

IDUG

**Let Me Make This Clear (Vol. 4):
Explaining Often-Misunderstood
Db2 for z/OS Concepts and Facilities**

Robert Catterall

IBM

X f in

@IDUGdb2
#IDUG_NA24

Session Code: PERF7 | Platform: Db2 for z/OS

Abstract:

Some aspects of Db2 for z/OS technology are plain as day; others, less so. Questions I get from Db2 users worldwide have highlighted for me things that a lot of folks – even experienced Db2 people – tend to misunderstand. From AI in Db2 to dealing with old Db2 client code, from data set encryption to RACF management of Db2-internal security, in this fourth iteration of my "Let me make this clear" presentation I'll try to take folks from "Uh" to "Oh!".

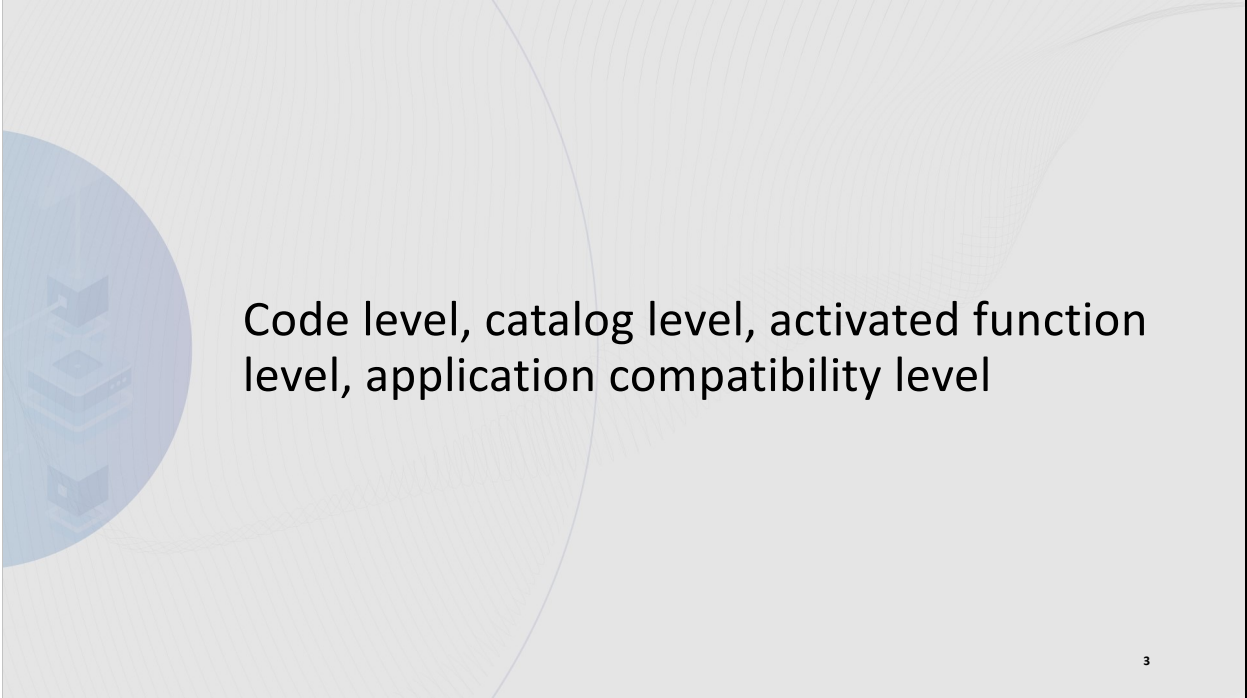
Robert Catterall
IBM
rfcatter@us.ibm.com

Agenda

- Code level, catalog level, activated function level, application compatibility level
- SQL Data Insights: AI in Db2
- “Old” Db2 client code in your environment
- The Db2 REST interface
- Db2 data set encryption
- RACF management of Db2-internal security

2

Here are the Db2 for z/OS-related items that I’m looking to clarify via this presentation.



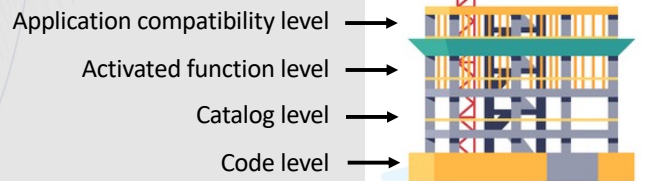
Code level, catalog level, activated function level, application compatibility level

3

These levels are all related to the continuous delivery mechanism for getting new Db2 for z/OS functionality out to users between new-version introductions. I'll try to clarify the meaning of each level and the relationships between levels.

The various “levels” of a Db2 for z/OS environment

- Consider floors of a building: can’t build the second floor without the first floor, can’t build the third floor without the second, etc.
- So it is with the various Db2 levels (introduced with Db2 12 and continuous delivery): each level **depends on the levels below it**



4

In explaining these Db2 for z/OS levels to people, I find it useful to consider the analogy of a building. In a Db2 for z/OS sense, what you’re building is a particular functional Db2 environment. You can’t add a given floor if the floors below aren’t there.

The ground floor: code level

- Refers largely to [currency of code](#) in a Db2 subsystem's load library
 - How so: each Db2 function level is associated with an APAR – when fix for APAR is applied to Db2 code (often via periodic upgrade of Db2 subsystem's maintenance currency to a new RSU level), that [code goes to a new level](#)
 - Example: on Friday, Db2 subsystem's code level is [131503](#) (V13, R1, function level 503)
 - On Saturday night, maintenance upgrade causes fix for APAR PH54919 (associated with Db2 13 function level 504) to be applied to Db2 code – [new code level is 131504](#)
 - Does that change anything in a functional sense? No, because the new functionality present with the 131504 code level [has not yet been activated](#)
- Of course the code level is the ground level:
 - FL504 introduced new AI_COMMONALITY built-in function – for that new function to be available, must be present in Db2 subsystem's [code](#)
- Check Db2 subsystem's code level with -DISPLAY GROUP command

5

Db2 for z/OS doesn't do what it does by way of magic – it does what it does through its code, which is the code in the Db2 load library (the code that is loaded into memory when a Db2 subsystem is started). If you'd like to use a Db2 feature that was introduced with function level 504 of Db2 13, your Db2 code level is going to have to be at least 131504 (meaning, version 13, release 1, function level 504). No Db2 feature can be used if the Db2 code that makes the feature usable isn't present in the system – that's what makes the code level the first level of your Db2 “building.”

Note that you can always check the code level for a Db2 subsystem by checking the output of the Db2 command -DISPLAY GROUP (in spite of its name, this command is as relevant for a standalone Db2 subsystem as it is for a Db2 data sharing group).

Second floor: the catalog level



- Sometimes, new capabilities provided by a Db2 function level have **catalog dependencies** (i.e., catalog changes are required to support the new capabilities)
 - Example: Db2 13 function level 501 introduced a **utility history** capability
 - Utility history information has to be recorded somewhere, so capability requires the SYSIBM.SYSUTILITIES table – added when **catalog level goes to V13R1M501**
 - When code level is 131501 and catalog level is V13R1M501, can function level 501 features be used? NO, because **function level has not yet been activated**

6

The Db2 catalog is a set of tables in which Db2 stores information about the environment in which Db2 is operating: information about tables and indexes, about IDs and associated privileges and authorities, about packages, etc. Sometimes, a new Db2 feature has a catalog dependency, meaning 1) that information pertaining to the feature has to be stored somewhere in the Db2 catalog, and 2) that the catalog itself has to be at a certain level for there to be a place in the catalog for the storage of the information (the new catalog level might introduce a new table, and/or a new column for an existing table, and/or a new data type for an existing column).

