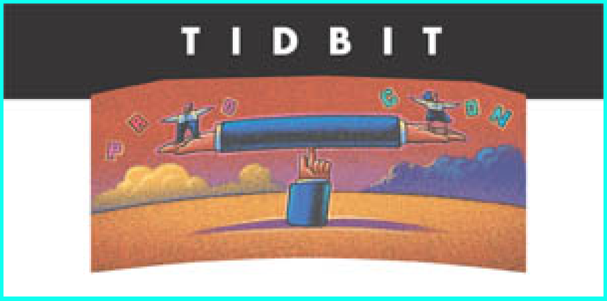
SOLUTIONS JOURNAL

The IDUG Solutions Journal
Volume 8 Number 1 (May 2001)



TIDBIT
A Challenging History: From Tortise to Hare

Before the birth of DB2, IT (DP back then) had no effective or efficient means of storing or retrieving data (or maybe it just seemed that way). People stored their information chiseled on stone tablets or, sometimes, in database managers like IMS. In a 1970 paper[1] to the ACM (Association for Computing Machinery), Dr. E. F. Codd proposed his relational model. Almost over night our industry went from nothing (or rather from ordinary database management systems) to DB2. There was a short stop at SQL/DS on our way to DB2 but we won't mention that. It was still a huge step for mankind's data. Mind you, those of you around back then know this giant data leap did not occur without a little pain. These were rough years.

Being a true IMS bigot, IBM dragged me kicking and screaming into DB2's world in the early 1980s. I knew this fancy new database management system would never last. Heck, it could barely run. And even if it did last, DB2 would never compare to or replace IMS. Thank goodness IBM paid no attention to me. So, into the world of DB2 I was thrust. It was difficult to erase all the IMS stuff from my brain to make room for all this new stuff. I discovered I didn't need that much room for DB2. DB2 seemed fairly simple and direct, and did what you expected. This would prove to be a real dilemma for me during my first few years with DB2. I would no longer be able to razzle dazzle anyone with the complexities of how IMS worked. Instead, I was stuck with DB2. It turned out DB2 was pretty cool. However, DB2 still had its challenges. It was slow, error prone, handled large amounts of data poorly and consumed enormous amounts of real storage. In addition, no one wanted to write (or maybe they simply refused to write) set processing code. Everyone did things the way they had always done them prior to DB2. The first few years were long and hard. People just didn't seem to understand. For example, we gave students the option to leave after the DB2 class overview without being charged. Once while teaching a class in New York City in 1985, I had a student tell me he was leaving because this class was not for him. He had been working with Dbase 4 for a year and didn't want to step back two levels to DB2.

DB2 knew what and where its weaknesses were. The problems IBM didn't find, user organizations were quick to point out. Over the next dozen or so years, DB2 would fix itself. Starting in DB2's first release, DB2 Version 1 Release 2, sequential prefetch was added. Now, we live and die by that feature. What better way to access thousands of sequential pages the quickest way possible? Can anyone even remember when we had to think about making buffer pools large enough (1000 or more 4K buffers) just so we would enable 32-page prefetch?

Or even 1000-page buffer pools? With prefetch, lock escalation and durations were added for more control over DB2's locking while accessing large amounts of data. I need to regress here for just a minute. When I first started presenting on DB2, I used to speak a lot on locking. That was in the mid-1980s. Seventeen years later, I am again presenting on locking for the 2001 NA IDUG conference. Talk about "what goes round comes around." However, I should get back on subject. Release 2 of DB2 also gave us EXPLAIN. EXPLAIN described the possible access path chosen by the optimizer. This promoted more efficient SQL. Release 2 also delivered the LOAD REPLACE option, DFHSM support, and reorganization with a no logging option. These features would be very important someday, if you have an occasion to access a bazillion rows of data. DB2 wasn't finished. Users still needed to process expanding amounts of data and in 1987 we saw the delivery of Version 1 Release 3. V1.3 would reduce LOAD REPLACE to the partition level and replace VSAM ESDS page sets with VSAM LDS support. Release 3 also increased the DBD's size to greater than 64K, a very big deal back then that today is a pain in the backside. The time then came for DB2 to abandon MVS/370 support in Version 2.1. Having only MVS/XA and MVS/ESA to support, DB2 took advantage of larger buffer pools. V2.1 also delivered substantially faster sort processing for large amounts of data, referential integrity, segmented table spaces and the QUIESCE utility. Throw in multiple index access in Version 2.2 and DB2 started to fly, regardless of the amount of data you tried to throw at it. Then, in late 1990, DB2 Version 2.3 added sequential detection and hardware-assisted sorts. It looked as though DB2 might win the war against large data stores. DB2 was now doubling, tripling, and in some cases, quadrupling its performance while reducing CPU consumption. What more could you ask of a database manager in the first 10 years of its life? Everything!

