IBM Z Tech Bytes – May 2025

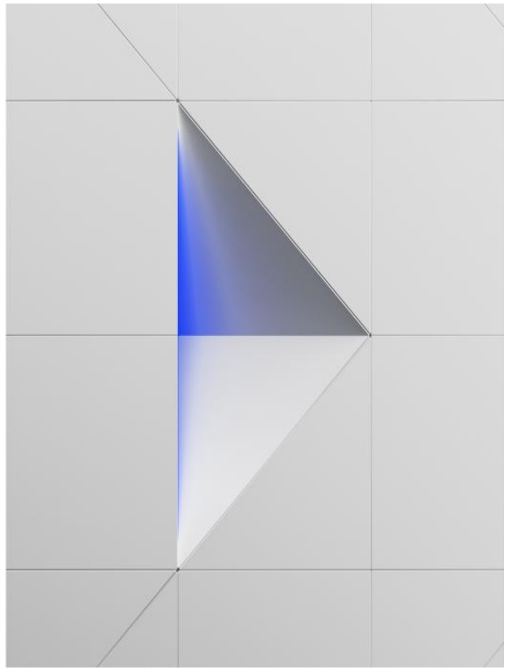
IBM Washington Systems Center

Db2 13 for z/OS Latest Features

Tori Felt
Victoria.felt@ibm.com

IBM

Contents



Autobind phase-in

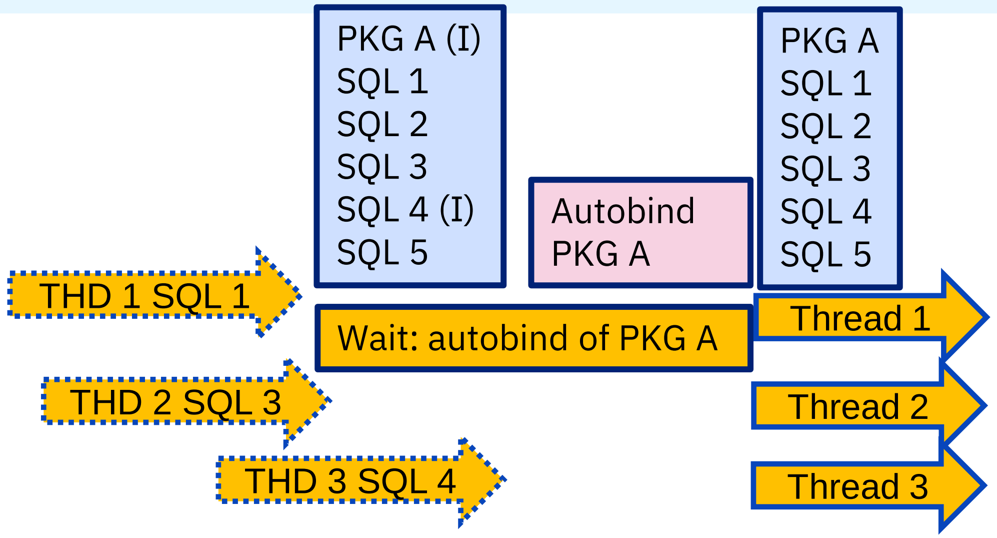
Package invalidation – reducing impact of autobind (1|3)

If package invalidated, next request to execute will trigger autobind

Previous behavior (V13R1M503 and prior)

– Autobind can significantly impact workload

- Package that triggered autobind has to wait for autobind to complete
- Other requests for same package also have to wait on autobind completion
- If autobind fails (infrequent)
  - Package marked inoperative
  - Explicit rebind required



