Craig S. Mullins

Return to Home Page

Vol. 12, No. 1 (April 2005)

From IDUG Solutions Journal...

The Buffer Pool

On VSAM and DB2

By Craig S. Mullins

Lately I've been getting a lot of questions from people looking to convert their VSAM data to DB2. So I thought I'd put together my thoughts on the topic for my *IDUG Solutions Journal* column so I can share them with everybody.

First of all, let's discuss the basics. VSAM and DB2 are very different technologies. VSAM is a file access method and DB2 is a database management system (DBMS).

VSAM, or Virtual Sequential Access Method, is a methodology for the indexed or sequential processing of records on direct access devices. There are three ways to access data in a VSAM file: random (or direct), sequential, and skip-sequential. ***Random access*** is enabled using a search argument to directly access data and ***sequential access*** is accomplished by processing one record at a time. ***Skip-sequential access*** is a combination of random and sequential: the first record is obtained randomly and from there on each subsequent record is processing one after the other. With VSAM though, direct access to data can only be provided using a pre-defined key: there must be a primary key (there can be multiple, alternate keys).

Of course, this simple introduction to VSAM just skims the basics. There is much more that you will need to know if you are using VSAM or considering whether to convert from VSAM to DB2. Sometimes the only motivating factor driving a conversion effort is that VSAM is old technology and DB2 is newer. That is not a very useful criterion for converting. Of course, if you are having trouble trying to hire folks who are knowledgeable about the older technology, then the

reasoning makes a little more sense. But I have never heard that line of reasoning with regard to VSAM; it is more common with regard to converting from IMS to DB2.

A better motivating factor for converting from VSAM to DB2 is to take advantage of the facilities and capabilities of a DBMS. Let's examine the core benefits of using a DBMS.

The Advantages of a DBMS

Being a DBMS comes with advantages (and overhead). A DBMS is a software package designed to create, store, and manage databases. The DBMS software enables end users or application programmers to share data. It provides a systematic method of creating, updating, retrieving and storing information in a database. DBMSs also are responsible for data integrity, data access control, and automated rollback, restart and recovery.

The main advantage of using a DBMS is to impose a logical, structured organization on the data. A DBMS delivers economy of scale for processing large amounts of data because it is optimized for such operations.

Using a DBMS provides a central store of data that can be accessed by multiple users, from multiple locations. Data can be shared among multiple applications, instead of new iterations of the same data being propagated and stored in new files for every new application. Central storage and management of data within the DBMS provides:

- Data abstraction and independence.
- Data security.
- A locking mechanism for concurrent access with ACID properties (ACID is an acronym for atomicity, consistency, isolation, and durability).
- An efficient handler to balance the needs of multiple applications using the same data.
- The ability to swiftly recover from crashes and errors.
- Robust data integrity capabilities.
- Simple access using a standard API.
- Uniform administration procedures for data.

Furthermore, a DBMS offers the ability to provide many views of a single database schema. A view defines what data the user sees and how that user sees the data. The

DBMS provides a level of abstraction between the conceptual schema that defines the logical structure of the database and the physical schema that describes the files, indexes, and other physical mechanisms used by the database. Users function at the conceptual level — for example, by querying columns within rows of tables — instead of having to figure out how to access data using the many different types of physical structures used by the DBMS to store the data.

When a DBMS is used, systems can be modified much more easily when business requirements change. New categories of data can be added to the database without disruption to the existing system. With DB2, for example, adding a new “field” is as simple as issuing an ALTER statement to add the new column to the table. Performing a similar task in VSAM is much more difficult — especially if the file does not have any unused “filler” area at the end.

A DBMS provides a layer of independence between the data and the applications that use the data. In other words, applications are insulated from how data is structured and stored. The DBMS provides two types of data independence:

- Logical data independence; that is, protection from changes to the logical structure of data.

- Physical data independence, meaning protection from changes to the physical structure of data.

As long as programs use the API (application programming interface) to the database as provided by the DBMS, developers can avoid changing programs because of database changes. With DB2, this API is SQL and there is no other approved way of accessing DB2 data.

Furthermore, *ad hoc* access to your data is very, very difficult — perhaps nigh on impossible – when using VSAM alone. *Ad hoc* access to data in a DBMS is simple, just code up some SQL. This offers better data availability and access to your end users.

Of course, a DBMS must perform additional work to provide these advantages, thereby bringing with it the overhead. A DBMS will use more memory and more CPU than a simple file storage system. But remember, you are accomplishing a lot more with the DBMS.

Converting from VSAM to DB2?