As they say, never leave well enough alone. Now that DB2 ran faster, we started finding more data to store in DB2, more reasons to access that data, and more different ways to use the data being stored. Terms like "mission critical" and "high availability" started creeping into DB2's marketing literature. IBM had created their own nightmare, a DB2 that accessed data so well that everyone wanted to use it for everything. It was once again time for DB2 to step up to the challenge. In 1993 another version of DB2 appeared. Version 3 delivered such innovative concepts as hardware compression, allowing us to make those large amounts of data appear small. But just in case there were any doubts about DB2's capabilities, V3 threw in more buffer pools (both 4K and 32K) and the introduction of a completely new caching mechanism; hiperpools. To maximize your fun with these new buffer pools and hiperpools, DB2 would allow you to dynamically alter their characteristics. Although DB2 could now keep more data in memory, we still needed to get to that data faster. So, DB2 developed a better method of I/O scheduling and a new concept called parallelism, but just I/O in the beginning. Not enough? DB2 Version 3 added partition independence giving you the ability to run utilities while accessing all of this newfound data via SQL. Can you say drains and claims?

With data demands skyrocketing because of the increased interest in warehousing and client/server processing, DB2 had little time to rest on its laurels. Performance improvements were needed and DB2 Version 4 was ready for the challenge. CPU parallelism became part of the product. On the off chance that one DB2 was not enough power to handle your data access and availability needs, true data sharing became a reality. However, of all the DB2 Version 4 enhancements, I think type 2 indexes were most significant. This new index manager would solve all kinds of data insert, update and retrieval issues.

What could possibly top type 2 indexes and data sharing? How about a 1 Terabyte table space? Talk about mega-data. Version 5, now DB2 Server for OS/390, would push DB2 into the terabyte world and DB2 would never look back. With terabyte table spaces came sysplex query parallelism and a smarter RUNSTATs that gave DB2's optimizer better statistics. While on the subject of optimizers, Version 5 also gave us runtime reoptimization of host variables. This feature would ensure that DB2 gave you the best access path possible. Of course, with that much data, the old REORG would never do, so DB2 now supplied an online REORG.

In 1999 we saw another version of DB2, again with a new name, DB2 UDB Server for OS/390 Version 6. If you have diverse data needs or just a need to store lots of data, you will just love DB2 Version 6. Sixteen terabyte table spaces, large object support, user defined types and functions, triggers and Java imbedded SQL is just a short list of some of the features V6 delivers. In fact, by the time you read this paper, DB2 Version 7 will probably be generally available. DB2 has come a long way since June 7, 1983. Almost all of its performance and data handling woes have been remedied. Now it can start to really concentrate on functionality.

DB2 has always given us the impression that it was "data-size challenged" and that fact has fascinated us since DB2's beginnings. Whether the amount of data we needed to access was small, large or extremely large, DB2 just loved to put us through that simplicity curve. Everything was just so easy to do in DB2, it tended to lull us into not worrying about data size until it was too late. I remember teaching classes back in the early 1980's and asking how large a table anyone in the class was using in production environment. I once had one student say that they had just implemented a table with over 100,000 rows. The entire class of over 90 students was left in awe and disbelief. They couldn't imagine someone having the nerve to process that much data in a DB2 Production environment. They all knew this customer couldn't care about response time. They probably assumed the customer was happy just having a query complete correctly.

In the beginning, we faced many challenges with DB2. The shear amount of data often messed up all kinds of processing. It was a challenge dealing with indexes during reorganizations and recoveries. We weren't sure how to build indexes to maximize our query performance yet. SQL was in its infancy. We stressed how easy it was to write SQL and spent very little time educating users on writing efficient SQL. As for online transaction processing, who cared? OLTP wasn't a very high concern in the beginning. Or at least it seemed like it wasn't. Even the idea of transaction processing has changed significantly. Anyone remember the first announcement letter for DB2? It introduced DB2, DXT and QMF to the world. DB2 was constantly referred to as a decision support database in its first years, maybe because DB2 and QMF were announced together. For a time, the DB2 detractors claimed DB2 would never be able to support OLTP because it was designed for decision support. DB2 has since turned into one of the best (if not the best) OLTP database managers. At a major conference a few years ago, one presenter, someone supposedly in the know, spoke about the latest advances in OLAP and other decision support areas. He commented that DB2 could never survive in the decision support arena because it was written as an OLTP database. If they would only pay attention to history.

It's funny how things never really change. We face many of the same DB2 challenges today that we did in the "beginning of relational time." Now, instead of 100,000+ rows in a single table, we are dealing with tables in the multiple terabyte size. With all the data challenges we face, both changes in data size and the way we use data, DB2 always managed to adjust and excel. When we conquer our current set of challenges, and we will, DB2 will just start examining ways it can handle our next set of challenges. It won't be too long before we are facing table sizes in the petabytes, zetabytes, or whatever else our need for data throws at us. By the way, I know that my time line may have been a bit exaggerated (they call that artistic license) and I didn't even attempt to list every significant improvement made to DB2 over the years. I also know the power and flexibility of DB2 could never be exaggerated, no matter how much data you need to store.

1. A Relational Model of Data for Large Shared Data Banks. CACM 13(6): 377-387(1970), Copyright © 1970 by the ACM, Inc.

About the author

Willie Favero has been a database professional for more than 25 years, the last 17 years primarily with DB2. He is currently a Principal Consultant with BMC Software. Willie can be reached at wfavero@attglobal.net

---

International DB2 Users Group
401 N. Michigan Ave.
Chicago, IL 60611
(312) 321-6881