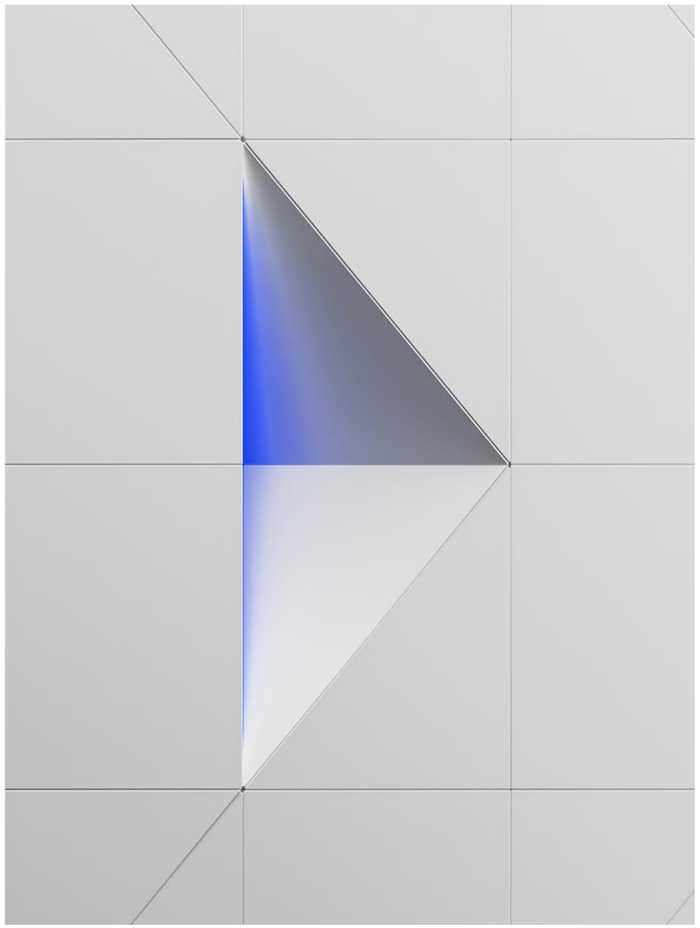
The Five Highest-Value Features of Db2 13 for z/OS

Tridex
September 12, 2024

Mark Rader, IBM Z WSC
mrader@us.ibm.com

Tori Felt, IBM Z WSC
victoria.felt@ibm.com

Agenda



SQL Data Insights

Utility execution history
- FL 504 Object level history

Application specific lock timeout
- CURRENT LOCK TIMEOUT

Online conversion from PBG to PBR

AUTOBIND phase-in

Questions

Summary

SQL Data Insights

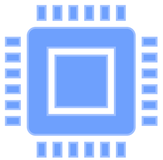
SQL Data Insights … new feature of Db2 13

An industry-leading relational database with embedded AI capabilities

Infuses AI directly into your database on existing data to **discover hidden information**

Minimizes complexity of deploying AI into your applications

Single model used for a range of inferencing tasks over multiple fields



Exploits IBM Z **zIIP and SIMD**

SQL Data Insights … extract greater value from Db2 data

Ease of Use

–Build Neural Network powered relationship maps using unsupervised training over (unlabeled) structured data

–Simply select data, enable training and Db2 for z/OS builds a data relationship model

–Apply relationship maps and built-in AI-related functions within any SQL statement

