Screening in a Virtualized Production Environment

–

True DB2 performance simulation covering

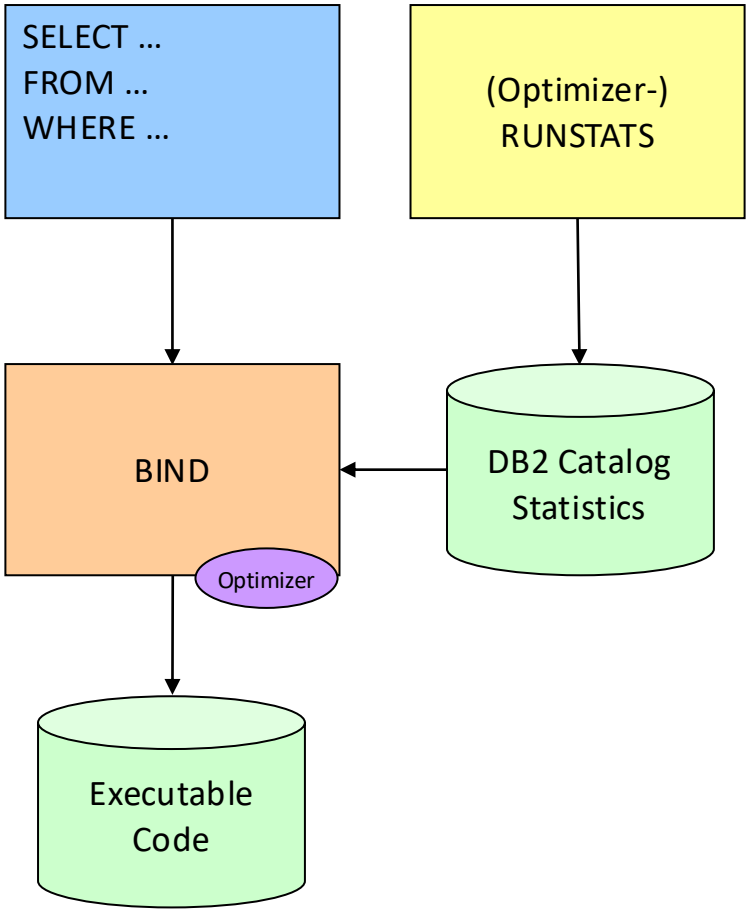
- Environment changes
- Schema changes
- Application changes

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

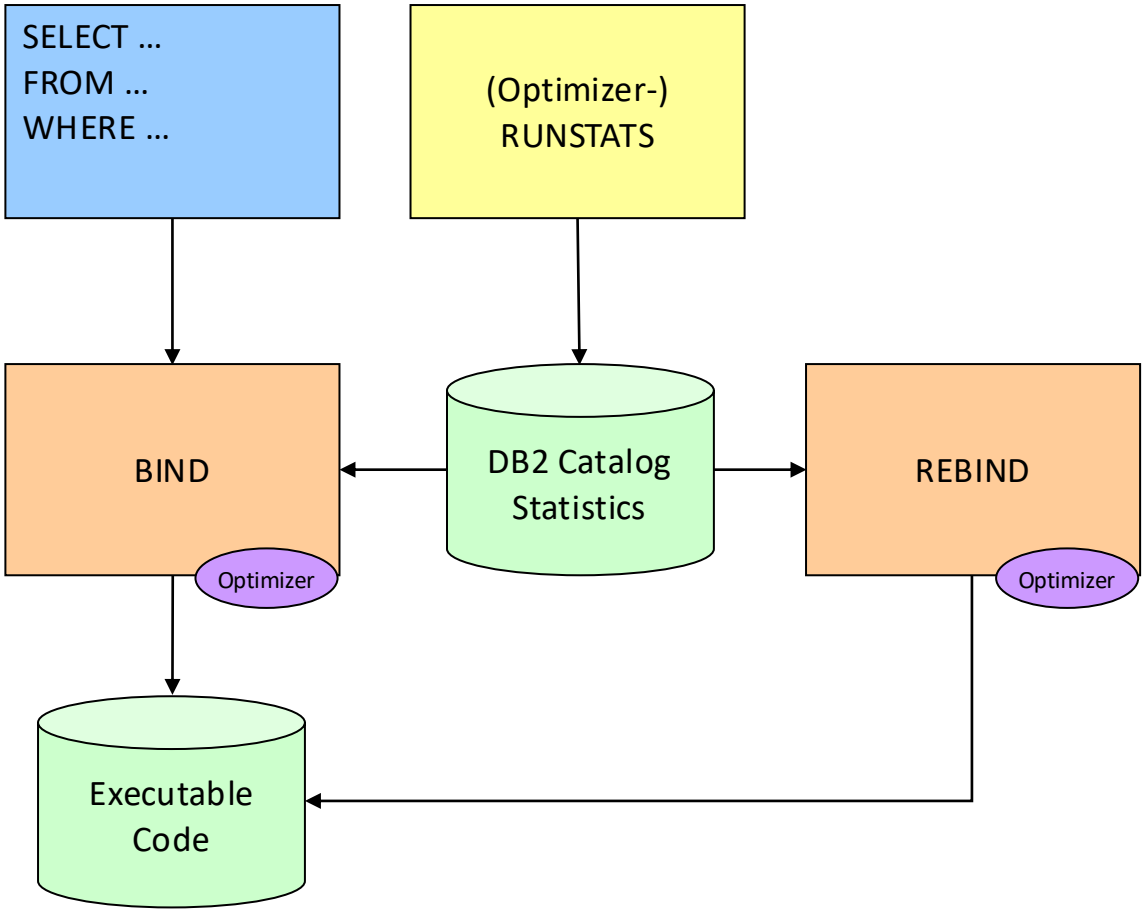
Agenda

- What the DB2 Optimizer needs to make a choice
- How to model a production environment
- Compare access paths resulting from
  - Buffer pool changes
  - RID expansion
  - CPU changes
- Test “what if” you drop an index, or change the sort order of an index