Example: the utility history feature of Db2 13 function level 501 will cause Db2 to write a record with useful information about utility execution (the particular utility executed, the ID that invoked the utility, the utility start time, elapsed and CPU times, etc.) every time a Db2 utility is executed. That information has to go somewhere, and that “somewhere” is the SYSUTILITIES table that is added to the catalog when the catalog level goes to V13R1M501; thus, even if the Db2 code level is 131501, the utility history feature can’t be used if the catalog level is not at least V13R1M501.

More on the catalog level

- Not every new function level has new catalog dependencies
 - When this is true, function level requires that catalog be at level associated with *most recent previous function level that did have catalog dependencies*
 - Example: Db2 13 FL503 requires V13R1M501 catalog level (FL503 has no catalog dependencies, and same is true for FL502)
- Mechanism for updating catalog level: CATMAINT utility
 - Example: `CATMAINT UPDATE LEVEL (V13R1M501)`

7

If the new features introduced by a Db2 function level don't have new catalog dependencies, we say that the function level does not have a catalog dependency. That means that there is not a new catalog level associated with the function level, and THAT means that the catalog doesn't have to go to a new level to support use of the new Db2 features introduced with the function level in question, as long as the catalog is at the level of the most recent previous function level that DID have a catalog dependency.

Example: the features introduced by Db2 13 function level V13R1M503 do not have any catalog dependencies, nor do the features introduced by function level V13R1M502. Does this mean that the features of Db2 13 function level 503 can be used with any catalog level? No. In fact, use of the Db2 13 FL503 features requires that the catalog level be V13R1M501. Why? Because activation of Db2 13 FL503 implies activation of all prior Db2 13 function levels, and one of those prior function levels is V13R1M501, and FL501 has catalog dependencies that make catalog level V13R1M501 necessary.

Third floor: activated function level

3

- For Db2 continuous delivery to work, “turning on” of functionality introduced with new function level had to be made **asynchronous** with the adding of function-enabling code to Db2 load library
 - Why? Because if that were not true, sysprogs would be (understandably) hesitant to upgrade maintenance level of a Db2 subsystem
- The means by which “add new code” and “turn on new code” were made asynchronous events: **-ACTIVATE FUNCTION LEVEL** command
 - Example: **-ACTIVATE FUNCTION LEVEL (V13R1M504)**
- To activate function level X, code level must be at least X and catalog level must be at least X (or at most recent previous level that had catalog dependencies, if X has no catalog dependencies)

8

Let's say that you have a Db2 13 subsystem with a code level of 131504 and a catalog level of V13R1M504 (there is a V13R1M504 catalog level because features introduced with Db2 13 function level 504 have catalog dependencies). Does that mean you can use Db2 13 FL504 features in that environment? No, unless function level V13R1M504 has been activated for the Db2 system. Why this activation business? Why can't Db2 13 FL504 features be used if they are there in the Db2 code and the catalog level is what it needs to be? Here's why: if new Db2 features became instantly available for use as soon as the Db2 code got to a certain level (assuming the catalog were at the at the required level), Db2 systems programmers would – understandably – be perhaps leery of updating the maintenance level of a Db2 subsystem, out of concern that a new Db2 code level (which would happen through application of the PTF for the APAR associated with said Db2 code level) would introduce features and functions the Db2 team had not yet examined and were not yet ready to support. The fact that activation of a new Db2 function level is a separate event versus the Db2 code going to the associated new level means that Db2 code currency can be taken forward (a good practice) without causing new features and functions to suddenly be available for use in the Db2 environment.

Top floor: application compatibility level



- Suppose Db2 subsystem's code level is 131504, catalog level is V13R1M504, and activated function level is V13R1M504
 - Can a program in this environment use the AI_COMMONALITY function, which was introduced with Db2 13 function level 504?
 - NO – unless the program's package has APPLCOMPAT(V13R1M504)

9

Even when the code level, catalog level (as applicable) and activated function level stars have all aligned, new features associated with a Db2 function level cannot be used unless relevant Db2 packages have been bound or rebound with the application compatibility level that matches the function level of interest – and remember: whenever a SQL statement is executed in a Db2 for z/OS system, whether static or dynamic in nature, there is always a package involved. What that means: you can't get away from this package APPLCOMPAT requirement.

A little more about the top floor

- APPLCOMPAT: application compatibility level for the program executing the package
 - A few things to note:
 - APPLCOMPAT(X) means that the program can use [SQL syntax, functionality](#) up through function level X
 - For [DRDA requester applications](#), relevant APPLCOMPAT will typically be that of the IBM Data Server Driver packages (i.e., the packages whose default collection is [NULLID](#))
 - APPLCOMPAT is relevant to [DDL statements](#) (e.g., ALTER and CREATE) – pay attention to APPLCOMPAT value of packages related to [SPUFI](#), [DSNTEP2](#), etc.

10

Here's the point on this slide that I most want to emphasize: the APPLCOMPAT value of the IBM Data Server Driver / Db2 Connect packages is really important. Why? Because DRDA requester applications often issue Db2 for z/OS-targeted SQL statements via the JDBC or the ODBC driver that is provided by the IBM Data Server Driver / Db2 Connect, and in such cases the package associated with execution of the SQL statement will be an IBM Data Server Driver / Db2 Connect package. That means that for the DRDA requester application, its effective APPLCOMPAT value will be the APPLCOMPAT value of the IBM Data Server Driver / Db2 Connect package used when the client-issued SQL statement is processed. The default collection for the IBM Data Server Driver / Db2 Connect packages is NULLID. Suppose you have a Db2 13 subsystem with function level V13R1M505 activated. If the packages in the NULLID collection are bound with (for example) APPLCOMPAT(V11R1) then by default your DRDA requester applications will not be able to issue SQL statements with syntax or options introduced after Db2 11 for z/OS. Is that what you want? Probably not. Check the APPLCOMPAT value of your NULLID packages, and see if it's what you want it to be.



SQL Data Insights: AI in Db2

11

On now to SQL Data Insights, a new feature introduced with Db2 13. I find that some clarification can be helpful here because SQL Data Insights provides query capabilities unlike anything we've had before with Db2 for z/OS.

Some basic facts about SQL Data Insights (SQLDI)

- Introduced with Db2 13 for z/OS, available via function level **V13R1M500**
 - A new SQL Data Insights-related built in function was introduced with FL504
- With regard to installation, SQL Data Insights has its own FMID (HDBDD18), but the functionality is part of the **base Db2 13 code** – not a separate product
 - Has some software prerequisites, but all of those are no-charge (example: the IBM Z Deep Neural Network Library, aka ZDNN, provided via application of z/OS APARs)

12

Main point here: SQL Data Insights is part of Db2 13 for z/OS – it is not a separate product. I say this because when we (IBM folks) talk about SQL Data Insights we sometimes talk about “AI in Db2,” and that causes some people to get SQL Data Insights confused with IBM Db2 AI for z/OS (aka Db2ZAI). Db2ZAI works with Db2 for z/OS, but it is a separate software product – separately licensed. Your license for Db2 13 for z/OS entitles you to use SQL Data Insights.

