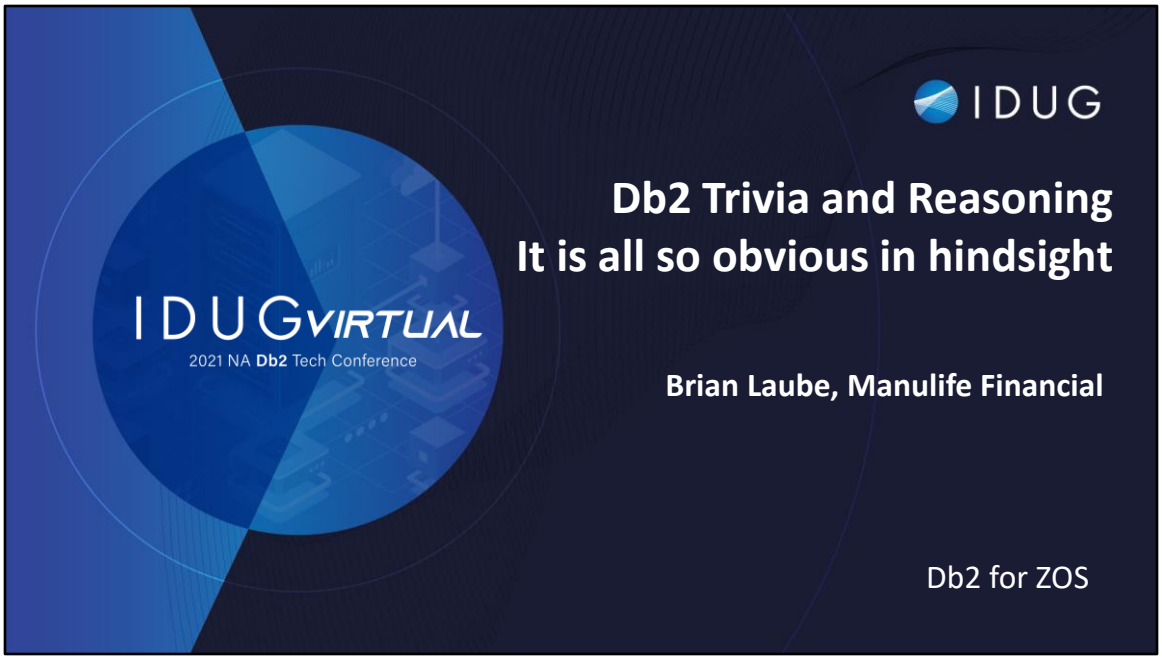

IDUG
IDUG VIRTUAL
2021 NA Db2 Tech Conference
Db2 Trivia and Reasoning
It is all so obvious in hindsight
Brian Laube, Manulife Financial
Db2 for ZOS

My audience for this presentation

- This presentation is for beginner and intermediate DBAS & it is for developer and application support community.
- The purpose is to re-clarify some obvious-to-me terms and make sure we are on the same page.
- I also go through what I consider some great modern SQL tips and best practices along with WHY my gentle-reader should consider them in their work life!

- I expect some of this will be repeats for YOU (the clever people who come to conferences) but there might be some new tips or at least some tips on WHY certain things are useful! I really like to understand WHY! You should too!
- Maybe I can share some useful Real-life experiences!

Agenda

- General mainframe terms
- IBM support and more terms
- RFE and AHA
- What is a disaster? What does it mean to recover?
- Encryption. Different layers.
- My favorite **SQL tricks and tips!**
  - What you need to know! What you should be doing!
  - What to think about during data-zaps -> especially in prod!
  - IBM supplied programs that do something useful: DSNTEP2, DSNTEP4, DSNTIAUL and DSNTIAD. They are not “utilities”
- The future of application development. NATIVE SQL stored procedures. And REST services!

General mainframe terms <used too loosely> (1|3)

- Db2 subsystem is everything about your Db2
- Db2 and “database” are not interchangeable (not in my world)
- I prefer to think of Db2 as the subsystem or instance… which contains multiple databases. Most of the databases are for the applications!
  - In my world… the databases are listed in catalog table SYSIBM.SYSDATABASE. Those are the databases inside my Db2.
- Db2 subsystem exists all the time.
  - Db2 instance only exists when the Db2 is up and running. The Db2 instance is all the memory and the started tasks and related datasets attached to Db2.
  - The Db2 subsystem is the definition of the “Db2” to the operating system (ZOS). It is always defined… no matter if it is stopped or not running now. The subsystem is the collection of all the datasets.

There are too many terms that people through around … and definitions are important!
Sometimes I must stop people and make sure we are speaking the same language.

In other technologies… I think they through around the term database in other technical contexts.

General mainframe terms <used too loosely> (2|3)

- REGION means what you want it to mean. But often… people say REGION when they mean a specific CICS started task.
- Most application databases have one primary or logical user. The application database may also be named after the application. But every database is often used by other applications. Just a little or a lot. Not all users of the application database are the same application! *Do not make assumptions!*
- ENVIRONMENT is the logical collection and logical relationships between the database and ALL the applications that use it.
  - The prod ENVIRONMENT is really all the real applications.
  - The non-prod ENVIRONMENT (say a TEST environment… say T1, but not T2) may have a subset of the applications and interfaces as the prod database. After all, it is often not necessary and too complicated to simulate everything about prod in non-prod

A non-prod environment may really only be the CICS application plus most of the batch. But the non-prod environment may not have all the relationships to some important distributed application server.

General mainframe terms <used too loosely> (2|3)

- JCL and JOB
- Your confusion on the difference confuses me when we talk.
- JCL is job-control-language. It is a script with many steps.
- You 'submit' the JCL to create a process that is the JOB that does all the steps.
- *The JOB is an instance of that one time when the JCL was submitted!*

- You don't fix jobs. You fix JCL.

The difference between JCL and JOBB is not really important… but some people mix them up…. So let us get it clear.

The different parts of IBM

- IBM software sells you software
- IBM support supports the software and

- If IBM runs your data center … then their system DBAS or staff really don't have a magic or special way to get support for IBM software. They get the support that you (indirectly) pay them to get.
- If IBM runs your data center. They may already own some IBM software that they let you use. But if you want extra software (say some Db2 Tool) then you (or your data center) must talk to IBM software to buy it and then your IBM data center people may need to install it!
- Basically… not all IBM is the same.. Be careful who you blame.

Although… go ahead and blame someone. Point the finger

What to do when you have a Db2 question?

- First, read the IBM knowledge center. The KC is reference, not how to do something. But it is the final word on how something should work (not why you would do it).
- If you want to know why something exists or why an option exists, then
  - Attend an IDUG conference! Attend sessions! Learn from others!
  - Read the IDUG-listserv for past discussions on the topic.
  - IBM REDBOOKS often explain "why" we should do something
  - Review popular "Db2 blogs" including IDUG content committee! (**)
- But basically… bookmark and use the Db2 Knowledge Center (KC). Trust the definitions. But don't expect it to tell you WHY or answer all your questions

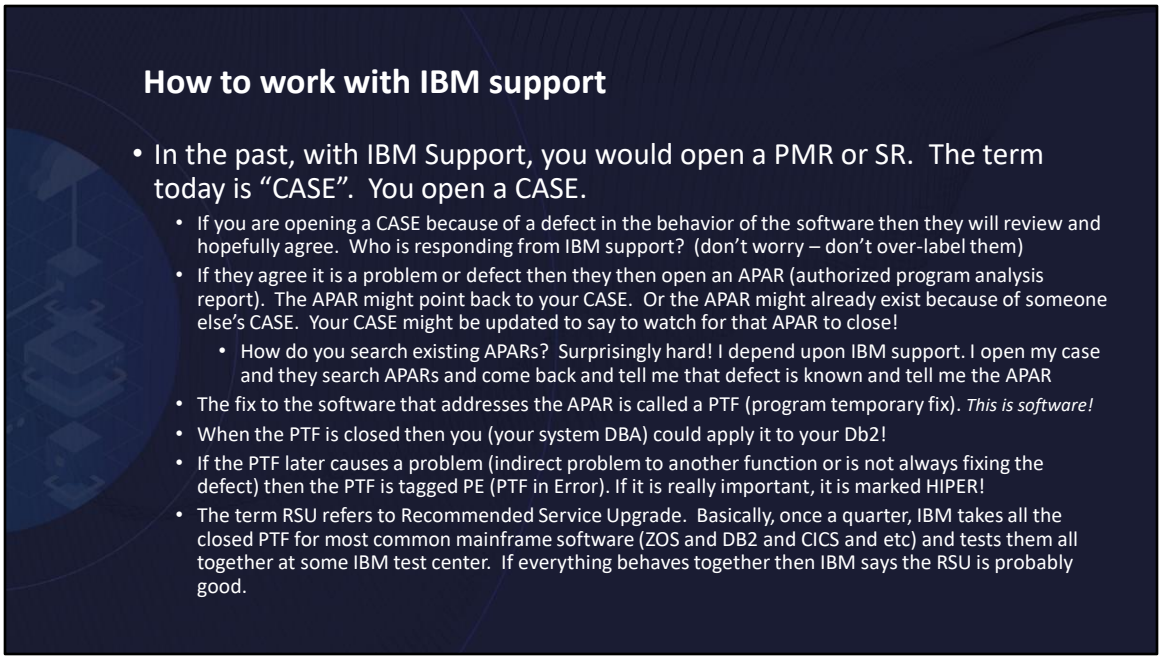
(**) I am a member of the IDUG content committee.

