

DATABASE WARS: Part 1 -- Memoirs of a SAS Technical Support Analyst

**Kathe Richards, Technical Support Analyst
Texas Comptroller of Public Accounts**

I had hoped to write a really technical paper with lots of tricky code and helpful hints for people facing the thrills of database migration. As it turns out, I haven't been able to get many of my neat tricks to work yet. (Maybe by the time I deliver this paper, some of them will pan out.) So instead, I offer herewith some things I've learned during this experience so far, in case it might be helpful to someone else. It's a poor substitute for tricky code, but here goes.

Lesson 1: It helps to pick the right tools.

This is not our first database conversion. We started this round with a good bit of our "enterprise" information in System 2000 databases. Many of the application supporting detail files are VSAM, however. Our production systems were written primarily in traditional programming languages, COBOL and IBM assembler. A very large chunk of what people call "management reporting" as well as some "departmental applications" was written by end users in operating divisions using SAS. SAS was introduced to the Comptroller's office by the Revenue Estimators back at the dawn of the computer age and has grown to be the language of choice for people who need to get something done in a hurry and can't wait for the traditional "workload prioritization" process. The System 2000 databases are not made available to the users as a matter of policy. Data from them is supplied by regularly scheduled extracts.

Among the first things mentioned when the move to DB2 started looking like a real possibility was the user reporting function. How were we going to support that? And of course, the first thing considered was SAS. It was immediately discarded. We were running release 5.18 at the time and the DB2 database access that came with that version had some features that made the professionals uncomfortable. It would be necessary to implement with a good deal looser security than they had planned on.

At this same time we were also introducing Natural, a fourth generation application development language. There was an "MIS reporting" tool available with this software, and someone suggested that this might be the magic substitute. Someone seriously underestimated the kind of "reporting" that was going on out there. The package was brought in, introduced to the key users, and the users laughed.

Back to the drawing board. Well, how about letting the users have Natural itself? Everybody was lined up for the week-long classes, set up and encouraged to go for it. The users were underwhelmed. They already had a fourth generation tool. SAS.

Well, how about QMF? That's what DB2 wants you to use. Two problems turned up with this. First, a good bit of what people needed to do involved matching data that was in DB2 with data that wasn't. As a reporting language, QMF is not bad for what's in DB2 but doesn't do much for you outside. And then, there was the inevitable instance or two where a user produced a Cartesian product of two large tables. Capacity planners and resource utilization monitors got nervous.

So we asked the users what they wanted. And they said SAS. By this time, Release 6.06 was available. Many of the objections of the professionals were now resolved. We brought in SAS/Access Interface to DB2, the users evaluated it, we held training classes, and it is now our tool of choice.

Lesson 2: Learning the tool is the easiest part.

Even though most of our people are batch users in a Roscoe environment, nobody had much trouble picking up enough TSO and Display Manager to create View Descriptors. Most of them would then use the View Descriptors on the batch side. (This was before 6.07.)

Relational concepts and database concepts were pretty easy for our user team. SAS users don't seem to have much trouble understanding these things, maybe because conceptually it's not all that different from SAS. The software handles the library and storage details and a table is a SAS dataset.

The biggest issue was how to let the users know what is in the black box called DB2. Since we had to have some data to read, we got a listing of all the public tables on the test side. Our general policy is that users have read access to any agency data that is not specifically restricted by law, such as personnel data. They didn't have any good way to know what was where within DB2, however. To fill this need, our database administrators built us some tools.

They preferred to supply us with a master library of Access Descriptors and View Descriptors. Although this turned out to be an enormous effort for them, it has been a very nice benefit for us. They have the information first when something is going to change and they have taken responsibility for updating the master library. During training, we encourage users to build their own View Descriptors because of performance considerations. We want people to avoid selecting all columns and all rows on a table if the data is not really needed.

The DBAs also supplied us with a database containing cross reference tables indicating which access descriptors and view descriptors belong to which DB2 tables and another table reflecting actual data elements. At this point, we had tools to use if we knew which production database held the information we wanted. We still didn't have a satisfactory solution for the problem of what to do when we don't know which production database the information is in. We were handling this on a case by case basis with the data administrators.

An additional problem turned up when we discovered that there were multiple DB2 views with identical access descriptors. This happened when a base table was subset for a production application. There was no easy way for a user to tell what the subsetting conditions were, so they didn't know which one to use. We are working on a piece of tricky code to handle this problem. You can read one of the DB2 SYSIBM tables to get the information. We are building a utility to make this easy. (Maybe you can beat me to it.) For now, we go to one of the database administrators who runs a quick QMF job for us.