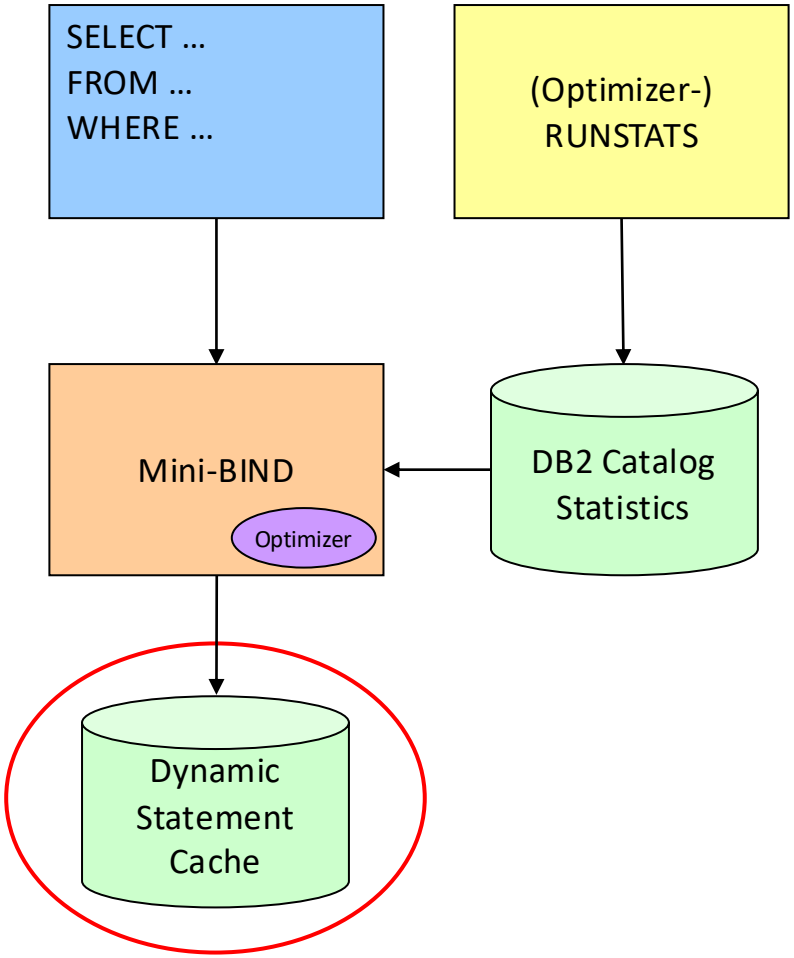
What the optimizer needs to make a choice



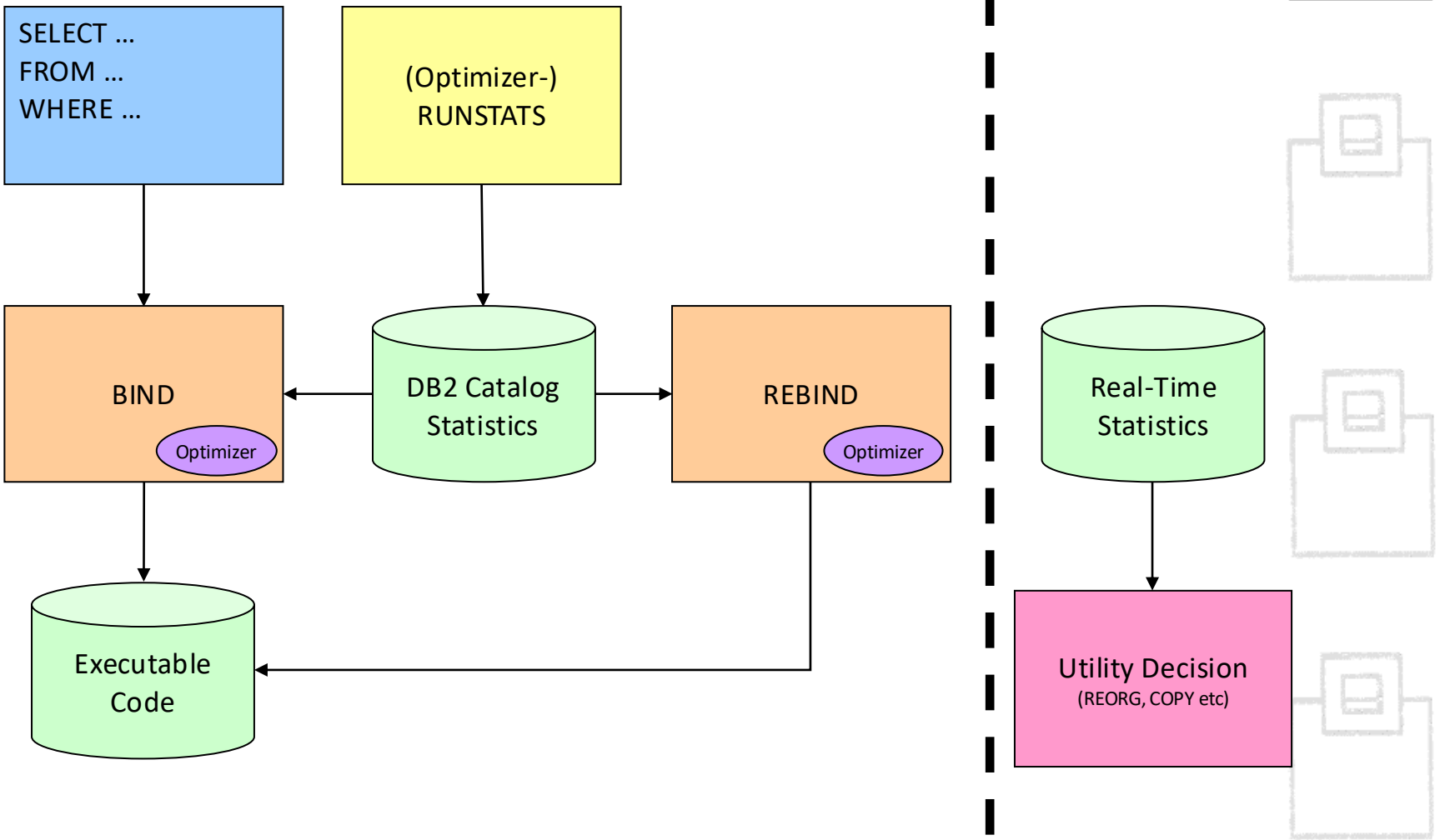
What the optimizer needs to make a choice