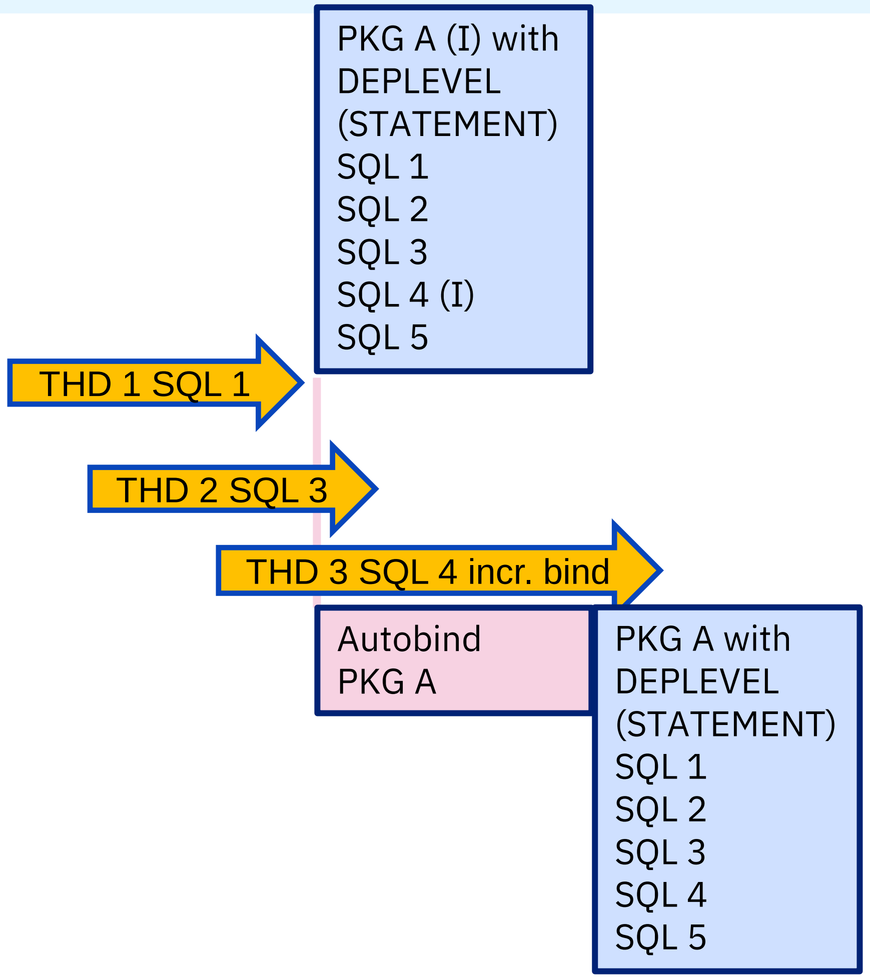
Package invalidation – reducing impact of autobind (2|3)

FL 502

FL 504

New behavior

- Package BIND/REBIND with DEPLEVEL(STATEMENT) option FL 502
  - Record statement-level dependencies in SYSPACKSTMTDEP
- If ALTER invalidates package bound with DEPLEVEL(STATEMENT), next package request triggers autobind
  - Autobind in background (phase-in) FL 504
  - Package executes before autobind completes
    - Non-invalidated statements as usual
    - Invalidated statements with incremental bind



PKG A (I) with DEPLEVEL (STATEMENT) SQL 1 SQL 2 SQL 3 SQL 4 (I) SQL 5
THD 1 SQL 1
THD 2 SQL 3
THD 3 SQL 4 incr. bind
Autobind PKG A
PKG A with DEPLEVEL (STATEMENT) SQL 1 SQL 2 SQL 3 SQL 4 SQL 5

Package invalidation – reducing impact of autobind (3|3)

FL 500

FL 502

FL 504

FL500: CATMAINT can be executed to take catalog level to V13R1M501 level

– One of the new tables in V13R1M501 catalog is SYSPACKSTMTDEP

FL502: packages can bound/rebound with new DEPLEVEL(STATEMENT) option

– That causes statement-level dependencies to be recorded in SYSPACKSTMTDEP

FL504: if ALTER causes invalidation of package bound with DEPLEVEL(STATEMENT), next request to execute package still triggers autobind, BUT…

– Autobind done in background, and package can still be executed even before autobind completes: non-invalidated statements execute as usual, invalidated statements incrementally bound when executed

– When autobind completes, newly-regenerated package phased in (similar to rebind phase-in functionality of Db2 12 FL505)

– If autobind fails, package gets "advisory rebind" status and can still be executed (non-invalidated statements execute as usual, invalidated statements incrementally bound when executed)

- SYSPACKAGE: OPERATIVE = 'R'
- Explicit rebind will put package back in valid state

Statement-level dependency infrastructure (Catalog change)

FL 500

FL 502

Previous behavior

- Db2 tracks application dependencies at package level
- An operation on any object requiring invalidation results in the entire package marked as invalid; even when only a subset of SQL statements in that package needs to be invalidated
- This is broad and limits Db2 flexibility to enhance and improve invalidation processing