Lesson 3: Nothing happens as quickly as you think it's going to.

Having chosen our tool and trained our team, we found that there was not much happening. This was because there was not much to use it on. While the tool selection project was going on, a few data stores were converted, the most significant of which was the Taxpayer Information file. There were also a few small new tax applications that had been written completely in DB2. The old Taxpayer information file, a VSAM extract of the System 2000 database, which contained name, address, and related information about taxpayers, continued to exist. There was a major reanalysis of our tax processing systems underway. There was also a major development effort going on in the financial systems. Suddenly legislative implementation descended upon us as it does every couple of years and things slowed down.

Conversion of the data is typically one of the last things that is going to happen in a project. Lots of development professionals will tell you this, but very few other people believe it. Until the data moves into another home, nobody is going to look for it there.

Lesson 4: When it does start happening, nothing is as easy as it looks.

Another reason things didn't happen very fast was that unless the tables held information people couldn't get anywhere else, they would continue to get it the old way. Would you rewrite all your code if you didn't have to?

Besides that, when an information store is converted from an S2K or VSAM format to a DB2 database, the data doesn't come out in a one-to-one relationship to where it was. We model the data first. One S2K database or VSAM file can be expected to turn into many (sometimes VERY many) tables. Without a comprehensive data dictionary (and maybe even with one) it is not easy to convey this information.

Finally, the first good-sized application that involved a complete conversion appeared. The application involved taking a process supported by user-developed interactive and reporting systems, some production systems and a lot of manual processes and turning it into a production DB2 application system. In the process, the primary key for most of the data changed from taxpayer number to case number. While this supported a much more sophisticated business need within the application, most other applications are based on taxpayer number. Once you got outside the application, things that used to be simple got really complicated. For instance, finding a name and address for someone, an operation that used to take one simple merge, now took half a dozen table joins. The user took a look at this and said "Are you kidding me?" The lead programmer, missing the point completely, immediately said "I knew SAS couldn't do it. They'll all just have to learn to use QMF."

Another thing that happened pretty fast is that people used to living by the safe and tidy rules of the SAS world started having to live by the rules in the DB2 world. This meant an introduction to all the interesting DB2 return codes that inform us when we have violated security, offended syntax, or caught DB2 asleep.

And then there is SQL. The existence of PROC SQL in SAS and DB2 SQL in the access facility caused confusion. People who already knew SQL before using PROC SQL (and maybe didn't completely understand SAS) had some problems sorting things out. People who didn't know SQL wondered if they needed to learn it.

Lesson 5: Performance is always an issue.

One of the first things that we found out during the evaluation period for SAS/DB2 was something that everybody else seems to know. This stuff takes a lot of resources. Our shop has been critically short of resources for some time now and the data police keep a close eye on what's going on.

Our first big shock was a simple job that ran in less than a minute of CPU time against VSAM and now took 18 minutes of CPU time in DB2. We called SAS immediately, who supplied some advice and debugging suggestions. We found that the code was inexplicably generating a table scan of a 2.6 million row table three times every time it executed. We tried various coding strategies and produced similar results. But we were unable to reproduce these results with any other tables and the technical support people at SAS were unable to reproduce our problem at all. With release 6.07, fortunately, this problem has disappeared.

Other problems were solved by release 6.07 and the advent of SQL pass-through, also. Table joins between two DB2 tables can now take place inside DB2. The batch facility for creating access descriptors and view descriptors has been a benefit for those of us who didn't much like managing large libraries of view descriptors. We define the view descriptors as WORK.xxx and they go away at the end of the job.

All problems have not disappeared, however. If you need to match a small amount of external data with a large amount of DB2 data by a field that is indexed, there is no easy way to take advantage of the index. We are working on some tricky code for this, but again have not quite been successful.

We also found that production tables are often not structured or indexed well to support reporting needs. We've handled these on a case by case basis, also. There are trade-offs involved with having indexes, especially on large tables (and you don't need them as much on small tables.)

The database administration team has implemented some strategy to handle the cases where the reporting needs and processing needs are in direct conflict and to manage the security administration more easily. We have a Production Environment and a Query Environment as well as a Test Environment, Acceptance Test Environment and Maintenance Environment. Users of the SAS/DB2 Access Interface facility go through the Query environment and can read Production data through Distributed Data Facility (DDF), a DB2 facility that links environments. (Figure 1 is a diagram of this configuration.) So far, this has been a satisfactory arrangement. We had one series of incidents in which VTAM and the DDF seemed to get out of sync in their messages to each other and VTAM would take

over the world. This was remedied by placing a governor on DDF. The problem was then referred to IBM.

