INTRODUCTION TO VSAM

William J. Atkinson and Paul A. DeSanctis

HAYDEN

TP3
A3

8361134

Introduction to VSAM

William J. Atkinson
Philadelphia College of Textiles and Science
and
Paul A. DeSanctis
Temple University

HAYDEN BOOK COMPANY, INC.
Rochelle Park, New Jersey

To Tracey and Paul

Books must be read as deliberately
and reservedly as they are written.
Henry David Thoreau

Library of Congress Cataloging in Publication Data

Atkinson, William J.
Introduction to VSAM.

Bibliography: p.
Includes index.
1. Virtual computer systems. I. DeSanctis, Paul A., joint author. II. Title.
QA76.9.V5A86 001.64 80-17325
ISBN 0-8104-5159-X

1 2 3 4 5 6 7 8 9 PRINTING

80 81 82 83 84 85 86 87 88 YEAR

Introduction to VSAM

Preface

When writing a book on the subject of computer programming, or on any technical subject for that matter, the author must make a very fundamental decision concerning the type of treatment to give that subject. He can choose to treat it at a purely theoretical level and leave to his reader the problem of transforming the theory into practice. Such treatment, while highly beneficial to a reader in an academic or research sphere, is hardly of much value or use to the reader in the working arena who must produce results quickly based on what he has read. On the other hand, the author can choose to treat his subject at a very practical level, drawing on theory only when necessary to support or justify certain basic steps in the development of the material. This practical or technique-oriented approach removes from the reader the burden of having to spend countless hours trying to bridge the gap between theory and application. It starts him off with a working knowledge of the subject, which, it is hoped, will excite his curiosity and spur him to delve into the subject at a more theoretical level as his experience grows. With this thought in mind, we have chosen the second approach for this book.

Our subject, of course, is the Virtual Storage Access Method, also known by the acronym VSAM. Specifically, we are concerned with the VSAM as it is applied in a 370 DOS/VS environment running under COBOL and COBOL-CICS. (Although we've built our presentation upon DOS/VS, most of the principles discussed in this book apply under OS/VS as well.) The subject is presented through the consideration of a very basic and real problem—the problem of converting the system data base from the Indexed Sequential Access Method, ISAM, to VSAM. The justification for this approach will become clear in Chapter 1, which introduces the notion of VSAM and explains, among other things, the need for it. But before becoming too deeply involved in a discussion of the book's contents and prerequisites, let us say a few words about its objectives.

The first objective is to familiarize the manager of a 370 DOS/VS COBOL-driven system with the basic capabilities and advantages of VSAM and suggest to him a workable approach for converting his files from ISAM to VSAM. The second objective is to impart to the COBOL programmer a thorough knowledge of the programming techniques needed to support VSAM files. This is done by stepping through a set of carefully formulated COBOL programs designed for use against ISAM files and showing how those same programs would have to be modified for use against VSAM files. By establishing a visible parallel between the old ISAM and the new VSAM, the concepts of the latter become easier to grasp for the experienced, ISAM-oriented programmer, who in the end must do the work.

Chapter 1 is addressed primarily to the manager and requires little in the way of preparatory technical knowledge. It is advisory more than it is didactic. Chapter 2 is to be shared by manager and technician alike. In very basic terms, it presents a conceptual overview of VSAM with emphasis on storage management and file structure. It also includes important technical considerations regarding the creation and maintenance of VSAM files. Chapter 3 and the rest that follow it are addressed primarily to the programmer and require an adequate understanding of COBOL programming and a working knowledge of ISAM. (For the reader who feels that his or her knowledge of ISAM is a little rusty, we have included a brief review of it in Appendix III.)

Of particular importance is Appendix II, which contains the sample programs around which the text is built. These programs have been satisfactorily tested by the authors against live VSAM files. If the reader does no more than study those sample programs, he or she will nonetheless stand to gain much insight into the workings of VSAM files.

From time to time within various chapters the reader is referred to the manufacturer's literature for a more detailed or extended discussion of a particular idea or technique. So that the reader knows what literature we mean, a list of selected readings and references is provided in the back of the book.

