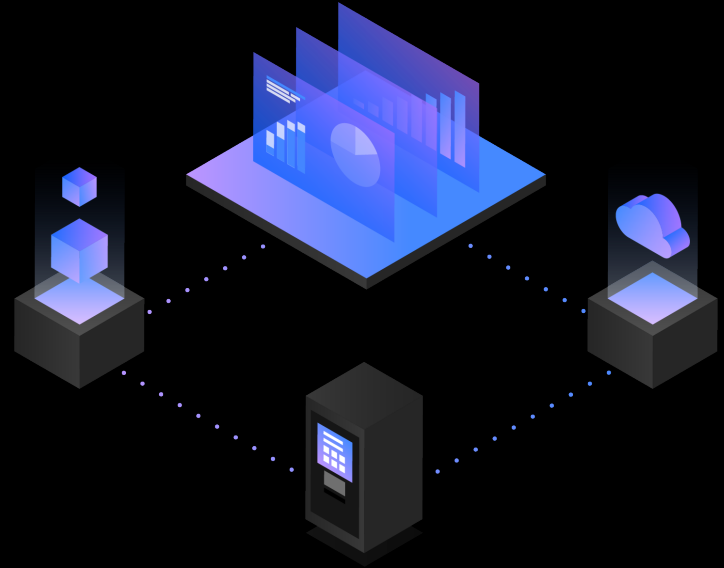


Let Me Make This Clear (Vol. 4): Explaining Often-Misunderstood Db2 for z/OS Concepts and Facilities

New England Db2 Users Group

March 21, 2024

Robert Catterall, IBM
Principal Db2 for z/OS Technical Specialist



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
- Temporary tables

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

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

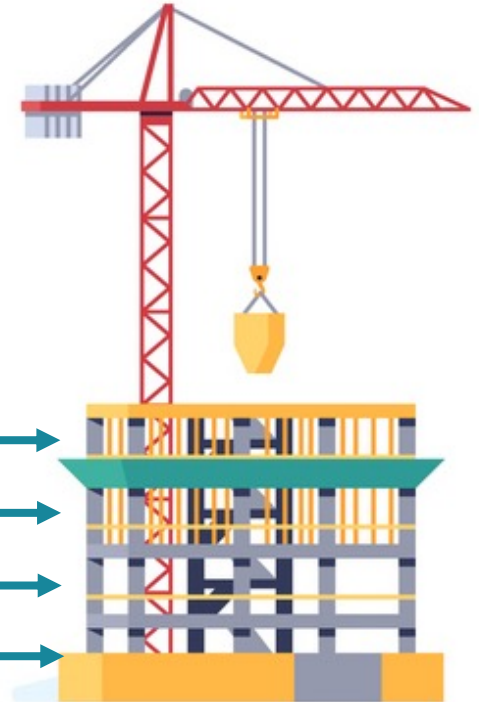
- Think about the floors of a building: you can't build the second floor without the first floor, you can't build the third floor without the second, etc.
- So it is with the various Db2 levels (which were introduced with Db2 12 and continuous delivery): each level **depends on the levels below it**

Application compatibility level →

Activated function level →

Catalog level →

Code level →



The ground floor: code level

- Refers largely to the [currency of code](#) in a Db2 subsystem's load library
 - How so: each Db2 function level is associated with an APAR – when fix for that APAR is applied to the 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, FL503)
 - Saturday: maintenance upgrade causes fix for APAR PH54919 (associated with Db2 13 FL504) 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 (among other things) new AI_COMMONALITY built-in function – for that function to be available, has to be present in Db2 subsystem's [code](#)

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**

More about the second floor

- Not every new function level has catalog dependencies
 - For such a function level, the catalog level has to be at least the one associated with the most recent previous function level that did have catalog dependencies
 - Example: **Db2 13 FL503** requires catalog level **V13R1M501** (FL503 has no catalog dependencies, and neither does FL502)
- Mechanism for updating catalog level: CATMAINT utility
 - Example: **CATMAINT UPDATE LEVEL (V13R1M501)**

Third floor: activated function level



- For Db2 continuous delivery to be practical, “turning on” of functionality introduced with new function level had to be made **asynchronous** with the adding of function-providing code to Db2 subsystem’s 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: Db2 command **-ACTIVATE FUNCTION LEVEL**
 - Example: **-ACTIVATE FUNCTION LEVEL (V13R1M504)**
- Successful activation of function level X requires that code level be at least X and that catalog level be at least X (or most recent previous level with catalog dependencies, if X has no catalog dependencies)

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)***

More about the top floor

- Package's APPLCOMPAT value specifies application compatibility level for program executing the package
 - A few things to note:
 - APPLCOMPAT(X): 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) as well as to DML statements (e.g., SELECT and UPDATE), so pay attention to APPLCOMPAT value of packages related to **SPUFI, DSNTEP2**, etc.