Db2 13 behavior

- Provide more granular dependency & validity tracking infrastructure, laying foundation for enhancements such as reduced impact of invalidated packages and improved DDL & static DML concurrency
- New DEPLEVEL BIND/REBIND option determines recording of statement level dependencies in addition to package level dependencies (FL 502)
- New system parameter PACKAGE_DEPENDENCY_LEVEL (SPRMPKGDEPLVL) sets DEPLEVEL default
- New catalog tables SYSPACKSTMTCOPY and SYSPACKSTMTDEP (Catalog V13R1M501)

Utility execution history

Utilities History Table – overview

FL 501

FL 504

- New catalog tables: **SYSIBM.SYSUTILITIES** (501 catalog level) **SYSIBM.SYSOBJEVENTS** (504 level)
- New ZPARM **UTILITY_HISTORY** – possible values:
  - **NONE** – Default value (preserves existing behavior – typical default for new ZPARM)
  - **UTILITY** – Directs Db2 to insert a row into SYSIBM.SYSUTILITIES at the start of each utility execution (this functionality is available when the activated function level is V13R1M501 or higher)
  - **OBJECT** – In addition to inserting utility execution information in SYSIBM.SYSUTILITIES, Db2 will insert a row into SYSIBM.SYSOBJEVENTS for each object (page set or partition) processed by a utility
    - This functionality is available when the activated function level is V13R1M504 or higher
    - Note: prior to activation of function level V13R1M504, information about an object processed by a utility can be obtained from the SYSCOPY catalog table (for a utility that generates SYSCOPY information) – SYSCOPY can be joined with SYSUTILITIES via the EVENTID column that appears in both tables
- Clean-up: at present, removal of rows from SYSUTILITIES and SYSOBJEVENTS is via user-issued DELETEs

Utilities History Table – normal flow

FL 501

```
/ DB2COPY JOB DB2ADM …
/ STEP1 EXEC DSNUPROC,UID='COPYTS' …
/ SYSIN DD *
LISTDEF COPYLIST
         INCLUDE TABLESPACE DSN8D13A.DSN8S13E
         INCLUDE TABLESPACE DSN8D13A.DSN8S13D
COPY LIST COPYLIST …
```

1 When the utility driver begins execution, a row is INSERTed

EVENTID	NAME	JOBNAME	UTILID	USERID	STARTTS	STARTLOGPOINT	CONDITION
1001	COPY	DB2COPY	COPYTS	DB2ADM	2022-04-05 13:26	…001F8C16A04…	*blank*

2 After utility-in-progress states are set, the row is UPDATEd

EVENTID	NUMOBJECTS	LISTNAME
1001	2	COPYLIST

3 When the utility terminates, the row is finally UPDATEd

EVENTID	ENDTS	ELAPSED TIME	CPU TIME	ZIIP TIME	SORT CPUTIME	SORT ZIIPTIME	RETURNCODE	CONDITION
1001	2022-04-05 13:28	418295	22910	0	0	0	0	E

Remark: The table columns and data is simplified for display purpose.

Utilities History Table – special cases

FL 501

When a utility ABENDs, the row is **not** updated. The utility is in a stopped state.

EVENTID	ENDTS	RETURNCODE	CONDITION
1002	NULL	NULL	*blank*

Issue –DIS UTIL command to determine if active or stopped

When a utility is RESTARTed, the corresponding row is UPDATEd like this:

EVENTID	RESTART	JOBNAME	USERID	GROUP_MEMBER
1002	Y	blank		DSNB

When a utility completes after RESTART, the row is finally UPDATEd like this:

EVENTID	ENDTS	ELAPSEDTIME	CPUTIME	ZIIPTIME	RETURNCODE	CONDITION
1002	2022-04-05 13:28	418295	22910	0	0 | 4 | 8	E

When a –TERM UTIL or STA DB(…) SP(…) ACCESS(FORCE) command terminates a *stopped utility*, the corresponding row is updated like this:

EVENTID	ENDTS	ELAPSEDTIME	RETURNCODE	CONDITION
1002	2022-04-05 13:28	418295	NULL	T | F

ELAPSEDTIME includes the time the utility was in stopped state

When a –TERM UTIL command is issued on an *active utility*, the corresponding row is updated like this:

EVENTID	ENDTS	ELAPSEDTIME	RETURNCODE	CONDITION
1002	2022-04-05 13:28	418295	8	T

Utilities History Table – operational aspects

FL 501

- New messages are added to the utility job output
  - **DSNU3031I** UTILITY HISTORY COLLECTION IS ACTIVE. LEVEL: UTILITY, EVENTID: *event-id-number*
  - **DSNU3032I** ERROR DURING UTILITY HISTORY COLLECTION, RETURN CODE X'*return-code'* REASON CODE X'*reason-code'*
- SQL INSERT, UPDATE and DELETE are allowed on SYSIBM.SYSUTILITIES table, e.g. for cleanup processing (example in Db2 13 and More Redbook) or tools integration
- It is recommended to use ISO(UR) when querying SYSIBM.SYSUTILITIES to avoid contention
- Users can define indexes on the table as needed to optimize query performance
- Utility history information is not collected for utilities executed on SYSIBM.SYSUTILITIES table, its index and tablespace, for RECOVER or REBUILD INDEX on catalog and directory objects, for objects in a restrictive state and when executing in preview mode

Utilities History Table – Sample queries

FL 501

- *“Show all utilities that started/stopped between midnight and 6am”*
- *“Show all utilities that ended with one or more errors (RC >=8) in the last 24 hours”*
- *“Show the top 10 CPU-consuming utility executions in the last 7 days”*
- *“Show restarted utilities in active or stopped state”*
- *“Show the most recent successful execution of REORG TABLESPACE for a specific table space or REORG INDEX for a specific index space”* (joining data in SYSUTILITIES and SYSCOPY using the EVENTID column)
- SQL and more sample queries are available in the Db2 13 for z/OS and More Redbook (SG24-8527-00)

Granular specification of security requirements

More granular specification of security requirements for DDF application: the problem

- Security requirements (e.g., use of AT/TLS – aka SSL – encryption, use of certificates vs. passwords for client authentication) vary for different Db2 for z/OS client-server applications…
  - Db2 control mechanisms were often of the "big switch" variety – "on" or "off" for all client-server applications
- Problems with "big switch" controls:
  - Inflexible – Db2 for z/OS can be set up to require use of SSL encryption for *ALL* network-connected applications to support SSL encryption
  - Risk – Increased risk of application disruptions (what if some application doesn't "get it right" at the outset?)

The Db2 13 solution

- Db2 13 extends the profile table functionality for specific security requirements in a very granular way for client-server applications
  - APARs PH48764 and PH57811
- Different security requirements can be specified for different categories of Db2 client-server applications (e.g., JDBC, ODBC, REST) *and for specific application servers within a category*
  - The Db2 profile tables (DSN_PROFILE_TABLE, DSN_PROFILE_ATTRIBUTES) have long been use-able for granular control of a Db2 client-server workload (e.g., to set connection and thread limits for specific applications)

Granular client-server application security control

- Example:

SYSIBM.DSN_PROFILE_TABLE

PROFILEID	LOCATION	ROLE	AUTHID	PRDID	COLLID	PKGNAME
101	1.2.3.4	null	null	null	null	null

SYSIBM.DSN_PROFILE_ATTRIBUTES

PROFILEID	KEYWORDS	ATTRIBUTE1	ATTRIBUTE2	ATTRIBUTE3
101	MONITOR JDBC CONNECTIONS FOR SECURITY	EXCEPTION	1	1

- What this set-up means:
  - JDBC-using app running on app server at address 1.2.3.4 must (ATTRIBUTE1) authenticate with an ID and a password or passphrase (ATTRIBUTE2) and must use SSL encryption (ATTRIBUTE3)

    ** Other category choices: CLI (usually referring to ODBC driver), REST, DSN (referring to another Db2 for z/OS system), DB2CONNECT (referring to Db2 Connect "gateway" server), and * (meaning, "Otherwise…")*
  - Security requirement for categories not referenced is based on value of TCPALVER in ZPARM

Reference info: https://www.ibm.com/docs/en/db2-for-zos/13?topic=support-discovering-controlling-secure-tcpip-connectivity-profile-tables

Monitor connections for security: AUTHID support

Extended support for using authentication IDs with profile monitoring was released with APAR PH63652.