This work would not have been possible without the cooperation, advice, and support of numerous others, including professional colleagues, friends, and members of our families. To all of them we are deeply indebted.

In particular, we thank Mr. George Slotnick, data-processing manager, lecturer, and friend, for providing us with the opportunity to work extensively with VSAM. We also thank Joe Plunkett, roving programmer and consultant, whose highly imaginative mind helped to develop many of the VSAM programming techniques discussed in this book. We would be remiss in not mentioning our publisher, whose editors and technical reviewers provided many valuable suggestions for improving the manuscript.

Nor could we forget to thank our wives, Maria Atkinson, who put in countless hours at her typewriter laboriously transforming hundreds of pages of scribbled notes into perfectly readable form, and Marguerite DeSanctis, who read the completed manuscript with the same vigorous energy—although not with the same vicarious interest—with which she read *Oliver's Story.* And last but not least, we thank our young children, who gave up more than one trip to Baskin-Robbins so that "daddy could work on his book."

WILLIAM J. ATKINSON
PAUL A. DESANCTIS

Contents

Chapter 1

Introduction to Virtual Storage Access Method

The Inevitable Decision

Sooner or later in this age of rapidly changing computer technology, the manager of every data-processing installation using the IBM 370 series computer in a DOS/VS-driven real-time environment will be faced with the decision to convert his ISAM files to VSAM. When that time comes, he will find himself in that decidedly unpleasant bind of having to bring his installation up to the current state of the art without interrupting the flow of the user's data stream. In all probability, the decision to convert will be made quickly. In addition, Murphy's Law will in all likelihood prevail, and the decision will come at the worst possible time when the manager is beset with a multiplicity of other problems. Consequently, he will not have much time to spend learning all the why's and wherefore's of VSAM, but this won't stem the tide of the many, often troubling questions that will descend on his mind begging for answers. Some of the questions will be of an administrative nature and will fall within the realm of management such as:

- What is VSAM and why do we need it in the first place?
- What are the advantages of VSAM over other access methods?
- How does VSAM compare cost-wise with ISAM?
- What type of files does VSAM support?
- How can the conversion be implemented most expeditiously?

Other questions will be of a technical nature and will more properly fall within the realm of systems and programming such as:

- What are the effects of VSAM on existing user software?
- What are the actual coding changes needed in CICS[1] and COBOL?
- How will JCL be affected?

[1]CICS, which stands for Customer Control Information System, keeps track of all the files it uses in its own File Control Table (see page 101), which is part of the CICS monitor.

These questions and others like them will have to be answered and answered promptly, thoroughly, and correctly if the conversion is to be made with a minimum of downtime and inconvenience to the user. Every attempt will be made in this chapter to deal with the administrative questions concerning VSAM. The remaining chapters of this book will deal with the technical questions concerning its successful implementation. Without any further expenditure of words, let us get started.

The Need for VSAM

It is quite possible that your present system using ISAM is humming along smoothly. You finally have it perfected to the point where it is doing all that you and the user had hoped it would. The programming staff knows how to work comfortably with it and fully understands its file-handling capabilities and limitations. Response time is good under ISAM, and logically there seems to be no good reason to abandon your present file-handling software package in favor of another. But think about the situation for a moment, and you will soon realize that you will have to convert to VSAM whether you want to or not, given, of course, that you stay with the same manufacturer.

VSAM is IBM's latest and, purportedly, best access method for users under DOS/VS. According to the manufacturer, it offers more functions, better performance, and better data integrity and security than DOS/VS, SAM, DAM, and ISAM. Their claims may be all true, but such claims in themselves might not be enough to convince the more conservative customer that he should convert to VSAM. But, if he waits long enough, he won't have a choice in the matter, since the day will come when IBM will no longer support ISAM and other similar access methods once VSAM gains a firm foothold as it seems destined to do. Already, IBM is marketing its 3350 disk unit to replace the current 3330 unit. While the bigger and faster 3350 unit will tolerate ISAM, its full file-handling capabilities can be achieved only with VSAM. VSAM is IBM's coming file-management facility, and it will soon be here to stay.