More on Db2 questions

- If you do research and cannot find a good answer to your Db2 question:
  - Post the question on the Db2-listserv. This often gets a fast response. Not always a good response… but someone usually responds in a day or two… if there was no response then your question was perhaps vague or unclear.
  - The guaranteed way to get some answer is to open a CASE with IBM support. If they don't understand then they will try to clarify with you.
    - I use IBM CASES for clarification questions! I am always happy in the end!

How to work with IBM support

- You need an IBM userid. Just sign up on the IBM website. Create your own ID … usually based upon your work email .
  - Once you have an IBM ID. You go under “support access” and click “request access” to associate your ID with one of your companies accounts with IBM. In a simple term.. this “account” must be the one paying for the software for which you want support!
  - How to find the account numbers? Really… ask someone else in your company who uses IBM Support to tell you the important account number(s)
  - Once you request access to the account then someone (usually, inside your company) must approve. This person is not necessarily a technical person or someone you know… (they probably just pay the bill)



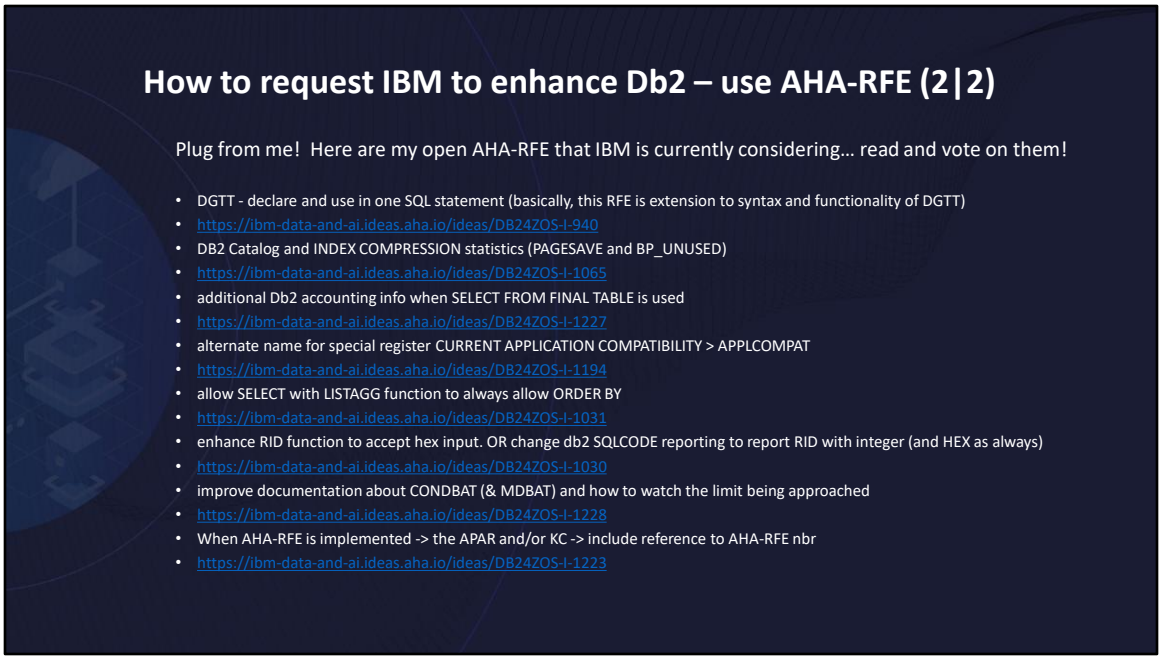
Stop saying PMR and SR. The term is CASE

Who responds from IBM support? A person! I hear the term “level 2” support. I am not sure what that means!. But don’t worry so much.
The initial responder is someone from support whose main job is to answer CASES. They then read the documentation (KC) and make sure what you are reporting is correct. They might search APAR or other CASES for similar scenarios. They might try to recreate your CASE.
If they cannot “resolve” your CASE then they might escalate to someone else. They might have a name but I am not sure of their title! Is is someone from the lab of people who actually write and create the Db2 software? Or pershaps it is someone who is more experienced from IBM support. Don’t worry abou the title. Just worry about the answer.

How to request IBM to enhance Db2 – use AHA-RFE (1|2)

- If you finally decide that Db2 (or any IBM software) should consider an enhancement or change to make it more useful to you (and others) then make a Request for Enhancement (RFE) via the AHA website to make those suggestions
- *You need an IBM userid to vote on or add new!*
- Please make a very clear description so others can review and understand vote and agree with you
- Please make a very focused request with something understandable and clear and logical
- https://ibm-data-and-ai.ideas.aha.io/ideas?project=DB24ZOS
- When review the AHA-RFE from others… vote for those which you understand and think are useful
- Not all AHA-RFE are equal. Some are easy to implement. Some are probably lots of work. Some have lots of benefit to everyone. Some might benefit only those working with that particular aspect of Db2. You can still vote for something if you understand and appreciate it even if does not help you!
- For example, an enhancement to reduce a type of problem you have never experienced. Or an enhancement to a SQL feature (say procedures or REST) that you don't use in that way. But if you understand then vote on it.

https://ibm-data-and-ai.ideas.aha.io/ideas?project=DB24ZOS



DGTT - declare and use in one SQL statement (basically, this RFE is extension to syntax and functionality of DGTT)
https://ibm-data-and-ai.ideas.aha.io/ideas/DB24ZOS-I-940

DB2 Catalog and INDEX COMPRESSION statistics (PAGESAVE and BP_UNUSED)
https://ibm-data-and-ai.ideas.aha.io/ideas/DB24ZOS-I-1065

additional Db2 accounting info when SELECT FROM FINAL TABLE is used
https://ibm-data-and-ai.ideas.aha.io/ideas/DB24ZOS-I-1227

alternate name for special register CURRENT APPLICATION COMPATIBILITY > APPLCOMPAT
https://ibm-data-and-ai.ideas.aha.io/ideas/DB24ZOS-I-1194

allow SELECT with LISTAGG function to always allow ORDER BY
https://ibm-data-and-ai.ideas.aha.io/ideas/DB24ZOS-I-1031

enhance RID function to accept hex input. OR change db2 SQLCODE reporting to report RID with integer (and HEX as always)

https://ibm-data-and-ai.ideas.aha.io/ideas/DB24ZOS-I-1030

improve documentation about CONDBAT (& MDBAT) and how to watch the limit being approached
https://ibm-data-and-ai.ideas.aha.io/ideas/DB24ZOS-I-1228

When AHA-RFE is implemented -> the APAR and/or KC -> include reference to AHA-RFE nbr
https://ibm-data-and-ai.ideas.aha.io/ideas/DB24ZOS-I-1223

Db2 application development – TIP 1 (1|3)

- Db2 NATIVE SQL STORED PROCEDURES are the future of Db2 application development. They are cool and easy to use and understand. Modern (non-mainframe) programmers seem to like them.
- Why are they the future? Because of SQLPL (SQL procedure language)! SQLPL allows you to have variables and loops and IF and the usual 3GL stuff. As a result, you can put real application logic inside the procedure and make the database engine (Db2) do heavy bursts of application processing that requires lots of data. This minimizes I/O and compartmalizes applications functions.
  - If you are using procedures then you really MUST use procedure VERSION. I don't care what VERSION name you use (heck, a simple strategy is a release name that includes date in the name. All related procedures changing at the same time should use the same name).
  - Do I have to spell it out? You create a new version of a procedure in advance. Later, you ALTER the procedure to activate the desired VERSION. If something goes wrong, you still have the old version and switch back is easy.
  - IBM Data Studio and Microsoft VSCODE with DB2 Z developer extension both include a debugger.

VSCODE with the Db2 extension is only useful for developers. Not a DBA

Db2 application development – TIP 1 (2|3)

- Do you use NATIVE SQL STORED PROCEDURES in your mainframe today? No? Too bad!
- As a pro-active DBA then you should setup your Db2 so that it could handle it if you wanted to use it in the future. Some GRANTS and what-not are required.
- And if you are going to proof-of-concept stored procedure debugging (which you really should do) then you will need a new WLM environment for debugging (only debugging) along with the appropriate security resource rules (a pain).
- Build some procedures for yourself!
- I built a bunch of useful-to-me procedures And then I debugged them. I also thought about how to do procedure application tracing or logging.
- My example procedure was a SQLPL SCRIPT to find all my IDENTITY columns and validate the current next value from the catalog against the actual max value in the table! And then my procedure generate sample ALTER statements to reset the next identity value for that column in that database. Why was this useful to me? In some applications, we copy the data around a lot but cannot reset the identify values because of RI. So this procedure was useful to reset. Although, later I just set my dev identity values to start at 10 billion and prod started at 1. So now they never cross each other when moving data from prod to dev.

VSCODE with the Db2 extension is only useful for developers. Not a DBA

Db2 application development – TIP 1 (3|3)

- When beginning to use application procedures. Think about effective application tracing. You should do “application logging” with an autonomous procedure!
  - Consider using a simple application TRACE_ON table.
  - The procedures read the table to see if tracing is required (basically some flag).
  - If the flag is set then use another procedure to INSERT into the application TRACE_LOG table some useful information about the procedure.
  - Of course, the procedure that inserts into the TRACE_LOG table is autonomous!
  - To be clear. An automous procedure can insert into the TRACE_LOG table and then commit and keep the changes. If the main procedure then (later?) decides to ROLLBAKC then at least the LOG info is kept.

