The IDUG Solutions Journal
Spring 2000 - Volume 7, Number 1

Tidbit

You Said What?

by Willie Favero

While watching ER recently on TV, a family member commented on the "foreign" language doctors and nurses spoke in the emergency room. It sounded like meaningless garble. My daughter quickly pointed out to me that I must not listen to myself talk very often. She is so correct. Our industry - and many others - are terrible about "shortening" words and expressions into acronyms, abbreviations, and initials. We speak in strange groupings of vowels and consonants just like those ER doctors. So, for this TidBit column, I'm going to examine "our language" - how we have gone from English (or should it be US American) to IT talk. Gee! How many acronyms is that in one paragraph? ER (Emergency Room), TV (television), US (United States), and IT (Information Technology).

We have become accustomed to using so many phrases that are made from non-words. It makes it difficult, to say the least, to converse with people outside our profession. Even sadder is how difficult it can be to talk with folks in our profession. We always assume that everyone else knows exactly what we are saying and what we mean, even when the sentence is full of acronyms. After all, we are all in the same business. But are we really? The breadth of an individual's knowledge of our secret language differs greatly among IT professionals. Recently, I commented that a product was going GA. Across the meeting table someone asked, "What is a GA?" Generally Available. Everyone knows that! Except, of course, for the person with whom I was meeting. How often do we lose a person (or persons) we are talking with because we decide to use an acronym; I mean abbreviation, I mean initials. Which one is it anyway?

Acronyms, abbreviations, and initials! Do they actually have different meanings? Do you use them correctly or simply flip-flop among them, using all three interchangeably? According to the dictionary (see Figure 1), an abbreviation is a shortened version of a word or phrase. Dr. and Tues. are examples of abbreviations for doctor and Tuesday.

Acronyms, on the other hand, are words formed by the initial letters or group of letters from a compound term. DBRM and IMS are acronyms for database request module and Information Management Systems. Of course, IMS could also be initials. So which is DB? It represents the word database. You figure it out.

How often do you use an acronym and have no idea what it means? How often are you caught? It seems that I am caught more often. Either my audiences are getting better about stopping me to ask or I am getting lazier about looking things up before I use them. Of course, it could be both. Sometimes you know what the acronym means, you just don't know what each letter stands for. Look at terms like CSA (common service area) and ECSA (extended common service area) that are thrown around everyday in DB2 land. (Did you know there is a commonly used acronym, TLA, which actually describes acronyms? TLA stands for "three letter acronym.")

Perhaps over time, it won't even matter what the individual letters stand for. When was the last time you heard someone refer to a DBA as a database administrator or use the phrases "International Business Machines," "Database 2", or "Boulett, Moores, and Cloer"? That last one is a great example. Even looking at the words some will still not know to what it refers. However, if you use only the abbreviation, most of you will quickly recognize the name of my employer, BMC! DR. Da is a great word made from the acronym DRDA (Distributed Relational Data Architecture). We just mentioned IBM. Isn't it IBM Corporation? So why don't we abbreviate it IBMC? As for its meaning, I have heard many. There is, of course, International Business Machines, the one everyone knows. Other favorites include Itty Bitty Machine Company and I've Been Moved. There are a few more, but they may not be appropriate for this article.

Another great acronym is MIPS. Most think it stands for Million of Instructions Per Second, a method of categorizing the speed of a computer. However, if you've been in this business for a while, you probably know that a better description for MIPS is Meaningless Indication of Processor Speed. There are still other acronyms that some of us use every day, whose meanings might occasionally escape us, sometimes intentionally, because the acronym does a much nicer job of describing what we are trying to do. Take the next two acronyms, for example: APARs and PTFs. An APAR is an authorized program analysis report, a method of reporting a code problem or error to IBM. A PTF, or program temporary fix, is the actual code change that fixes the problem. Usually there is not much "temporary" about it. Once installed, it is there to stay, usually forever, unless of course, the code fix is in fact broken. Then we need to put the PTF, or fix, in PE status and develop another PTF to fix the first PTF. Believe it or not, this can occur so many times that we end up with a PE Chain, or list of PTFs needed to fix the first PTF. Confused yet? Perhaps now you have a little more sympathy for your systems programmer.

Some of the words used to make up acronyms will probably disappear in the future. These newly invented words just "work" too well. Let's look at some new terms like LOBs, BLOBs and CLOBs, all pronounced just as they look. It's inevitable! LOB, BLOB, and CLOB are just so much easier to say than Large Objects, Binary Large Objects, and

Character Large Objects. DBCLOBs, pronounced "db-clobs", is the acronym for Double Byte Character Large Objects, a case where the acronym is far easier to say than the actual word. The problem here is that the words we have formed from the acronyms sound so "cool,"everyone will use them and eventually no one will use the actual words again. We all saw this happen with Structured Query Language. Yes, SQL! No one would even think of using the words for fear they might sound like newbees to the relational database world. However, most folks think nothing of dropping the S-Q-L acronym in favor of the more popular "sequel" pronunciation. Some of them probably have no idea there is actually a company in Silicon Valley that deals in computers named Sequel.