How SQLDI functionality is enabled

- Using SQLDI GUI, Db2 DBA directs Db2 to build a “**model table**” (aka vector table) based on a given table in the database (for example, the CUSTOMERS table)
- Once the model table has been built by SQLDI, a program (or query tool) can use the **Db2 built-in AI functions** to query data in the CUSTOMERS table
 - Model table built by SQLDI from data in the CUSTOMERS table is used in execution of built-in AI functions, but is **never referenced in query**

13

A few things related to Db2 building a model table (also referred to as a vector table) for a given Db2 for z/OS base table:

- This process is initiated for a given table via the GUI that is part of SQL Data Insights functionality.
- A DBA, using the GUI, could tell Db2 to build a model table for an entire base table (i.e., for all columns of the table).
- A DBA could tell Db2 to build a model table using a subset of the columns of a base table (e.g., 40 of a table’s 50 columns), because the business might determine that certain columns of the base table are not important for comparative purposes.
- A DBA could tell Db2 to build a model table based on a virtual table that is actually a view defined on a join of three (for example) actual tables.
- A DBA could tell Db2 to build a model table based on a virtual table that is actually a VSAM file – this VSAM file could be made to appear as a Db2 for z/OS table by the IBM product called Data Virtualization Manager (aka DVM).
- A model table, built by Db2 for a given base table, is used in the execution of queries that include the Db2 built-in AI functions associated with SQL Data Insights, but that model table will never be referenced in such queries – it’s logically invisible in that sense.

How SQL Data Insights changes the game

- Provides user/program with to “similar to this” or “dissimilar to that” insights – *without user having to tell Db2 what is meant by “similar” or “dissimilar”*
 - Previously, you had to tell Db2 what you mean by “similar” or “dissimilar” via things such as LIKE and NOT LIKE predicates
 - With SQLDI, Db2 (using model table built from data in base table) detects *patterns of similarity* (or dissimilarity) in data – patterns a human might be challenged to discern



Here’s the main reason that SQL Data Insights is so significant an advance in Db2 for z/OS query functionality: in the past, if you wanted Db2 for z/OS to return rows similar to, or dissimilar to, a given row, *you had to tell Db2 what you mean by “similar to” or “dissimilar to,”* perhaps using LIKE or NOT LIKE predicates. That basically means that you had to tell Db2 what you wanted Db2 to tell you! With SQL Data Insights, you tell Db2 to return rows that are similar to or dissimilar to a given row (or set of rows), and Db2 figures that out through its ability to detect patterns of similarity or dissimilarity within and across the rows of a table – patterns that could be very hard for a human being to detect (especially if the table in question holds millions or hundreds of millions of rows or more). What’s more, Db2 actually calculates a “similarity score” for a row, so when you say you want the rows for the 20 entities (an entity being the primary subject of a table – a product, a customer, a supplier, an account, an order, whatever) that are most similar to entity X, in order of “similar-ness,” that’s what you’ll get (the closer to 1 the similarity score is for a row, the more similar it is to the basis-of-comparison row, and the closer to -1 the similarity score is, the more dissimilar the row is to the basis-of-comparison row).

A usage scenario

- Policy holder 12345 has been caught submitting **fraudulent insurance claims** – which of the **millions** of other policy holders might be doing the same?
 - With SQLDI, user could tell Db2, “Show me the 20 policy holders **most similar** to 12345,” or “Show me the 30 claims that are **most similar** to this set of 3 fraudulent claims that were submitted by 12345”
 - Let fraud analysis team do deep-dive analysis of that **small result set**, versus trying find a few needles in a **giant haystack**

15

What’s on the slide is an example of a “reduce the size of the haystack” benefit of SQL Data Insights functionality, when analysts would otherwise be looking for “a needle in a haystack.” Think about this scenario involving submission by a bad guy of fraudulent insurance claims (e.g., for damage to insured property, when that alleged damage in fact never occurred). Some bad guys are really good at covering their tracks when they do this kind of thing. The insurance company’s fraud analysis team might have finally nailed such a bad guy after much effort, and then might think, “Whew boy – we have millions of other policy holders. Which others of those folks might also be engaged in this type of fraudulent claim activity?” Well, with SQL Data Insights you could have Db2 provide for you the 25 (for example) rows in a POLICY HOLDER table that are most similar to policy holder 12345, and/or the 30 claims from a CLAIMS table that are most similar to a set of three fraudulent claims submitted by policy holder 12345. Then you can take that manageable set of rows to the fraud analysis team and say, “Hey, guys. The DBMS is telling us that these policy holders (or these claims) are most similar to policy holder 12345 (or to fraudulent claims submitted by that policy holder). You might want to do a deep dive analysis of these smaller result sets to see if they can help you uncover more instances of fraudulent activity.”

Another usage scenario

- You have identified three customers – X, Y and Z – that are among your very best
 - With SQLDI, you could then tell Db2, “Show me the 25 customers that are *least similar* to the set of X, Y and Z”
 - Let the customer relations team analyze that manageable set of customers
 - How is it that they are dissimilar to our best customers? Could we take actions that would help us to get more business from these customers?

16

Here’s a SQL Data Insights use case of the “dissimilar to” variety. Maybe your organization is a retailer, and maybe your customer relationship management team has identified 3 customers that are among your very best – maybe these 3 customers have bought a lot from the company each year for several years running, and maybe they’ve bought from several of your different lines of business, and maybe they always pay on time, and so on. With SQL Data Insights, you tell Db2 to do this: “Hey, Db2. Here are 3 of our best customers. Consider that set, and return to me the 30 customers that are least similar to this set of 3 great customers.” Then, hand that set of 30 rows over to the customer relationship management folks, and let them analyze the data. How are these 30 customers dissimilar from the 3 really great customers? Could your company take any actions (special offers – whatever) to change some of those differences in ways that would potentially make these folks better customers?

SQL Data Insights – the built-in Db2 functions

FL 500

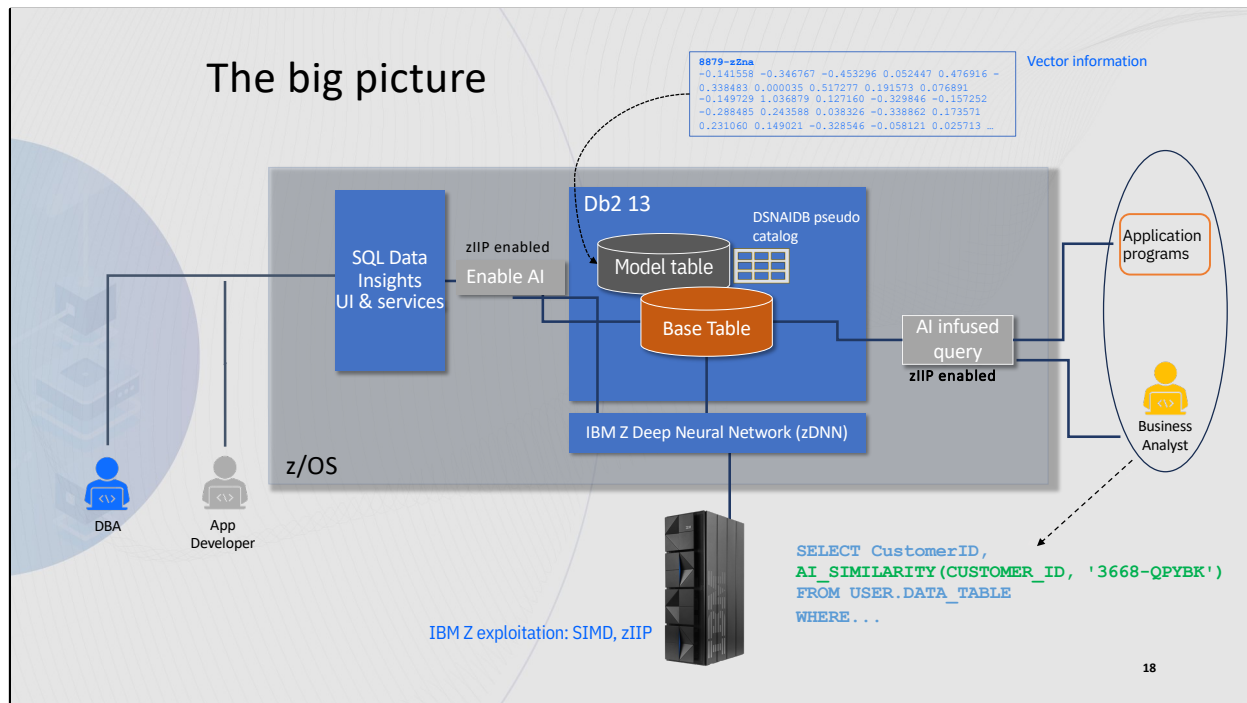
Function	Description
AI_SIMILARITY	Returns the entities that are most similar to (or dissimilar to) a particular entity
AI_SEMANTIC_CLUSTER	Returns the entities that are most similar to (or dissimilar to) a given set of up to three entities
AI_ANALOGY	Consider the relationship between value X in COL1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