The Advantages of VSAM

According to the manufacturer, VSAM is an efficient, easy to use, and easy to control access method that provides fast retrieval and storage of data on direct-access devices. It can be used much the same as ISAM is used except that VSAM has many additional advantages:

1. High performance due to the efficiently organized index and to the dynamic allocation and management of free space. File reorganizations are not needed as often as they are with ISAM mainly because

of VSAM's capability to redistribute or re-use disk space made available when records are dynamically deleted.

2. Simplicity of use and optimization of storage space. VSAM maintains a catalog of the physical and logical attributes of a file, thus simplifying job-control specifications. In addition, VSAM controls record blocking, thus optimizing the block length to suit the particular file storage device.
3. Central control over the files. Access Methods Services[2], a companion software set of VSAM, controls the definition, creation, access, deletion, and space management of all files defined in the VSAM catalog.
4. Protection of data. VSAM provides security against accidental obliteration of data as well as protection against unauthorized deletion or alteration of those data. This is done by the optional use of customer-created passwords stored in the VSAM catalog.
5. Device independence. VSAM, because of its address-initiating and tracking capability, frees the application programmer of the need to be concerned with storage devices and device addresses of his files. No longer does the programmer have to worry about where his files are. VSAM knows and VSAM can find them.
6. CICS compatibility. CICS-written transactions for ISAM-supported files are totally compatible with VSAM file structures. The only change required when adapting a CICS system to VSAM is to change the File Control Table (FCT) entry. More will be said of this particular change later.

VSAM has other advantages that are not listed above but that can be found in the manufacturer's literature. Since it is not our purpose to present a theoretical discourse on VSAM, we will move forward to areas of more practical importance.

Cost Effectiveness of VSAM

Attempting to measure the cost effectiveness of any new software system package is indeed a risky undertaking, and the effort expended on any serious cost-effective study is usually not justified by the results, which are more often than not imprecise and misleading. Certainly, this is true with VSAM. VSAM has a higher rental fee than does ISAM, and also the hardware to support VSAM is more expensive; for example, its 3300 series disk packs carry a rather high price tag. Conversion to VSAM is not necessarily going to produce any actual dollar savings for the

[2]Access Methods Services (AMS) commands are the reserved words within the AMS instruction set for performing the various file-manipulating functions.

renter. In fact, it will produce a greater dollar expense, and the leasee might justifiably wonder if VSAM offers any tangible benefits to offset its extra cost. Fortunately it does. Let us examine some of those cost-offsetting benefits.

The first and perhaps most important cost-offsetting feature of VSAM is its highly efficient utilization of disk space. If the 3350 series disk unit is to be used, then VSAM offers an important space-saving advantage. The reason for this is that the 3350 disk unit has been designed with VSAM in mind in that it supports that particular file-management facility most efficiently. More specifically, in order to optimize utilization of disk space on the 3350 pack, VSAM must be used. Although ISAM can be used to drive the 3350 disk unit, it is not nearly as efficient, and tests have shown that up to one-third of the pack's storage capacity is lost with ISAM. Even in the case of the more currently popular 3330 disk pack, VSAM utilizes the available space more efficiently than does ISAM. Consider, for example, the following actual case-study, which demonstrates clearly the space-saving efficiency of VSAM over ISAM.

In April of 1978, a leading metropolitan police department in the mid-Atlantic region converted its files from ISAM to VSAM. The system is built around an IBM 370-148 mainframe with a configuration of twelve 3330 disk units attached. The particular file cited here is the department's Wanted Persons file, which at the time of conversion held 83,000 records and required 68 cylinders of disk space. That same file, when converted to VSAM, required only 58.5 cylinders of disk space, which represents a reduction of 15%. Even more important, however, was the saving realized through the freeing of allocated disk space. Although the Wanted Persons file under ISAM used 68 cylinders for actual record storage, an additional 132 cylinders had to be allocated to provide for file growth over the next several years. In effect, these 132 cylinders of reserved space represented a sizable waste of resources, since they couldn't be used for any other file application. But with VSAM, it is not necessary to allocate reserve space for file growth. VSAM acquires disk space only as needed from the floating pool of free space that is available to any application. When a particular file needs extra extents, VSAM fetches them dynamically from the available data space so that there is no need to hold extra areas of reserved disk space for each application. This is not to say that VSAM does not require a space reserve to provide for future file growth. It does. However, VSAM manages its reserve a lot more efficiently than ISAM does.