What about words, I mean acronyms, which suffer from mistaken identities? A BMP is a batch message-processing program if you grew up with IMS. Of course, everyone who grew up working with Windows on their home computer will know it as a binary map image. This one also falls into the category of Why Isn't It? Why is it BMP and BMI? One of my personal favorites, mostly because of where I once resided and the group I dealt with at IBM, is STL. If you use STL in my presence, I will immediately assume Santa Teresa Lab. STL is IBM's internal designation for the home of DB2 for OS/390. However, many, especially those who travel in the Midwest, will think it means St. Louis, MO. And others, Windows experts (not me by any means), might assume it means "certificate trust list." Another personal favorite is CAF- call attach facility. That is unless you are part of the Favero family. Around my house, CAF stands for only one thing, Catherine Ann Favero. Catherine is my 12-year-old daughter. Yes, that really is her name! No, we did not plan it that way, it just happened. There are other more subtle examples. PPT can be the processing program table to a CICS programmer and the program properties table to an MVS systems programmer. BTW (that stands for By The Way in e-mail and chat room talk), what ever happened to MVS? One day I was running DB2 under MVS, and then the next day, DB2 was suddenly running under OS/390. I hardly noticed the difference.

There are so many examples. CEC is a central electronic complex, which, as we all know, is the same as a CPC, central processing complex, which should not to be confused with a CPC, conference planning committee (that group of people that puts together IDUG every year). And how do you use an ATM? Is it a machine (automated teller machine) that allows you to withdraw money from your bank account whenever you need it? On the other hand, is it a dedicated-connection switching technology (asynchronous transfer mode) that can be an integral part of a high-speed network?

CICS is one acronym that everyone loves to make into a word. At one point, only those technicians for Great Britain referred to CICS with the word "kicks." The rest of us tried to stick with just the acronym, but it was futile. The popular pronunciation "kicks" has won. We have done so well at calling it "kicks" that I had to make sure I had the correct words - Customer Information Control System! Another one that has had its day in the sun is IRLM. IRLM started out as an acronym within an acronym, IMS Resource Lock Manager. Then the meaning of the words that made up the acronym actually changed. One day IRLM suddenly stood for Internal Resource Lock Manager. You do not see that happen

very often.

Some acronyms end up being duplicated in the same product. In DB2, we spent years referring to the database allocation table as DBAT. One day, I read some documentation on the new distributed features of DB2, which mentioned DBATs. At the time, the context of how the acronym was being used did not make sense. Turned out that DBAT in the distributed environment meant database access thread.

Loosely following that same thread, how can we forget SPUFI, or SQL Processor Using File Input? Come on, who could have dreamed up a process using those words and then create such a great acronym? It almost sounds like the acronym was created first - then they found some words to match the acronym. Of course, no one would ever do that. Would they? Years ago, I heard a rumor that SPUFI really stood for System Programmer User Friendly Interface. Then someone realized that no self-respecting system programmer would be caught dead relying on panels to do their job. Now they had this great acronym and nowhere to use it. Might be true, might not! We will never know.

Not only acronyms but whole words are invented for our computer language. Once we come up with an acronym, we have to make that acronym into a word. DB2 is loaded with cute little words that roll off our lips without the slightest regard for what they mean. Take for example "deckle gen" or DCLGEN. It refers to declaration generator. When was the last time you heard anyone in DB2 land use that phrase? And there are oldies but goodies like Eb-sa-dic (EBCDIC) and as-key (ASCII). I'm pretty sure EBCDIC stands for extended binary-coded decimal interchange code and ASCII is the acronym for American Standard Code for Information Interchange. You never know though - I could be mistaken. I cannot even remember the last time I actually referred to these things by their full names.

Let's examine a few commonly used abbreviations/acronyms and determine if we really need them. Is it better to say DBRM or Database Request Module? It may take longer to type, but DBRM is no faster to say than database request module. In addition, if you were speaking with someone new to DB2, you might keep that person on track. On the other extreme, some acronyms and their associated words have simply outlived their usefulness. One that comes to mind immediately is DASD (direct access storage device). Why can't we just call them disk like the rest of the world?

How many of you have a time sharing option on resort property and use it to enter information into MVS or OS/390 or whatever it is this week. That option is, of course, TSO.

Have you tried figuring out DSN yet? You see it enough. It is the first three letters of every module distributed with the DB2 product. It is the name given to the TSO command processor and the DB2 TSO, or batch, attachment. Give up? It doesn't mean a thing. It is just an arbitrary string of three characters picked by someone at IBM to designate DB2. It is no different from DFS and DCS for IMS and CICS.

Then there is DB2 UDB!
The U is for Universal and DB is for Database. Actually, when you look back on history this one may even make some sense. Although we spell database as one word now (this was a major debate in the old days), we still abbreviate it as if it was two. A technical editor who used to review my articles would always change database to data base (two words rather than one). Therefore, some checking seemed in order. It turned out that back in the DB2 Version 1.1 announcement materials, other than when it was used as part of a product name, all references to database were two words. At some time, not sure when, the two words merged into one. Then of course, the opposite happened to table space. Many documents I review, even today, still carry references to tablespace (a single word). This is completely accurate for commands and SQL statements. However, somewhere around DB2 Version 1.3, the single word tablespace became two words, just the opposite of database.

