))
          AS SORT_POOL_SIZE,
  FROM PLAN_TABLE
  WHERE QUERYNO=12345°;
```

° must be unique

Setting up a Precheck System - environment

Production Modelling

- → DSN_PROFILE_TABLE
- → DSN_PROFILE_ATTRIBUTES SORT_POOL_SIZE
  - Apply on test by

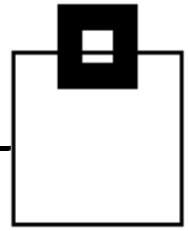
```
INSERT INTO SYSIBM.DSN_PROFILE_TABLE
  VALUES (1234);

INSERT INTO
SYSIBM.DSN_PROFILE_ATTRIBUTES
  VALUES (1234, 'SORT_POOL_SIZE',NULL,
  307200);
```

profile ID°

Desired size to simulate

° must be unique

Setting up a Precheck System - environment



Production Modelling

- → DSN_PROFILE_TABLE
- → DSN_PROFILE_ATTRIBUTES MAX_RID_BLOCKS
  - Gather from production by

```
EXPLAIN ALL SET QUERYNO=12345°
FOR SELECT * FROM SYSIBM.SYSDUMMY1;

SELECT
HEX(SUBSTR(IBM_SERVICE_DATA,13,4))
          AS MAX_RID_BLOCKS,
  FROM PLAN_TABLE
  WHERE QUERYNO=12345°;
```


° must be unique

Setting up a Precheck System - environment

Production Modelling

- → DSN_PROFILE_TABLE
- → DSN_PROFILE_ATTRIBUTES MAX_RID_BLOCKS
  - Apply on test by

```
INSERT INTO SYSIBM.DSN_PROFILE_TABLE
  VALUES (1234);

INSERT INTO
SYSIBM.DSN_PROFILE_ATTRIBUTES
  VALUES (1234, 'MAX_RIDBLOCKS',NULL,
  307200);
```

profile ID°

Desired size to simulate

° must be unique

Setting up a Precheck System - environment

Production Modelling

- → DSN_PROFILE_TABLE
- → DSN_PROFILE_ATTRIBUTES bufferpools
  - Gather from production from DSNTIP1 panel
  - Apply on test by

```
INSERT INTO SYSIBM.DSN_PROFILE_TABLE
  VALUES (1234);