SQL terminology 1 – result set and rows

- SQL SELECT result set contains what you want it to contain.
  - It may contain rows from the table. If you used SELECT * and did not join to another table.
  - But really, the rows in the result set are rows in the result set.
  - And if you want the result set to "look" useful to you then think about the tool you are using. SPUFI and DATA STUDIO will display the DECIMAL column data in a pretty numeric form that is easy to read. The same SELECT run through DSNTIAUL will put the DECIMAL data in DECIMAL (packed) output. And that could be fine. Depends who is reading or looking at that data! Think about it
  - If you want your output to always look "pretty" and easy to read then consider using Db2 SQL Functions to convert all the data to CHAR and concatenate it all together!
    - I do this all the time. My SELECT uses functions to make the data pretty. Then I run it through DSNTIAUL and the output is easy to read by the subsequent process/program/application
    - For example, I may use SELECT to generate result output that looks like DB2 commands. The subsequent step can use the DSNTIAUL output with no tweaking! (more later)
    - Same thing is often done with application stuff! Generate control cards or even generate SQL itself.

Disaster Recovery – What does it mean today? (1|2)

- Back in the day, we would "backup" everything!
  - For Db2 itself > system DBAS would "backup" the catalog and BSDS and logs and what-not.
  - And the application DBAS would "backup" all the tablespaces inside the application database.
- Then we would go to the DR-site and recover everything from scratch! Fun times.
  1. The ZOS guys would "recover" the LPAR on some new machine in the DR data center.
  2. And then the system dbas would "recover" the empty Db2 subsystem.
  3. And the application DBAs would "recover" the application databases
  4. In parallel, the application support would recover all their important datasets.
- Eventually we would all think we were ready, and we would bring up CICS and run batch (through the job scheduling software which was itself on the LPAR).
- We would validate everything was recovered and available and working as expected.
- Then we knew we could RECOVER the whole LPAR and everything inside it!
- *This all took time. It was good practice of (fortunately) rarely used utilities in real life prod-lpar.*

Just

Disaster Recovery – What does it mean today? (2|2)

- NOWADAYS, we have data mirroring of disk and tape. All data is mirrored from one set in the real-prod data center to a copy of the disk and tape in the other data center. In the event of a "disaster" we spin up a new DR-test-lpar (not same name as real the LPAR) and point to the copy of the data. Everything fires up and everything works. No recovery of anything!
  - So a modern DR test is a test of our ability to recover from loss of data center (falling meteorite, etc). This is an important validation.
- But we must not forget "application disaster". We must continue to practice regular recovery of stuff like databases (or CICS datasets or whatever) in case of "application disaster". The application disaster often necessitates recovery to a point in time in the past (often a very recent point in time).
- So just because our modern DR-test is fast and efficient and reliable… we must not forget "application recovery"!
- The best place to "practice" application recovery? Inside the DR-test-lpar AFTER you have validated the DR-test-lpar is OK and everything was mirrored, and it all looks ok!
- ***add a day or two the end of your DR test. Ask yourself scenarios, like some application batch caused a bunch of tables to be deleted or corrupted. Opps! How will you recover? Or imagine the storage team had a typo and deleted Db2 datasets on some important DASD driver? Opps! How will you recover? Do you have JCL ready? Do you remember all the little steps and gotchas? Practice APPLCIATION RECOVERY.***
  - ***After all… why else do we COPY our Db2 application databases?***

Just because you have a successful DR-test for a major disaster (loss of disaster) does not mean you can recover your application database

SQL opinion – 1 – DISTINCT

- I have never met a SELECT with DISTINCT that I liked
  - I never use DISTINCT
  - If I am given SQL SELECT to review and it contains DISTINCT then I ask why? Are they truly worried about duplicate rows? If yes, then why? Do they really understand the data and their query? (probably not)
  - In place of DISTINCT… I almost always use SELECT .. COUNT(*) with GROUP BY.
    - Why?
    - It is essentially the same effort by Db2 to deal with DISTINCT or COUNT(*) with GROUP BY.
    - The result set is same number of rows.
    - The info of the result set is the same… except now you have COUNT(*).
    - You the option to look at the COUNT(*) and think if the value is reasonable or expected!

Please disagree with this opinion and tell me why.

SQL opinion – 2 – SELECT without ORDER BY

- Every SELECT must have an ORDER BY
- A SELECT without ORDER BY is suspicious.
  - One exception to the rule is if the SELECT always returns 1 or zero rows.
  - But one must be very sure that is always the case!
- Otherwise, is the application depending upon ORDER of the result set? If not … then really? Are they sure? Some cases do exist… but having consistent order is usually helpful!
- Did the application get the result set in desired order by coincidence?
  - If yes, then they have been lucky so far! Stop depending upon luck!
  - So have ORDER BY unless you can tell me why not!
- ***I had a (fun ?) problem a few years. An old program was rebound and started returning the data in a different order and the application did not process the whole result set because the logical last record was in the middle. And it was not consistent. The change of result set order depended where we ran the query! It was very tricky to find root cause.. The SELECT CURSOR with no ORDER BY!!! Arghhh.***

Even i

IBM/Db2 supplied sample programs that do something useful!

- IBM supplies several sample programs that do something useful.
- We all use them.
- They are called DSNTEP2, DSNTEP4, DSNTIAUL, and DSNTIAD
  - https://www.ibm.com/support/knowledgecenter/en/SSEPEK_12.0.0/appdevsamp/src/tpc/db2z_db2prodaidsamps.html
  - The knowledge center cavalierly refers to these sample programs as utilities.
  - They are useful sample programs. I call them programs. Not utilities.
- They are "sample" because IBM Db2 delivers the source code. You could look at the source (if you wanted) to understand how to write similar programs using assembler or PL1. You could even copy the source and tweak it to your own (twisted) requirements.

The word utility is confusing and incorrect. Very misleading.

DSNTEP2 and DSNTEP4

- DNSTEP2 is the world's simplest program to feed SQL and produce very simple "report" output. If the SQL is SELECT then you get a "report" with simple page numbers and column headers.
- You use DSNTEP2 to "look" at the result of a SQL statement
- You should never use DNSTEP2 as a form of "backup". Why do that? What do you think you will do with this kludgy report?
- DSNTEP2 does single row fetch.
  - But who cares? Sure, multi-row fetch is more efficient for large result sets. But who wants to print a large result set with DSNTEP2? Not me! What would be the point!
- DNSTEP4 is identical to DSNTEP4 except it does multi-row fetch!
  - I never use DNSTEP4 because I do not process large results with DSNTEP2 (see point above!)

Batch SPUFI (minor digression)

- What do you mean when you say “BATCH SPUFI” ???
- SPUFI is very much an ISPF panel delivered by Db2 with the DB2-interactive ISPF panels. It is not a batch program! It produces a formatted output of SQL scripts.
- Most people really mean DSNTEP2 when they say batch SPUFI.
- But DSNTEP2 and SPUFI have very different outputs
- So just say the real batch program name… just say DSNTEP2!

DSNTIAD

- DSNTIAD is like DSNTEP2 but worse. It process all SQL statements except SELECT.
- DSNTEP2 can process all the SQL that can be done by DSNTIAD… and DSNTEP2 can also do SELECT. Hence, I always use DNSTIAD
  - I suppose one could use DNSTIAD for DDL… but I am not sure the point.

DSNTIAUL

- The very useful “unload program”. Not “unload utility”!!!
  - It has the option to feed in a simple table name and it puts a SELECT * in front and then issues the SELECT. (I never use that option)
  - I just always use the option to give DSNTIAUL a SELECT statement as input
    - Obviously, you can easily write your own SELECT * FROM TABLE
    - And then you can use ORDER BY with your SELECT
    - And of course, you can use any SELECT … with joins or complex WHERE
  - The output of DSNTIAUL has 2+ outputs (DSNTEP2 has 1+ important output)
    - SYSPUNCH has a generated LOAD card that *could* be used to LOAD the result set data into a target table. The generated LOAD is rarely/never used as input to LOAD utility without tweaking for the desired cases.
      - Even if you don’t LOAD the result set output later. This SYSPUNCH is your only “description” of what the output result set looks like! Important!
    - SYSRECnn is the output that contains the result set of each SELECT from the input SQL script. The data looks exactly like the result set! 1 wide row! Not necessarily easy to read. And you can’t read it and understand without having the corresponding SYSPUNCH nearby to know the layout!

I TRY TO NOT SAY “USE UNLOAD”. VAGUE. I ALWAYS SAY UNLOAD PROGRAME OR UNLOAD UTILITY.