- No need to know IP addresses or other LOCATION-based filtering criteria
- Wildcards are supported

PROFILE ID	**LOCATION**	**ROLE**	**AUTHID**	**PRDID**	**COLLID**	**PKGNAME**
24	Null	Null	**USER24**	Null	Null	Null
25	Null	Null	**USER***	Null	Null	Null

How this changes the game

- Flexibility: Easy to implement different security requirements for different client-server applications
- Phased implementation of new security controls for client-server applications
    - If you want to go from password-based to certificate-based client authentication, you can do that starting with one application running on one app server
    - When you have it working for that one application, you have a template that can be used to implement the change for other applications
    - Reduces risk versus “big switch” approach

Now we have lots of switches…

And one more thing…

- The value of warning vs. exception mode
  - Exception mode: if security requirement not met, Db2 will refuse connection request
  - Warning mode: if security requirement not met, Db2 allows connection request, issues informational message
- Warning-mode security requirements are a great way to "get the lay of the land"
  - Scenario: suppose you want to switch from password-based to certificate-based authentication for your JDBC-using Db2 client-server applications
  - Implement profile table-based requirement for certificate-based client authentication for JDBC-using applications in warning mode
  - The information provided by Db2 will help you scope the effort for this transition – are some JDBC-using apps already using certificate-based authentication?

Enhanced data availability when clearing data partitions with LOAD

Enhanced data availability when clearing data partitions with LOAD

FL 506

Several techniques available for clearing a data partition:

- SQL DELETE on all qualifying records
- REORG DISCARD WHEN on qualifying records
- LOAD PART REPLACE with empty/dummy input on target partition

➔ Different requirement based on data availability, logging volumes, recoverability, performance and resource consumption

Enhanced data availability when clearing data partitions with LOAD

FL 506

Common use case: reuse oldest partition of PBR table space (e.g. partitioned on time-series basis)

Typical procedure to reuse a data partition:

- Unload/archive data in oldest partition that is to be reused
- LOAD PART REPLACE with empty/dummy input on target partition
  - Non-logging deletes of all rows/keys associated with target partition
- ALTER TABLE… ROTATE PARTITION FIRST TO LAST

Current behavior:

Data unavailability window = elapsed time of the LOAD utility

- Clearing of partition for table space (and partitioned indexes, if any) -> fast!
- Delete entries in NPIs -> *much slower!*
- Meanwhile, target objects under UTUT (Utility restrictive state, exclusive control) access and drain all by LOAD utility
  - NPI access restricted; entire table space is non-updatable if NPI is unique

Enhanced data availability when clearing data partitions with LOAD

FL 506

New behavior – Reduce total data unavailability time for LOAD REPLACE clearing of partition

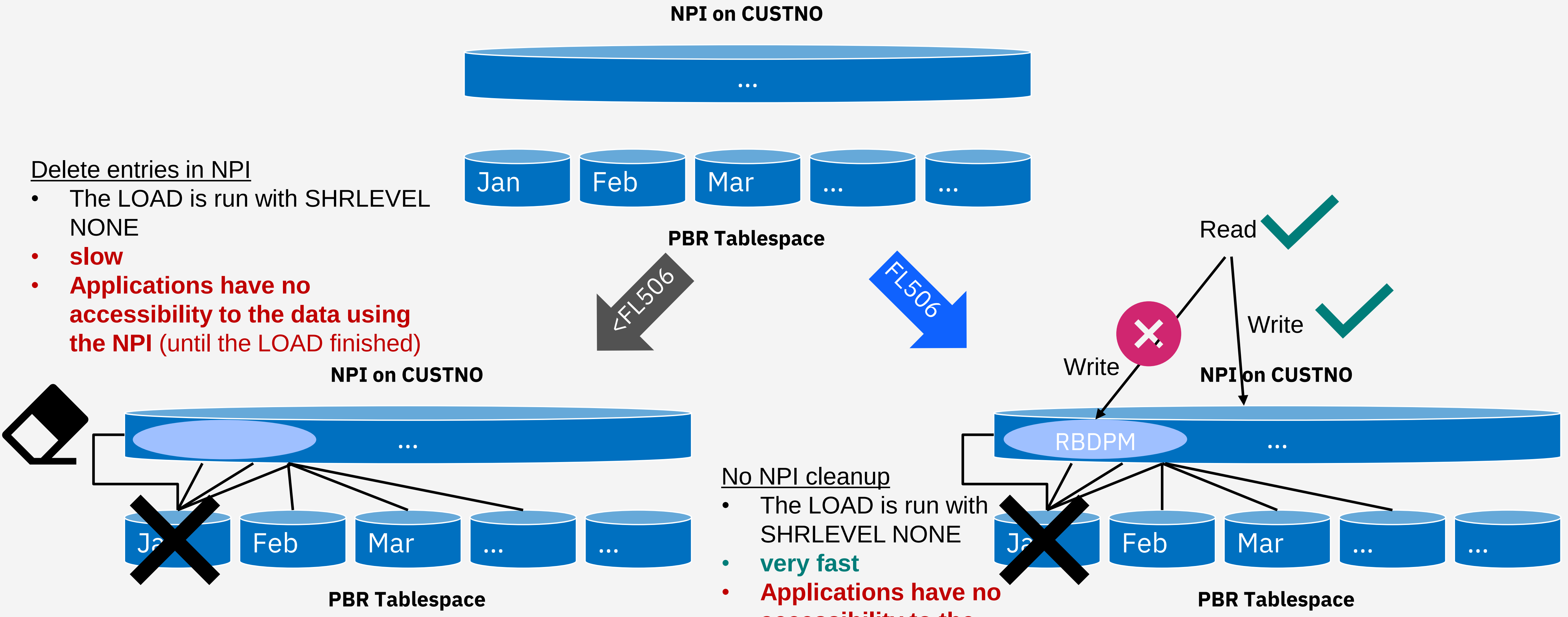
– FL506 introduces new LOAD option:
  - LOAD INTO TABLE PART REPLACE INDEXDEFER NPI NOKEYDELETE SHRLEVEL NONE