FL 504

17


Here are the four built-in SQL Data Insights functions (the first three are available when function level V13R1M500 is activated, and the fourth came out with function level 504, in the fall of 2023). The table provides a brief explanation of the capability of each function. Using these functions in a query is really easy: it's just standard SQL syntax for a built-in Db2 function (name the function, open parenthesis, input to the function, close parenthesis) – and, as previously mentioned, the vector table that Db2 uses in executing these functions for a given base table is never referenced in a query (the query just references the base table of interest).

Note that **AI_ANALOGY** is a “fill in the blank” kind of function (consider X in column COL1 and Y in COL2: what COL2 values would make the relationship with Z in COL1 most analogous to the “X in COL1 / Y in COL2” relationship?), while **AI_COMMONALITY** returns the values furthest from, or closest to, the “center value” of a given table column.



A few comments on this slide:

- In the box in the upper right corner you have an example of what a vector table row might look like: a distinct value of a column of the base table (in this case, 8879-zZna is a customer ID), followed by 1280 bytes of vector information (numerical values, calculated by SQL Data Insights using a neural network model, that are used for comparative purposes). Note that there will be a row in the vector table for every distinct value of every base table column included for consideration in building the vector table.
- Note the example query on the lower right side of the slide. As I have previously pointed out, the syntax is standard for a query that references a built-in function, and the vector table associated with the base table is not referenced in the query.
- The main Z hardware features exploited by Db2 SQL Data Insights are SIMD (single instruction, multiple data), which allows a single operation to be performed simultaneously for several data elements; and zIIP processors (building of a model table is zIIP-offload-able, and so is execution of one of the built-in SQL DI functions).



“Old” Db2 client code in your environment

19

Some DBAs think that the presence of old Db2 client code in their environment is more of a problem than it actually is (“client code” in this case refers to the software, such as the IBM Data Server Driver and Db2 Connect, that enables DRDA requester applications to send SQL statements over a TCP/IP network connection to a Db2 for z/OS server).

“Old” Db2 client code: what and why

- What I mean: Db2 client code (IBM Data Server Driver / Db2 Connect) that is older than the 11.1 release
- Why am I talking about “pre-11.1” when anything “pre-11.5” is out of support?
 - A. Because there is a lot of pre-11.1 client code out there
 - B. Because, if the IBM Data Server Driver packages (default collection: NULLID) have an APPLCOMPAT value greater than V12R1M500, a connection request from a pre-11.1 Db2 client [will fail](#)

20

While any Db2 client code prior to the 11.5 release is out of support, what I’m referring to as “old” is Db2 client code older than the 11.1 release. Why is that? Here’s the deal: if you take the APPLCOMPAT value of the packages associated with Db2 client code (on the Db2 for z/OS side, those packages by default are in the collection called NULLID) higher than V12R1M500, an attempt to connect to Db2 for z/OS from a pre-11.1 Db2 client will fail.

A fairly widely-held misconception

- “We can’t go to Db2 13, because we have old Db2 client code in our environment”
 - Me: “How is that old Db2 client code holding you back?”
- “With that old Db2 client code, we can’t take APPLCOMPAT for the NULLID packages above V12R1M500”
 - Me: “Why is that a problem?”
- “If we can’t go above APPLCOMPAT(V12R1M500) for the NULLID packages, we can’t go to Db2 13”
 - Me: “NOT TRUE – in a Db2 13 system, a package (including a package in NULLID) can have an APPLCOMPAT value as low as V10R1

21

What’s on this slide basically reflects an actual conversation (via email) I had not too long ago with a Db2 for z/OS systems programmer. This individual first of all was under the impression that the old Db2 clients in his environment would keep him from being able to migrate his organization’s Db2 12 subsystems to Db2 13 – this because he couldn’t take APPLCOMPAT for the NULLID packages above V12R1M500 without causing connection failures for applications using pre-11.1 Db2 client code.

I explained that you could successfully migrate from Db2 12 to Db2 13 with packages (whether in NULLID or some other collection) that have APPLCOMPAT values as low as V10R1; so, the old Db2 client code in his environment would NOT block him from migrating Db2 12 systems to Db2 13. With that point made, I acknowledged that pre-11.1 Db2 client code could indeed be problematic...

Old Db2 client code won't block move to V13, but...

- You don't want developers of DRDA requester applications to be **blocked from using SQL syntax and functionality** introduced after Db2 12 FL 500, do you?

What you can do about it...

22

While old Db2 client code won't block a migration from Db2 12 to Db2 13, it could be a headache for developers of DRDA requester applications. Why? Well, what if you keep APPLCOMPAT for the NULLID packages at or below V12R1M500, to keep connection requests issued through pre-11.1 Db2 clients from failing? That would mean that developers of DRDA requester applications could not use SQL syntax and functionality introduced after Db2 12 function level 500 with their programs (that would mean, for example, no use of the built-in functions associated with SQL Data Insights, and no use of SET CURRENT LOCK TIMEOUT to specify an application-specific lock timeout value – these SQL enhancements were introduced with Db2 13 function level 500).

Is there a way to accommodate pre-11.1 Db2 client code and enable developers of DRDA requester applications to take advantage of the latest SQL syntax and functionality available in your Db2 for z/OS environment? Yes...

