Total environment simulation

Workload Replay in an agile world

Ulf Heinrich
SEGUS Inc
Agenda

- Testing, virtualizing and simulating – the aspects of reliable quality assurance
- Db2 database/object cloning – what’s state of the art and what’s beyond
- XML commander – the comprehensive automation of flexibility, covering
  - FTP/routing
  - JCL
  - ISPF file tailoring, panels, messages
  - Db2 commands
- Different flavors of (pro-active) testing and how it can be automated:
  - Anomaly alerting based on Incompatibility Change Indicators (ICIs)
  - Dynamic/static access path change detection e.g. Plan Management
  - Cloning exploiting Backup System
  - Workload-KPI verification using SQL replay and KPI comparison
- Real world experience highlighting the benefits of automated testing
The aspects of reliable quality assurance

Agile development requires near-time delivery
  - Continuous Delivery (CD) is an approach to produce software in short cycles
  - CD ensures that changes can be released at any time, considering building, testing and releasing faster and more frequently
  - Key is a focus on more incremental updates
  - CD requires a straightforward and repeatable deployment
The aspects of reliable quality assurance

- If your shop can’t accepted outages for hours/days...
  - Make sure you consider
    - Time to detect anomalies
    - Time to analyze effect and origin
    - Time to evaluate a forward and a backward strategy
    - Time to fix/recover

... you gotta TEST, TEST, TEST

- Carefully
- Thoroughly
- Rigorously
The aspects of reliable quality assurance

Db2 comes with the right capabilities to protect your applications

- Always have IFCID 376 look for potential incompatibilities
  - Use APPLCOMPAT to minimize affected SQL from Db2 updates
- Use EXPLAIN to precheck access path changes
- Use BACKUP SYSTEM, or CONSISTENT COPY to have a consistent base for cloning
- Use efficient monitoring traces and be aware of applications being affected by changed behavior
What’s required for a virtual environment

1st DDL:

- 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’s required for a virtual environment

**2nd STATISTICS:**

- `SYSIBM.SYSCOLDIST`
- `SYSIBM.SYSCOLSTATS`
- `SYSIBM.SYSCOLUMNS`
- `SYSIBM.SYSINDEXES`
- `SYSIBM.SYSINDEXPART`
- `SYSIBM.SYSKEYTARGETS` (same as `SYSCOLUMNS`)
- `SYSIBM.SYSKEYTGTDIST` (same as `SYSCOLDIST`)
- `SYSIBM.SYSROUTINES`
- `SYSIBM.SYSTABLES`
- `SYSIBM.SYSTABLESPACE`
- `SYSIBM.SYSTABSTATS`
- `SYSIBM.TABLESPACESTATS` °
- `SYSIBM.INDEXSPACESTATS` °

- degree of parallelism only and, after APAR PK62804, also “sometimes” used to bound filter factor estimates
- only for dynamic SQL – so far
What’s required for a virtual environment

- **3rd ENVIRONMENT:**

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

Optimizer
How to virtualize an environment – DDL

1. Data Source Explorer

2. Generate DDL
Options
Select the model elements that you want to include in the DDL script.

Generate:
- Fully qualified name
- Quoted identifier
- DROP statements

Generate DDL statements:
- CREATE statements
- Comments

3. Generate DDL
Objects
Select the model objects that you want to include in the DDL script.

Generate:
- Check constraint
- Foreign key constraint
- Indexes
- Primary key constraint