Another important cost-offsetting feature is experienced in the area of file maintenance. Recall that VSAM, by virtue of its capability to reuse deleted record space, considerably reduces and, in some cases, eliminates the need for file reorganizations. In those applications in which there is little or no net file growth, the file can go indefinitely without being reorganized. Even in those applications in which there is considerable net file growth, file reorganizations may still be needed but not as often as would be needed with ISAM. This is not to say that VSAM requires less file maintenance than ISAM. It is just that VSAM performs its file-maintenance function dynamically in a continuing interactive process that is part and parcel of its normal file-management function. ISAM, on the other hand, requires that the file be shut down for maintenance, and it is this resulting downtime that makes file reorgs expensive in terms of reduced file accessibility and in terms of the amount of background processing needed to accomplish the reorg. VSAM files rarely have to be shut down for maintenance, and the few reorgs that may be necessary can be accomplished with easy-to-use utility programs that are part of the Access Method Services package. These utility programs are covered in detail in a later chapter.

A third cost offsetting feature derives from the VSAM characteristic that its file-maintenance function is for the most part internal to the system and, hence, requires fewer manual steps. This means less operator involvement and, therefore, less risk of losing the file to operator error. It is difficult to place a dollar value on a particular file, but when that file is thought of in terms of the amount of time and work that went into its creation, it is not hard to see how costly the loss of that file can be, not to mention the ensuing embarrassment. But with the added file security that VSAM provides, the chances that a data-processing manager will have to face that kind of loss are considerably lessened.

VSAM File Support

VSAM files, or *data sets* as they are frequently called, are of three main kinds: (1) *entry-sequenced,* (2) *key-sequenced, and* (3) *relative-record. Entry-sequenced data sets* are those files whose records are kept in the same physical order in which they are entered. These data sets are nothing more than ordinary sequential files. For example, a record with key sequence number 5 will precede in physical order on the file a record with key sequence number 4, if record 5 was entered before record 4. With entry-sequenced data sets, then, records are time-sequenced rather than key-sequenced.

Key-sequenced data sets are those files whose records are kept in sequential order according to record key. Here, as in ISAM, there is a definite relationship between the sequential order of a record key and the logical order of that record on the file. The record with key sequence number 5 used in the above example will always follow in logical order the record with key sequence number 4. In other words, records on key-sequenced data sets are kept strictly in order of record key regardless of entry sequence. Note that this file structure is completely analogous to the indexed sequential structure of ISAM files. How key-sequenced files are organized under VSAM is a question that will be dealt with in the next chapter.

Relative-record data sets are direct-access files whose records are stored according to their relative record numbers. The address of each record is computed relative to the address of the first record on the file. There are several drawbacks of relative-record files that should be noted. First, they do not support variable-length records and, second, relative-record files require that the user programs accessing them keep track of the record addresses. These drawbacks are not experienced with the first two file structures. Both entry-sequenced and key-sequenced data sets support variable-length records and key-sequenced data sets keep track of record keys so there is no need for the application programmer to concern himself with record addresses.

Implementing the ISAM-to-VSAM Conversion

The most important question that the data-processing manager must deal with once the decision to convert from ISAM to VSAM has been made is how the conversion can be done most expeditiously at the least expense and inconvenience to the user. There is no simple answer to this question. Much will depend on the personnel resources at the manager's disposal. One of the options available to the manager, particularly if he is hard-pressed for programming personnel, is to engage an independent software firm to do the conversion. The cost of this, however, can be exorbitant and, worse yet, the manager will lose the opportunity to afford his technical staff the benefit of first-hand experience with VSAM. The do-it-yourself approach is probably the most sensible. In the end, the conversion will have cost less and also the technical staff will have had an invaluable learning experience in the area of VSAM techniques and practices.

If management decides to do the conversion in-house, drawing on its own resources, then the question arises of how to plan and implement the conversion effort. Several alternatives are available. One of them is the committee approach wherein a panel of in-house experts convene to study the applications that are candidates for conversion to VSAM. This committee would then formulate a strategy and submit its recommendations to the data-processing manager, who can either accept or reject the recommended strategy. In all probability, he would accept the strategy and implement it with the hope of completing the task in several months.