The gist of most of the questions I've been hearing lately are why and/or how to convert from VSAM to DB2. So let's try to get our arms around the situation. Usually, a company has an existing, VSAM legacy system, written in COBOL and accessed in batch (and perhaps online in CICS). Then, someone at that company gets the bright idea to convert that system to use DB2. Which, in turn, causes people to ask: "Should we do this?" and "How can we do this?"

Tackling the "should" question first, the answer is "it depends" (of course). Most people asking these questions already use DB2 for other applications and employ trained DB2 professionals (programmers, DBAs, system programmers, etc.). If you do not already use DB2, then re-read the beginning portion of this article because that explains the primary benefits of using a DBMS like DB2, instead of a file system like VSAM.

But the "should" question goes deeper than just DBMS versus file system. The VSAM application programs are already written and (probably) running well. Any conversion you attempt will disrupt the apple cart, at least somewhat. If your existing application is working well and there are not a lot of outstanding requests for

changes or many requests for ad hoc reports, then you should probably leave well enough alone – meaning, do not convert the VSAM data or applications. The trouble of converting is unlikely to produce sufficient benefits to make the conversion worthwhile.

There are several other issues with VSAM that might be driving the desire to convert to DB2. It can be difficult to share VSAM data across different processes while ensuring data integrity. If data growth is an issue, there are architectural limits that impose a maximum size on a VSAM file, whereas DB2 imposes higher limits. And change management is more difficult with VSAM, especially as DB2 online schema evolution continues to evolve.

All right, so what if we still want to convert because of the above reasons, or some other compelling reason (management dictate, data integration requirements, etc.)? That is when we get to the “how” question. And that is the stickier of the two questions.

Most converters, at this point, are looking for techniques or products that will allow them to convert the data and leave the applications untouched. If

pursued without knowledge of the application, this strategy can produce very bad results. Because DB2 uses SQL it accesses data a set-at-a-time. VSAM, on the other hand, accesses data a record-at-a-time. So there is an impedance mismatch between the application that is already written, and the new data storage mechanism – DB2. If you simply convert VSAM calls to DB2 SQL statements then you most likely will not take advantage of the power of SQL. You will not be joining data. You will not be formulating predicates properly because VSAM only accesses data by keys. You may be reading data that your programs do not require. This will result in diminished application performance – and nobody wants that, do they?

Indeed, the biggest problem that VSAM professionals encounter when moving to DB2 is treating the DB2 data like it is in a flat file. A mentality shift is required to think in sets instead of files, rows instead of records, and putting as much work as possible into the SQL predicates to allow DB2 to work as efficiently as possible.

Conversely, sometimes the VSAM proponents denigrate DB2 by calling it a pig. Yes, there is additional overhead

when using DB2 instead of VSAM. DB2 does more than VSAM, so the overhead is warranted. Does that mean that VSAM outperforms DB2? Absolutely not!

If you understand DB2 and use it appropriately, its performance will be excellent. If you use DB2 like VSAM, its performance will stink. Think about it this way: compare a DB2 SELECT of four columns in a clustering index against the application code needed to access the same data by reading the entire VSAM file. In such a scenario, properly coded DB2 will undoubtedly outperform properly coded VSAM requests.

Flexibility is another important concern. DB2 is flexible and VSAM is not. If you do not believe that, then think about what it would take to add an index to existing data. With DB2, you add the index, rebind the program, and DB2 will take advantage of it without having to change any application code. With VSAM you would have to explicitly code requests to use the new index – not very flexible, is it?

And the robustness of the environment is another consideration. Running concurrent updates against the same VSAM file in batch and online is not nearly as

efficient as doing the same with DB2. DBMS locking and ACID properties make such situations a clear-cut advantage for DB2.

DB2 Uses VSAM

Finally, with all of the above said and done, remember that DB2 needs VSAM. DB2 uses underlying VSAM linear data sets for its table spaces. However, keep in mind that all control over that data is done by DB2. Reading, writing, buffering, and so on, all are controlled by DB2, and not by VSAM components or commands. (Of course, some DBAs still use IDCAMS to define the underlying data sets, but most DBAs these days use storage groups thereby enabling DB2 to define the VSAM data sets itself.)

Summary

Hopefully this short synopsis of VSAM and DB2, along with the advantages of the database approach over the flat file approach, has reinforced concepts and knowledge that you already had. If you are looking for additional information on VSAM, consider reading the

IBM redbook titled ***VSAM Demystified*** (SG24-6105). You might also want to look into IBM's VSAM Transparency options if you are tasked with converting VSAM applications to DB2. And good luck with your VSAM to DB2 conversions...

From *IDUG Solutions Journal*, April 2005.

Home.