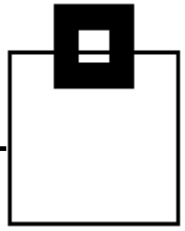
What the optimizer needs to make a choice



What the optimizer needs to make a choice



What the optimizer needs to make a choice

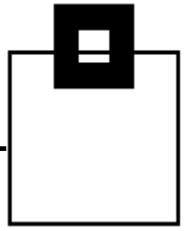


1st DDL:

```
SELECT …
FROM …
WHERE …
```

- Which objects are referenced in the SQL
  - SELECT <columns> FROM <table> or <view> or …
  - WHERE <local predicates>
  - ORDER BY or GROUP BY or UNION or … <columns>
- Which objects are defined and how
  - INDEX
  - PARTITIONING

What the optimizer needs to make a choice



2nd STATISTICS:

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

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

SYSCOLSTATS
COLCARD
HIGHKEY/HIGH2KEY
LOWKEY/LOW2KEY

SYSINDEXES
CLUSTERING*
CLUSTERRATIOF
DATAREPEATFACTORF
FIRSTKEYCARDF
FULLKEYCARDF
NLEAF
NLEVELS

SYSINDEXPART
LIMITKEY*

SYSINDEXSPACESTATS
NLEAF (SHRLEVEL REFERENCE)
NLEVELS (SHRLEVEL REFERENCE)

SYSTABSTATS
CARDF
NPAGES

SYSROUTINES
CARDINALITY*
INITIAL_INSTS*
INITIAL_IOS*
INSTS_PER_INVOC*
IOS_PER_INVOC*

SYSTABLES
CARDF
EDPROC*
NPAGES
NPAGESF
PCTROWCOMP

SYSTABLESPACE
NACTIVEF

SYSTABLESPACESTATS[2]
NPAGES (SHRLEVEL REFERENCE)
TOTALROWS (SHRLEVEL REFERENCE)

* Columns are not updated by RUNSTATS
_ Columns are not updatable
2 RTS not updated if TABLESAMPLE SYSTEM AUTO/YES

What the optimizer needs to make a choice

3rd ENVIRONMENT:

- CP speed
- # of CPs
- BPs
- RID pool
- Sort pool

How to model a production environment

Create a Baseline…

- DDL
- STATISTICS
- ENVIRONMENTAL data

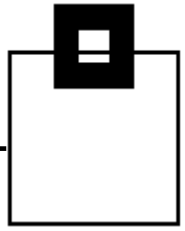
… allow to select any existing SQL along with the current access path details for comparison

- Production PLAN_TABLE
- Dynamic Statement Cache (DSC)

… and consider workload data

- DSC
- SSC / Monitor

How to model a production environment

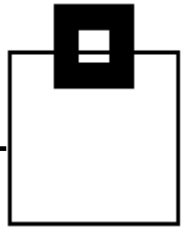


- A Baseline represents a given system, or subset

- Do multiple Baselines to allow covering
  - Different systems
  - Different amounts of data
  - Different distribution
  - Different workload, or time frames

- Apply naming conversions if you need to allow
  - Multiplications of Baselines (e.g. for different users)
  - Merging multiple production environments

How to model a production environment



→ A test/simulation environment that behaves exactly like the production environment

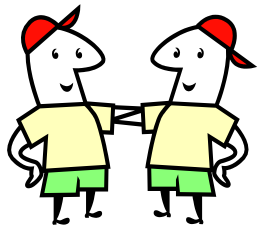
→ This allows to pre-check the results from a
- New application version
- New DB2 version / Function Level (or APARs affecting performance)
- New statistics
  - RUNSTATS (especially with dynamic SQL)

→ Keep control of your environment!
- Quickly identify and understand performance changes (degradation)
- Protect your production environment