DSNTEP2 and DSNITAUL and “data zaps” (1|9)

- All applications have the occasional need to run DML and “data zap” table data. This is how we change data outside of the application!
- The DML is often prepared and tested and validated in non-prod databases
- We might use a workstation tool for ad-hoc SQL development to work on the SQL (Data Studio, VSCODE, or any number of workstation db tools)
- But usually (most people?) want to run data zaps against the prod data using one-shot JCL and a job that can then be properly archived and audited in the future! (and then I don’t have to run any zaps in prod myself…. Better a oneshot)

DSNTEP2 and DSNITAUL and "data zaps" (2|9)

- I have seen many different types of one-shot JCL.
- What is the best thing to do around the zap?
  - Backup first (how?). Display current contents? There is no one best answer!
  - It depends upon your application and what you are zapping and what you consider is possible for backout?
  - If a data zap goes wrong… the most common first solution is another data zap to correct the problem! FIX FORWARD!
    - It may be a simple prepared and ready UPDATE to undo the previous UPDATE. Or another UPDATE to zap rows to be "ineffective"
    - It maybe a simple DELETE to remove the recently INSERTed rows.
    - It all depends… no one solution. But think in advance!
  - My second solution to data zap gone wrong is to use my LOG ANALYZER tool to generate UNDO SQL and then run that. <practice your LOG ANALYZER>
    - This is useful when your zap actually zapped more then you expected (for example semi colon in wrong place! Big opps)
    - Or the ZAP/UPDATE had a surprising impact to the application and it must be backed out.

DSNTEP2 and DSNITAUL and "data zaps" (3|9)

- A 3rd or final solution would be to replace the contents of the zapped table with a known good backup!
- I dis-like the term 'backup'. It means different things.
- This is easy to do with semi-static reference tables. But if it is a live and constantly data-changing application table… then you must be VERY careful about "restoring" the table!
- If you might "restore" your table then consider these forms of "backup" before the data-zap:
  - QUIESCE
  - FULL IMAGE COPY
  - UNLOAD UTILITY
  - DSNTIAUL
- But think! Why do any of the above if you won't use the "backup" then why do it?
- Basically, a data zap with fix-forward (more DML) backout plan is a decent plan.

DSNTEP2 and DSNITAUL and “data zaps” (4|9)

- In any case, simple ‘backup’ is easy with semi-static reference tables. But if it is a live and constantly data-changing application table… then you must be VERY careful about “restoring” the table! You probably don’t want to “restore” that table!
  - A “restore” could be LOAD REPLACE or RECOVER !
- If you might “restore” then consider these forms of “backup” before the data-zap:
  - QUIESCE
  - FULL IMAGE COPY
  - UNLOAD UTILITY
  - DSNTIAUL
- But think! Why do any of the above?
- Using UNLOAD or DSNTIAUL is handy if you might want to “look” at the data before the change. I suppose that is possible (but how often do you really do that?)

DSNTEP2 and DSNITAUL and "data zaps" (5|9)

- One could use DSNTEP2 for data zap DML. But consider DSNTIAUL
- If your data zap script has SELECT * before the zap to somehow display "useful" information before the zap then ask yourself why and how much might you see?
- If the result set is 'small' then DSNTEP2 is maybe an okay method for running that script. But if the result set might large… why not run it in DSNTIAUL? Then the output goes to a flat file that is not easy to read. But, in a pinch, it could be read.
- And of course, DNSTIAUL can process DML. So why not let it!

- BUT…even better… use "modern" SQL to zap your data.
- Use SELECT FROM FINAL TABLE (or SELECT FROM OLD TABLE) around or outside your UPDATE or DELETE.
  - This great SQL SELECT option lets you SELECT exactly the data you are changing.
  - You will not need SELECT * before and after the data zap.
  - Just SELECT from the DML!!!
- It might actually be useful. AND it will make the inquisitive auditor happy!

DSNTEP2 and DSNITAUL and "data zaps" (6|9)

- I had a recent case where we needed to "zap" a varying amount of data in a huge production table. We did not want to unload the whole table before hand (no point).
- We considered SELECT of the relevant columns before and after the UPDATE. But the output of the SELECTS was going to hard to read and take real meaning without lots of comparisons.
  - And should we use DNSTEP2 or DSNTIAUL?
  - People like to use DSNTEP2 for zaps… but really.. The output is a formatted and simple report. But will you look at it? And is it useful for helping with a backout? (probably no)
- After some discussion, we decided on SELECT FROM FINAL TABLE. The "trick" or thing to remember when the DML is UPDATE is that you might want to see the row's PRE and POST change value. EASY. Use "include" to add some columns!
- We then used DSNTIAUL and produced a useful output of every row changed with the primary key columns, a few other name columns and then pre and post values for the columns that changed.
- Really.. .this is all pretty elegant and this how we should all do DATA ZAPS today

DSNTEP2 and DSNITAUL and “data zaps” (7|9)

```
-- I AM A NICE GUY.  GIVE ALL 'DESINGERS' A 10% RAISE
SELECT EMPNO, WORKDEPT, FIRSTNME, LASTNAME
, JOB, SALARY, PREV_SALARY
FROM FINAL TABLE (
UPDATE EMP INCLUDE (PREV_SALARY DECIMAL (9,2))
  SET PREV_SALARY = SALARY
     ,SALARY = SALARY * 1.10
WHERE 1=1
  AND JOB='DESIGNER'
)
ORDER BY EMPNO
;
```

EMPNO	WORKDEPT	FIRSTNME	LASTNAME	JOB	SALARY	PREV_SALARY
000150	D11	BRUCE	ADAMSON	DESIGNER	27808.00	25280.00
000160	D11	ELIZABETH	PIANKA	DESIGNER	24475.00	22250.00
000170	D11	MASATOSHI	YOSHIMURA	DESIGNER	27148.00	24680.00
000180	D11	MARILYN	SCOUTTEN	DESIGNER	23474.00	21340.00
000190	D11	JAMES	WALKER	DESIGNER	22495.00	20450.00
000200	D11	DAVID	BROWN	DESIGNER	30514.00	27740.00
000210	D11	WILLIAM	JONES	DESIGNER	20097.00	18270.00
000220	D11	JENNIFER	LUTZ	DESIGNER	32824.00	29840.00
200170	D11	KIYOSHI	YAMAMOTO	DESIGNER	27148.00	24680.00
200220	D11	REBA	JOHN	DESIGNER	32824.00	29840.00

DSNTEP2 and DSNITAUL and “data zaps” (8|9)

```
-- I HAVE DECIDED ... THIS DEPT IS LAME.  DELETE IT!
SELECT *
FROM OLD TABLE (
DELETE FROM DEPT
WHERE 1=1
  AND DEPTNO='J22'
)
ORDER BY DEPTNO
;
```

```
DEPTNO DEPTNAME          MGRNO ADMRDEPT LOCATION
------ ----------------- ----- -------- ----------------
J22    BRANCH OFFICE J2  NULL  E01
```

DSNTEP2 and DSNITAUL and “data zaps” (9|9)

- Interesting caveat about using SELECT FROM FINAL/OLD TABLE!
- The accounting trace information for the data zap will change SELECT (or OPEN/FETCH) counters but nothing for the counter of the quantity of UPDATE performed by that thread!
- This is a bit misleading.
- The accounting trace output will show ROWS_UPDATED > 0.
- Basically, if reviewing accounting trace output (say in your online monitor OR Db2 performance table for DB2PMFACCT_GENERAL) then you may find thread history shows no UPDATE (or DELETE or INSERT) but the ROWS_changed information will be greater then zero. So you must be aware that this is how you find the thread history of the usage of this type of activity!

Vote for my AHA-RFE to enhance accounting trace for when SELECT FROM FINAL/OLD TABLE is used

https://ibm-data-and-ai.ideas.aha.io/ideas/DB24ZOS-I-1227

SQL TIP 2 – why WHERE 1=1 clause?

- As you see from my previous SQL examples, I like to use WHERE 1=1 to begin most ad-hoc SQL WHERE clause!
- Why? Most often I have a series of WHERE "AND" conditions. Sometimes I comment out one or two and re-run. If the condition that needs to be commented is the first condition, then I must be more careful. But if the first condition is "1=1" then I probably won't change it! And then I can use the easy manual "- -" (dash dash comment) in my TSO editor.. Or if I am in Data Studio use the easy "ctrl-slash" to have Data Studio auto-comment the selected lines.
  - I run some SQL over and over and tweak all the time. So this 1=1 works for me.
  - I do not recommend using 1=1 in real application SQL. Just ad-hoc stuff!

SQL Tip 3: let Db2 track last data change (1|1)

- It is handy if application tables have columns to track things like:
  - LAST_CHN_TS and LAST_UPD_USERID
    - On INSERT, it is easy enough to default to meaningful value.
    - On UPDATE, these values can be maintained by the application (who I don't trust to be consistent) or triggers.
- But modern Db2 has DDL options which make it super-easy for Db2 to track key info so you don't have to do anything. Your application people do not even need to know!
- Read the KC…. But basically use 'GENERATED ALWAYS' *on all important tables!*
  - *Consider adding these useful (almost invisible) columns to all important tables!*

```
ALTER TABLE TCCUSTOMER
ADD LAST_UPD_TS TIMESTAMP NOT NULL
     GENERATED ALWAYS FOR EACH ROW ON UPDATE AS ROW CHANGE TIMESTAMP
ADD LAST_UPD_USERID VARCHAR(128)
     GENERATED ALWAYS AS (SESSION_USER)
ADD LAST_DML_TYPE_CD CHAR(1)
     GENERATED ALWAYS AS (DATA CHANGE OPERATION)
;
```

If you can… begin to add these useful columns to all 'important' tables!