–Readily interpret underlying reasons for insight

Major Benefits

–No deep data scientist skills required

–Rapid time to develop and deploy AI

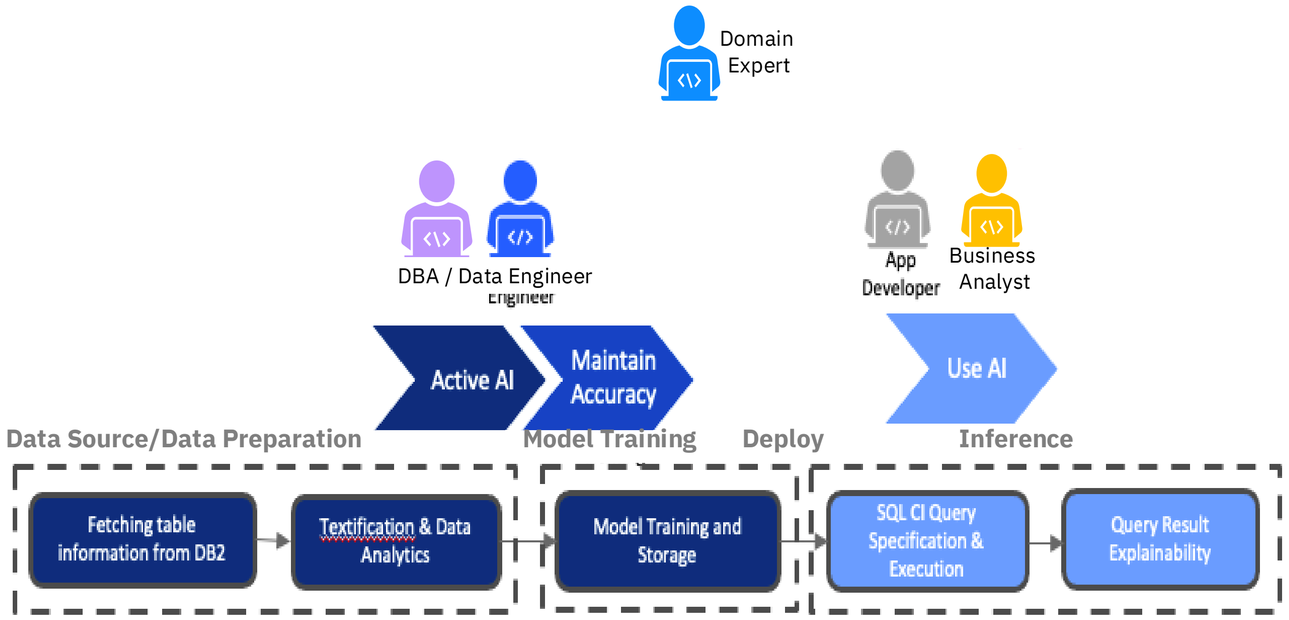
–No specialized architecture

–Efficient AI scoring (elapsed time, CPU, throughput)

–Highly efficient retraining and redeployment

–No data latency

–Model can address multiple questions



Applicable to a broad range of enterprise critical domains: Finance, Insurance, Retail, Security, HR, IT Management, Data Integration, etc.
(Entity Resolution; Data Cleansing)

SQL Data Insights – built-in Db2 functions

FL 500

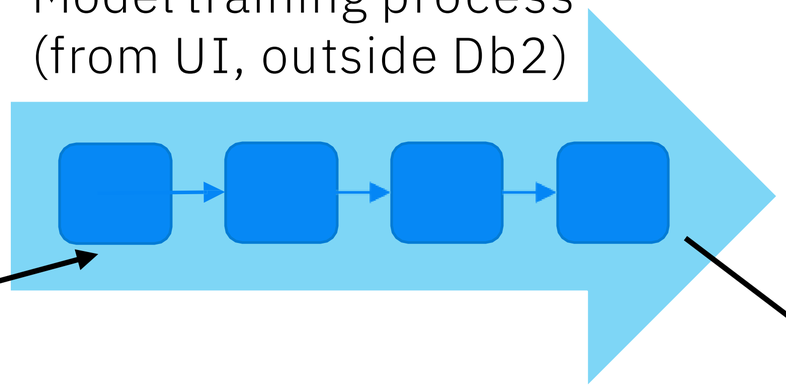
Function	Description
AI_SIMILARITY	Returns the entities that are most similar to (or dissimilar to) a particular entity
AI_SEMANTIC_CLUSTER	Returns the entities that are most similar to (or dissimilar to) a given set of up to three entities
AI_ANALOGY	Consider the relationship between value X in COL1 and value Y in COL2, and return the most analogous COL2 values if the COL1 value is Z
AI_COMMONALITY FL 504	Returns the values of a column that are outliers with respect to all rows in a table

SQL Data Insights … examples of functions

Built-in function - example	What the query does
SELECT V.VENDOR_NAME, AI_SIMILARITY(VENDOR_NAME, 'IBM CORPORATION') FROM VENDORS V ORDER BY 2 DESC FETCH FIRST 10 ROWS ONLY;	Returns the 10 vendors that are most similar to IBM Corporation
SELECT V.VENDOR_NAME, AI_SEMANTIC_CLUSTER(VENDOR_NAME,'IBM CORPORATION','AMAZON', 'MICROSOFT') FROM VENDORS V ORDER BY 2 DESC FETCH FIRST 10 ROWS ONLY;	Returns the top 10 vendors that are most similar to the cluster of vendors comprised of IBM Corporation, Amazon and Microsoft
SELECT V.SERVICE_COUNTRY, AI_ANALOGY('IBM CORPORATION' USING MODEL COLUMN VENDOR_NAME, 'USA' USING MODEL COLUMN SERVICE_COUNTRY, 'SAMSUNG' USING MODEL COLUMN VENDOR_NAME,SERVICE_COUNTRY) FROM VENDORS V ORDER BY 2 DESC FETCH FIRST 10 ROWS ONLY;	Returns the 10 service countries that, when paired with the vendor Samsung, are most analogous to the service country USA as it is paired with IBM Corporation
SELECT AI_COMMONALITY(C.DRIVERS_LICENSE_NUMBER) AS SCORE, C.* FROM SDILAB.INSURANCE C ORDER BY SCORE ASC FETCH FIRST 5 ROWS ONLY;	Returns the top 5 drivers licenses that are outliers with respect to all rows in the SDILAB.INSURANCE table