– LOAD skips scan and delete of entries in NPIs

– Sets new exception state, RBDPM (rebuild-pending empty), on affected logical part of NPI
  - Physical entries for replaced partition remain in NPI tree structure
  - Db2 treats logical partition in RBDPM as empty on index access
  - Associated data partition must remain empty –> -904 with 00C900F6 on INSERT
  - COPY INDEX, CHECK INDEX blocked; be wary of using DSN1COPY
  - Recommended: REORG INDEX SHRLEVEL CHANGE to remove RBDPM

Enhanced data availability when clearing data partitions with LOAD

FL 506



Delete entries in NPI
- The LOAD is run with SHRLEVEL NONE
- **slow**
- **Applications have no accessibility to the data using the NPI** (until the LOAD finished)

No NPI cleanup
- The LOAD is run with SHRLEVEL NONE
- **very fast**
- **Applications have no accessibility to the data using the NPI** (until the LOAD finished)

Recommended:
REORG INDEX SHRLEVEL CHANGE to remove RBDPM

New syntax for multi-row INSERT

New syntax for multi-row INSERT

- Here is the familiar Db2 for z/OS syntax for insert of a single row into a table:

```
INSERT INTO EMPLOYEE
(EMPNO, FIRSTNAME, LASTNAME, WORKDEPT)
VALUES
   ('000206', 'ELIZABETH', 'GRACE', 'A11’);
```

- Db2 for z/OS has had multi-row INSERT capability since Db2 V8

Multi-row INSERT introduce in Db2 V8

- Db2 for z/OS has had multi-row INSERT capability since Db2 V8
  - This involved use of host variable arrays, declared and populated by the row-inserting program
  - There would be one host variable array for each column of the target table, and the multi-row INSERT statement might look like this

```
INSERT INTO EMPLOYEE (EMPNO, FIRSTNME, LASTNAME, WORKDEPT)
  VALUES (:hva1, :hva2, :hva3, :hva4) FOR 3 ROWS
  NOT ATOMIC CONTINUE ON SQLEXCEPTION;
```

Multi-row INSERT

- Multi-row INSERT capability introduced in Db2 13 FL506:

```
INSERT INTO EMPLOYEE
(EMPNO, FIRSTNAME, LASTNAME, WORKDEPT)
VALUES
   ('000206', 'ELIZABETH', 'GRACE', 'A11'),
   ('000207', 'JACK', 'JOHNSON', 'B13'),
   ('000208', 'JENNIFER', 'WHITE', 'D15');
```

- A *more-intuitive, more programmer-friendly* way to get the efficiency benefits of multi-row insert, versus the host variable array form

- Db2 for Linux/UNIX/Windows already supported the Db2 13 FL506-introduced syntax – good for developers who work with both of these members of the Db2 family

Questions

?

Summary

- Auto-bind phase-in
  - Less application disruption due to ALTER activity
- Utility execution history – object-related information
  - Operations benefits, and possible cost reductions
- More-granular specification of security requirements for DDF-using applications
  - Easier to roll out required authentication and encryption changes for DDF apps
- Enhanced data availability when clearing data partitions with LOAD
  - Smoother operations, less application disruption
- New syntax for multi-row INSERT
  - Programmer productivity, application portability, application performance
- FL 507 is now available!! https://www.ibm.com/docs/en/db2-for-zos/13.0.0?topic=levels-function-level-507

Thank you