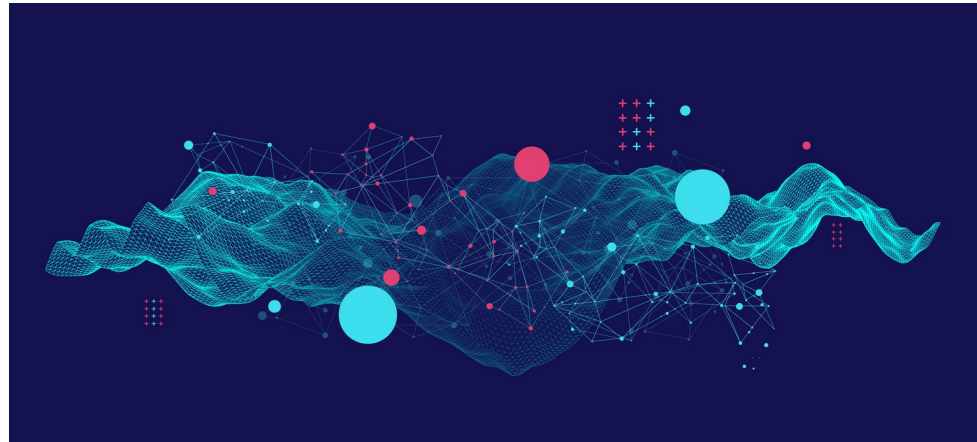
SQL Data Insights: AI in Db2

Some basic facts about SQL Data Insights (SQLDI)

- Introduced with Db2 13 for z/OS, available via function level [V13R1M500](#)
- 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 are no-charge (example: IBM Z Deep Neural Network Library, aka ZDNN, provided via application of z/OS APARs)
- Putting the functionality to use:
 - Using SQLDI GUI, Db2 DBA directs Db2 to build “[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 user via 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](#)

How SQL Data Insights changes the game


- Provides user/program with answers to “similar to this” or “dissimilar to that” – *without user having to tell Db2 what “similar” or “dissimilar” means*
 - 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