Sometimes an acronym can come back to haunt you. We used to joke that IBM had acronym police to make sure you did not call a product something that might end up sounding inappropriate if it were turned into an acronym. Years ago, we used to refer to something internally called System Application Development. That term later became commonly known as AD/Cycle. Just think what a great acronym it (SAD) would have made if the process had not been changed. Then there is the near to hearts DB2 Universal Database. Using the current method of referring to things by their initials, we might have ended up with the acronym DUD. Ouch!

We are all teachers every day of our lives. You do not need to speak at a user group meeting or conference or stand in a classroom to teach others. Every time you answer a question of one of your co-workers or reply to a message on the DB2-L listserver, you are taking the role of a teacher. As a teacher, you must ensure that your "pupils" understand your answer. One way you can help ensure that is to explain all those little letters you use. You will be surprised at how often you will be thanked for the extra explanation.

So remember SUWIAP, pronounced "sue-ip" of course! Or "Start Using Words Instead of Acronyms."

Back to Text

Figure 1: The actual definitions from Merriam-Webster Dictionary
According to Merriam-Webster Dictionary on the Internet:

Acronym - a word (as NATO, radar, or snafu) formed from the initial letter or letters of each of the successive parts or major parts of a compound term

Abbreviation - a shortened form of a written word or phrase used in place of the whole; amt is an abbreviation for amount

Initials - the first letter of a name; or the plural: the first letter of each word in a full name

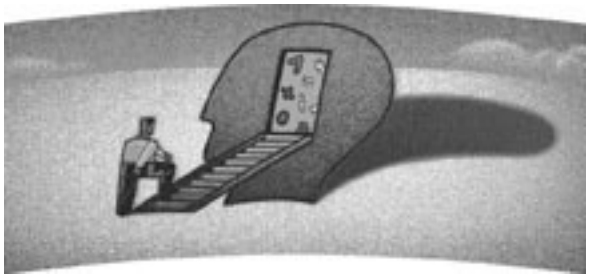
ABOUT THE AUTHOR

Willie Favero has been a database professional for more than 20 years, the last 14 years primarily with DB2. He is a software consultant with BMC Software, and a senior DB2 instructor for IBM. He can be reached via email at wfavero@ibm.net.

International DB2 Users Group
401 N. Michigan Ave.
Chicago, IL 60611
(312) 644-6610

The IDUG Solutions Journal
Spring 2000 - Volume 7, Number 1

Developer's Corner



Storage and DB2 UDB for OS/390

by Roger Miller

Storage has many meanings, including processor memory and disk storage. The storage hierarchy is changing very rapidly. Some old problems are being resolved, but other new problems occur at the new scale. Two gigabytes of real and virtual memory seemed infinitely large as we moved from 24 bits to 31 bits, but now many customers find that two gigabytes are not enough. If you have large buffer pools, dynamic statement caching or large numbers of concurrent users, you will want to read Mary Petras' article in this issue about managing the virtual storage size.

Today, storage hierarchy consists of virtual buffer pools, buffer pools in data spaces, hiperpools, global buffer pools in the Coupling Facility, then storage controller caches and data on the disk. Sometimes, the difference in speed between some layers is a small factor, but sometimes it can scale up to more than a thousand times. Balancing the storage hierarchy requires careful work in order to understand the work loads, storage sizes, and relative speeds.

Today's disk subsystems are more complex than the computer systems of only a few years ago. Our applications demand more space, better performance, and greater availability. Fiber connections and channels now deliver data rates of 100 megabytes per second. Disk units use much larger caches, more advanced controllers, RAID techniques and parallelism to deliver higher performance. The disks have higher density and much larger capacities. Cache controllers are making disks more and more virtual.

The latest IBM disk is called the Enterprise Storage Server (ESS), code named "Shark."

I'm happy to report that ESS performance is outstanding. We have had early units in our DB2 lab for some time; details of our results have been published in the ESS Performance White Paper, which you can find on the web: See the DB2 for OS/390

home page or the red books page for the red paper. The ability to have parallel access to a single volume provides a substantial improvement in performance for concurrent access, a reduction in IOS queuing, and a reduction in the amount of management time needed. Parallel Access Volumes are the next step in making disk more virtual. See www.storage.ibm.com for more on this topic.

We are seeing compound growth rates for disk space at 89 percent per year! If we started with a "middle sized" one terabyte back in 1999, then by the year 2004, we will be managing 24 terabytes. If your database is larger than one terabyte now -- or it's growing faster than the average, you will have even more to manage. This kind of growth suggests that the current management must change. We MUST automate. We MUST organize by groups and exceptions. We need to use System Managed Storage and take advantage of the virtual disk.

We're working hard on the next enhancements and plan to describe Version 7 changes at IDUG's May conference in Dallas. See you there.

Meanwhile, please see our red books, *Storage Management with DB2 for OS/390* (SG24-5462) and *DB2 UDB for OS/390 and Continuous Availability* (SG24-5486), for additional details.

ABOUT THE AUTHOR

Roger Miller is a lead DB2 strategist at IBM's Santa Teresa Laboratory. He has spent the last 17 years working on product strategy, design, and development.