A way to deal with old Db2 client code

- **BIND COPY** the NULLID packages to alternate collection (e.g., COLL_X) with **APPLCOMPAT(V12R1M500)**
- Using **Db2 profile tables**, create one or more profiles based on **product ID (PRDID)** of pre-11.1 Db2 clients in your environment
 - Output of **-DISPLAY LOCATION** command will show product IDs of Db2 clients in your environment
- For each such profile, have attribute that directs Db2 to **automatically** issue **SET CURRENT PACKAGE PATH = COLL_X** when pre-11.1 Db2 clients connect to system
 - After doing that, take APPLCOMPAT for NULLID packages **as high as you want** – old Db2 clients are automatically using IBM Data Server Driver packages in COLL_X

23

The way to accommodate pre-11.1 Db2 client code and enable developers of DRDA requester applications to leverage newer Db2 for z/OS SQL syntax and functionality is to have more than one collection for the Db2 client packages (this is increasingly common, with the difference between the packages in NULLID and the packages in one or more alternate collections being their bind specifications – APPLCOMPAT is of course a package bind specification); so, you can BIND COPY the packages from NULLID into an alternate collection with an APPLCOMPAT specification of V12R1M500 (or at least not higher than that), and if you point pre-11.1 Db2 client-using applications at that alternate collection then they'll be able to connect successfully to the Db2 for z/OS system, and then you can take APPLCOMPAT for the NULLID packages (the packages that will be used by applications that utilize Db2 client software at the 11.1 level or higher) as high as you want.

How do you point client applications utilizing pre-11.1 Db2 client code to a package collection other than NULLID? The easiest way to do that is via the Db2 profile tables – specifically, with one or more profiles based on the “product ID” (abbreviated as PRDID) values associated with pre-11.1 Db2 clients (-DISPLAY LOCATION command output provides you with these PRDID values).

Example related to preceding slide

- Suppose -DISPLAY LOCATION shows 9.7 ODBC driver in your environment

```
LOCATION          PRDID
::FFFF:1.2.3.4  SQL090703
```

Note: for JDBC driver, PRDID will be something like JCC03580 – that's a 3.58 driver version, which relates to Db2 9.7 client, as shown on [this page](#) in online documentation

- In SYSIBM.DSN_PROFILE_TABLE

PROFILEID	PRDID	PROFILE_ENABLED
99	SQL090703	Y

Note: wildcard can be used with PRDID value, so SQL09* will cover 9.7, 9.5 and 9.1 ODBC drivers

- In SYSIBM.DSN_PROFILE_ATTRIBUTES

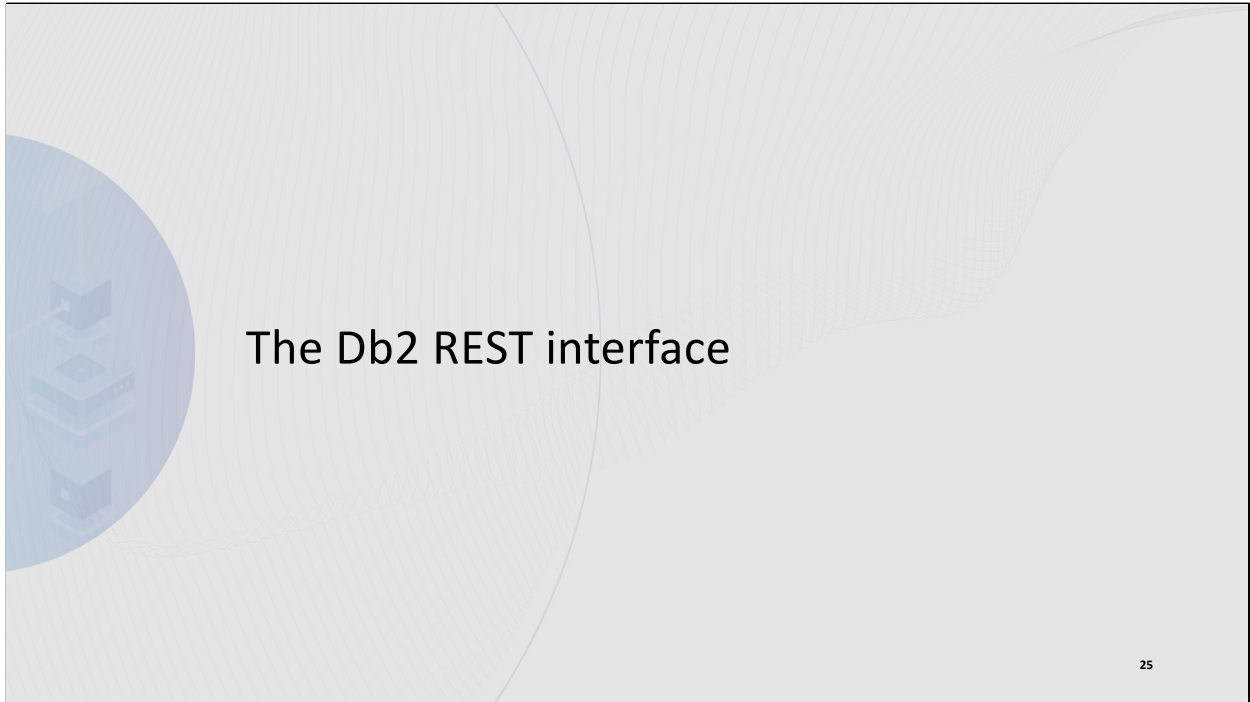
PROFILEID	KEYWORDS	ATTRIBUTE1
99	SPECIAL_REGISTER	SET CURRENT PACKAGE PATH = COLL_X

24

This slide shows how a PRDID associated with a pre-11.1 Db2 client (obtained via the command -DISPLAY LOCATION) can be used as an identifier for a profile, the attribute of which is a SET CURRENT PACKAGE PATH statement that will be automatically executed when a connection request comes from an application using the old Db2 client associated with the PRDID. That SET CURRENT PACKAGE PATH will make the alternate collection of Db2 client packages (bound with an APPLCOMPAT value not higher than V12R1M500) the default for the DRDA applications that use the old Db2 client associated with the PRDID value.

As noted on the slide, identifying the release level of the Db2 client associated with an IBM ODBC driver (this will have SQL as the first three characters of the PRDID value) is pretty easy – it's right there in the PRDID value. To determine the Db2 client release level associated with a given IBM JDBC driver (first three PRDID characters: JCC) requires looking up that information on the documentation for which a hyperlink is provided on the slide.

Also keep in mind that – as noted on the slide – the use of a wildcard in the PRDID value specified for a profile can enable one profile to be associated with several releases of a given Db2 client type (e.g., the ODBC driver).



To me, it's not so much that the REST interface to Db2 for z/OS is not understood by Db2 people in a technical sense – rather, I get the impression that a lot of Db2 people don't fully appreciate the importance of leveraging the REST interface to Db2 for z/OS.

Underappreciated by some Db2 for z/OS people

- Some folks have not seen how important the Db2 REST interface can be...
 - REST architectural style can be ideal for [seamlessly integrating z/OS-based data services with cloud-based applications](#)
 - For client-side developers, particulars of data server completely abstracted – same is true of server-side developers
 - That abstraction is good for development and deployment [productivity](#) and [agility](#)
 - Major [broadening of programming languages](#) that can access Db2 for z/OS data: if a program written in some language can issue a REST request, it can access Db2

26

“Hybrid cloud” is way beyond the concept stage – it’s being advantageously implemented by more and more organizations, across industries. The idea is to dispense with an “either/or” mindset with regard to on-premise versus cloud-based systems, and to instead deploy an IT infrastructure that enables seamless integration of application- and data-serving systems across on-prem and cloud-based systems. This approach delivers the development/deployment agility and flexibility associated with “cloud-native” applications, while enabling organizations to maximize the value of both on-prem and cloud-based systems.