SR of SESSION_USER is preferred to USER

SQL Tip 4: A use case example for system temporal tables (1|5)

- You don't need a big application requirement to find uses for temporary tables… just think of your application data questions that are hard to answer!
- In my world, we have an important customer table called TCUST with column called EMAIL_ADDR. All of a sudden, there was a lot of question about the quality of EMAIL_ADDR and who was setting the data value.
- Some researched showed that half-dozen (maybe six) distinct applications had the potential to update that column. And some did the update periodically. *It was a popular table. The question was complicated!*
- Every once in a blue moon we discover an unexpected EMAIL_ADDR value. We would try to figure when this value was set and by whom. Unfortunately, sometimes it was set a long time ago.

SQL Tip 4: A use case example for system temporal tables (2|5)

- We considered code-reviews and our Query Monitor Tool and our Log Analyzer and our Db2 monitor (and our trace outputs).
- All these tools provide some answers to some questions… but none was a complete answer.
- We decided to use the Db2 concept of temporal tables!
- Basically, we add a parallel (identical) history table beside the main table. Use the magic DDL to relate the two tables. Anytime there is DML on a row, the current version is written to the history table before actually updating the main table.
- There is no impact to the applications.
- There is a performance impact (extra INSERTs across to the history table and what-not) but the cost is relative to the amount of data change to the main table in question. It is all relative. But 100s of thousands per day is not unreasonable or impactful.

SQL Tip 4: A use case example for system temporal tables (3|5)

- Obviously, you have to think about eventual purge of the history table. But keep as much history as might be useful for YOUR data questions. We are keeping many many months…. Because space is cheap and the overall impact is manageable.
- The history table does not need the same indexes as the main table. It only needs whatever indexes might help you with your later data investigation questions. Heck, maybe no indexes… just scan it. It depends upon your data investigation questions!

SQL Tip 4: Use case for system temporal tables (4|5)

- How to do it? Much documentation exists. But here are the highlights.
  1. Add 3 columns to main table.
  2. Create history tablespace.
  3. Create history table like main table.
  4. Create an index on the history table including primary key of main plus the new SYS timestamps
  5. Alter main table to add period for system time
  6. Alter main table to add versioning and use the history table!

```
ALTER TABLE TCUST ADD COLUMN SYS_START TIMESTAMP(12) NOT NULL
GENERATED ALWAYS AS ROW BEGIN;

ALTER TABLE TCUST ADD COLUMN SYS_END TIMESTAMP(12) NOT NULL
GENERATED ALWAYS AS ROW END;

ALTER TABLE TCUST ADD COLUMN TRANS_ID TIMESTAMP(12)
GENERATED ALWAYS AS TRANSACTION START ID;

ALTER TABLE TCUST
ADD PERIOD SYSTEM_TIME(SYS_START, SYS_END);
```

```
CREATE  TABLESPACE ZCUSTH  IN dbname
 USING STOGROUP stogroup PRIQTY -1   SECQTY -1
 yada yada
 ;

CREATE TABLE TCUST_HIST LIKE TCUST IN DYC01P.ZCUSTH;

CREATE UNIQUE INDEX X01CUSTH  ON TCUST_HIST
 (pk_fld1, pk_fld2,SYS_START,SYS_END)
 yada yada
 ;

ALTER TABLE TCUST ADD VERSIONING USE HISTORY TABLE TCUST_HIST
ON DELETE ADD EXTRA ROW
;
```

You may see that I use PRIQTY -1 SECQTY -1. YES. That is intentional. No downside unless you love to micro-manage

SQL Tip 4: Use case for system temporal tables (5|5)

- The end result is history table on my important customer table!
  - **I love it!** I am able to answer all sorts of questions that I did not know I had before. The cost is low to watch the change of a customer record over time… so very valuable!
  - My recommendation for everyone is to think about it… can this concept help you in your world? Try it!
  - You don't even need to use the SQL syntax for query from temporal tables. Just query that useful history table directly!

```
select 'current' as vers
, cust_id, EMAIL_ADDR, CUST_FIRST_NAME, SPONSOR_NAME, CHN_USER_ID, CHn_Ts, LAST_UPD_TS, SYS_START
from TCUST c
where cust_id = 3118591
union all
select 'history' as vers
, cust_id, EMAIL_ADDR, CUST_FIRST_NAME, SPONSOR_NAME, CHN_USER_ID, CHn_Ts, LAST_UPD_TS, SYS_START
from TCUST_HIST h
where cust_id = 3118591
order by last_upd_Ts desc
;
```

Notice UNION ALL … (more on that later)

SQL TIP 5 – Fun with SQL and find parallel value w GROUP BY (1|3)

- Has this happened to you? (people who often create SQL SELECT)
- How can one find parallel values for a GROUP BY?
- When you think about it… this type of question can occur all the time.
- For example, you query SYSCOPY to find the most recent image copy for a tablespace in a database. But you want the name of the tablespace!
- Another example, query the "employee" table to find most recent hire_date and the name of that hire!

- I wrote a IDUG content blog on this question and solutions
- https://www.idug.org/blogs/tony-andrews1/2020/12/15/sql-tricks-part-4-finding-parallel-values-with-gro

SQL TIP 5 – Fun with SQL and find parallel value w GROUP BY (2|3)

- The initial and obvious (and expensive/slow)solution is simple.
- First query and find the GROUP_BY and then query main table again

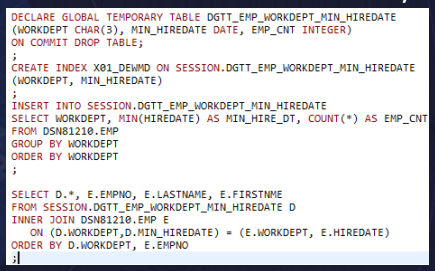
```
WITH CTE_A AS (
SELECT WORKDEPT, MIN(HIREDATE) AS MIN_HIRE_DT, COUNT(*) AS EMP_CNT
FROM DSN81210.EMP
GROUP BY WORKDEPT
ORDER BY WORKDEPT
)
SELECT A.WORKDEPT, A.EMP_CNT, A.MIN_HIRE_DT
     , E.EMPNO, E.LASTNAME, E.FIRSTNME
FROM CTE_A A
INNER JOIN DSN81210.EMP E
  ON (A.WORKDEPT, A.MIN_HIRE_DT) = (E.WORKDEPT,E.HIREDATE)
ORDER BY WORKDEPT, E.EMPNO
;
```

WORKDEPT	EMP_CNT	MIN_HIRE_DT	EMPNO	LASTNAME	FIRSTNME
A00	5	1958-05-16	000110	LUCCHESI	VINCENZO
B01	1	1973-10-10	000020	THOMPSON	MICHAEL
C01	4	1971-07-28	000130	QUINTANA	DOLORES
D11	11	1966-03-03	000200	BROWN	DAVID
D21	7	1966-11-21	000230	JEFFERSON	JAMES
E01	1	1949-08-17	000050	GEYER	JOHN
E11	7	1964-09-12	000310	SETRIGHT	MAUDE
E11	7	1964-09-12	200310	SPRINGER	MICHELLE
E21	6	1947-05-05	000340	GOUNOT	JASON
E21	6	1947-05-05	200340	ALONZO	ROY


```
WITH CTE_A AS (
SELECT WORKDEPT, MIN(HIREDATE) AS MIN_HIRE_DT, COUNT(*) AS EMP_CNT
FROM DSN81210.EMP
GROUP BY WORKDEPT
ORDER BY WORKDEPT
)
SELECT A.WORKDEPT, A.EMP_CNT, A.MIN_HIRE_DT
     , E.EMPNO, E.LASTNAME, E.FIRSTNME
FROM CTE_A A
INNER JOIN DSN81210.EMP E
  ON (A.WORKDEPT, A.MIN_HIRE_DT) = (E.WORKDEPT,E.HIREDATE)
ORDER BY WORKDEPT, E.EMPNO
;
```

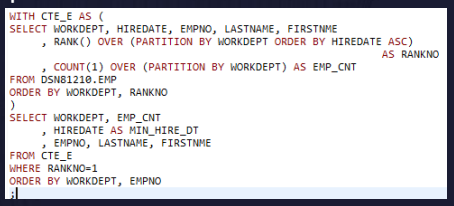
SQL TIP 5 – Fun with SQL and find parallel value w GROUP BY 3|3)