About IDUG | Conferences | DB2 Resources | Regional User Groups |Solutions Journal | Vendor Directory

Home | Contact Us

International DB2 Users Group
401 N. Michigan Ave.
Chicago, IL 60611
(312) 644-6610

The IDUG Solutions Journal
Spring 2000 - Volume 7, Number 1

THE INTELLIGENT OPTIMIZER

by Dr. Pat Selinger

In this column, I've been covering some of the DB2 Universal Database Version 6 features. IBM has been investing in new and enhanced optimization features for more than 25 years. This series of columns on Version 6 features will provide a detailed description of some particularly interesting enhancements for Version 6 across several platforms.

In the last issue, I discussed one feature that has generated some DBA interest as well as some comments from the competition -- DB2 UDB for OS/390's so-called optimizer hints capability, added in Version 6. We talked about why DB2 added this feature and how to use it, and I gave you my view that DBAs will find this capability most useful as a plan preserving mechanism when service is applied or when migrating to new releases of DB2.

In this column, I will cover the Index Advisor feature of db2advis, which provides a set of recommended indexes for a DB2 query workload. I'll give you details on how to use this tool and some advice about what to do (and more importantly what NOT to do) to maximize the value of this capability.

What the Index Advisor does

The Index Advisor makes it easier for DBAs to identify the best indexes for one or more queries and tables with less time and resources. Why would you use the Index Advisor?

1. You want to examine a single query to figure out the best indexes to improve its performance. Many experienced DBAs can do this without using a tool, but when the query becomes very long and complex, it's good to have tools that can handle the complexity and the large number of possible combinations of columns.

2. You want to receive recommendations of the best indexes for a large set of queries and take into account the cost of updating, the size of the indexes and take advantage of cost-benefit tradeoffs in a formal, repeatable way. In this case, the goal of the advising engine is to identify a set of indexes that minimizes the total workload cost.

3. You want to evaluate whether a specific set of additional indexes will improve performance on a query or workload without actually having to take the time and resources to create those indexes.

The Index Advisor, and the Index SmartGuide user interface that it works with, leads DBAs through the process of selecting the best indexes for a given query workload. The steps are:

1. Gathering the workload

2. Running the Index Advisor, db2advis, to get recommendations

3. Determining the constraints on the index selection process

4. Choosing which of the recommended indexes to create.

The Index Advisor was built to run with DB2 UDB for Unix, Windows, and OS/2. Initial experiments are underway to evaluate how well this same Index Advisor does at recommending indexes for DB2 for OS/390 workloads. Preliminary results look promising, but for the purposes of this column, I'll cover what the Index Advisor does and how it works with DB2 UDB for Unix, Windows, and OS/2. Let's go through these steps, one at a time.

Gathering the workload

The Index Advisor takes as input one or more SQL statements (selects, updates, inserts, deletes) on one or more tables, along with their frequencies. You can type all these SQL statements into a user input file or into an explain table called the ADVISE_WORKLOAD table, which is created when you run the EXPLAIN.DDL script found in the misc subdirectory of the sqllib subdirectory. Alternatively, you can use the Index Advisor SmartGuide to help you gather the workload. If you do that, the user interface offers you the opportunity to fill in ADVISE_WORKLOAD (WORKLOAD_NAME, STATEMENT_TEXT, FREQUENCY, WEIGHT) from the following sources:

- DB2 get snapshot monitor commands to capture the SQL statements currently in the dynamic statement cache (plus their frequencies),

- Selected SQL statements from the static SQL used in one or more recently used packages, obtained from the SYSCAT.STATEMENTS catalog view.

- Additions or deletions you manually make to this list by INSERTing or DELETEing ADVISE_WORKLOAD rows.

It's best to run the Index Advisor for multiple tables at one time. That way, you can include join queries and make tradeoffs between indexes on different tables. For example, in a join query it might be beneficial to have an index on the join column for one table or on the join column for the other table, but having both indexes may not be necessary. By considering many tables and queries at once, you will use the Index Advisor to determine whether your entire query workload would benefit from having an index on either one or the other join columns rather than both.

Running the Index Advisor, db2advis, to get recommendations

What we have described above is the SQL statement workload input to the Index Advisor. The advising engine uses this workload information in conjunction with the database information to recommend indexes. This is all the input that is needed when the Index Advisor is run with the Index Advisor SmartGuide in RECOMMEND INDEXES mode. The recommended indexes from this invocation will be placed in another explain table called ADVISE_INDEX, which has 43 columns, most of which are identical to columns in SYSCAT.INDEXES, such as COLCOUNT, NLEAF, NLEVELS, FIRST2KEYCARD, CLUSTERRATIO. The advising engine uses the ADVISE_INDEX table to put its current working set of recommended indexes in while it is examining multiple SQL statements in the query workload (contained in the ADVISE_WORKLOAD table.) ADVISE_INDEX also includes a column called CREATION_TEXT, which will contain the CREATE INDEX statement for any recommended indexes.

In addition to the RECOMMEND INDEXES mode, the Index Advisor can also be invoked in EVALUATE INDEXES mode. This means that you have set the CURRENT EXPLAIN MODE special register to EVALUATE INDEXES. This mode is available for invocations of db2advis on the command line. In this mode, it accepts as input a set of candidate indexes in the ADVISE_INDEX table. If you insert the description of one or more candidate indexes that do not currently exist into the table ADVISE_INDEX, you must set the column USE_INDEX="Y" for those indexes. Then the advising engine will examine those indexes as candidates and will determine if they benefit the dynamic SQL statements executed while that special register is set to EVALUATE INDEXES.