This seems all well and good, but there is one serious shortcoming with this approach, and that is getting the committee members to agree on a specific strategy. More often than not they will become hopelessly deadlocked on small technical details and never arrive at a mutually acceptable plan for implementation. It is an all too well-known truism in data processing that two or more data technicians attempting to work their way through a forest will invariably stray from the beaten path and become hopelessly entangled in the underbrush. It would profit the manager to give no more than fleeting consideration to the committee approach. In the world of data processing, management or anything else by committee usually results in failure or limited success at best.

What approach, then, should the manager take to ensure successful implementation of the ISAM-to-VSAM conversion? The best approach, at least in the authors' experience, is the project approach with a project leader reporting directly to the data-processing manager. The project leader should be selected from the pool of in-house technical personnel, and little concern should be expended over that individual's relative standing in the installation. The project leader should be selected primarily on the basis of his technical ability and on his overall knowledge of the system and its applications. He needn't be a systems programmer as such, but he should have a basic understanding of file structures and data bases. Even more important, the candidate for the position should have a broad knowledge of most, if not all, of the user applications. This knowledge will pay handsome dividends in the way of time savings when it comes to identifying all of the files and associated programs that must be modified in order to implement the conversion to VSAM. Finally, the project leader should be someone with the knack for getting along well with his co-workers, since much of his efforts will be directed toward selling the idea of VSAM to the rest of the technical staff.

Once the project leader has been selected, he should be permitted a direct line of communication to the data-processing manager. Furthermore, he should be responsible to the manager alone and not be handicapped by the inconvenience of having to report to intermediate managers or other supervisors. The conversion project is far too important to let it be jeopardized by the intrusions of middle managers, who will in all likelihood have their own private notions of how the job should be done. This arrangement, which has come into vogue under the name of *matrix management*, can be quite effective, but it must have total top-down commitment. The project leader cannot be effective without the complete backing of management.

As soon as possible after the project leader has been selected, he should receive formal training in VSAM. The manufacturer offers several good courses in VSAM concepts and techniques that are designed to impart a thorough working knowledge of this file-management facility to the trainee. It is almost imperative that the project leader have this specialized training. He might not return from it with all of the answers, but at least he will have developed enough savvy to do independent research from technical manuals once back at the shop. One more thing should be said about training, and it is this: It is not necessary to send all programming personnel who will be working on the conversion project to school for this specialized training. It is quite expensive and cost ineffective, since anything the application programmer has to know about VSAM can be gotten from the later chapters of this book or from the already trained project leader.

With the selection and training of the project leader out of the way, and with the installation of the VSAM package, the conversion process is ready for implementation. The question now is, "How?" Generally, the conversion should be done one application at a time. Begin with the easier applications, and, as experience working with VSAM is gained, move on to the more complex applications. Save for last those applications that share several common data bases. By then, the conversion team will have a sizable store of knowledge and know-how to draw upon, thereby making the task that much simpler.

What is the best plan of action to adopt in order to implement the conversion to VSAM most expeditiously? That, of course, depends on many factors including previous commitments to the user, personnel availability, and machine time availability for the purpose of testing. But, whatever the plan of action, a "best" one will comprise the following steps:

1. Identify all files that must be converted to VSAM together with the associated programs. Divide the files and associated programs into groups by application.
2. For each program within a particular application, identify the nature and extent of the necessary coding changes. Some programs may require nothing more than a change in the SELECT and FD statements. Others may require major surgery in the logic portions of the program depending on the program function. Much of the conversion project's success will depend on knowing exactly how much work must be done on each program.
3. Document, on an application-by-application basis, all work that has to be done in order to accomplish the conversion. Obtain approvals when needed.
4. Select technical personnel on a rotating basis to do the programming modifications as defined by the project leader. Make sure that the assigned personnel understand that, in the beginning, they will have to follow the project leader's instructions to the letter. As they gain experience, they will become more adept and will be able to work independently with VSAM.
5. When the coding changes are completed and when the new VSAM files are defined and created in the test system, test each applica-