SQL Data Insights ... model representation

Model training process
(from UI, outside Db2)

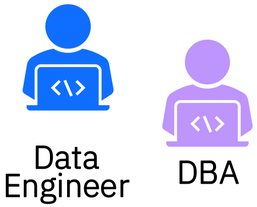


A model is a Db2 table containing encoded vectors for each distinct entity in the source table

SQL semantic functions retrieve the vectors to calculate their results

USER.DATA_TABLE

CUSTOMER_ID	GENDER	BILLING	...
3668-QPYBK	F	auto	...
...	...	...	...

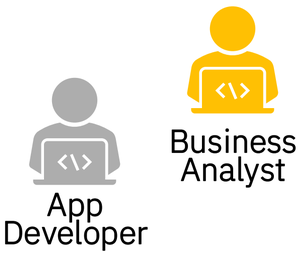


DSNAIDB.<generated vector table name>

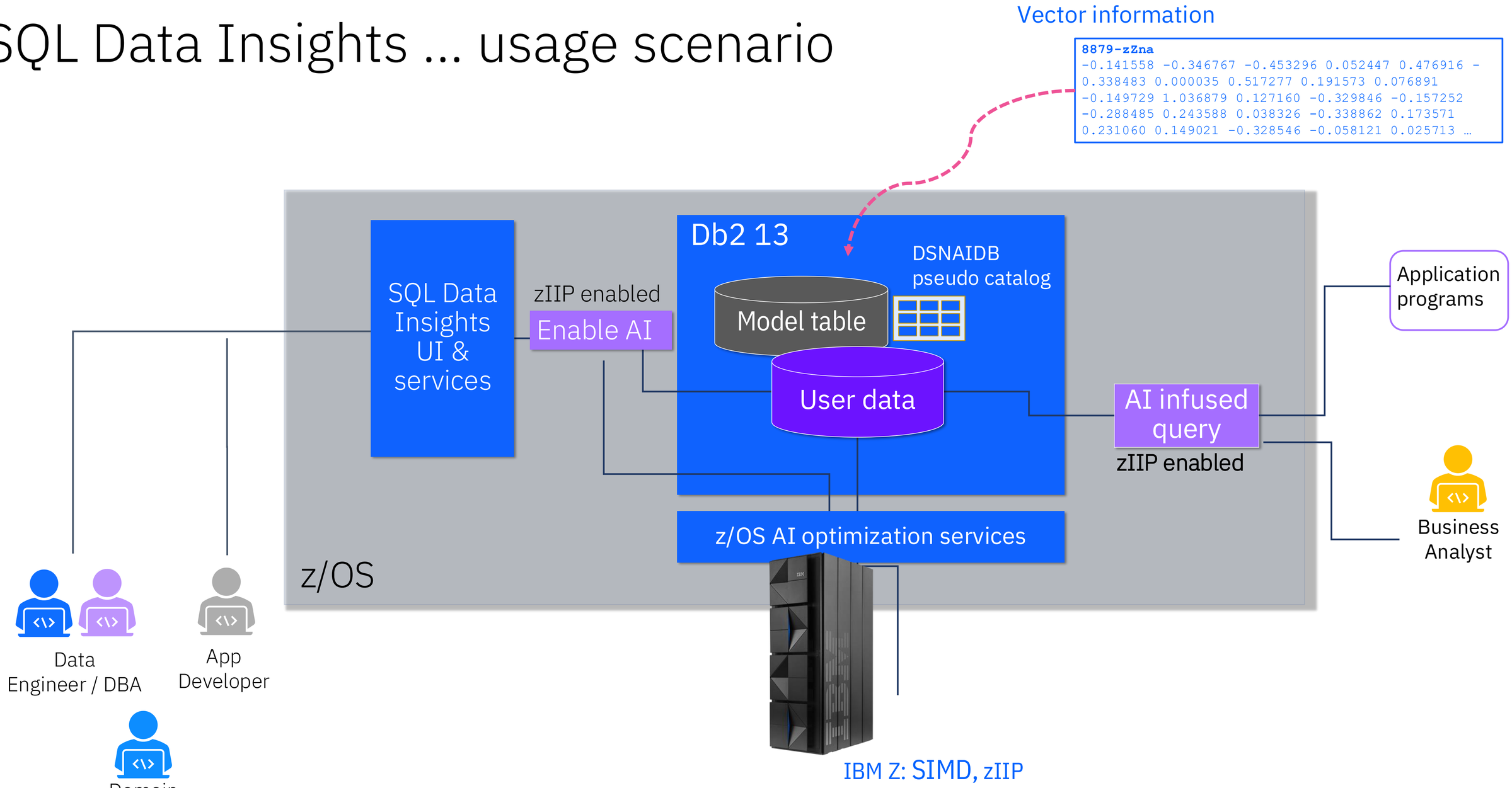
Column	Value	vector
CUSTOMER_ID	3668-QPYBK	<1280 byte vector>
GENDER	F	<1280 byte vector>
...	...	...