To summarize, you can either run the Index Advisor in RECOMMEND INDEXES mode (the default) or in EVALUATE INDEXES mode set by special register. In the latter case, the advising engine will evaluate the candidate indexes you inserted into ADVISE_INDEX for all SQL statements executed dynamically while the special register is set. In the RECOMMEND INDEXES case, you start with no suggested candidate indexes and allow the Index Advisor to provide all the recommendations for the workload of SQL statements provided to it, either in the command line of db2advis or in the ADVISE_WORKLOAD table.

Determining the constraints on the index selection process

The Index Advisor works by iterating through candidate indexes, evaluating the cost/benefit of each index for the workload of SQL statements. In EVALUATE INDEXES mode, it uses the candidate indexes you provide.

In RECOMMEND INDEXES mode, it constructs a set of candidate indexes from the columns that are either selected, or participate in certain local and join predicates, or in GROUP BY or ORDER BY clauses in the current SQL statement. The advising engine combines these columns into sets of multi-column indexes and then evaluates whether these indexes provide benefit to the workload of SQL statements. As each SQL statement in the workload is examined, it may contribute more index candidates to the solution set being evaluated. Because the benefit an index contributes is dependent on the other indexes being recommended, the Index Advisor ranks the indexes, by their current benefit/cost ratio and takes the top set of indexes. Then it selects a few of these recommended indexes, replaces them with other candidate indexes, and runs through the workload again, reevaluating the benefit/cost of this new set of indexes. If this new set is cheaper overall for the whole workload, it becomes the new set of recommended indexes. This iterates until the time limit for running the Index Advisor is reached.

You provide constraints on the operation of the Index Advisor in two ways:

1. You limit the total amount of space you want to use for indexes. A small amount of space will significantly limit the value of the Index Advisor provides. A nearly unlimited amount of space may be more than you want to support.

2. You limit how long the Index Advisor iterates through candidate index sets by specifying a time limit for the Index Advisor to run. This time limit should be long enough to allow the Index Advisor to explain each SQL statement in the workload on each iteration, for as many iterations as you believe it will take to determine a good set of indexes.

With 17 queries on five tables for the one GB TPC-D workload, we found that 90 seconds and 2.5GB of index space was sufficient to converge on the best set of indexes. For your workloads, you will need to experiment to determine how quickly the set of indexes converges. A good rule of thumb is that indexes need to be two to three times the size of the raw data for a complex query and/or decision support workload. This number is often much smaller for an OLTP workload. My advice is to be liberal about the amount of space you want to use for indexes. Over-constraining the amount of space significantly detracts from the value you will get from the Index Advisor.

Choosing which of the recommended indexes to create

The output of the Index Advisor goes to your terminal screen, the ADVISE_INDEX table, and an output file, if desired. As we said before, the ADVISE_INDEX table includes a

column called CREATION_TEXT, which will contain the CREATE INDEX statement for any recommended indexes. The Index Advisor SmartGuide provides an interface to create these indexes or, alternatively, you can manually retrieve the index create text by issuing SELECT CAST(CREATION_TEXT as CHAR(200)) FROM ADVISE_INDEX.

If you were generous with the space constraint for the Index Advisor and you are actually more constrained on indexing space, you may choose to take the list of recommended indexes and eliminate one of the recommended indexes before creating them. You can do this by examining the explain table output to see which queries used the index you want to eliminate and decide how important that query is in your workload.

Advice on getting the most from the Index Advisor

The important point about getting the most value from the Index Advisor is to remember what you already know about the query optimizer. A rich and accurate set of detailed statistics on the data provides the optimizer more information to work with; in almost all cases, it will result in better query plans. The better your information and statistics on the data, the more accurately the Index Advisor can use the query optimizer to recommend or evaluate indexes for your query workload. Tables with no or few statistics will degrade the overall quality of the index recommendation. Specifically, this means that you need to have detailed distribution, correlation, and cardinality statistics on the columns most likely to participate in recommended indexes. If you have frequently used join columns on which there are no statistics, then the cost information stored in the ADVISE_INDEX table about a candidate index including that column will not have accurate information. In such a situation, the Index Advisor cannot provide a good estimate of the value of that index.

How quickly the Index Advisor converges on an optimal set of indexes is very workload and data schema dependent. As you use this tool with variations on your data and schemas, you will become more expert at knowing how long your query workloads take to run explain and how many times it takes to iterate with different candidate sets of indexes. The advising engine is very efficient on each individual SQL statement. In one invocation of the query optimizer, it recommends all the indexes that would benefit that individual statement based on already existing indexes in the solution set. The total number of explain cycles on different solution sets required to converge on the best set of indexes will depend on your workload and data.

Given the tremendous difference in query execution times between good and poor plans (due to having the wrong set of indexes), it's definitely worth taking the time to let the index advising engine run long enough to find the right set. You may have concerns about doing this on your production or even your test system, based on the number of catalog reads and the CPU time the optimizer requires. If you do, try running DB2LOOK and the Index Advisor SmartGuide on that system, then copy that information onto another DB2 system that you can use with less impact on your operational systems.