Lesson 6: The laws of nature have not been repealed.

Things get harder before they get easier. Everything takes as long as it takes. Users are suspicious of programmers (often with good reason) and programmers don't understand users. Management looks at how expensive all this stuff is and feels nervous. Here we all stand, on various sides of this great black hole, peering in.

So what happens next? The big tax processing system integration project and database conversion project are now on hold pending review. New concepts like object oriented programming and client-server technology are drifting around in the air. We're still short of computer resources, time, technical staff and money. But what we've got in place is working pretty well. My guess is that we will continue on into this brave new world, and while nothing will be quite as magical as initially promised, when we look back in ten years, I bet we'll be impressed.

DB2 ENVIRONMENTS

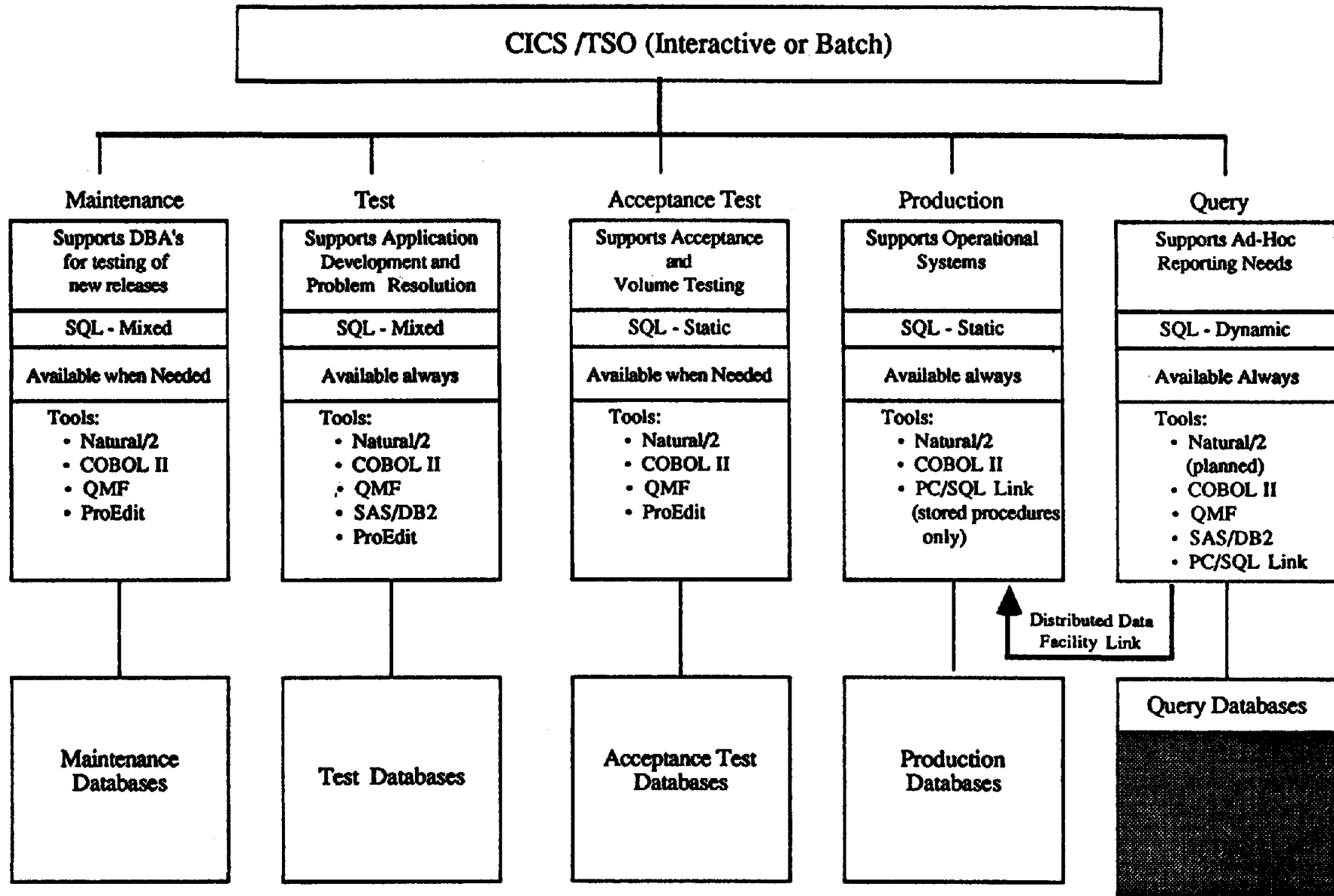


Figure 1