INSERT INTO
SYSIBM.DSN_PROFILE_ATTRIBUTES
VALUES(1234, 'BP0',NULL, 25000);
```

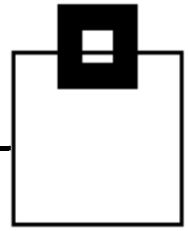
profile ID°

Desired BP to simulate

Desired size to simulate

must be unique

Setting up a Precheck System - environment

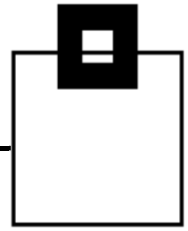


All changes can be done „on the fly“ with no restart of DB2

Before rushing off and changing everything think!

- Will my change affect anyone apart from me?
- Am I sure about that?
- Am I really sure?
- Ask just in case!

Setting up a Precheck System - environment



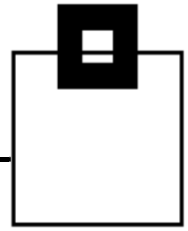
Production Modelling

- All simulation values are assigned to a **global** profile for a single DB2 subsystem
- Activation of a **global** profile by
  `-START PROFILE`
- Execute a dynamic EXPLAIN for the SQL queries you'd like to assess
- Refer to column REASON of the DSN_STATEMNT_TABLE
  → 'PROFILEID 1234,

Your assigned profile ID

- Refer to SYSIBM.DSN_PROFILE_TABLE
  → PROFILE_ENABLED = 'Y'

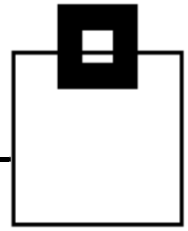
New Access Path Pattern V8



We encountered access path pattern of the following categories:

- UNCHANGED – Statements without AP changes
- IMPROVED – Statements with improved AP
- DB2 V8 specific patterns (examples)
  - V8 pattern 1 – SORT first QBLOCK
  - V8 pattern 2 – Tablespace scan instead of non matching index scan
  - V8 pattern 3 – usage of smaller index
  - …
- CHANGED – Statements with changed AP (not classified)
- WORSENED – Statements with degraded AP

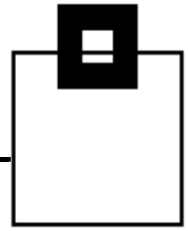
New Access Path Pattern 9



We encountered access path pattern of the following categories:

- UNCHANGED – Statements without AP changes
- IMPROVED – Statements with improved AP
- DB2 9 specific patterns (examples)
  - 9 pattern 1 – Usage of smaller index
  - 9 pattern 2 – Usage of bigger index, sort avoided
  - 9 pattern 3 – Usage of smaller index, index only get lost
  - …
- CHANGED – Statements with changed AP (not classified)
- WORSENED – Statements with worsened AP

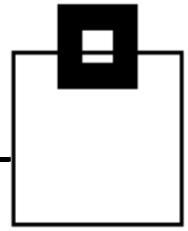
New Access Path Pattern 10



We encountered access path pattern of the following categories:

- UNCHANGED – Statements without AP change
- IMPROVED – Statements with improved AP
- DB2 10 specific patterns (examples)
  - 10 pattern 1 – Range list index scan
  - 10 pattern 2 – In memory in list
  - V10 pattern 3 – SQL pagination
  - …
- CHANGED – Statements with changed AP (not classified)
- WORSENED – Statements with worsened AP

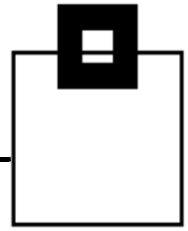
DB2 Optimizer Characteristics



Using a different index

- Comparing DB2 V7 and V8, the latter showed a tendency to use smaller indexes (no. of pages)
- In general DB2 9 shows the same tendency, but ..
- In a lot of cases DB2 9 changed access paths in the way to prefer a bigger index to avoid a physical sort (similar to V7)

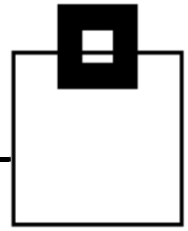
DB2 Optimizer Characteristics



Small tables

- With DB2 V8 a lot of matching or non matching index scans changed to a tablespace scans for small tables
- In a lot of cases this behavior changed back to an index usage with DB2 9 even for tables with only 1 page

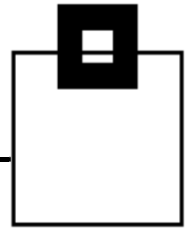
DB2 Optimizer Characteristics



Bad stats:

- With DB2 V8 we saw many bad access paths caused by bad statistics. For example, missing statistics for a new index lead to a tablespace scan, even if another index was used before