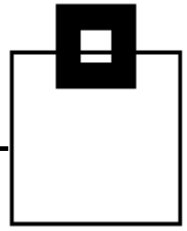
How to model a production environment

Gathering data for the simulation environment

- DB2P: Production
  - 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



How to model a production environment



Applying data to the simulation environment

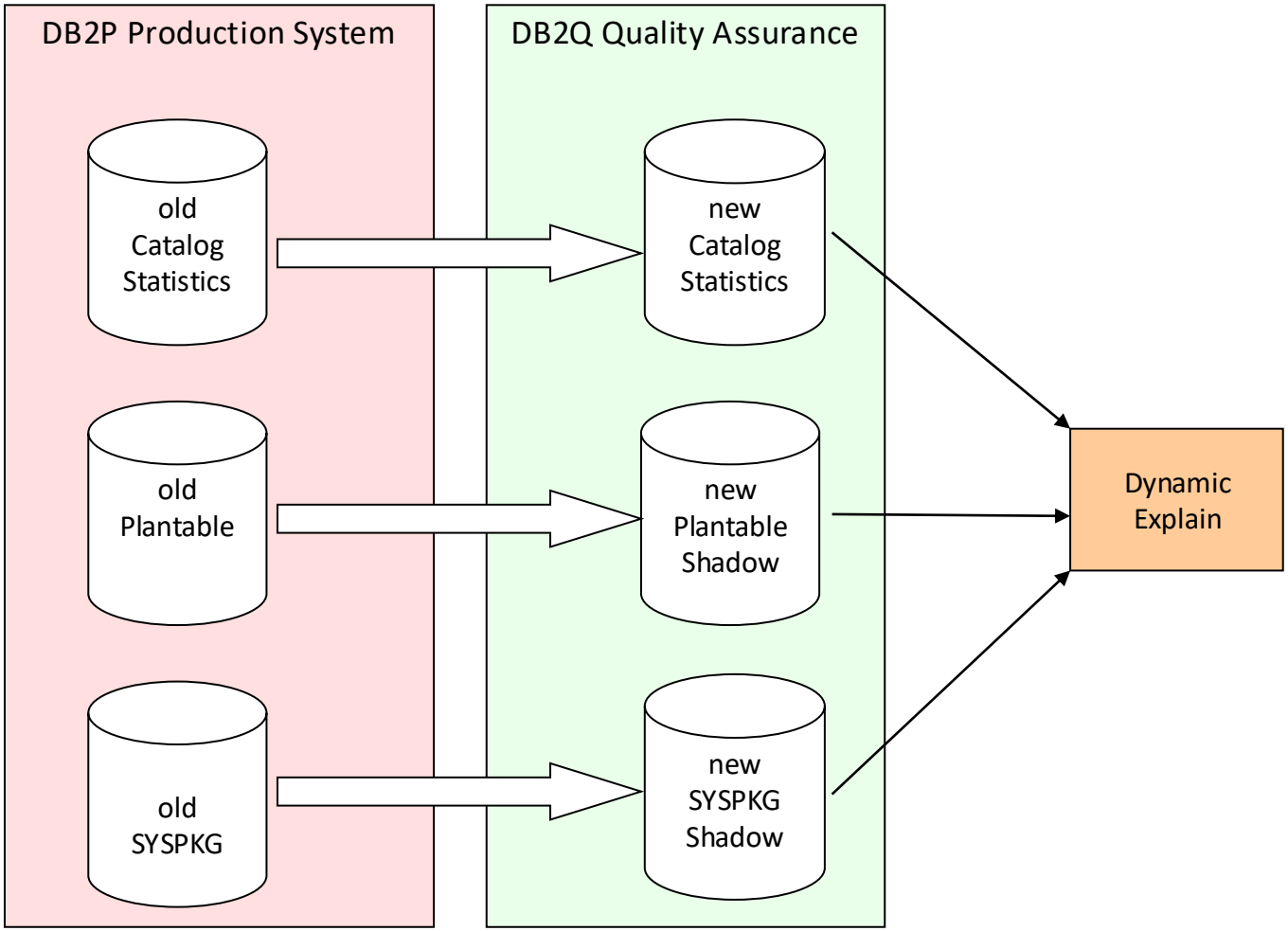
… Import the flat files:

- Apply naming changes if desired

- DB2Q: Quality Assurance
  - 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