SQL:

```
SELECT CustomerID,
AI_SIMILARITY(CUSTOMER_ID, '3668-QPYBK')
FROM USER.DATA_TABLE
WHERE ...
```



SQL Data Insights … usage scenario



Finding hidden information

I want to find other customers like this one

Customer ID	Gender	Senior Citizen	Dependents	Tenure	Phone Service	Multiple Lines	Internet service	Contract	Paperless billing	Payment method	Charges	Churn
001	Male	0	NO	2	YES	NO	DSL	Month-to-Month	YES	Mailed Check	120.22	YES

```
SELECT * ,
AI_SIMILARITY (Customer_ID, '001')
AS SimScore
FROM  TABLE WHERE ....
ORDER BY SimScore DESC
```

Ranked Similarity Results (most to least similar)

Sim Score	Customer ID	Gender	Senior Citizen	Dependents	Tenure	Phone Service	Multiple Lines	Internet service	Contract	Paperless billing	Payment method	Charges	Churn
0.80	004	Male	0	NO	1	YES	NO	DSL	Month-to-Month	YES	Mailed Check	48.55	YES
0.75	002	Male	0	NO	7	NO	NO	DSL	Month-to-Month	YES	Mailed Check	51.00	YES
0.70	006	Male	0	NO	3	NO	NO	DSL	Month-to-Month	YES	Mailed Check	49.80	YES
0.55	003	Female	0	NO	4	NO	NO	DSL	Month-to-Month	YES	Mailed Check	60.40	YES
0.35	005	Female	1	NO	1	NO	NO	DSL	Month-to-Month	YES	Credit Card	55.10	YES

SQL Data Insights … software & hardware requirements

Software:

- Db2 13 for z/OS BIFs
- SQL Data Insight UI and training services
  - separately orderable, no-charge feature of Db2 13 (FMID HDBDD18)
- z/OS maintenance
  - z/OS 2.4 and above and
  - Install 3 IBM neural network libraries

Separate install steps needed to enable SQL Data Insights – refer to IBM Documentation and IBM Redbooks

Hardware:

- zEC12 and above
- Z14 + leverages OpenBLAS library exploitation for AI with SIMD
- Both training and SQL execution are ZIIP eligible

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'*
    - *Note: utility will complete with RC 00 if this is the only problem*
- 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)

Application deadlock and timeout control

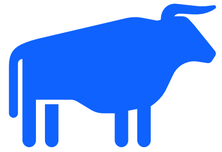
Application deadlock and timeout control

Current behavior

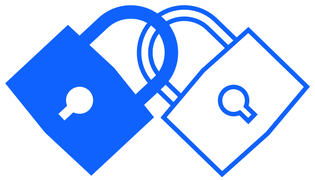
- Single subsystem parameter IRLMRWT
  - Seconds before resource timeout detected
  - Challenges:
    - No granularity by application process
    - Constrains multi-tenancy
    - Short response time application requirements constrained by single value across Db2 member or subsystem
- Deadlock resolution applies equally to all processes in the Db2 member or subsystem