- Jables
- Triggers
- Views

SELECT ...
FROM ...
WHERE ...
How to virtualize an environment – statistics

<table>
<thead>
<tr>
<th>Column/Target</th>
<th>SYSCOLDIST</th>
<th>SYSKEYTGTDIST</th>
<th>SYSCOLSTATS</th>
<th>SYSTABLESTATS</th>
<th>SYSINDEXEXPART</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CARD</td>
<td>COLGROUPCOLNO</td>
<td>COLCARD</td>
<td>CARD</td>
<td>LIMITKEY*</td>
</tr>
<tr>
<td></td>
<td>COLGROUPKEYNO</td>
<td>COLVALUE</td>
<td>HIGHKEY</td>
<td>CARDF</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>KEYVALUE</td>
<td>LOWKEY</td>
<td>NPAGES</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FREQUENCYF</td>
<td>NUMCOLUMNNS</td>
<td>SYSTABLESPACE</td>
<td>NACTIVE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>HIGHVALUE</td>
<td>NUMKEYS</td>
<td>NACTIVEF</td>
<td>NACTIVEF</td>
<td></td>
</tr>
<tr>
<td></td>
<td>LOWVALUE</td>
<td>QUANTILENO</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>STATSTIME</td>
<td>SYSTABLESPACESTATS</td>
<td>TOTALENTRIES</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>SYSINDEXES</td>
<td>SYSTABLESPACESTATS</td>
<td>TOTALROWS</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>COLGROUPCOLNO</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>KEYVALUE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>NUMCOLUMNNS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>NUMKEYS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>QUANTILENO</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>SYSTABLESPACESTATS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>SYSINDEXSPACESTATS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>SYSCOLUMNS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>SYSKEYTARGETS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>COLCARDDF / CARDF</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>HIGH2KEY</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>LOW2KEY</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>n/a / STATS_FORMAT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>COLCARD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>CARD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>COLGROUPCOLNO</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>COLGROUPKEYNO</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>COLVALUE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>KEYVALUE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>NUMCOLUMNNS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>NUMKEYS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>QUANTILENO</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>STATSTIME</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>SYSCOLDIST</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>SYSKEYTGTDIST</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>COLGROUPCOLNO</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>COLGROUPKEYNO</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>COLVALUE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>KEYVALUE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>NUMCOLUMNNS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>NUMKEYS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>QUANTILENO</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>SYSTABLESPACESTATS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>SYSINDEXSPACESTATS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>SYSCOLUMNS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>SYSKEYTARGETS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>COLCARDDF / CARDF</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>HIGH2KEY</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>LOW2KEY</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>n/a / STATS_FORMAT</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* Columns are not updated by RS – Columns are not updatable
How to virtualize an environment – hardware

Production Modelling

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

*Find DDL in member DSNTIJOS of your SDSNSAMP
# How to simulate changes – DDL

## How to reliably simulate index changes:

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

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

```sql
SELECT ... FROM ... WHERE ...```
How to simulate changes – DDL

Comparing KPIs before and after Index creation clearly shows whether an index helps or hinders Db2.
How to simulate changes – statistics

- Relationships exist among certain columns of certain tables:
  - Columns in SYSCOLUMNS, SYSCOLDIST, and SYSINDEXES
  - CARDF (SYSCOLDIST): CARDF is related to COLCARDF and FIRSTKEYCARDF and FULLKEYCARDF. It must be at minimum:
    - A value between FIRSTKEYCARDF and FULLKEYCARDF if the index contains the same set of columns
    - A value between MAX(colcardf of each col) and the product of all the columns COLCARDFs in the group
  - CARDF (SYSTABLES): CARDF must be equal or larger than any other cardinalities, such as COLCARDF, FIRSTKEYCARDF, FULLKEYCARDF, and CARDF in SYSCOLDIST
  - FREQUENCYF and COLCARDF or CARDF: The number of frequencies collected must be less than or equal to COLCARDF for the column or CARDF for the column group
  - FREQUENCYF: The sum of frequencies collected for a column or column group must be less than or equal to 1

- Refer to chapter 37 of: „Managing Performance“
How to simulate changes – hardware

- CPU simulation
  - 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
Db2 database/object cloning

- Instant Cloning for clone based code level checks:
  - Scope of Cloning: Object level
    - Fully exploiting instant copy technology (e.g. Flashcopy)
    - Supports DDL and/or data cloning
    - Flexible include/exclude of dependent objects
    - Powerful renaming capabilities

- Application Indexspaces
- Application Tablespaces
- Tables
- Synonyms
- Views
- Aliases
- Triggers
- Grants
- Stogroups
- Permissions
- RI
- Routines
- Sequences
- Databases
- Trusted
- Context
- Distinct
- Data
- MQT
- Roles
- Variables
Db2 database/object cloning

- Instant Cloning for clone based code level checks:
  - Scope of Cloning: Subsystem level
  - Fully exploit instant copy technology (e.g. Flashcopy – ESS, Timefinder, Snapshot)
  - DS→NDS, as well as cross-version cloning possible
  - Highly customizable and fully automated if driven by scripts, like a XML scenario scheme
Db2 database/object cloning

- Required Steps
  - Clone your source data
    - Dump via ADRDSSU
    - Split Mirror systems and break the mirror
    - FLASH Copy / Disk dump and then Restore
    - Any other method...
  - Stop the target system
  - Restore the source data (using the new ssid vcat)
  - Rename (if naming should be different and/or target is not isolated from source)
  - Adjust LOGs, BSDSs, DSNZPARM, DSNHDECP
  - Start target
  - Execute NEWCAT to adjust Db2 object names
Db2 database/object cloning

- XML controlled cloning
  - Due to its nature XML is a fully flexible, human- and machine-readable language
  - It may, or may not have elements and/or attributes, but has to be well-formed only
  - Since a complex cloning procedure may, or may not have individual steps, including some very customer specific tasks, XML is a perfect choice to drive a cloning scenario