- Solution 2 – use DGTT – logical - fast
- Solution 3 – use OLAP – tricky – super fast



```
DECLARE GLOBAL TEMPORARY TABLE DGTT_EMP_WORKDEPT_MIN_HIREDATE
(WORKDEPT CHAR(3), MIN_HIREDATE DATE, EMP_CNT INTEGER)
ON COMMIT DROP TABLE;
;
CREATE INDEX X01_DEWMD ON SESSION.DGTT_EMP_WORKDEPT_MIN_HIREDATE
(WORKDEPT, MIN_HIREDATE)
;
INSERT INTO SESSION.DGTT_EMP_WORKDEPT_MIN_HIREDATE
SELECT WORKDEPT, MIN(HIREDATE) AS MIN_HIRE_DT, COUNT(*) AS EMP_CNT
FROM DSN81210.EMP
GROUP BY WORKDEPT
ORDER BY WORKDEPT
;

SELECT D.*, E.EMPNO, E.LASTNAME, E.FIRSTNME
FROM SESSION.DGTT_EMP_WORKDEPT_MIN_HIREDATE D
INNER JOIN DSN81210.EMP E
   ON (D.WORKDEPT,D.MIN_HIREDATE) = (E.WORKDEPT, E.HIREDATE)
ORDER BY D.WORKDEPT, E.EMPNO
;
```



```
WITH CTE_E AS (
SELECT WORKDEPT, HIREDATE, EMPNO, LASTNAME, FIRSTNME
     , RANK() OVER (PARTITION BY WORKDEPT ORDER BY HIREDATE ASC)
                                                        AS RANKNO
     , COUNT(1) OVER (PARTITION BY WORKDEPT) AS EMP_CNT
FROM DSN81210.EMP
ORDER BY WORKDEPT, RANKNO
)
SELECT WORKDEPT, EMP_CNT
     , HIREDATE AS MIN_HIRE_DT
     , EMPNO, LASTNAME, FIRSTNME
FROM CTE_E
WHERE RANKNO=1
ORDER BY WORKDEPT, EMPNO
;
```

```
DECLARE GLOBAL TEMPORARY TABLE DGTT_EMP_WORKDEPT_MIN_HIREDATE
(WORKDEPT CHAR(3), MIN_HIREDATE DATE, EMP_CNT INTEGER)
ON COMMIT DROP TABLE;
;
CREATE INDEX X01_DEWMD ON SESSION.DGTT_EMP_WORKDEPT_MIN_HIREDATE
(WORKDEPT, MIN_HIREDATE)
;
INSERT INTO SESSION.DGTT_EMP_WORKDEPT_MIN_HIREDATE
SELECT WORKDEPT, MIN(HIREDATE) AS MIN_HIRE_DT, COUNT(*) AS EMP_CNT
FROM DSN81210.EMP
GROUP BY WORKDEPT
ORDER BY WORKDEPT
;

SELECT D.*, E.EMPNO, E.LASTNAME, E.FIRSTNME
FROM SESSION.DGTT_EMP_WORKDEPT_MIN_HIREDATE D
INNER JOIN DSN81210.EMP E
  ON (D.WORKDEPT,D.MIN_HIREDATE) = (E.WORKDEPT, E.HIREDATE)
ORDER BY D.WORKDEPT, E.EMPNO
```

```
;


WITH CTE_E AS (
SELECT WORKDEPT, HIREDATE, EMPNO, LASTNAME, FIRSTNME
    , RANK() OVER (PARTITION BY WORKDEPT ORDER BY HIREDATE ASC)
                                        AS RANKNO
    , COUNT(1) OVER (PARTITION BY WORKDEPT) AS EMP_CNT
FROM DSN81210.EMP
ORDER BY WORKDEPT, RANKNO
)
SELECT WORKDEPT, EMP_CNT
    , HIREDATE AS MIN_HIRE_DT
    , EMPNO, LASTNAME, FIRSTNME
FROM CTE_E
WHERE RANKNO=1
ORDER BY WORKDEPT, EMPNO
;
```

SQL TIP 6 – modern WHERE or ON clause matching

- Has everyone noticed that IBM DB2 now allows a different form of matching? You can match on "sets" of column names.
- This is interesting. And I think I like it. *I want to remind others (i.e. YOU) to consider this syntax as a solution to your future SQL!!*
- One reason I like it… you can have 3 or 5 columns within the brackets and then match over a short two lines of SQL code. It looks obvious to me!
- If you go back to my first SELECT in previous TIP… you can see me using it

```
WITH CTE_A AS (
SELECT WORKDEPT, MIN(HIREDATE) AS MIN_HIRE_DT, COUNT(*) AS EMP_CNT
FROM DSN81210.EMP
GROUP BY WORKDEPT
ORDER BY WORKDEPT
)
SELECT A.WORKDEPT, A.EMP_CNT, A.MIN_HIRE_DT
     , E.EMPNO, E.LASTNAME, E.FIRSTNME
FROM CTE_A A
INNER JOIN DSN81210.EMP E
  ON (A.WORKDEPT, A.MIN_HIRE_DT) = (E.WORKDEPT,E.HIREDATE)
ORDER BY WORKDEPT, E.EMPNO
```

SQL Tip 7 – call procedure in batch via DSNTEP2! (1|2)
> Convert the procedure to a UDF !

- Stored Procedures are powerful and fun
- I have always wanted an easy way to call from BATCH JCL. But easy is subjective. CALL is not allowed in DSNTEP2! (SPUFI neither). Data Studio is possible… but harder
- One can create a (simple) UDF with SQLPL to call the procedure and parse the result up and return one value from the UDF back to the caller
  - You then "call" the procedure via the UDF by SELECT the UDF via SYSDUMMY1.

```
SELECT CURRENT TIMESTAMP AS CUR_TS
, DBCDBD1.UDF_DDF_HOSTNAME() AS UDF_DDF_HOSTNAME
, DBCDBD1.UDF_DB2_DETAILS() AS UDF_DB2_DETAILS
FROM SYSIBM.SYSDUMMY1;
```

- https://www.idug.org/blogs/emil-kotrc1/2021/03/29/how-to-call-a-db2-stored-procedure-in-spufi-or-dsn

https://www.idug.org/blogs/emil-kotrc1/2021/03/29/how-to-call-a-db2-stored-procedure-in-spufi-or-dsn

SQL Tip 7 – call procedure in batch via DSNTEP2! (2|2)
> Convert the procedure to a UDF !

- My first USE case for my first UDF was that I wanted to use SQL to find my Db2 server/hostname. There are tons of fun special registers about Db2 itself. But none for hostname. **BUT, the Db2 command DISPLAY DDF DETAILS shows hostname! So I made a UDF to call the IBM supplied procedure that allows one to run a Db2 command. My UDF parsed the result and returned the hostname**
- I then did the same thing for Db2 command DISPLAY GROUP to find the catalog level and function level and all that fun stuff.
- For real details … see my IDUG content committee blog post!
- https://www.idug.org/blogs/emil-kotrc1/2021/03/29/how-to-call-a-db2-stored-procedure-in-spufi-or-dsn
  - *I was going to show the sample code for how to CREATE a UDF that calls a procedure … it is not complicated… but it is too much for a powerpoint! Goto the IDUG content blog post and copy the sample!*

```
-- find special register details and other fun stuff
select CURRENT TIMESTAMP AS SR_CUR_TS

, DBCDBp1.UDF_DDF_HOSTNAME() AS UDF_DDF_HOSTNAME  -- my UDF
, DBCDBp1.UDF_DB2_DETAILS() AS UDF_DB2_DETAILS    -- my UDF

, GETVARIABLE('SYSIBM.SYSTEM_NAME') AS SYSTEM_NAME
, GETVARIABLE('SYSIBM.SSID') AS SSID
, GETVARIABLE('SYSIBM.VERSION') AS DB2_VERSION  -- version, but no FL!
, GETVARIABLE('SYSIBM.NEWFUN') AS newfun_FOR_COMPILE

, GETVARIABLE('SYSIBM.PACKAGE_SCHEMA') as PACKAGE_SCHEMA_COLL
, GETVARIABLE('SYSIBM.PACKAGE_NAME') as PACKAGE_NAME
, GETVARIABLE('SYSIBM.PACKAGE_VERSION') as PACKAGE_VERSION
, GETVARIABLE('SYSIBM.PLAN_NAME') as plan_name

, CURRENT SERVER AS SR_CUR_SERVER_LOCATION -- actually LOCATION name
, CURRENT APPLICATION COMPATIBILITY AS SR_APPLCOMP  -- for this package
, CURRENT SQLID AS SR_SQLID
, CURRENT SCHEMA AS SR_SCHEMA
, SESSION_USER  AS SR_CUR_USER
, CURRENT CLIENT_WRKSTNNAME AS SR_CL_WRKSTN
, CURRENT CLIENT_ACCTNG AS SR_CL_ACCT
, CURRENT CLIENT_APPLNAME AS SR_CL_APPLNAME
, CURRENT DEGREE AS SR_DEGREE
from sysibm.sysdummy1;
```

https://www.idug.org/blogs/emil-kotrc1/2021/03/29/how-to-call-a-db2-stored-procedure-in-spufi-or-dsn

-- find special register details and other fun stuff
select CURRENT TIMESTAMP AS SR_CUR_TS

, DBCDBp1.UDF_DDF_HOSTNAME() AS UDF_DDF_HOSTNAME -- my UDF
, DBCDBp1.UDF_DB2_DETAILS() AS UDF_DB2_DETAILS   -- my UDF

, GETVARIABLE('SYSIBM.SYSTEM_NAME') AS SYSTEM_NAME
, GETVARIABLE('SYSIBM.SSID') AS SSID
, GETVARIABLE('SYSIBM.VERSION') AS DB2_VERSION -- version, but no FL!
, GETVARIABLE('SYSIBM.NEWFUN') AS newfun_FOR_COMPILE