From a Db2 for z/OS perspective, leveraging Db2’s REST interface can be one of the most effective steps you can take to optimize integration of Db2 data across a hybrid cloud IT infrastructure. A key reason for this: the REST architectural style completely abstracts the technical particulars of a service-providing system from a client-side developer’s perspective, and similarly abstracts client-side platform particulars from the perspective of a server-side developer. This can majorly boost application development productivity and deployment agility. The REST interface also opens up Db2 data access to a greater variety of programming languages: if a program written in language XYZ can issue a REST request, that program can access Db2 for z/OS data.

More Db2 REST interface goodness

- **No need for Db2 client code** (e.g., IBM Data Server Driver) on client application side, because client application issues REST requests, not SQL (SQL is server-side)
- **Secure**: SQL is static (application auth ID only needs EXECUTE privilege on package associated with Db2 REST service)
 - Also, REST interface can make it **easier to implement SSL encryption** for application
- **Cost effective**: because REST interface is an extension of Db2 DDF functionality, execution of SQL invoked via REST request is **up to 60% zIIP-offload-able**

27

I previously talked about the challenges that can be presented by older releases of Db2 client code in a Db2 for z/OS client-server environment. When the REST interface to Db2 for z/OS is utilized, those challenges do not exist because Db2 client code is not needed (it's not needed because the client-side programs are not issuing SQL statements – they're issuing REST requests).

Utilizing the REST interface to Db2 for z/OS involves invocation of server-side, static SQL statements via client-issued REST requests. Static SQL provides a security advantage: related applications require only the execute privilege on associated Db2 packages, versus the table privileges (SELECT, INSERT, UPDATE, DELETE) that are required for successful execution of dynamic SQL statements.

Do you want to use SSL (i.e., AT/TLS) encryption for a Db2 for z/OS client-server application? That can be easier when the REST interface versus the SQL interface (i.e., the DRDA interface) to Db2 for z/OS is used, in part because of the challenge of getting the connection string right when the SQL interface is used.

Because the Db2 for z/OS REST interface is an extension of DDF functionality, you get the zIIP-offload advantage that comes with access to Db2 via DDF.

Some people are skeptical of Db2 REST scalability

- In fact, Db2 REST services are **highly scalable**, capable of supporting 1000s of trans per second for one subsystem (given adequate processing capacity)
 - One organization tested average per-tran CPU cost using REST interface vs. DRDA requester interface, and found that in REST case CPU cost was < 1% greater

28

Do you sacrifice CPU efficiency, throughput and scalability when you use the REST architectural style for a Db2 for z/OS client-server application? No. As shown on the slide, one organization found that the difference in z/OS-side CPU consumption for the same client-server transaction implemented using the REST and SQL interfaces to Db2 was less than 1% (in different scenarios, the slight edge in CPU efficiency could go to the REST request-issuing or the SQL-issuing application). I've seen the Db2 monitor data for a REST-using Db2 client-server transaction that is executed at one site 14.5 million times a day, with an average elapsed time of 453 microseconds. Another organization implemented a new Db2 for z/OS client-server application using Db2's REST interface, and saw transaction volumes double within a year, with zero scalability issues.

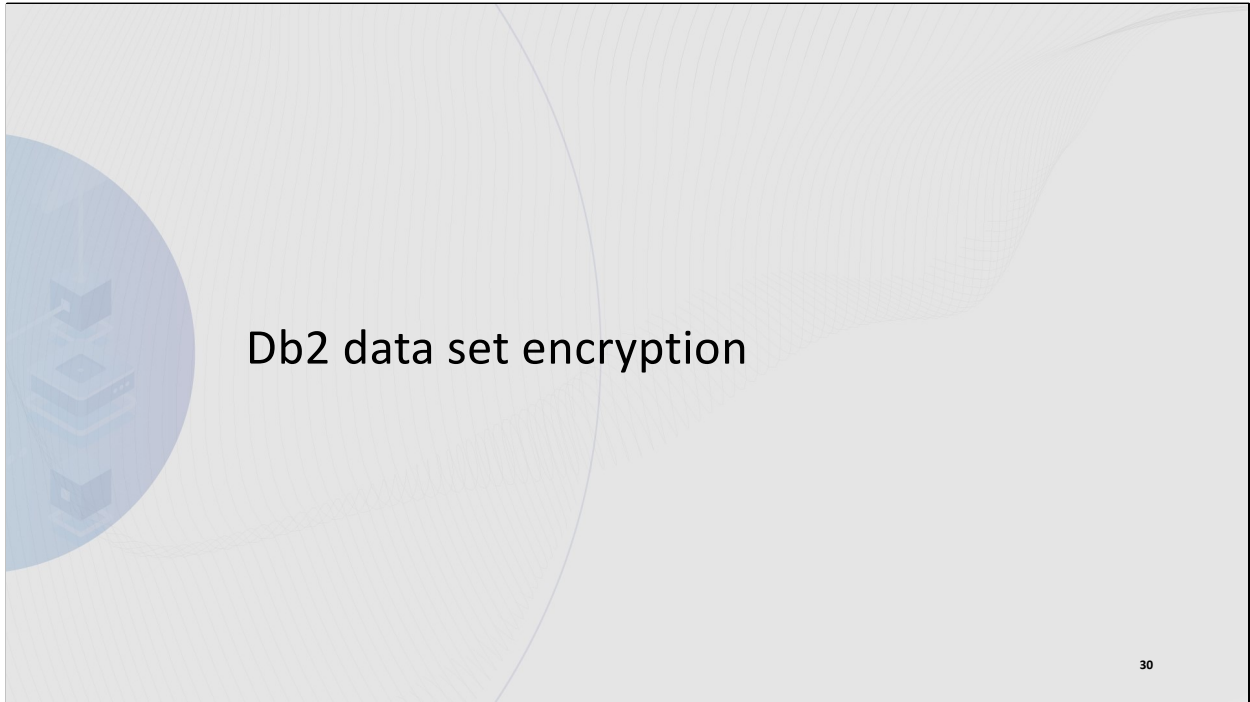
Obviously, the more involved a transaction is, the higher server-side CPU and elapsed times are expected to be. My point is this: whether said transaction involves client-issued REST requests versus client-issued SQL statements should not significantly impact efficiency, throughput and scalability. You can driver very high transaction volumes, very efficiently, with the Db2 for z/OS REST interface.

But what about stateless nature of REST?

- Won't each REST transaction involve creating and then terminating a connection to the Db2 system?"
 - Nope – The initial REST request from a given app server will create a connection to Db2, but that connection **will be retained** (in an inactive state) **for up to 15 seconds**
 - Result: if another REST request comes in from that app server within 15 seconds (likely for a higher-volume application), **Db2 will re-use existing connection**
 - Also: REST requests will **re-use Db2 DBATs** (DDF threads) in the DBAT pool
 - Also: when ID and password (or certificate) for a REST request are authenticated by RACF (or equivalent), Db2 **caches that in memory** for another 3 minutes

29

One reason some people doubt the scalability of the REST interface to Db2 for z/OS: they are aware that the REST architectural style is stateless in nature, and that a client application simply issues a REST request, as opposed to explicitly connecting to a service-providing system and then issuing a REST request targeting that system. This causes some folks to conclude that a connection to a Db2 for z/OS system must be established and then terminated for each Db2-targeting REST request. In fact, that is not the case. As shown on the slide, the first Db2-targeted REST request from a client application (coming from a particular application server) will indeed involve creation of a connection to the Db2 for z/OS system, but when that transaction completes *the connection will not be immediately terminated*; instead, the connection will be preserved for up to 15 seconds. What that means: if another request for the application comes from the same app server within 15 seconds of the first request (highly likely in a high-volume situation, which is our concern where scalability is concerned), the “already there” connection will be reused. As noted on the slide, REST scalability also benefits from the re-use of DDF threads in the DBAT pools (same as DRDA requester applications) and a Db2 13 enhancement that reduces the need for Db2 to check with RACF when several REST requests, each separated by less than 3 minutes, come from one application running on a given app server.