```xml
<name>validate datasets</name>
<description>Check installation specific datasets</description>
</menuitem>
- <menuitem>
  <name>Gather information</name>
  <description>Get all needed information</description>
</menuitem>
- <menuitem>
  <name>Stop DB2</name>
  <description>Stop target DB2</description>
</menuitem>
- <menuitem>
  <name>Restore</name>
  <description>Restore volumes</description>
</menuitem>
```
Db2 database/object cloning

- FTP and Routing
  - Usually source and target subsystem reside on different LPARs, machines, or even locations
  - A fundamental part of cloning automation is to take care of routing and transferring required data, no matter if being logged in on source, or target
    - Temporary/Workfile datasets are shared via FTP
    - Commands are routed by adding the system name
      \[
      \text{ROUTE LPRS, } /F \text{ DB2S, STOP DB2}
      \]
    - Jobs can be route either via
      \[
      \text{ROUTE XEQ LPRSNJE, or SCHENV=LPRSDB2S}
      \]
Db2 database/object cloning

- JCL, Panels, Commands, Messages
  - ISPF tailoring services are a great vehicle to prepare skeleton JCL, panels, commands and messages for automation, but keep the flexibility for a variety of individual systems and clone runs

```sql
)DEFAULT )&?!<\>
)SETF JOBDESC1 = &STR(Get UTILITY info of source system)
)TB 12
)IM HSSJOC
)DEFAULT )&?üä|ö
)SET N = 1
//*
//DISUT EXEC PGM=IKJEFT01,DYNAMNBR=20
)IM HSCSSTP
//SYSPRINT DD SYSOUT=&SYOUTDST
//SYSTSIN DD *
DSN SYSTEM(&DB2ID)
-DIS UTILITY(*)
-DIS DATABASE(*) SPACENAM(*) LIMIT(*) RESTRICT
END
//SYSTSPRT DD DISP=(NEW,PASS,DELETE),
```

Imbed jobcard
Desired sysout
Desired Db2 system
Imbed steplib
Db2 database/object cloning

- A clone-environment is designed for automated testing and should be isolated and automatically scratched afterwards.
- However, since we have potentially sensitive production data, consider auditing the entire system, like:
  - SELECTs (against sensitive data)
  - Modifications (INS/UPD/DEL)
  - DDL
  - DCL
  - Utilities
  - Commands
  - Assignment, or modification of a user ID/authorization – especially privileged users
- You may want to terminate the system when unauthorized access occurs.
Automated testing – anomaly alerting

The scope and the environment for automated testing can be chosen flexibly, but of course SQL replay is intended for a test environment.

As a result, the testing scope and associated cost matches the needs. E.g. for a non-critical system with a small change to test a quick access path precheck might be enough. For a highly critical financial OLTP system, facing a major Db2 update, we better run a fully automated clone with a complete replay test that covers multiple workload sets over night.

<table>
<thead>
<tr>
<th>Application Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICI Detection</td>
</tr>
<tr>
<td>AP Check</td>
</tr>
<tr>
<td>Instant Cloning</td>
</tr>
<tr>
<td>Capture/Replay</td>
</tr>
</tbody>
</table>
Automated testing – anomaly alerting

In recent versions, IBM has modified the behavior of certain Db2 functionality
- Built-in Functions (BiFs)
- Reserved Words
- SQL Return Codes
- Deprecated Functionality

It is important that you are aware of, and track, the incompatibilities that may cause issues/problems... why?
- Applications no longer function
- Applications function differently
- The results of your SQL SELECT statements can change
Access path change detection

- (RE-)BIND EXPLAIN only explains based on static SQL optimization rules
  - A feature to populate details of an existing, or a new package without affecting the access path
    - Good for gathering explain data when previously bound with EXPLAIN(NO)
    - Reliable way to prescreen access paths for static packages

- EXPLAIN STATEMENTCACHE explains dynamic statements in the DSC

- EXPLAIN ALL <stmt> explains based on dynamic SQL optimization rules
  - A feature to populate details of a new statement without affecting the access path
    - Reliable way to prescreen access paths for dynamic SQL
  - E.g. RTS # pages/rows = 0
    - EXPLAIN: tablespace scan
    - BIND EXPLAIN(ONLY): index access
Access path change detection

- Automatically and reliably check access path changes

![Diagram showing the process of access path change detection]

1. Plan Table
2. Comparison
3. Impact-Analysis
4. New Access Path Pattern
5. Improved or equal
6. Worsened
7. Report
8. New behavior
## Access path change detection

- **Access Path Check:**
  - Static & Dynamic SQL

### Access Path Check Table

<table>
<thead>
<tr>
<th>STMT No.</th>
<th>Section Number</th>
<th>DSC STMT ID</th>
<th>Impact</th>
<th>Serviceunits Old</th>