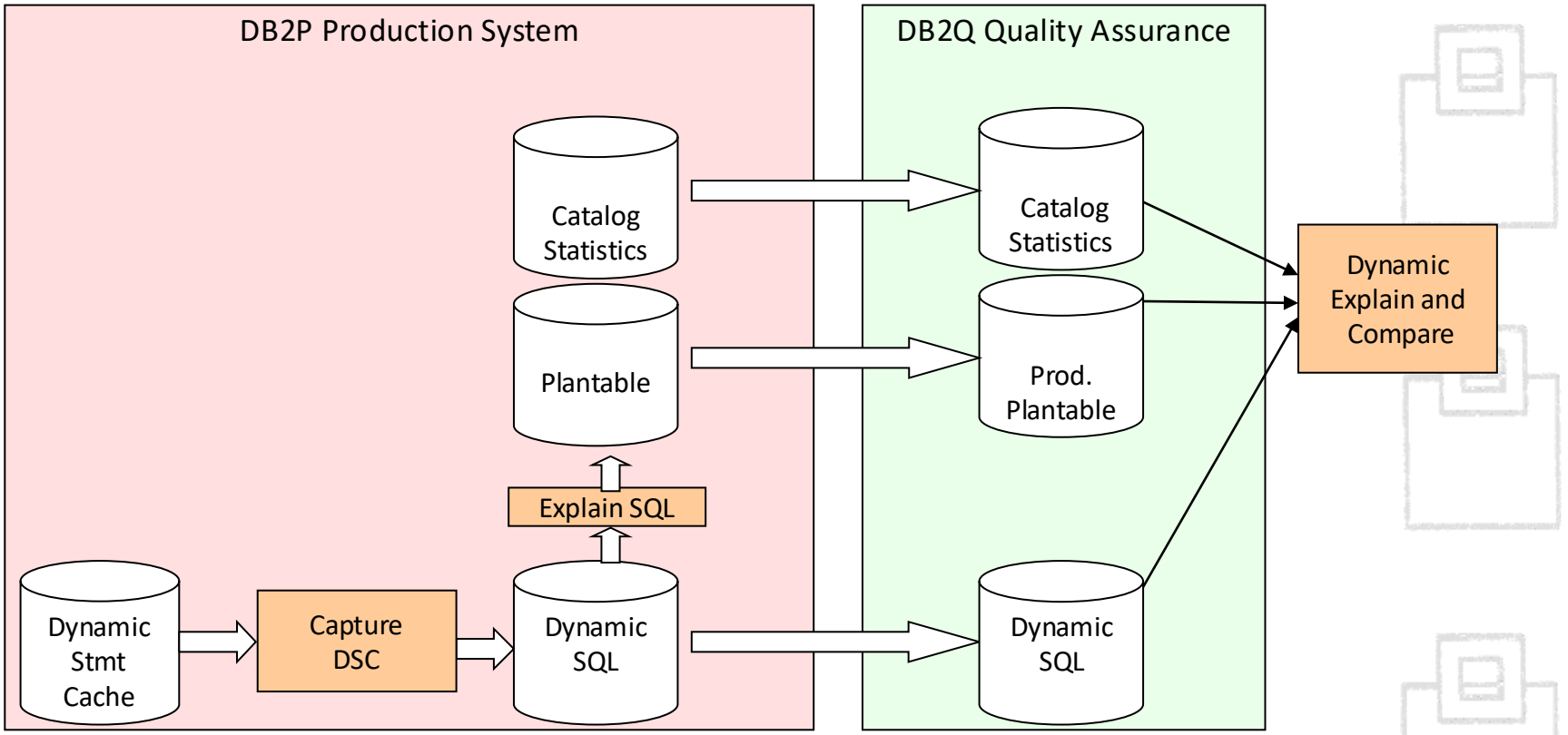
→ A true simulated environment, ready for any desired analysis/precheck

How to model a production environment

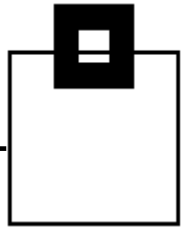


How to model a production environment

Dynamic SQL management and protection:

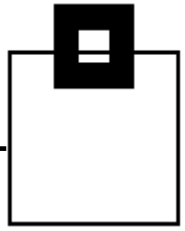


Compare Access Paths – Environment changes



- CPU simulation
  - Copy production metrics 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
- Accelerator

Compare Access Paths – Environment changes

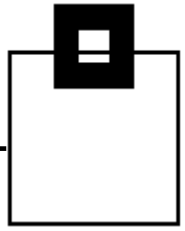


Production Modelling

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

*Find DDL in member DSNTIJSG of your SDSNSAMP

Compare Access Paths – Environment changes



Production Modelling

→ ZPARM SIMULATED_CPU_SPEED

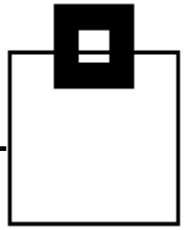
- µs of task or SRB execution time per SU
- OFF or an integer from 1 – 2147483647
- Gather from production by

```
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

Compare Access Paths – Environment changes



Production Modelling

→ ZPARM SIMULATED_CPU_COUNT*

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

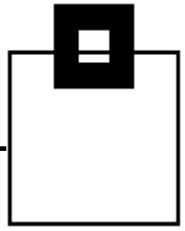
```
SET CURRENT DEGREE='ANY';
EXPLAIN ALL SET QUERYNO=12345°
FOR SELECT * FROM SYSIBM.SYSDUMMY1;


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

Compare Access Paths – Environment changes



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

Compare Access Paths – Environment changes

Production Modelling

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

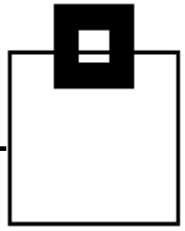
```
INSERT INTO SYSIBM.DSN_PROFILE_TABLE
  VALUES (1234);

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

profile ID°

Desired size to simulate

° must be unique

Compare Access Paths – Environment changes



Production Modelling

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

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

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


° must be unique

Compare Access Paths – Environment changes

Production Modelling

- → DSN_PROFILE_TABLE
- → DSN_PROFILE_ATTRIBUTES MAX_RIDBLOCKS
  - 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

Compare Access Paths – Environment changes

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);
```

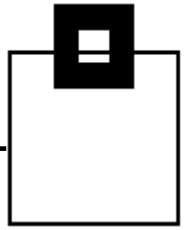
profile ID°

Desired BP to simulate

Desired size to simulate

° must be unique

Compare Access Paths – Environment changes

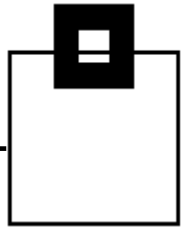


Production Modelling

… and there‘s more simulation supported via SYSIBM.DSN_PROFILE_ATTRIBUTES

- Parallelism
  - CURRENT_DEGREE
- Accelerator
  - ACCEL_NAME_EXPLAIN
  - CURRENT_QUERY_ACCELERATION
  - CURRENT_GET_ACCEL_ARCHIVE

Compare Access Paths – Environment changes



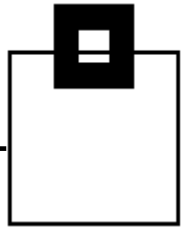
All changes can be done "on the fly" with no restart of DB2

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!