I'd like very much to hear your feedback from using the DB2 Index Advisor -- how many iterations it took, and what kinds of improvements you found in your query execution after you added indexes recommended by the Index Advisor. Please send your feedback to me at pgs@us.ibm.com. I'll publish your advice and experiences so that you can all learn from one another.

ABOUT THE AUTHOR

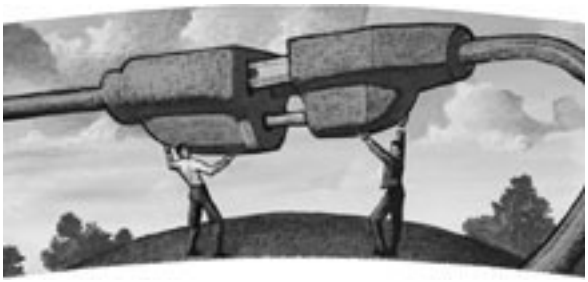
Dr. Pat Selinger has headed IBM's Data Technology Institute for 13 years. An IBM Fellow, she was recently elected to the U.S. National Academy of Engineering. Numerous awards honor her pivotal work on relational databases.

About IDUG | Conferences | DB2 Resources | Regional User Groups |Solutions Journal | Vendor Directory

Home | Contact Us

International DB2 Users Group
401 N. Michigan Ave.
Chicago, IL 60611
(312) 644-6610

The IDUG Solutions Journal
Spring 2000 - Volume 7, Number 1



Technical Tip
Stored procedure version control in DB2 Version 6 with WLM and Schemas

by Ramesh Bala

With the increasing use of stored procedures in systems development, it has become important to have a feasible development and support environment for stored procedures. With the DB2-managed stored procedure address space, it is possible to reference only one version of the load module for execution. It is quite limiting for a small to moderate sized shop, with one development DB2 subsystem, to achieve meaningful development and testing of stored procedures. Intended for DBAs and DB2 systems programmers, this article examines a strategy to employ Work Load Manager (WLM) and the recently available schemas to provide true and hassle-free version control.

After a high level discussion of the prerequisites, we will look at them in greater detail.

Figure 1 illustrates the relationship between the schemas, named, for example, UNIT, SYSTEM and QA with WLM Application environments named UNIT, SYSTEM and QA. Notice the stored procedure UNIT.SP1 in schema UNIT executes in Application environment UNIT, SYSTEM.SP1 executes in Application SYSTEM, and QA.SP1 in Application environment QA. Each application environment has a different concatenation of load libraries and hence can invoke different versions of the stored procedure SP1.

What is WorkLoad Manager?

Workload Manager (WLM) is a component of OS/390; it is a solution for managing workload distribution and distributing resources to workloads. Check for a process called WLM in your SDSF active panel, and if you find a process called WLM operating, your shop has implemented WLM. WLM has two modes of operation: goal mode and compatibility mode. In goal mode, WLM can fire up address spaces on its own to meet

demand. This is critical for version control, since we need more than one address space for supporting multiple versions of the stored procedure. In compatibility mode, the address spaces are started manually. WLM is the only means by which we can have multiple stored procedure address spaces in a DB2 subsystem; that is why it is the key part of this implementation.

Enter Schemas

Schemas, introduced in DB2 Version 6, are a handy way to group objects together. The objects could be stored procedures, triggers, or functions. Objects are created into a "Schema" and use the Schema name as the qualifier. This name extends the concept of "Qualifier" to these relatively new objects. Using multiple schema names is central to the idea to establishing multiple WLM-managed stored procedure address spaces, since only with multiple schema names can one define multiple stored procedure definitions having different WLM environment names.

Resource recovery services

Resource Recovery Services (RRS) is an MVS feature that coordinates two-phase commit processing. It's a system-logger application that records events related to protected resources.

If your shop has implemented WLM, check for a process called RRS in your SDSF active panel. WLM-managed stored procedure address spaces utilize RRS attachment facility (RRSAF) to interface with DB2, not the Call attach Facility (CAF). Thus stored procedures executing in traditional DB2-managed address spaces will need to be recompiled for the WLM environment.

Language interface module DSNRLI

Stored procedures in the WLM environment interface with DB2 using the RRSAF by calling the new language interface module DSNRLI. If the traditional interface module DSNALI is invoked in this environment, the application issues a -981 SQLCODE. Later, we will look at a mechanism by which we at SAIF Corporation, (an insurance company based in Salem, Oregon), were able to overcome the recompilation requirement and also painlessly adapt our stored procedures in the new WLM environment.

Definitions in WLM

Listed below are some of the important definitions in WLM. Though one may not need to use most of these, it is helpful to be aware of them when creating some of the new definitions (See Figure 2).

Workload: A group of related work meaningful for the installation. All the

work started by a set of applications is a workload. We could define a workload for exclusively stored procedures.

Service definition: One set of definitions of your workload and its performance goals.

Service policy: A set of service policies is a service definition. Only one service policy can be active at any given time. That is, DAYTIME service policy is active during the day, while a NIGHTTIME policy is active during the night.