Now for a look at Db2 for z/OS data set encryption – specifically, by way of the data set encryption feature of z/OS.

What I'm talking about: z/OS data set encryption

- This is about encryption of data “at rest” (i.e., on disk)
- It's a feature of the operating system, [introduced with z/OS 2.3](#)
- [Application-transparent](#): data is automatically encrypted when written to disk, automatically decrypted when read into memory
- With IBM z14 and later generations of the mainframe, [very CPU-efficient](#) (with z14, encryption processing moved from card-based to [on-chip](#))
 - There are organizations that have encrypted 1000s of Db2 data sets using this feature – additional CPU cost of SQL execution can be 1% or less
 - [Bigger Db2 buffer pools](#) further reduce CPU cost of encryption – fewer read I/Os mean reduced data decryption activity

31

Encrypting Db2 for z/OS data “at rest” (i.e., on disk) by way of the data set encryption feature of z/OS is becoming a more and more widespread practice. Here are some reasons for that:

- *It's application-transparent.* A solution often becomes much more attractive if it can be implemented with no requirement to change application code.
- *It's a very CPU-efficient encryption mechanism.* This is particularly true when the mainframe server platform is a z14 or later, the reason being the huge improvement in encryption performance when that functionality went from card-based to on-the-chip (that happened with the z14). As noted on the slide, bigger Db2 buffer pools further enhance the CPU efficiency of at-rest encryption of Db2 data. How's that? Well, data encrypted on disk is automatically decrypted when read into memory. Bigger Db2 buffer pools mean fewer read I/Os, and that means less CPU spent on data decryption.
- *It got even easier to implement for Db2 data thanks to enhancements introduced with Db2 12 function level 502.* These enhancements will be explained momentarily.

So, what needs clarification?

- Some folks are not clear on the mechanics of how this feature works with Db2...
- With z/OS data set encryption, a **key label** (the external “handle” of an encryption key) is associated with a data set *at data set create time*

How that can be done...

32

The essential element of z/OS data set encryption is pretty easy to explain: the enabling mechanism is the assignment of an encryption key label to a data set *when the data set is created* (a label is a way to reference an encryption key without having to reference the key itself, which of course has to be very carefully secured). The next slide shows ways in which assigning a key label to a data set can be accomplished.

Options for associating a key label with a data set

- Via IDCAMS when creating the data set
- By way of a RACF data set profile
- With an SMS data class specification
- Via **KEY LABEL** clause in Db2 **CREATE** or **ALTER TABLE** or **STOGROUP**
 - Do-able in Db2 13 system, or in Db2 12 system with function level 502 or higher activated
 - Note: Db2 package through which CREATE or ALTER statement is executed **must have APPLCOMPAT value of V12R1M502 or higher**
 - **ENCRYPTION_KEYLABEL** in ZPARM provides key label for catalog and directory data sets, and for archive log data sets (when archiving to disk – can subsequently migrate to tape)

33

The first three options listed on this slide were the only options for assigning an encryption key label to a Db2 data set prior to Db2 12 function level 502.

Function level 502 of Db2 12 provided a new option for enabling z/OS data set encryption for Db2 data sets. It added the KEY LABEL option for the CREATE and ALTER TABLE and CREATE and ALTER STOGROUP statements. [Specifying KEY LABEL in a CREATE TABLE or ALTER TABLE statement is do-able for a table in a universal table space. KEY LABEL in a CREATE or ALTER STOGROUP statement applies to any object associated with the STOGROUP, though for a table space associated with STOGROUP XYZ, the KEY LABEL value of CREATE or ALTER TABLE will override the KEY LABEL value specified for STOGROUP XYZ.]

Function level 502 of Db2 12 also introduced the new ZPARM parameter **ENCRYPTION_KEYLABEL**. The value provided for this parameter is the key label that will be associated with Db2 catalog and directory data sets, and with archive log data sets *when those data sets are allocated on disk*. z/OS data set encryption cannot be used for a data set that is allocated on tape; however, if an archive log data set allocated on disk is encrypted, it will remain encrypted if it is subsequently migrated to tape (maybe via HSM).

How does existing Db2 data get encrypted?

- Generally speaking, via **online REORG** (could also be LOAD REPLACE or REBUILD INDEX or RECOVER)
- Think about it: what does Db2 do for an online REORG?
 - Answer: Db2 **creates shadow data sets** for table space and indexes – if those data sets are associated with a key label, data that goes into data sets will be encrypted
 - When key label is associated with a Db2 table space's data set(s), online REORG of table space will encrypt everything –
 - Data in table space data set(s)
 - Data in associated index data set(s)
 - Data in data sets of associated “auxiliary” table spaces (e.g., LOB or XML table spaces)

34

I mentioned earlier that z/OS data set encryption is enabled for a given data set when an encryption key label is assigned to the data set at data set creation time. How, then, is data in an existing Db2 data set encrypted? That is usually done via online REORG of a table space for which z/OS data set encryption is to be utilized (LOAD REPLACE, REBUILD INDEX or RECOVER would also do the job). The important thing to remember about online REORG in a data set encryption context is this: the utility starts out by creating shadow data sets for the target table space (and for associated indexes and for auxiliary objects such as LOB or XML table spaces). When those shadow data sets are created, the key label related to the table will be associated with the shadow data sets, and data written to those data sets will be encrypted as a result. At the end of the utility's execution, what had been the shadow data sets will become the new “original” data sets for the objects processed by the online REORG job, and voila: the data in the data sets is encrypted on disk.

A little more on Db2 data set encryption

- What if you want to encrypt [active log data sets](#)?
 - Probably created a long time ago, and there's no such thing as an "online REORG" for those objects – key label is assigned at data set create time, so what do you do?
 - Easiest approach: [leverage Db2 13 online removal of active log data sets](#)
 - Suppose you have 20 pairs of active log data sets
 - Create 20 new pairs of encrypted active log data sets, dynamically add them to the log inventory via NEWLOG option of -SET LOG command (do-able since Db2 10)
 - After older unencrypted active log data sets have been archived, dynamically remove them via [REMOVELOG option of -SET LOG command \(Db2 13, FL500\)](#)
 - If not at Db2 13 FL500, remove unencrypted log data sets with DSNJU003 utility (Db2 subsystem [has to be down](#) when executing that utility)

35

I mentioned earlier that the ENCRYPTION_KEYLABEL parameter in ZPARM introduced with Db2 12 function level 502 can be used to provide an encryption key label that will be associated with archive log data sets when those data sets are allocated on disk. What about encrypting active log data sets? As noted, z/OS data set encryption works by way of an encryption key label that is associated with a data set when the data set is created. More than likely, the active log data sets of a Db2 subsystem were created some time ago. That means that encrypting a subsystem's active log data sets will require replacing the current unencrypted active log data sets with new encrypted active log data sets (data sets for which an encryption key label was provided at data set create time). Prior to Db2 13 function level 500, this active log data set encryption procedure would have required execution of the Db2 change log inventory utility (DSNJU003), which cannot be executed unless the Db2 subsystem is down. With Db2 13 function level 500, old unencrypted active log data sets can be replaced with new encrypted active log data sets while Db2 remains up and running, thanks to the new REMOVELOG option of the Db2 command -SET LOG (a nice complement to the NEWLOG option of -SET LOG that was introduced back with Db2 10 for z/OS).