<th>Serviceunits New</th>
</tr>
</thead>
<tbody>
<tr>
<td>2547</td>
<td>13</td>
<td>2547</td>
<td>DEGRADED</td>
<td>85</td>
<td></td>
</tr>
<tr>
<td>799</td>
<td>2</td>
<td>796</td>
<td>DEGRADED</td>
<td>152</td>
<td></td>
</tr>
<tr>
<td>804</td>
<td>3</td>
<td>804</td>
<td>DEGRADED</td>
<td>152</td>
<td></td>
</tr>
<tr>
<td>812</td>
<td>4</td>
<td>812</td>
<td>DEGRADED</td>
<td>152</td>
<td></td>
</tr>
<tr>
<td>820</td>
<td>5</td>
<td>820</td>
<td>DEGRADED</td>
<td>152</td>
<td></td>
</tr>
<tr>
<td>671</td>
<td>2</td>
<td>671</td>
<td>DEGRADED</td>
<td>152</td>
<td></td>
</tr>
<tr>
<td>679</td>
<td>3</td>
<td>679</td>
<td>DEGRADED</td>
<td>152</td>
<td></td>
</tr>
<tr>
<td>687</td>
<td>4</td>
<td>687</td>
<td>DEGRADED</td>
<td>152</td>
<td></td>
</tr>
<tr>
<td>695</td>
<td>5</td>
<td>695</td>
<td>DEGRADED</td>
<td>152</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td>DEGRADED</td>
<td>130</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td>DEGRADED</td>
<td>95</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
<td>DEGRADED</td>
<td>82</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>6</td>
<td>DEGRADED</td>
<td>149</td>
<td></td>
</tr>
<tr>
<td>968</td>
<td>7</td>
<td>968</td>
<td>IMPROVED</td>
<td>171</td>
<td></td>
</tr>
<tr>
<td>1167</td>
<td>9</td>
<td>1167</td>
<td>IMPROVED</td>
<td>174</td>
<td></td>
</tr>
<tr>
<td>235</td>
<td>1</td>
<td>235</td>
<td>IMPROVED</td>
<td>178</td>
<td></td>
</tr>
<tr>
<td>1194</td>
<td>2</td>
<td>1194</td>
<td>IMPROVED</td>
<td>175</td>
<td></td>
</tr>
</tbody>
</table>

### Diagram

The diagram shows the REBIND Analysis summary with different colored sections indicating the level of change:
- **Improved** (green)
- **Changed** (yellow)
- **Improved AF props** (blue)
- **Changed AF props** (red)
- **Worsened AF props** (pink)
- **Unchanged AF props** (gray)

*Note: The diagram is a visual representation of the changes detected in the access path check.*
Workload KPI verification at a glance

Workload Capture/Replay with KPI verification:

→ Workload Capture/Replay adds application level testing and automates executing sets of captured workload.

- Highly efficient IFCID (OPx) capturing to catch SQL for automated execution in the isolated, cloned environment.
- Workload sets can be saved to represent quarter’s end, year’s end and other specific workload patterns.
- Tested workload is automatically compared on a KPI level (e.g. # of getpages, rows returned, rows processed…) to report only anomalies.
Workload KPI verification

- Using IFCIDs along with OPx buffers delivers in-depth information without the overhead of SMF processing
- Correlation headers add detailed authentication data

- IFCID 316* and 318 externalize data from the Dynamic Statement Cache (DSC), even when a flushing situation occurs (LRU, RUNSTATs, ALTER, DROP, REVOKE, …) (+317* for the full SQL statement)
- IFCIDs 400* and 401 externalize data from the EDM pool – let’s call it the Static Statement Cache – even when a flushing situation occurs (EDM pool is full) (+SYSPACKSTMT for the full SQL statement)

*This IFCID is not really an IFCID but more of a „switch“ to enable externalization of static SQL metrics
Workload KPI verification

- Counters
  - More than 100 KPIs, like
    - executions, getpages, IX/TS scans, rows processed/examined, ...
- Zero-Counters
  - Failure indicators that should always be zero, like
    - RID list overflow, RID list append, RID pool failure, ...
- Timings
  - More than 80 KPIs, like
    - CPU/elapsed time, claim/lock/latch wait time, thread read/write
- Identification
  - About 20 IDs, like
    - SQL ID, end user, workstation, transaction, ...
- Environmental
  - Metadata, like collection ID, currentdata, isolation level, ...
Workload KPI verification

- SQL workload that has been captured can be divided into two groups:
  1. Re-executable statements
     - SELECT A, B, C FROM MYTABLE WHERE B = 'B'
  2. Non re-executable statements
     - SELECT A, B, C FROM MYTABLE WHERE B = ?
     - SELECT A, B, C FROM MYTABLE WHERE B = :B

- SQL with literals and host variables needs to be prepared for re-execution
Workload KPI verification

In dynamic SQL statements host variables are represented as question marks (parameter markers). There are *typed* and *untyped* parameter markers.

- Typed parameter markers are specified with their target data type (via `CAST`).
- Untyped parameter markers are specified in the form of a single question mark.
Workload KPI verification

To make non re-executable statements executable, we replace parameter markers and host variables with real values, considering the characteristics of the affected column.