, GETVARIABLE('SYSIBM.PACKAGE_SCHEMA') as PACKAGE_SCHEMA_COLL
, GETVARIABLE('SYSIBM.PACKAGE_NAME') as PACKAGE_NAME
, GETVARIABLE('SYSIBM.PACKAGE_VERSION') as PACKAGE_VERSION
, GETVARIABLE('SYSIBM.PLAN_NAME') as plan_name

```
, CURRENT SERVER AS SR_CUR_SERVER_LOCATION -- actually LOCATION name
, CURRENT APPLICATION COMPATIBILITY AS SR_APPLCOMP  -- for this package!
, CURRENT SQLID AS SR_SQLID
, CURRENT SCHEMA AS SR_SCHEMA
, SESSION_USER  AS SR_CUR_USER
, CURRENT CLIENT_WRKSTNNAME AS SR_CL_WRKSTN
, CURRENT CLIENT_ACCTNG AS SR_CL_ACCT
, CURRENT CLIENT_APPLNAME AS SR_CL_APPLNAME
, CURRENT DEGREE AS SR_DEGREE
from sysibm.sysdummy1;
```

SQL tip 8 – Db2 has no SQL-PL for dynamic SQL but try DGTT

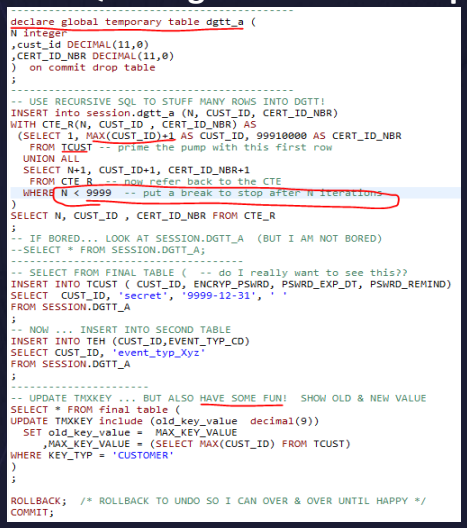
- Db2 Z has no easy SQL-Procedure Language (PL) for ad-hoc SQL-PL in dynamic SQL. Not like some other RDMS (Db2 LUW, SQLserver, Oracle)
- If you want SQL-PL such as looping and IF logic in your SQL script then you must stuff it into a UDF or procedure and then call it.
- BUT, as an alternative. I have found Declared Global Temporary Tables are interesting alternative./workaround You can stuff the DGTT with data (use SQL INSERT, LOAD utility does not work with DGTT) and then reference the DGTT. Then you can use SELECT and INSERT or MERGE or whatever with the DGTT and do lots of stuff at once. Use SELECT FROM FINAL TABLE to get answers of what was done.
- The following IDUG content blog discusses DGTT. Not as an alternative to looping or IF (future blog)
- https://www.idug.org/blogs/brian-laube1/2020/11/04/temporary-tables-in-db2-along-with-a-painful-lesson

SQL tip 8b – example of Db2 SQL using DGTT to not loop

I had a SQLSERVER person give me a SQL script to run against Db2. They tried to use SQLPL and variables and loops.

Basically, they wanted to INSERT many (9999) NEW related rows into a few application tables. Starting with some max ID value. And then at the end update the record table for ID.

It seemed obvious to them… but it was going to be complicated with Db2 SQL until we used a DGTT.



```
-------------------------------------------
declare global temporary table dgtt_a (
N integer
,cust_id DECIMAL(11,0)
,CERT_ID_NBR DECIMAL(11,0)
)  on commit drop table
;
-------------------------------------------
-- USE RECURSIVE SQL TO STUFF MANY ROWS INTO DGTT!
INSERT into session.dgtt_a (N, CUST_ID, CERT_ID_NBR)
WITH CTE_R(N, CUST_ID , CERT_ID_NBR) AS
 (SELECT 1, MAX(CUST_ID)+1 AS CUST_ID, 99910000 AS CERT_ID_NBR
   FROM TCUST -- prime the pump with this first row
  UNION ALL
  SELECT N+1, CUST_ID+1, CERT_ID_NBR+1
   FROM CTE_R  -- now refer back to the CTE
  WHERE N < 9999  -- put a break to stop after N iterations
)
SELECT N, CUST_ID , CERT_ID_NBR FROM CTE_R
;
-- IF BORED... LOOK AT SESSION.DGTT_A  (BUT I AM NOT BORED)
--SELECT * FROM SESSION.DGTT_A;
-----------------------------------
-- SELECT FROM FINAL TABLE (  -- do I really want to see this??
INSERT INTO TCUST ( CUST_ID, ENCRYP_PSWRD, PSWRD_EXP_DT, PSWRD_REMIND)
SELECT  CUST_ID, 'secret', '9999-12-31', ' '
FROM SESSION.DGTT_A
;
-- NOW ... INSERT INTO SECOND TABLE
INSERT INTO TEH (CUST_ID,EVENT_TYP_CD)
SELECT CUST_ID, 'event_typ_Xyz'
FROM SESSION.DGTT_A
;
---------------------
-- UPDATE TMXKEY ... BUT ALSO HAVE SOME FUN!  SHOW OLD & NEW VALUE
SELECT * FROM final table (
UPDATE TMXKEY include (old_key_value  decimal(9))
  SET old_key_value =  MAX_KEY_VALUE
     ,MAX_KEY_VALUE = (SELECT MAX(CUST_ID) FROM TCUST)
WHERE KEY_TYP = 'CUSTOMER'
)
;

ROLLBACK;  /* ROLLBACK TO UNDO SO I CAN OVER & OVER UNTIL HAPPY */
COMMIT;
```


```
-------------------------------------------
declare global temporary table dgtt_a (
N integer
,cust_id DECIMAL(11,0)
,CERT_ID_NBR DECIMAL(11,0)
)  on commit drop table
;
-------------------------------------------
-- USE RECURSIVE SQL TO STUFF MANY ROWS INTO DGTT!
INSERT into session.dgtt_a (N, CUST_ID, CERT_ID_NBR)
WITH CTE_R(N, CUST_ID , CERT_ID_NBR) AS
 (SELECT 1, MAX(CUST_ID)+1 AS CUST_ID, 99910000 AS CERT_ID_NBR
   FROM TCUST -- prime the pump with this first row
  UNION ALL
  SELECT N+1, CUST_ID+1, CERT_ID_NBR+1
   FROM CTE_R  -- now refer back to the CTE
  WHERE N < 9999  -- put a break to stop after N iterations
)
SELECT N, CUST_ID , CERT_ID_NBR FROM CTE_R
```

```
;
-- IF BORED... LOOK AT SESSION.DGTT_A  (BUT I AM NOT BORED)
--SELECT * FROM SESSION.DGTT_A;
-------------------------------------
-- SELECT FROM FINAL TABLE (  -- do I really want to see this??
INSERT INTO TCUST ( CUST_ID, ENCRYP_PSWRD, PSWRD_EXP_DT, PSWRD_REMIND)
SELECT  CUST_ID, 'secret', '9999-12-31', ' '
FROM SESSION.DGTT_A
;
-- NOW ... INSERT INTO SECOND TABLE
INSERT INTO TEH (CUST_ID,EVENT_TYP_CD)
SELECT CUST_ID, 'event_typ_Xyz'
FROM SESSION.DGTT_A
;
----------------------
-- UPDATE TMXKEY ... BUT ALSO HAVE SOME FUN!  SHOW OLD & NEW VALUE
SELECT * FROM final table (
UPDATE TMXKEY include (old_key_value  decimal(9))
  SET old_key_value =  MAX_KEY_VALUE
     ,MAX_KEY_VALUE = (SELECT MAX(CUST_ID) FROM TCUST)
WHERE KEY_TYP = 'CUSTOMER'
)
;

ROLLBACK;  /* ROLLBACK TO UNDO SO I CAN OVER & OVER UNTIL HAPPY */
COMMIT;
```

SQL tip 8c – how to stuff lots of data into DGTT?

I had another ZAP where we had to stuff thousands of rows into a DGTT before doing "stuff" with the DGTT later in the SQL script. The raw data came from someone's EXCEL.

It seemed easiest to use a DGTT for the subsequent processing. We transformed the EXCEL data into hundreds of SQL INSERT into DGTT before the main script processing using the DGTT.

- It turns out that executing thousands of simple SQL INSERT of 1 row (using VALUES) into a DGTT using Data Studio is annoying slow (all the network back and forth).
- First lesson, was running thousand simple SQL (of any type) is WAY faster via DSNTEP2.
- But do not forget, Data Studio is pretty and colourful and "easy" to develop SQL
  - smart-insert and completing the column names & easy "visual explain"
- But Data Studio is slow(ish) for long scripts of many statements (and besides, it is not good for official record keeping > we JCL and oneshot JOBS for official tracking)

We changed the 1000s of inserts into 2 or 3 insert where we used SELECT from SYSDUMMY1 and UNION ALL to put many rows into the DGTT all at once. Much more efficient.