App1

App2



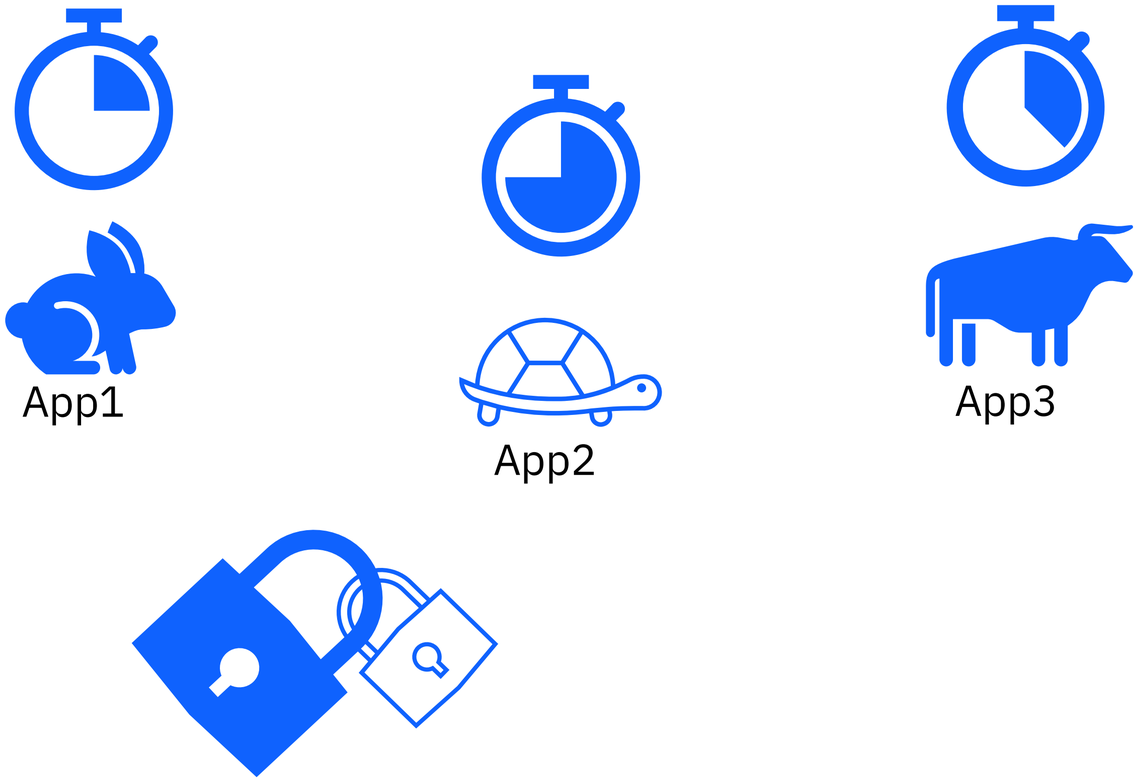
App3



Application deadlock and timeout control

New behavior

- New special register — FL 500
  - CURRENT LOCK TIMEOUT
    - Set at application or SQL statement level
- New global variable — FL 501
  - DEADLOCK_RESOLUTION_PRIORITY
    - Weighting factor in resolving deadlocks with other threads
- Either or both can be set with system profile monitoring



Application deadlock and timeout control - details

CURRENT LOCK TIMEOUT

FL 500

- INTEGER, range -1 to 32767
  - -1 – no timeouts; wait until lock released or deadlock detected
  - 0 – application does not wait for a lock
  - 1-32767 – seconds to wait for a lock
    - Limited by subsystem parameter SPREG_LOCK_TIMEOUT_MAX
  - NULL / not specified – use IRLMRWT subsystem parameter
  - DSNT376I modified to include special register settings, if applicable

DEADLOCK_RESOLUTION_PRIORITY

FL 501

- SMALLINT, range 0-255
- Process with highest priority wins

SPREG_LOCK_TIMEOUT_MAX

FL 100

* IRLMRWT now online changeable

LOCK TIMEOUT MAX

```
DSNTIPI                          IRLM PANEL 1
 ===>

 Enter data below:

    1 INSTALL IRLM       ===> YES       IRLM is required for DB2. Should the
                                        IRLM distributed with DB2 be installed?
    2 SUBSYSTEM NAME     ===> IRLM      IRLM MVS subsystem name
 *  3 RESOURCE TIMEOUT   ===> 30        Seconds to wait for unavailable resource
    4 PROC NAME          ===> IRLMPROC  Name of start procedure for IRLM
    5 U LOCK FOR RR/RS   ===> YES       Lock mode for update cursor with
                                        RR or RS isolation. YES or NO
    6 X LOCK FOR SEARCHED U/D ===> NO     Use X lock for searched updates or
                                          deletes. NO, YES, or TARGET
    7 START IRLM CTRACE  ===> NO        Start IRLM component traces at startup
                                        Blank, NO, YES, or 10 - 255
    8 IMS BMP TIMEOUT    ===> 4         Timeout multiplier for BMP. 1-254
    9 DL/I BATCH TIMEOUT ===> 6         Timeout multiplier for DL/I. 1-254
   10 RETAINED LOCK TIMEOUT ===> 0      Retained lock timeout multiplier. 0-254
   11 LOCK TIMEOUT MAX      ===> -1     CURRENT LOCK TIMEOUT special register
                                        maximum value. 0 to 32767 seconds, or -1

 PRESS:  ENTER to continue   RETURN to exit   HELP for more information
```