Service class: Classification of work that has the same performance characteristic. It is by the mechanism of the CLASSIFICATION RULE that WLM associates a service class with a piece of work that arrives at a subsystem.

Service class period: Makes up a SERVICE CLASS. This definition has performance objectives. The service class period has three attributes:

Importance: How important is the work?

Goal: Response time or Execution velocity. OLTP transactions have a response time goal, whereas batch typically gets a velocity goal.

Duration: Determines for how long the task will receive the level of importance and the specified goal.

Application environment: By far the most important definition related to stored procedures; it is probably the only definition that a DBA will need. An AE definition represents one or more stored procedures. It is a way to have WLM dynamically create and delete server address spaces.

Creating an application environment

To implement version control with WLM, we will have to create AE definitions in the existing SERVICE DEFINITION installed in your shop. Your systems programmer will need to grant you authorization to read the existing service definition. After you save this in a dataset, you can add the AE definitions, and then have the systems programmer install the modified service definition.

The ISPF application supplied to create and modify WLM definitions is typically in SYS1. SBLSCLI0 (IWMARIN0). Executing this application displays the following screen.

After extracting and saving this service definition in a dataset, use option 9 on this screen to create a new AE definition.

Use the following values for the new AE definition. The procedure name referred to is the JCL procedure name in SYS1.PROCLIB. This is the procedure that WLM will use to initiate an address space when there is demand for one. For version control we need to have three AEs invoking three different JCL procedures (with different load libraries).

This screen shows the different application environments for the three test environments. Each of them invokes a JCL procedure with a different load library concatenation.

Creating the schema definitions

In order to have three environments for development, we need to have three schemas defined in SYSROUTINES. Shown here is a sample DDL for the creating a stored procedure definition DBU008 in a schema called UNIT. Notice the WLM ENVIRONMENT name is the application environment that we just defined.

```
CREATE PROCEDURE UNIT.DBU008 (CHAR(172) INOUT) LANGUAGE COBOL
  EXTERNAL NAME DBU008 COLLID COLLD11
  RUN OPTIONS 'MSGFILE(OUTFILE),RPTSTG(ON),RPTOPTS(ON)'
  WLM ENVIRONMENT TDB2WLMD;
```

Shown below are the three schema definitions for the stored procedure in SYSROUTINES.

SCHEMA	OWNER	NAME	LANGUAGE	WLM_ENV
UNIT	DBS	**DBU008**	COBOL	**TDB2WSPD**
SYSTEM	DBS	DBU008	COBOL	TDB2WSPT
QA	DBS	DBU008	COBOL	TDB2WSPQ

Modifying the language interface module

In order to avoid recompiling the stored procedures for WLM-established address spaces, we at SAIF decided to invoke the language interface module dynamically. We went a step further and created a one line COBOL stub program called DSNHLI that transferred the call from the stub to DSNHLIR, an entry point in DSNRLI that handles dynamic calls. The load module of this stub was kept in a dataset that was concatenated ahead of the DB2 system run library in the WLM JCL procedure.

```
       IDENTIFICATION DIVISION.
```

```
PROGRAM-ID. DSNHLI.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 DSNHLIR   PIC X(8) VALUE 'DSNHLIR'. LINKAGE SECTION.
01 P   PIC X.
PROCEDURE DIVISION USING P.
*
  CALL DSNHLIR USING P.
  GOBACK.
```

Maintaining WLM-established address spaces

Following are some of the commands that a DBA needs to be familiar with in order to manage WLM-established address spaces. These commands are issued on the SDSF log or on the operator console.

- Display status of a Application environment
- D WLM, APPLENV=TDB2WSPD
- Start an Application environment
  VARY WLM,APPLENV=TDB2WSPD,RESUME
- Stop an Application environment
  VARY WLM,APPLENV=TDB2WSPD,QUIESCE
- Refresh modules in an Application AE
  VARY WLM,APPLENV=TDB2WSPD,REFRESH

Summary

By implementing WLM-established address spaces, Schemas, and making modifications to the language interface module DSNRLI and WLM JCL procedure, we at SAIF were able to implement an elegant stored procedure version control environment.

Figure 1. Schemas in one DB1 subsystem

Back to Text

Figure 2. Relationship among WLM Definitions

Back to Text

ABOUT THE AUTHOR

RAMESH BALA has been a database professional for more than six years, and has been associated with DB2 for more than 12 years. Formerly a consultant for IBM, he is currently a lead database analyst at SAIF Corporation, Salem, Oregon. He can be reached at RAMBAL@SAIF.COM.

About IDUG | Conferences | DB2 Resources | Regional User Groups |Solutions Journal | Vendor Directory

---

Home | Contact Us

International DB2 Users Group
401 N. Michigan Ave.
Chicago, IL 60611
(312) 644-6610

The IDUG Solutions Journal
Spring 2000 - Volume 7, Number 1

Technical Talk
New Features of Version 6

by Richard Yevich

Among the many new enhancements to Version 6 introduced over the last few months, several are very important to the way applications are built and SQL is used. From a planning perspective, the following may make a major difference in your future strategies.