Some usage scenarios

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

SQL Data Insights – the built-in Db2 functions

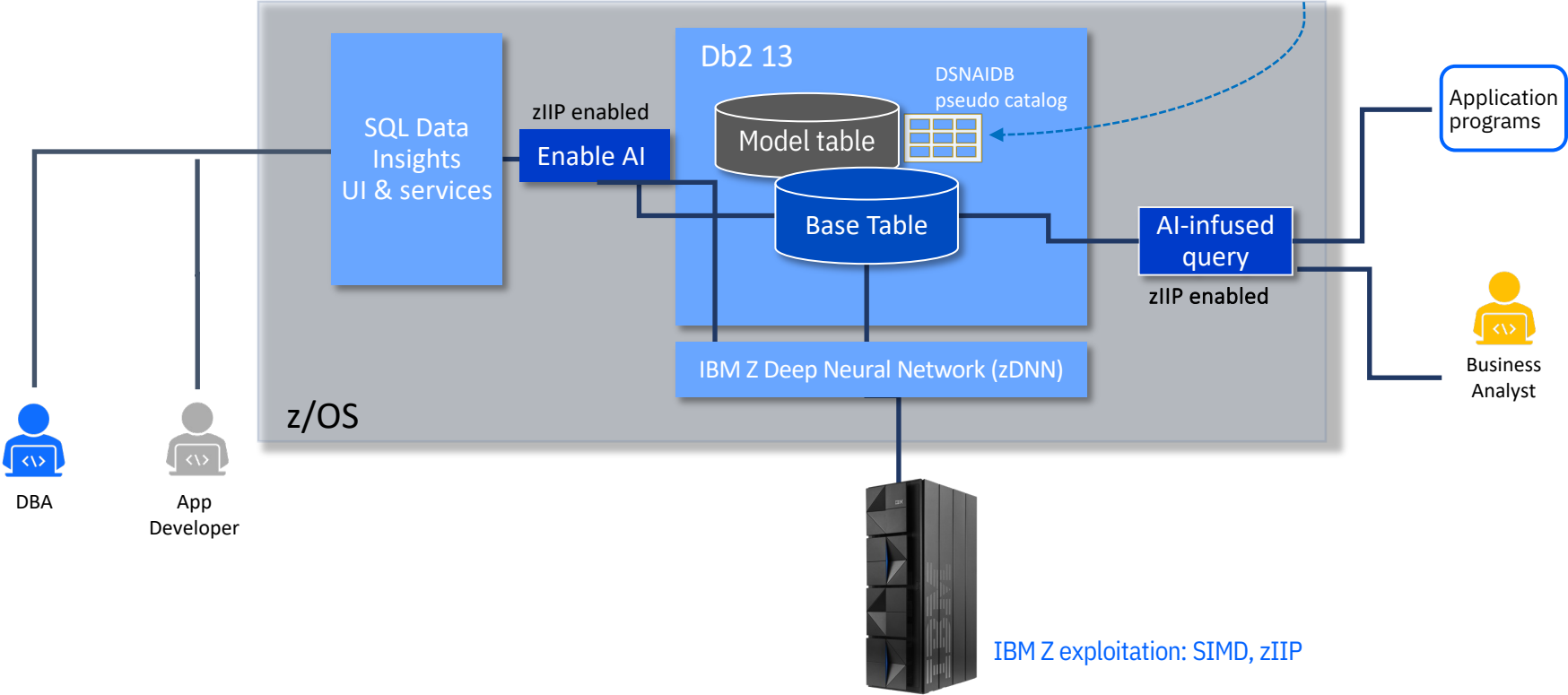
Function	Description
AI_SIMILARITY	Returns the entities that are most similar to (or dissimilar to) a particular entity
AI_SEMANTIC_CLUSTER	Returns the entities that are most similar to (or dissimilar to) a given set of up to three entities
AI_ANALOGY	Consider the relationship between value X in COL1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

The big picture

Vector information

```

8879-zZna
-0.141558 -0.346767 -0.453296 0.052447 0.476916
-0.338483 0.000035 0.517277 0.191573 0.076891
-0.149729 1.036879 0.127160 -0.329846 -0.157252
-0.288485 0.243588 0.038326 -0.338862 0.173571
0.231060 0.149021 -0.328546 -0.058121 0.025713 ...
    
```



“Old” Db2 client code in your environment

“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](#)

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

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

- You don't want developers of DRDA requester apps to be **blocked from using SQL syntax and functionality** introduced after Db2 12 FL500, do you?
- If you have old Db2 client code, and you want developers of DRDA requester applications to have access to latest Db2 SQL functionality:
 - **BIND COPY** packages in NULLID to 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 profile, have attribute that tells Db2 to issue **SET CURRENT PACKAGE PATH = COLL_X** when pre-Db2 11.1 clients connect to system
 - After doing that, make APPLCOMPAT of NULLID packages **as high as you want** – old Db2 clients are using IBM Data Server Driver packages in COLL_X