-1: any supported value can be specified in SET CURRENT LOCK TIMEOUT statement (default)

– NOTE: make sure you set this zparm to a non-default value

0-32767: maximum value that can be specified in a SET CURRENT LOCK TIMEOUT statement

Profile table support for local threads

Current behavior

- Profile tables can be used by DBA to set special registers and global variables
  - Distributed threads only
- Local applications cannot easily change:
  - Special registers
  - Global variables
- Application developer required to make changes to local applications

New behavior

- Profile tables enhanced
  - Local thread support in some situations
  - New special register:
    - CURRENT LOCK TIMEOUT — FL 500
  - New built-in global variable:
    - SYSIBMADM.DEADLOCK_RESOLUTION_PRIORITY — FL 501
  - New keyword:
    - RELEASE_PACKAGE — FL 500

Online conversion from PBG to PBR

Online conversion from PBG to PBR (1|5)

FL 500

- Partition-by-growth (PBG) is considered the default type of universal table space (UTS) and most table spaces converted to UTS are PBG
- PBG works well for small and medium sized tables
- For larger tables, partition-by-range (PBR) has several advantages over PBG:
  - Greater insert throughput
  - Enhanced query performance
  - Easier to maintain clustering within partitions
  - Ability to have partitioned indexes
  - Maximizes utility independence and parallelism

Online conversion from PBG to PBR (2|5)

FL 500

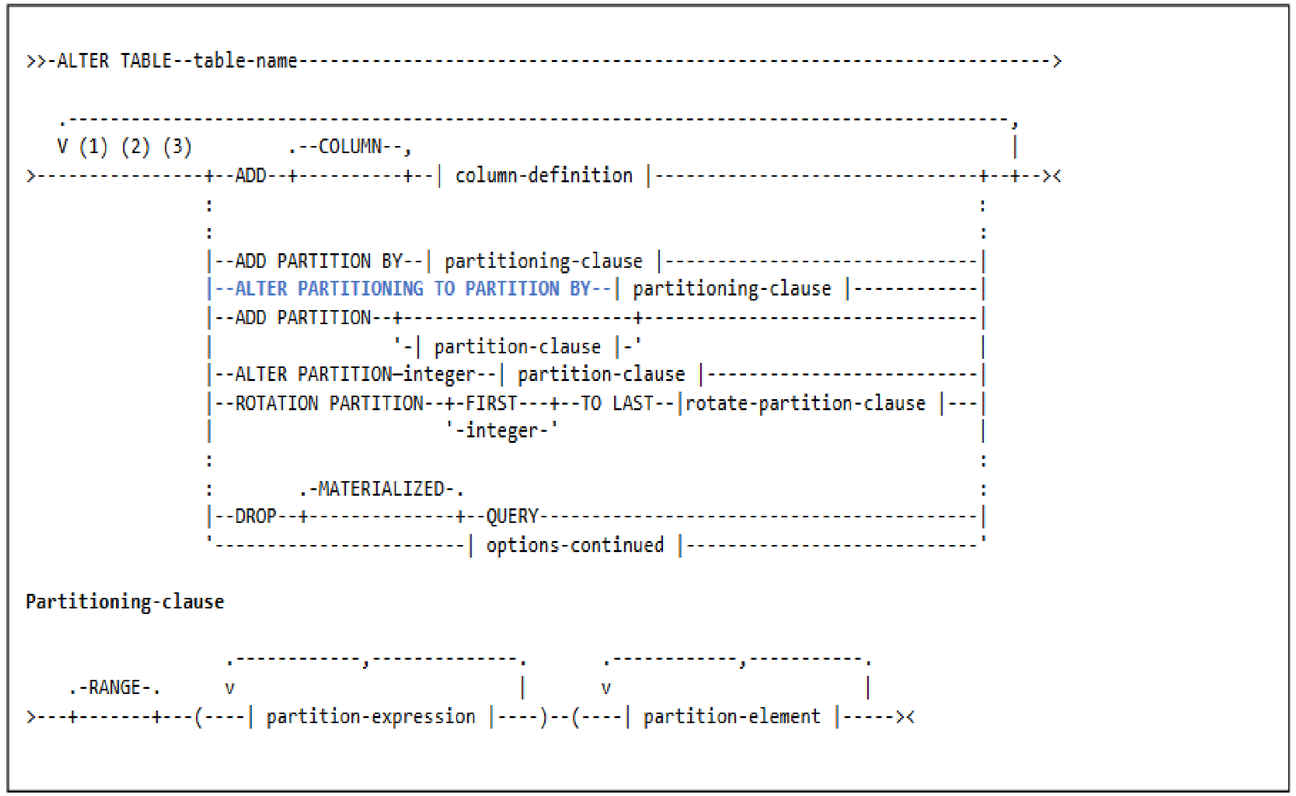
- Db2 13: introduced the capability to convert a table's partitioning scheme from partition-by-growth (PBG) table space to partition-by-range (PBR) table space online
- The partitioning scheme is altered directly to partition-by-range with relative page numbering
- The new PBR table space does not need to have the same number of partitions as the prior PBG table space. It can have more, the same or fewer partitions
- The existing indexes on the table are handled as a part of the conversion process
  - Db2 does not change any aspects or attributes of those indexes
- Users may create partitioned indexes on the table as desired after the conversion has been completed