(e.g. We don‘t want to set BP0 to 350.000 in test etc.)

Compare Access Paths – Environment changes

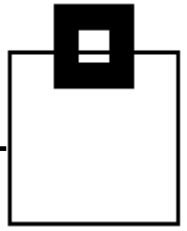


Production Modelling

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

Your assigned profile ID

Compare Access Paths – Schema changes



How to reliably simulate index changes?

E.g.

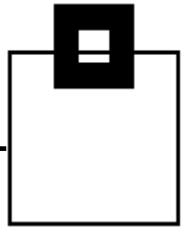
- Drop Index
- Create index
- Alter Index

→ Virtual Indexes are your friend

"Recommendation*: Do not manually insert data into or delete data from this table, it is intended to be used only by optimization tools.“

* DB2 for z/OS Managing Performance

Compare Access Paths – Schema changes



How to build an entry yourself:

DSN_VIRTUAL_INDEXES*	SYSINDEXES	Description
TBCREATOR	TBCREATOR	Auth. ID of owner/schema of table on which the entry is simul.
TBNAME	TBNAME	Name of the table on which the entry is being simulated
IXCREATOR	IXCREATOR	Auth. ID/schema of the owner of the index
IXNAME	IXNAME	Name of the index to simulate
ENABLE		Whether entry will be considered ('Y') or not ('N')
MODE		Whether the index is being created ('C') or dropped ('D')
UNIQUERULE	UNIQUERULE	Index is uniqueness: D for No (duplicates are allowed); U for Yes
COLCOUNT	COLCOUNT	The number of columns in the key
CLUSTERING	CLUSTERING	Whether the index is clustered ('Y' or 'N')
NLEAF	NLEAF	# of active leaf pages in the index, or -1 if unknown
NLEVELS	NLEVELS	# of levels in the index tree, or -1 if unknown
INDEXTYPE	INDEXTYPE	The index type: '2' - NPSI; 'D' – DPSI
PGSIZE	PGSIZE	Size, in bytes, of the leaf pages in the index: 4K, 8K, 16K, 32K
FIRSTKEYCARDF	FIRSTKEYCARDF	# of distinct values of the first key column, or -1 if unknown
FULLKEYCARDF	FULLKEYCARDF	# of distinct values of the key, or -1 if unknown
CLUSTERRATIOF	CLUSTERRATIOF	Clustering ratio, or -1 if unknown
PADDED	PADDED	Index keys padded for varying-length column data ('Y' or 'N')
COLNO1		Column # of the first column in the index key
ORDERING1		Ordering ('A' or 'D') of the first column in the index key
COLNOn		Column # repeated up to 64
ORDERINGn		Ordering (‘A’ or ‘D’) repeated up to 64

*Find DDL in member DSNTIJSG of your SDSNSAMP
needs to have the same schema name (authid) as the PLAN_TABLE

Compare Access Paths – Schema changes

How to build an entry yourself:

The easiest way for drop simulation (mode D) is to insert from sysindexes:

```
INSERT INTO <qualifier>.DSN_VIRTUAL_INDEXES
   (TBCREATOR, TBNAME, IXCREATOR, IXNAME, ENABLE, MODE,
   UNIQUERULE, COLCOUNT, CLUSTERING, NLEAF, NLEVELS, INDEXTYPE,
   PGSIZE, FIRSTKEYCARDF, FULLKEYCARDF, CLUSTERRATIOF, PADDED
   , COLNO1, ORDERING1
   , COLNO2, ORDERING2)
   SELECT TBCREATOR, TBNAME, CREATOR, NAME, 'Y', 'D',
   UNIQUERULE, COLCOUNT, CLUSTERING, NLEAF, NLEVELS, '2', 4,
   FIRSTKEYCARDF, FULLKEYCARDF, CLUSTERRATIOF, PADDED
   , 30, 'A'
   , 32, 'A'
   FROM SYSIBM.SYSINDEXES
   WHERE CREATOR = '<ixcreator>'
   AND NAME = '<ixname>'
   AND TBCREATOR = '<tbcreator>'
   AND TBNAME = '<tbname>'
```

active

drop

pgsize

type 2 IX

colno [1-64], ordering

Your desired index for drop simulation

Compare Access Paths – Schema changes

How to build an entry yourself:

Simulating a new index (MODE = 'C'):