Example related to preceding slide

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

LOCATION PRDID
::FFFF:1.2.3.4 SQL090703

- In SYSIBM.DSN_PROFILE_TABLE

PROFILEID	PRDID	PROFILE_ENABLED
99	SQL090703	Y

- In SYSIBM.DSN_PROFILE_ATTRIBUTES

PROFILEID	KEYWORDS	ATTRIBUTE1
99	SPECIAL_REGISTER	SET CURRENT PACKAGE PATH = COLL_X

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

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

The Db2 REST interface

Underappreciated by some Db2 for z/OS people

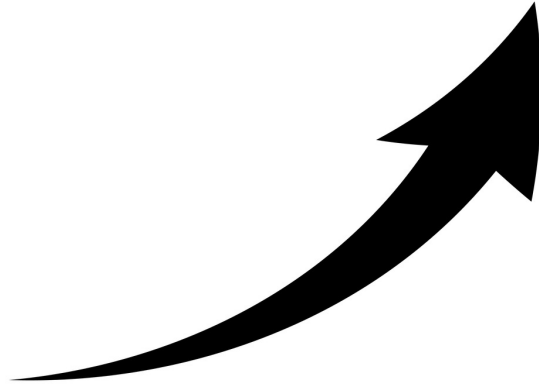
- Some folks have not seen how important the Db2 REST interface can be...
 - REST architectural style can be ideal for [seamlessly integrating z/OS-based data services with cloud-based applications](#) (for client-side developers, particulars of data server completely abstracted – same is true of server-side developers)
 - Good for development and deployment [productivity](#) and [agility](#)
 - Major [broadening of programming languages](#) that can access Db2 for z/OS data: if program written in some language can issue a REST request, it can access Db2
 - [No need for Db2 client code](#) (e.g., IBM Data Server Driver) on client application side, because client application issues REST requests vs. SQL (SQL is server-side)

More Db2 REST goodness...

- **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**

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



What about stateless nature of REST architectural style?

- “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

Db2 data set encryption

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 today that encrypt thousands of Db2 data sets using this feature – additional CPU cost of SQL statement 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

Some folks not clear on mechanics of how this 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*
- There are **several ways of doing this**:
 - 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** statement
 - Do-able in Db2 13 system, or Db2 12 with FL502 or higher activated
 - Also: Db2 package through which CREATE or ALTER statement is executed **must have APPLCOMPAT value of V12R1M502 or higher**
 - Note: ENCRYPTION_KEYLABEL in ZPARM provides label for catalog/directory data sets, and for archive log data sets (when archiving to disk)

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?
 - 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)

A little more on Db2 data set encryption

- Question: can you **compress and encrypt** data in a Db2 table space?
 - **Absolutely** – data is compressed in memory, then that compressed data gets encrypted when written to disk (reverse happens when data read into memory)
- What if you want to encrypt **active log data sets**?
 - **Leverage online removal of active log data sets** – Db2 13 feature
 - Suppose you have 20 pairs of active log data sets
 - Create 20 new pairs of encrypted active log data sets, dynamically add them to log inventory via NEWLOG option of -SET LOG command (Db2 10 feature)
 - 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)

Temporary tables

Lots of Db2 for z/OS people know there are temp tables, but...

- Not sure when it's good to use one, what type should be used
- Background: two kinds of Db2 temporary table
 - Declared global temporary table (DGTT)
 - Declared in an [application program](#)
 - Usable only by the process that declared the table
 - Created global temporary table (CGTT)
 - Created by a [Db2 DBA](#) (table definition recorded in Db2 catalog)
 - When a CGTT is referenced by an application process, that process gets its own instance of the table
 - Both DGTTs and CGTTs are physically provisioned in [Db2 work file database](#)

When temporary table makes sense versus permanent table

- When it is useful for an application process to have **its own SQL-accessible data store**, and when there is no need to persist data in that data store beyond an execution of the application process
- In such cases, temp tables have some **efficiencies** vs. permanent tables:
 - **DBA time-efficiency**: no persistent database object to manage – temp table **goes away** when application process using the table terminates
 - **Processing efficiencies**:
 - **No locking** at all for CGTT, **almost none** for DGTT
 - No **SYSCOPY inserts** (Db2 catalog) for things like mass DELETES
- Useful page in doc: differences between permanent table, CGTT, DGTT

<https://www.ibm.com/docs/en/db2-for-zos/13?topic=tables-distinctions-between-db2-base-temporary>

Key advantage of DGTTs: support for indexes

- Real-world scenario: at one site, an application process inserted a large number of rows in temporary table, and then needed to selectively delete a portion of those rows, using a DELETE with predicates
 - Right choice: declared global temporary table
 - Why: **index**, matching on predicates of DELETE statements, enable **efficient removal of selected rows**, versus repeated table scans that would otherwise be required



Key advantage of CGTTs: dynamic statement prep not needed

- Statements referencing a DGTT have to be dynamically prepared by Db2 for execution, because there is no information about table in Db2 catalog
- CGTT, on the other hand, is described in Db2 catalog, so static SQL statements referencing table can be prepared for execution at bind time
- Especially when execution of an application process will involve execution of a large number of simple SQL statements referencing temp table, this can make a big difference in CPU time

Executable form of static SQL statement?



CGTT: ready to go



DGTT: assembly required

Robert Catterall

rfcatter@us.ibm.com