Online conversion from PBG to PBR (3|5)

FL 500

– The ALTER TABLE statement has been enhanced with a new ALTER PARTITIONING clause

```
ALTER TABLE E8054.TB01
ALTER PARTITIONING TO PARTITION BY
  RANGE (COLINT, COLCHAR)
(PARTITION 1 ENDING AT ( 5, 'CCC'),
 PARTITION 2 ENDING AT (10, 'MMM'),
 PARTITION 3 ENDING AT (MAXVALUE,
  MAXVALUE))
```



Online conversion from PBG to PBR (4|5)

FL 500

- Determine a suitable partitioning scheme to use for the table, including the columns that will define the partitions and the limit key values for each partition, and evaluate the following considerations:
  - The number of partitions that will be created
    - Note that if the table to be converted is defined with DATA CAPTURE CHANGES, then the number of partitions in the new PBR table space cannot be less than the number of partitions in the old PBG table space
  - The data set size for each partition
    - Initial DSSIZE for the new PBR table space is inherited from the old PBG table space
    - Either ensure that each partition of new PBR table space can fit within that DSSIZE, or if necessary, alter the PBG table space to have a larger DSSIZE (if fix for APAR PH51359 is applied, DSSIZE change and ALTER PARTITIONING change can be put into effect via one online REORG; otherwise, those two changes will require two REORGs of the table space)
- You can use the RUNSTATS utility to collect useful statistics for planning the range-partitioning scheme

Online conversion from PBG to PBR (5|5)

FL 500

- REORG TABLESPACE SHRLEVEL REFERENCE or CHANGE must be run to materialize the ALTER TABLE ALTER PARTITIONING TO PARTITION BY RANGE pending definition change
- The entire table space needs to be reorganized to convert it from PBG to PBR after the pending definition change has been issued
- The high limit key for the last partition requires MAXVALUE for ascending key columns or MINVALUE for descending key columns
- Consider creating partitioned indexes on the table to support parallel processing advantages
- The materializing REORG invalidates dependent packages
- Table space cannot be recovered to a point in time prior to the materializing REORG
  - You can still run the UNLOAD utility on the old image copies of the table space or partitions created prior to REORG materialization for data mining or recovery purposes, and the LOAD utility can be used to reload the data into a different tablespace

Autobind phase-in

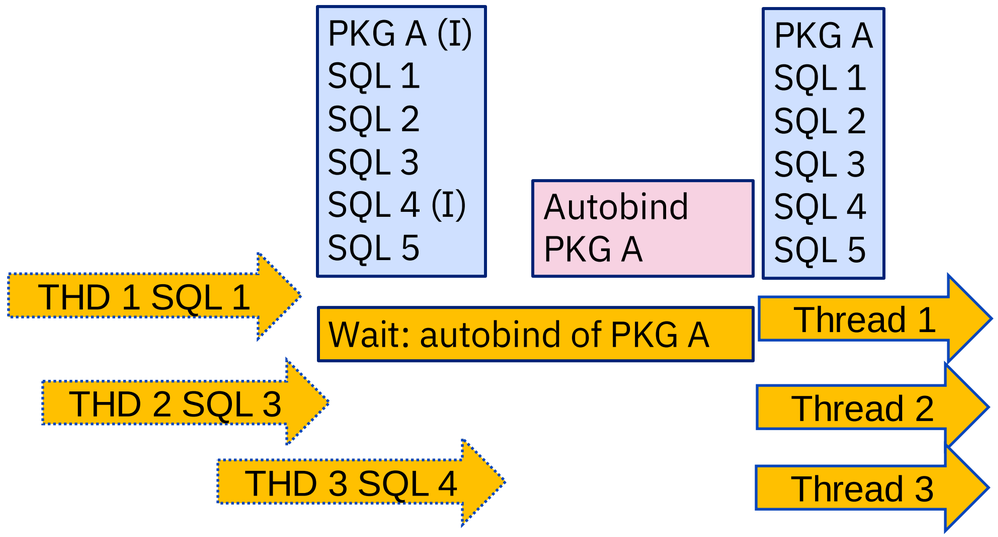
Package invalidation – reducing impact of autobind (1|3)