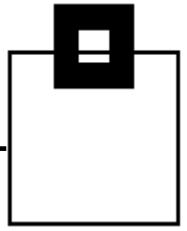
```
INSERT INTO <qualifier>.DSN_VIRTUAL_INDEXES
   (TBCREATOR, TBNAME, IXCREATOR, IXNAME, ENABLE, MODE,
    UNIQUERULE, COLCOUNT, CLUSTERING, NLEAF, NLEVELS, INDEXTYPE,
    PGSIZE, FIRSTKEYCARDF, FULLKEYCARDF, CLUSTERRATIOF, PADDED
   , COLNO1, ORDERING1
   , COLNO2, ORDERING2)
VALUES (
   '<tbcreator>', '<tbname>',
   '<ixcreator>', '<new-ixname>', 'Y', 'C', 'D', 2, 'N', -1, 2,
   '2', 4, -1, -1, -1, 'Y', <colno1>, 'A', <colno2>, 'D');
```

active

create

colno [1–64], ordering

Compare Access Paths – Schema changes



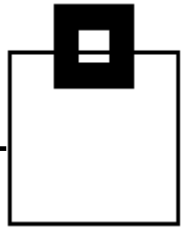
How to simulate an index change (ALTER):

There is no direct ALTER mode. The pattern is: virtually drop the existing index + virtually create the new one.

1. Virtually drop the existing index
   Insert a row with MODE = 'D' pointing to the real index (as shown on the DROP slide)
2. Virtually create the modified index
   Insert a second row with MODE = 'C' and the changed definition e.g. different column order, sort direction or clustering flag
3. Run EXPLAIN
   DB2 ignores the dropped index and considers the new virtual index
4. Compare access paths
   Compare PLAN_TABLE entries with the production baseline

→ Example: Test whether reversing the sort order of an index key column improves ORDER BY performance.

Compare Access Paths – Schema changes



How to activate simulation of an index change:

- Entries are active as long as ENABLE = 'Y'
- No restart required – changes take effect immediately
- Trigger EXPLAIN using SET CURRENT EXPLAIN MODE:

```
SET CURRENT EXPLAIN MODE = EXPLAIN;
SELECT ... FROM ... WHERE ...;
SET CURRENT EXPLAIN MODE = NO;
```

- Alternatively use: EXPLAIN ALL FOR <statement>
- Results are written to PLAN_TABLE / DSN_STATEMNT_TABLE
- Deactivate an entry: set ENABLE = 'N' – delete the row to remove the simulation permanently
- Works for both static (BIND/REBIND) and dynamic SQL
- Virtual Indexes and DSN_PROFILE_TABLE can be active at the same time*

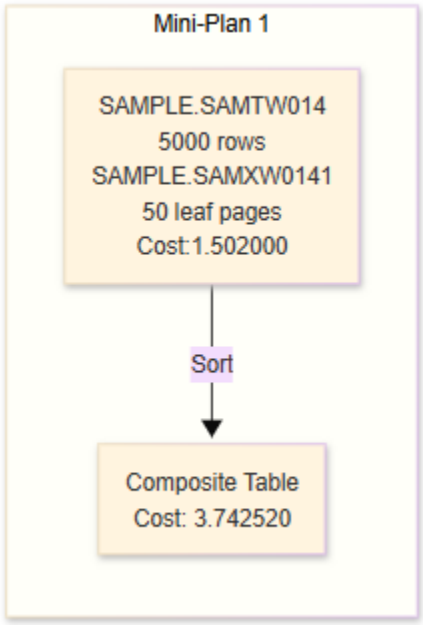
* DSN_VIRTUAL_INDEXES must have the same schema name as the PLAN_TABLE

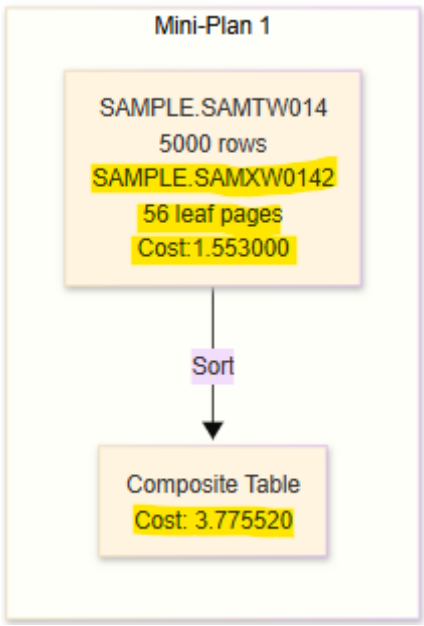
Conclusion

- Precheck any changes
  - Environment
  - Schema
  - Application
- Allow flexibility in developing and running your applications
- Insight out of the box into the effect of your desired changes before you apply them
- Cost efficiency and cost avoidance in development and operations

Last, but not least and worth to mention

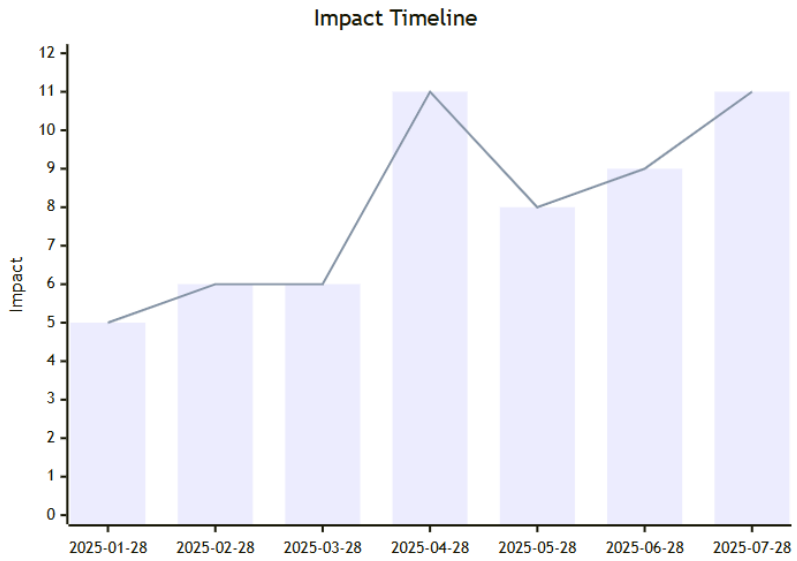
- Make the data easy to understand:
  - Comparing access plans is easier using visualization than reading PLAN tables