- With DB2 9 Indexes with 0 pages seem to be always used

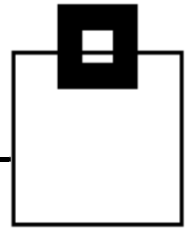
DB2 Optimizer Characteristics



Index size (part 2)

- DB2 9 tends to use the smallest index
- even if index only access gets lost
- in most cases the smaller ix was the partitioning index

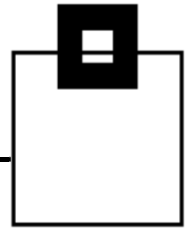
DB2 Optimizer Characteristics



Other changes

- Change Prefetch=S to D
- Changed join sequence (random feature?)
  - Bigger table outside
  - Smaller table outside
- MIX -> Hybrid Join

DB2 Optimizer Characteristics



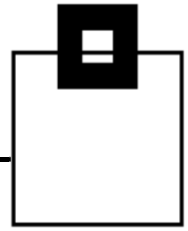
Other changes

- DB2 10 Range-List IX scan

  When SQL has multiple OR, IN or other predicates that reference the same index

Improves SQL processing against an index when multiple WHERE clauses can all reference the same index.

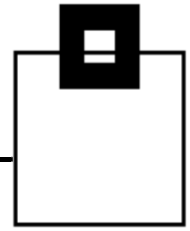
DB2 Optimizer Characteristics



Other changes

- DB2 10 Optimizer uses more parallelism
  - index reverse scan for a table
  - SQL subquery is transformed into join
  - multiple column hybrid join with sort composite
  - leading table is sort output and the join between the leading table and the second table is a multiple column hybrid join

DB2 Optimizer Characteristics

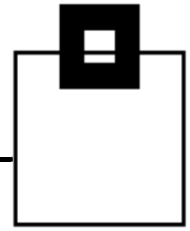


The DB2 10 Optimizer avoids further sorts

Multi-index access, list prefetch and sort

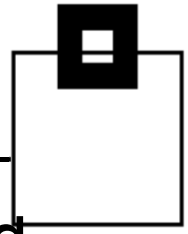
→ Single matching index access with sort avoided

Statistics



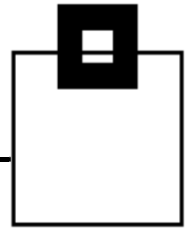
- REBINDs without distribution stats
  - Improved: 4 %
  - Degraded: 6,1 %
  - Unchanged: 79,9 %
  - Changed: 10 %
- REBINDs with distribution stats on everything
  - Improved: 13,2 %
  - Degraded: 0,9 %
  - Unchanged: 79,9 %
  - Changed: 6 %

Statistics



- REBINDs in a critical online application, well tuned
  - Improved: 4,9 %
  - Degraded: 1,3 %
  - Unchanged: 70,6 %
  - Changed: 23,2 % (not categorized)
- REBINDs in an uncritical application, batch
  - Improved: 4,2 %
  - Degraded: 5,8 %
  - Unchanged: 82 %
  - Changed: 8 %

Conclusion

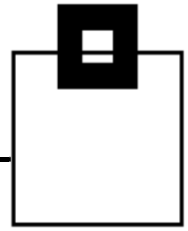


It's all about statistics...
– Garbage in, garbage out!

From DB2 V8 on we have seen significant impact on the quality and scope of statistics on the optimizers decision

If you provide proper statistics, the DB2 Optimizer provides a good access path

Conclusion



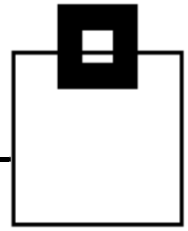
It's all about statistics… – Garbage in, garbage out!

V8 Recommendation:
Keep your Catalog Statistics current AND
gather DISTRIBUTION STATS

9 Recommendation:
Keep your Catalog Statistics current AND
gather DISTRIBUTION STATS AND
gather HISTOGRAM STATS AND
capture enhanced clusterratio/DRF

10 Recommendation:
Keep your Catalog Statistics current AND
gather DISTRIBUTION STATS AND
gather HISTOGRAM STATS

Summary



- IBM significantly improved the DB2 Optimizer with general as well as specific enhancements
- „Usual“ RUNSTATS doesn‘t provide detailed statistics to enable the Optimizer to find the best access path
- Access path decisions of the Optimizer in DB2 z/OS 10 (CM, NFM) are different from prior DB2 z/OS – we found certain patterns
- Requirements regarding quality and efficiency to maintenance procedures (RUNSTATS and REBIND) are higher for DB2 10 and upcoming versions
  - Growth of tables
  - Column Distribution Statistics
  - Histogram Statistics