- **Do I have to say it? If I see simple "UNION" then I always ask why? Why not "UNION ALL"? UNION ALL implies you know the "concatenated result sets" do not contain duplicates. If you know this is true (no duplicates) then use UNION ALL. Only use simple "UNION" if you really think duplicate rows might exist and you want to eliminate the extras!**
- The alternative to the DGTT (for this zap) was to CREATE a real table and LOAD up the EXCEL contents. But I don't like to CREATE these type of ad-hoc tables (I forget to drop them sometimes).

```
--------------------------------------------------------
DECLARE GLOBAL TEMPORARY TABLE SESSION.DGTTX
  (
  POLICYID CHAR(7)
 ,CVG_ID CHAR(3)
  )
ON COMMIT DROP TABLE;
--
INSERT INTO SESSION.DGTTX
SELECT '0111634','000'FROM SYSIBM.SYSDUMMY1 UNION ALL
SELECT '0111634','001'FROM SYSIBM.SYSDUMMY1 UNION ALL
SELECT '0080298','001'FROM SYSIBM.SYSDUMMY1 UNION ALL
SELECT '0080298','004'FROM SYSIBM.SYSDUMMY1 UNION ALL
SELECT '0113348','001'FROM SYSIBM.SYSDUMMY1 UNION ALL
SELECT '0120421','001'FROM SYSIBM.SYSDUMMY1 UNION ALL
SELECT '0106584','000'FROM SYSIBM.SYSDUMMY1 UNION ALL
SELECT '0118608','001'FROM SYSIBM.SYSDUMMY1 UNION ALL
SELECT '0118779','001'FROM SYSIBM.SYSDUMMY1 UNION ALL
SELECT '0118779','002'FROM SYSIBM.SYSDUMMY1 UNION ALL
SELECT '0118779','003'FROM SYSIBM.SYSDUMMY1 UNION ALL
SELECT '0118779','004'FROM SYSIBM.SYSDUMMY1 UNION ALL
SELECT '0118779','005'FROM SYSIBM.SYSDUMMY1 UNION ALL
SELECT '0118779','006'FROM SYSIBM.SYSDUMMY1 UNION ALL
SELECT '0118779','007'FROM SYSIBM.SYSDUMMY1 UNION ALL
SELECT '0118779','008'FROM SYSIBM.SYSDUMMY1 UNION ALL
SELECT '0115290','001'FROM SYSIBM.SYSDUMMY1 UNION ALL
SELECT '0115290','002'FROM SYSIBM.SYSDUMMY1 UNION ALL
SELECT '0120889','001'FROM SYSIBM.SYSDUMMY1 UNION ALL
SELECT '0068640','004'FROM SYSIBM.SYSDUMMY1 UNION ALL
SELECT '0068640','005'FROM SYSIBM.SYSDUMMY1 UNION ALL
SELECT '0068640','006'FROM SYSIBM.SYSDUMMY1 UNION ALL
SELECT '0068640','007'FROM SYSIBM.SYSDUMMY1 UNION ALL
SELECT '0121857','001'FROM SYSIBM.SYSDUMMY1 UNION ALL
SELECT '0100017','001'FROM SYSIBM.SYSDUMMY1 UNION ALL
SELECT '0100017','002'FROM SYSIBM.SYSDUMMY1 UNION ALL
SELECT '0100017','004'FROM SYSIBM.SYSDUMMY1 UNION ALL
SELECT '0100017','005'FROM SYSIBM.SYSDUMMY1 UNION ALL
SELECT '0100017','006'FROM SYSIBM.SYSDUMMY1 UNION ALL
SELECT '0100017','007'FROM SYSIBM.SYSDUMMY1 UNION ALL
SELECT '0100017','008'FROM SYSIBM.SYSDUMMY1 UNION ALL
SELECT '0113652','001'FROM SYSIBM.SYSDUMMY1 UNION ALL
SELECT '0113652','002'FROM SYSIBM.SYSDUMMY1 UNION ALL
SELECT '0113652','003'FROM SYSIBM.SYSDUMMY1 UNION ALL
SELECT '0113652','004'FROM SYSIBM.SYSDUMMY1 UNION ALL
SELECT '0116870','001'FROM SYSIBM.SYSDUMMY1 UNION ALL
SELECT '0116870','002'FROM SYSIBM.SYSDUMMY1 UNION ALL
SELECT '0116870','003'FROM SYSIBM.SYSDUMMY1 UNION ALL
```

The slide says all I have to say

Stop saying BIND when you mean REBIND (1|3)

- A program (COBOL) with static SQL inside is compiled (no one really pre-compiles anymore) to produce two outputs. The LOAD module and the DBRM.
  - The DBRM contains a copy of all the static SQL from the program.
- You BIND the DBRM to create a package.
  - The package is the executable instructions (access path) of that SQL in that particular schema (QUALIFIER… database).
  - This executable access path decision is made at that BIND TIME based upon current statistics.
- When you run the LOAD module via a plan, it finds the package. At run time Db2 makes sure the contoken (timestamp) inside the LOAD module matches the contoken of the package!
  - If they don't match then you get everyone's favorite SQLCODE … -805

WE should all know this basic stuff… but let me say it again! For my audience (not experienced DBAs)

Stop saying BIND when you mean REBIND (2|3)

- When the program needs to run in another database/schema/table-qualifier then you BIND the same DBRM against this other database/schema/table-qualifier.
  - The resultant package might, but not necessarily, have the same access path!
    - Even if the access path is different. The produced “result set” is always correct in every db!
    - If the access path is different then performance could be “different” but that is a whole other issue.
- And so on… you move the program up to other “environments” and you must BIND this DBRM

Stop saying BIND when you mean REBIND (3|3)

- When do you REBIND?
- You do a REBIND when something external to the program changes.
- The program is not changed (hence, the DBRM did not change).
- You use REBIND when :
  - the Db2 software itself was upgraded (smarter optimizer now?)
  - the Db2 objects have vastly different statistics today which could influence the optimizer to pick a whole new access path.
  - you added new indexes so new choices exist for the optimizer!
- To be explicitly clear. REBIND does not read the DBRM again. The package already knows the SQL statements (inside the catalog already!).
- ***My main point… don't willy nilly say BIND when you mean REBIND or vice-vesa***

Are you BINDING or REBIND because of -805?

- In general, REBIND does not help with -805
  - The reported contoken in the -805 SQLCA is from the LOAD module. You could be a keener and compare and double check with the package you think you should be using! If they matched then you would not be getting -805!
  - If were keen, you could read the -805 SQLCA for the common reason code of 02 or 03… and then think. 02 means the package was NOT found in plan list of packages. 03 means it was found but the package contoken did not match. What does this mean to YOUR situation? Ask yourself and answer it!
- Having said all that. If you don't want to think too much about -805. Just try a quick BIND. Just be sure to pick up the DBRM that matches the LOAD module that you are using. If all your ducks now line-up.. You should be able to use the package!

Application development tip #2 – REST … the real future!

- IBM has been pushing Db2 REST for several years. They have convinced me that REST is the future of application development.
- If you are not using REST yet, then try to find some time to make your own use case and prove to yourself how easy it is use. Especially by developers who know nothing about Db2.
- A Db2 REST service API can be a single SQL statement… and that statement can be "CALL" so you can call a procedure and do something complex! Fun
- The Db2 REST service API does not require any type of IBM client software on your computer/laptop/server. This is pretty handy!
  - Anyone who can see the mainframe on the network can invoke the service
- You can even use the Db2 SECPORT and secure network encryption from most tools on your 'computer'. No certificates required. Again , very easy.
- IBM has provided a Db2 REST service to create other Db2 REST services. This may be useful for dev ops and pipelines and what not. But to be honest, I use the BIND SERVICE Db2 command
  - My first REST service calls a procedure which does something useful for me. My procedure calls the procedure to issue a Db2 COMMAND of DISPLAY DDF DETAIL. The procedure parses up the command output and returns the outputs to the REST service. And then the computer program (browser) shows a pretty graph of DDF info every N minutes. (more next year)

ENCRYPTION – security – permission – authority (1|2)

- Encryption is a loaded word.
  - Encryption is not a substitute for permission to look at something. If you have permission, then encryption does not matter (and should not make it hard to look).
  - Encryption is protection from someone who does not have permission!
- We encrypt our data at rest. But this can done on both the disk itself and the dataset on the disk.
  1. Encrypting the disk is like encrypting your laptop disk. Easy. And we all do it! It protects the data from someone yanking the disk drive out of the box and running away and looking at via a new computer. The data would be encrypted and hard to read! Good… but this encryption of data at rest protects from one specific worry!
  2. In Db2 V12 FL505, IBM introduced encryption of tablespace datasets. This is another layer of encryption for data at rest. But all it really protects is someone who has permission to look at the outside of the dataset from looking inside the dataset. So the storage team could move datasets around but they can't look inside!
- So when the auditor asks if data at rest is encrypted… what do you say?

Say yes

ENCRYPTION – security – permission – authority (2|2)

- We can turn on the Db2 SECPORT and allow secure encrypted network connections to our Db2. Remember, this is inside our already secure network (VPN and whatnot – most mainframe Db2 are not visible via the internet). This helps protect from malicious people inside the network sniffing the network traffic and finding userids and passwords… or watch sensitive customer data go between Db2 and the client computers. An important risk…
- We can force the use of the encrypted network connection by setting ZPARM of TCPALVER to value SERVER_ENCRYPT.
  - But, think of all the client computers/laptops… all need to be updated! So much work…
- Data in memory Encryption! Even with dataset encryption… the data is unencrypted in memory and in the clear. In the case of someone peeking at the memory… or a DUMP occurs and people need to look at the DUMP (these people do not need, do not care, about the application data inside the DUMP)…. then in order to ensure this important data is encrypted… we should use encryption functions! The data remains encrypted in memory and in the application until explicitly decrypted!
- There is a good recap on Db2 Z security in the IDUG content blog (not from me!)
- https://www.idug.org/blogs/emil-kotrc1/2021/03/19/introduction-to-db2-for-zos-security

That is enough of my brain dump of what I think is important (this year). Let me know what you think!