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

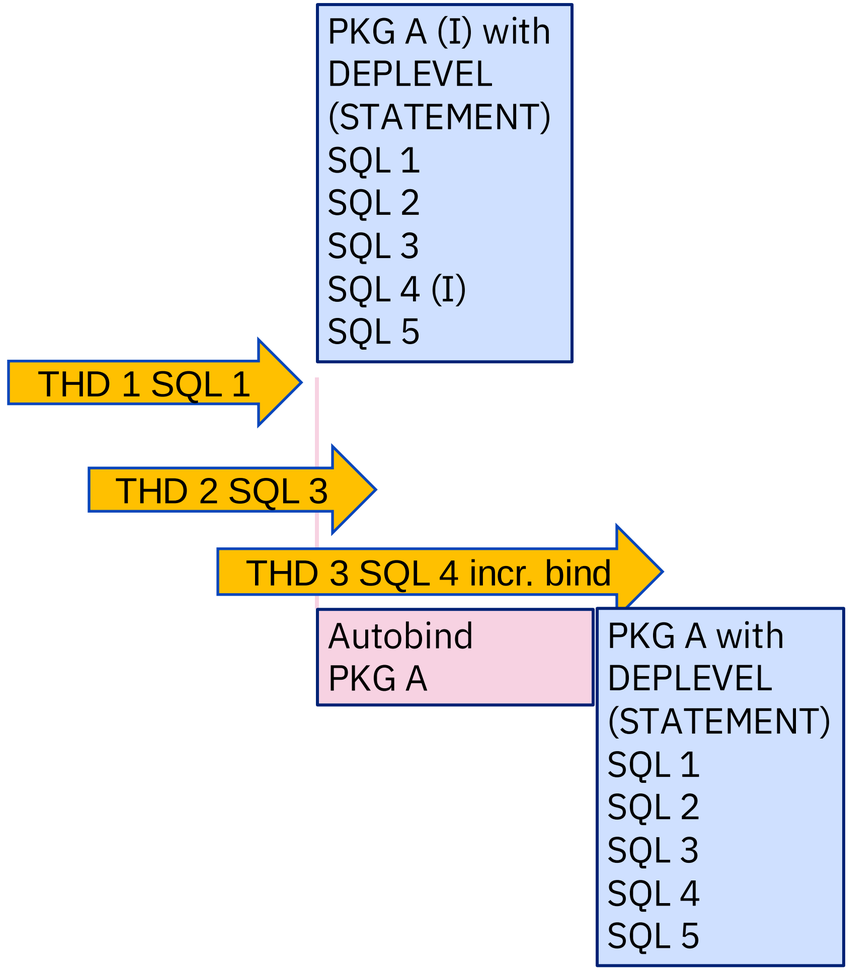


Package invalidation – reducing impact of autobind (2|3)

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



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

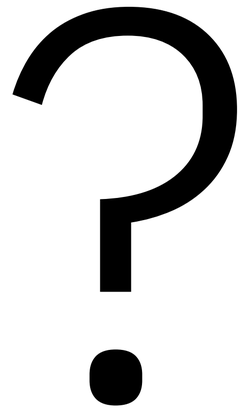
FL 502

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)



Summary

Db2 13 for z/OS provides many features for availability, resiliency, scalability, performance, security, operations, and application enablement

Top 5 high value features covered today:

- SQL Data Insight: new value from existing data
- Utility history: improved operations
- CURRENT LOCK TIMEOUT special register and DEADLOCK_RESOLUTION_PRIORITY global variable: flexibility, multi-tenancy
- UTS PBG to PBR via online REORG: scalability, performance
- Autobind phase-in: performance, operations

Db2 13 for z/OS delivers more features in each function level, and independent of function levels

- Explore what Db2 13 has to offer
- Take advantage of rich capabilities
- Share your experience with Tridex community!

Thank you