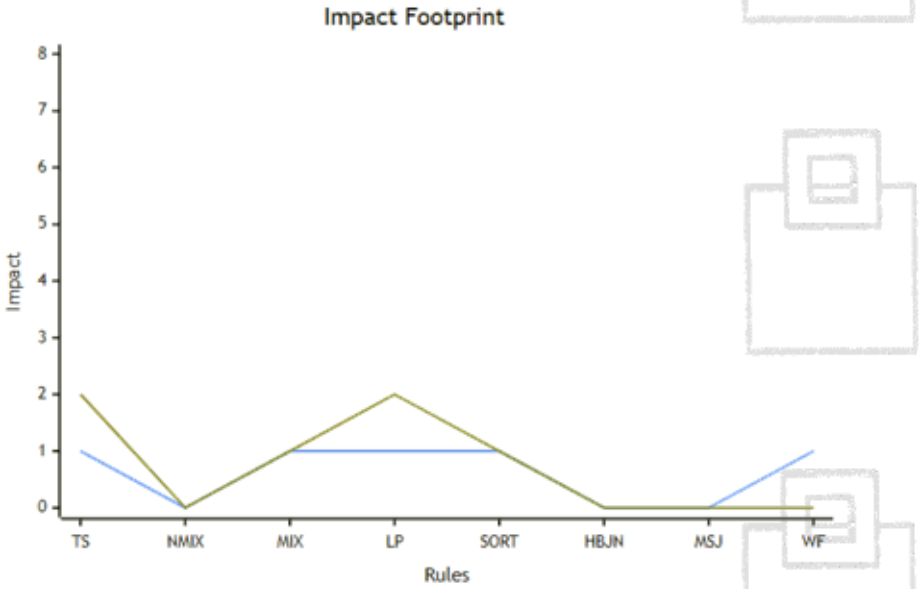




Last, but not least and worth to mention

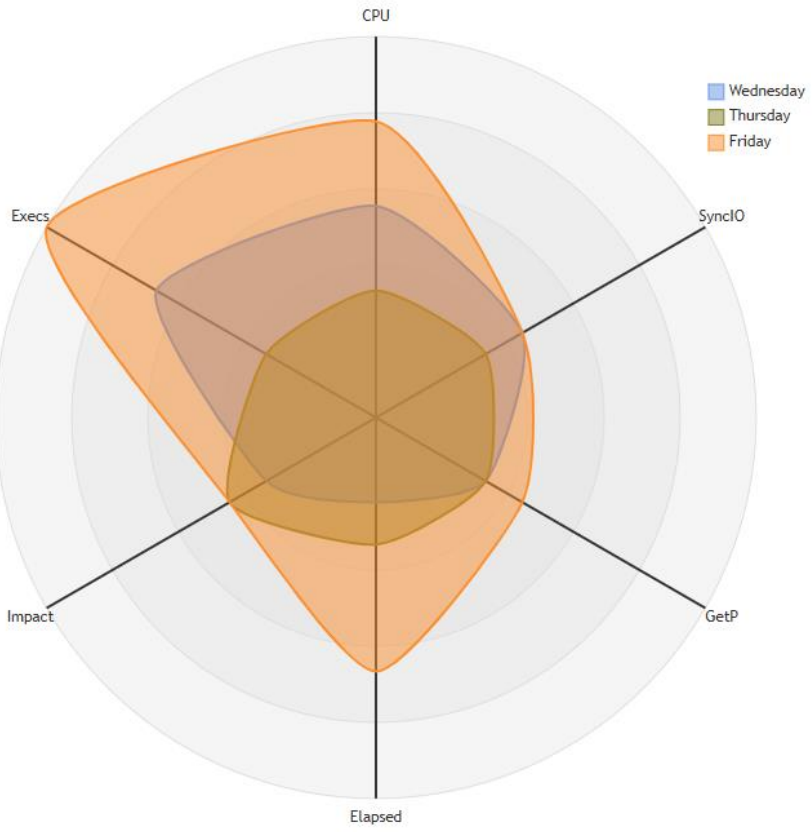
- Make the data easy to understand:
  - … your baselines allows highlighting performance trends from a quantity as well as a quality perspective



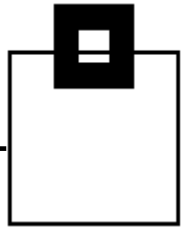


Last, but not least and worth to mention

- Make the data easy to understand:
  - Radar diagrams are great for pinpointing multiple values at different points in time

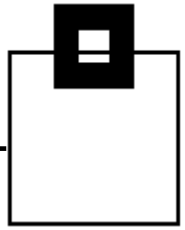


Assure Best Practices and precheck!



Before you apply

- Hardware modifications
- Environment setup changes
- New application release
- New DB2 code
  - Version
  - Function Level
  - APAR
- Index modifications
  - Create
  - Drop
  - Change



Ulf Heinrich

SEGUS, Inc

u.heinrich@segus.com