```
SELECT A, B, C FROM MYTABLE WHERE B = ?
```

A simple shot is a character = 'A' and numeric = 5 replacement. A more sophisticated solution checks catalog statistics for more solid replacements.
Workload KPI verification

- The captured statements can (and should) represent various workloads to cover a representative scope
  - Month’s end processing
  - Quarter’s end processing
  - Year’s end processing
  - Typical OLTP
  - Typical batch

... and can be bundled in workload sets to be individually chosen for testing.
Workload KPI verification

- Executing the captured statements dynamically doesn’t require the associated applications
  - There are typically no static programs/packages as part of the Db2 clone
- Consider parameters allowing to chose the number of executions per statement
- Ignore certain SQL errors/warnings, like
  - Duplicate key
  - Object exists
  - Grantee already has the permission
  - +100
  - ...

© 2018 SOFTWARE ENGINEERING GMBH and SEGUS Inc.
Workload KPI verification

- Before starting the mass execution, verify to be in our own cloned and encapsulated environment!
- The entire execution needs to be monitored (using OPx based, highly-efficient capturing technology) to gather comparison metrics and KPIs, like
  - CPU consumption
  - Access path pattern
  - Rows processes/examined
  - ...
for a before and after comparison of changes, like:
- New application release
- System changes
- Db2 APAR/PTF/new Db2 12 modification level
- Environment/hardware changes
The big benefit of KPI verification is the ability to handle massive amounts of testing without the overhead of having to review individual executions.
Workload KPI verification

Drill down capabilities allow looking into details, when anomalies are detected
Drill down capabilities allow looking into details, when anomalies are detected.
Workload KPI verification

BIF Usage is a major area of concern and occurs quite often. The ability to test multiple sets of workloads, detects even quarters-end, or years-end query issues before they occur in production.