And one more thing...

- Some people ask: can you **compress and encrypt** data in a Db2 table space?
 - Answer: **absolutely** – data is compressed in memory, then that compressed data gets encrypted when written to disk (reverse happens when data read into memory)

36

I have several times been asked the question seen on this slide. Db2 table space compression can definitely be used for a table space that is also encrypted via z/OS data set encryption. The key thing to remember here is this: data in pages of a COMPRESS YES table space is in compressed form in memory (data is decompressed when a row on the page is retrieved by a program, or when Db2 needs to examine a row – or a certain column or columns – to see if the row qualifies for a query's result set). When a page of a COMPRESS YES table space is written to disk, the data set encryption feature of z/OS will encrypt the compressed data on the page. When the page is read into memory, the compressed data in the page will be decrypted. No problem.



RACF management of Db2-internal security

37

And, lastly, a look at the use of RACF (or equivalent) to manage Db2-internal security (i.e., to manage Db2 privileges and authorities).

Misunderstanding related to terminology

- I might ask a Db2 for z/OS DBA, “Do you use RACF to manage Db2-internal security?” and DBA might respond, “Yes,” when in fact this is not the case – **why does that happen?**
- Could be due to confusion about Db2 **internal** vs. **external** security
 - Db2 external security: “Which IDs (of users or applications) **can connect** to this Db2 system, **and how** (e.g., via CICS, DDF, batch, etc.)?”
 - Db2 external security is basically always handled by RACF (or equivalent)
 - Db2-internal security: “Once an ID **has successfully connected** to this Db2 system, **what can that ID do?**”
 - This has to do with Db2 privileges (e.g., SELECT ON TABLE_X, EXECUTE ON PKG_Y) and authorities (e.g., SYSADM, DBADM)
 - Db2-internal security can be managed through Db2, **or through RACF** (or equivalent)

38

Even veteran Db2 for z/OS people regularly get crossed up on this topic. Some of these folks would say that RACF is used to manage Db2-internal security at their site when that is in fact not the case. Why the misunderstanding? Sometimes it is related to misunderstanding about Db2 external security (“What IDs can connect to this Db2 subsystem, and how?”), which is always managed by RACF (or equivalent), and Db2-internal security (“Once connected to this Db2 subsystem, what can an ID do?”), which can be managed through Db2 or through RACF (or equivalent).

Another source of confusion

- Some people will say, “Yes,” to “Do you use RACF to manage Db2-internal security” when RACF is just **involved** in management of Db2-internal security
- Common reason for that: **RACF group IDs** are used in managing Db2 privileges and authorities
 - That can be a really good practice (e.g., grant system DBADM to group ID PRODDBA, and connect IDs of 10 DBAs to that group ID), but **you’re still managing Db2-internal security through Db2**
- Here’s the deal: if you manage Db2 privileges and authorities with **GRANT and REVOKE statements**, you’re **using Db2** to manage Db2-internal security

39

As noted on the slide, some Db2 people will say (incorrectly) that Db2-internal security at their site is managed through RACF, when in fact it’s managed through Db2 but with privilege and authority grants often executed for RACF group IDs versus individual IDs. [Granting Db2 privileges and authorities to RACF group IDs can significantly simplify Db2 privilege management – if 10 people have a certain Db2-related role, the privileges and/or authorities needed to perform that role can be granted once to a RACF group ID associated with the role, and when the IDs of the 10 individuals are connected to the RACF group ID, they will have those privileges and/or authorities in their Db2 privilege set, because the RACF group ID will be a secondary Db2 authorization ID for the individuals.]

Here’s the deal: if the SQL statements GRANT and REVOKE are executed in your environment, Db2 is being used to manage Db2-internal security.

Using RACF to manage Db2-internal security

- Db2 exit **DSNXRXAC** lets RACF manage Db2-internal security
- When exit is **activated**, and ID SMITH wants to SELECT data from table T1, **Db2 drives the exit** and asks RACF if SMITH can do that
 - RACF checks to see if a) it has a **resource defined** that maps to SELECT privilege on T1, and b) if ID SMITH has been **permitted access to that resource profile**
 - If **a and b are true**, RACF tells Db2 that **SMITH can SELECT from T1**
 - If a is true and **b is not**, RACF tells Db2, “SMITH **cannot** do that”
 - If a is not true (no applicable resource defined), RACF defers to Db2
 - When RACF management of Db2 internal security **fully implemented**, the **authorization tables** in the Db2 catalog (e.g., SYSUSERAUTH, SYSPACKAUTH) can be **empty**, because Db2 is not using them

40

When RACF is used to manage Db2-internal security, how does that work? It works by way of an exit that Db2 provides for this purpose: when some ID wants to do some Db2 thing, Db2 drives that exit and asks RACF, “Can this user do this thing?” RACF will check its Db2-related resource profiles (which map to various Db2 privileges and authorities) and will respond to Db2 with “Yes,” “No,” or “I’ll defer to you on this one, Db2” (the latter response is returned from RACF when RACF does not have a resource profile that maps to the privilege or authority being checked).

As pointed out at the bottom of the slide, when RACF management of Db2 privileges and authorities has been fully implemented for a Db2 subsystem, that subsystem’s security-related catalog tables (e.g., SYSUSERAUTH, SYSTABAUTH, SYSPACKAUTH) can be empty, because they are not being used by Db2.

More on managing Db2-internal security with RACF

- For organizations that go this route, **what is the motivation?**
 - Usually, aim is to have **one team** manage Db2 internal **and** external security – in that case, makes sense for that one team to be **RACF team**
 - **Db2 team still involved**, advising RACF team on privilege/authorization requirements
- If you decide to do this, check value of **2 Db2 ZPARM parameters**:
 - AUTHEXIT_CHECK – if **set to DB2** (default: PRIMARY) then if package is **auto-rebound** RACF will check ID of **package's owner** for rebind authorization
 - Otherwise, ID checked will be that of **process requesting package execution** – could lead to **autobind failure** due to authorization error
 - AUTHEXIT_CACHEREFRESH – if **set to ALL** (default: NONE), **Db2 in-memory caches** of authorization information (e.g., package execution authorization) **will be refreshed** when authorization change made on RACF side

41

Sometimes I'm asked by a Db2 for z/OS DBA, "When an organization opts to manage Db2 privileges and authorities through RACF, why do they go that route?" In my experience, the decision to take this approach is motivated by a desire to have a single team manage all aspects of Db2 security – internal and external.

If your organization decides to implement RACF-management of Db2 privileges and authorities, I can tell you that it works well. The transition can be a little involved, owing in part to the fact that Db2 people and RACF people use different terminology to refer to the same thing, but again, the end result I'd expect would be success.

The two Db2 ZPARMs highlighted on the slide are important when RACF is used to manage Db2-internal security – definitely check on the values to which they are set if you decide to use RACF for managing Db2 privileges and authorities.



IDUG

**Let Me Make This Clear (Vol. 4):
Explaining Often-Misunderstood
Db2 for z/OS Concepts and Facilities**

Robert Catterall
rfcatter@us.ibm.com

Session code: PERF7

 Please fill out your session evaluation!


@IDUGdb2
#IDUG_NA24

I hope that the information In this presentation will be useful for you.