1. External SAVEPOINTs
2. Declared Temporary Tables
3. Identity columns
4. Update with Subselect
5. More than 15 table joins

External SAVEPOINTs

SAVEPOINTs enable milestones within a transaction (unit of recovery) to be demarked. An external SAVEPOINT represents the state of data and schema at a particular point in time. Data and schema changes made by the transaction after the SAVEPOINT is set can then roll back to the named SAVEPOINT, as application logic requires, without affecting the overall outcome of the transaction.

```
SAVEPOINT name ABC (other options)
ROLLBACK TO SAVEPOINT name
```

This enables milestones with a unit of recovery and represents the state of the data and

schema at a particular point in time. You can use ROLLBACK to restore to a SAVEPOINT. This will be useful when you have reached a point during a UNIT-OF-WORK where you need to back out without ROLLBACK over the entire UNIT-OF-WORK. You can name the individual SAVEPOINTs and then ROLLBACK to which point is required based on the application processing requirements. You can also skip over individual SAVEPOINTs.

Declared Temporary Tables

The DECLARE GLOBAL TEMPORARY TABLE statement defines a temporary table for the current session, not only for a UOW. The table description does not appear in the system catalog. It is not persistent and cannot be shared.

This statement can be embedded in an application program or issued through the use of dynamic SQL statements. It is an executable statement that can also be dynamically prepared. Each thread that defines a declared global temporary table of the same name has its own unique instantiation of the temporary table. When the thread terminates, the rows of the table are deleted, and the temporary table is dropped. Declared temporary tables make it possible to avoid some of the locking, DB2 catalog updates, and DB2 restart forward and backward log recovery issues that are associated with persistent base tables. No authority is required to issue the DECLARE GLOBAL TEMPORARY TABLE statement, but an authority will be required to use the new work table space where the table will be materialized.

Declared Temporary Tables may be useful for applications that need to extract data from various sources and use them in SQL joins, or for data that needs to be used repetitively, or kept separate from other OLTP processes. These tables could also be used as staging areas for data that comes from various sources (i.e., VSAM or IMS) so that the data could be made available.

Some examples of the syntax for Declared Temporary Tables are shown below.

```
DECLARE GLOBAL TEMPORARY TABLE SESSION.ACCOUNT
  LIKE CLAIMS.ACCOUNT
INSERT INTO SESSION.ACCOUNT
  SELECT * FROM CLAIMS.ACCOUNT
  WHERE ACCOUNT_NUMBERS > :host_var
DECLARE GLOBAL TEMPORARY TABLE SESSION.ACCOUNT
  AS
  SELECT * FROM CLAIMS.ACCOUNT
  WHERE ACCOUNT_NUMBERS > :host_var
```

Identity columns

The Version 6 refresh included support for the use of identity columns. These provide a

way for DB2 to automatically generate unique, sequential, and recoverable values for each row in a table.

This new identity column is defined with the AS IDENTITY attribute provided in column definition.

```
CREATE TABLE MY_TABLE
(ACCOUNT INTEGER NOT NULL
 GENERATED ALWAYS AS IDENTITY
 START WITH 100000
 INCREMENT BY 100
 CACHE 20
```

Each table can have only one identity column defined to it. Identity columns are ideally suited for the task of generating unique primary key values such as employee number, order number, item number, or account number. The columns must be a data type that is arithmetic and exact - this means that SMALLINT, INTEGER, BIGINT, or DECIMAL with a scale of zero can be used. It is also possible to use a distinct type based on one of these types. Single and double precision floating point data types are not listed since they are considered to be approximate numeric data types, not exact.

Identity columns can also be used to alleviate concurrency problems or problems caused by application generated sequence numbers.

Update with Subselect

We have all been trying to do this for years in SQL statements. Eventually, we realized that we could not Update with Subselect, although we still were able to when using other DB2 family members. Now, however, you can have an UPDATE with a subselect to allow for an update to be performed based on criteria from another table. For example:

```
UPDATE RECEIVABLE RC
 SET AMOUNT_PAID =
  (SELECT AMOUNT

   FROM INCOME IC
   WHERE IC.ACCT_NO = RC.ACCT_NO
   AND IC.INV_NO = RC.INVOICE_NO
```

More than 15 tables in a join

Exceeding the 15 table limit for a single SQL statement has become more of a requirement recently, because of some ERP products and to enable processing in some larger warehouses. Now DB2 has been changed to allow more than 15 tables. If the resources are available, it is possible to use up to 225 tables in a SQL statement.

Theoretically, it is possible to use up to 225 tables in a single join -- although this should be viewed as simply support for more than a 15 table join. This allows more complex SQL to be written.

PeopleSoft made this specific requirement, because in many situations, they needed to join more than 15 tables together to produce reports. In truth, many applications have needed more than 15 tables in a single SQL statement because they had a large number of subqueries validating codes and references, as well as multiple relationship existence checks.

These are among the new features that may have the most impact on DBMS conversions and new physical designs.

ABOUT THE AUTHOR

Richard Yevich is a principal with RYC Inc., and an internationally recognized consultant, lecturer, and teacher. He is a member of IBM's S/390 and DB2 Gold Consultant programs. He can be reached via email at richard_yevich@ryci.com.

International DB2 Users Group
401 N. Michigan Ave.
Chicago, IL 60611
(312) 644-6610