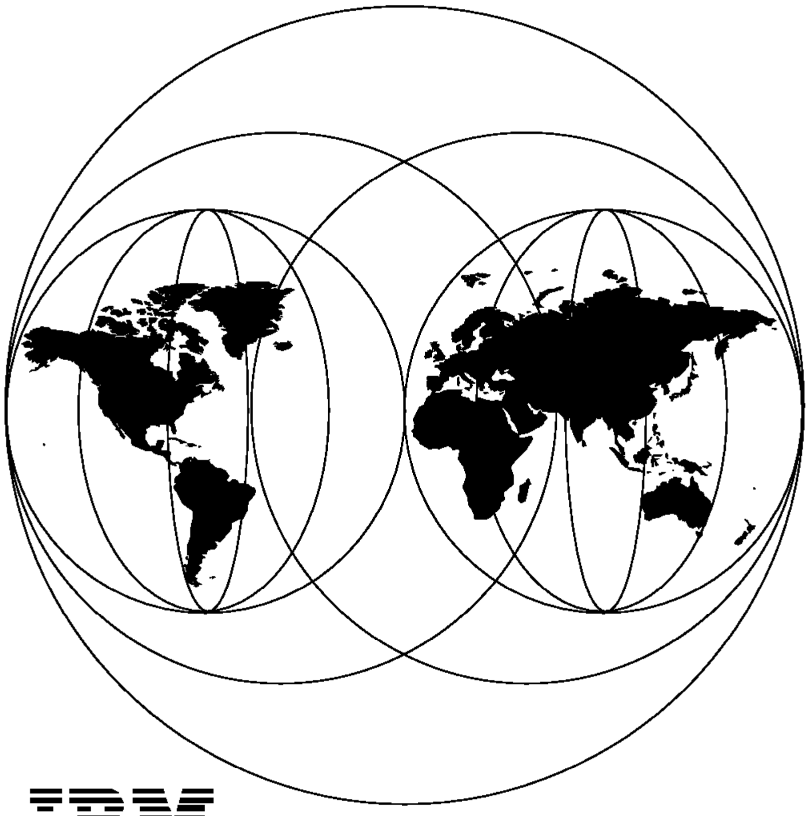
SG24-4931-00

How to Get to DB2 from VSE/VSAM Using DB2 VSAM Transparency for VSE/ESA

April 1997



IBM

International Technical Support Organization
Boeblingen Center

International Technical Support Organization

SG24-4931-00

How to Get to DB2 from VSE/VSAM Using DB2 VSAM Transparency for VSE/ESA

April 1997

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix C, “Special Notices” on page 113.

First Edition (April 1997)

This edition applies to Version 5 Release 1 of DB2 VSAM Transparency for VSE/ESA, Program Number 5697-B88, for use with the VSE/ESA operating system.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. 3222 Building 71032-02
Postfach 1380
71032 Böblingen, Germany

Contents

Figures

Preface

Many customers have already recognized the advantages of DB2 Server for VSE & VM over VSE/VSAM, but could not spend the effort to convert their business data and applications. With DB2 VSAM Transparency there is a tool that eases conversion to DB2. After the manual input of the data structures in VSAM and DB2, it performs the data migration automatically and offers transparent access to DB2 from the unchanged VSAM applications. DB2 VSAM Transparency can be used as a first step to exploit DB2; the applications can later be modified step by step to further optimize the data and application structure to the individual needs.

This redbook points to the critical areas in a conversion process, and in detail explains the usage of the DB2 VSAM Transparency. It describes the steps in using data migration and application Transparency, puts them in the context of the overall conversion process and gives invaluable hints and tips.

The redbook was written primarily for application programmers and system programmers, but also for database administrators, managers and all those who are responsible for planning and performing a database conversion. Basic knowledge of VSE, DB2, and application programming is assumed.

The Team That Wrote This Redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization Böblingen Center.

Gys Brummer from IBM South Africa.

Vijaykumar Singh from ML Sultan Technikon in Durban, South Africa.

Clara Yu from IBM China.

Eberhard Lange from the International Technical Support Organization Böblingen Center was the project leader.

We want to especially thank **Rolf Löben** from IBM Germany for his invaluable advice.

Comments Welcome

Your comments are important to us!

We want our redbooks to be as helpful as possible. Please send us your comments about this or other redbooks in one of the following ways:

- Fax the evaluation form found in "ITSO Redbook Evaluation" on page 129 to the fax number shown on the form.
- Use the electronic evaluation form found on the Redbooks Home Pages at the following URLs:

For Internet users	`http://www.redbooks.ibm.com`
For IBM Intranet users	`http://w3.itso.ibm.com/redbooks`

- Send us a note at the following address:

  `redbook@vnet.ibm.com`

Part 1. General Conversion Considerations

Part 1, "General Conversion Considerations" describes the conversion process in general. It highlights the problem areas and gives an idea where DB2 VSAM Transparency for VSE/ESA can help.

- Chapter 1, "Introduction"

  This chapter is an executive overview of a conversion from VSAM to DB2 on the VSE/ESA platform.

- Chapter 2, "Planning for Conversion"

  This chapter gives an overview of the planning necessary for a successful conversion. This includes aspects such as project phases, conversion methods, inventory needs, and coexistence strategies.

- Chapter 3, "Database Design"

  This chapter describes the task of converting VSAM files to a relational design.

- Chapter 4, "Testing"

  This chapter details testing methodology and some procedures with a concentration on system integration testing.

Chapter 1. Introduction

Converting to a database system is a major decision. It affects business flexibility, costs, and efficiency. Many customers have already recognized the advantages that DB2 Server for VSE & VM offers over keeping the enterprise data in VSE/VSAM files, but have lacked the chance to convert because this meant a large effort.

This chapter is an executive overview of a conversion from VSAM to DB2 on the VSE/ESA platform.

In this manual we use the term “conversion” for the whole project of moving data and applications from VSAM to DB2. The word “migration” names the process of moving the data from VSAM into DB2 tables.

1.1 Value of DB2 over VSAM

The advantage of a relational database lies in a data structure which is easy to understand. Therefore, the data and applications can be more quickly and cheaply adapted to the changing needs of your business. This is achieved with full integrity and with much higher consistency of your data than you can ever achieve with VSAM.

Additionally, a DB2 database combined with other IBM software provides you with the capability to build:

- Data warehouses
- Executive information systems
- Data marts
- Client/server applications
- Network computing applications
- Mobile computing applications.

To summarize:

A DB2 database makes *information* out of your *data*

Naturally, you will be looking for a way to get these advantages while leveraging your existing assets in information and applications. This can be achieved, as you will see, using the DB2 VSAM Transparency for VSE/ESA.

1.2 Management and Personnel

It is obvious that the project manager is a key person; many conversion projects failed because the project manager was not suited for the job.

Less obvious is the following fact: For a successful conversion it is crucial that the management understands the need for the conversion and backs the efforts and the costs involved. If the upper management does not support the project, a conversion project cannot be successful. The reason for this lies in the large effort, the long time and huge costs involved with it, and in the amount of problems that can arise even in spite of a thorough and careful planning. All this may lead the management to have high expectations which should be fulfilled in the short run - but they are not.

It is important that the management understands the purpose of the conversion to the relational database management system and values its **long** term advantages. Since in the short term there are high costs involved and no big advantages visible (actually none in the first phase), such projects often are deferred to never, or are begun and fail because of lacking management backup.

1.3 Conversion Effort

Disregarding differences between individual customer installations, generally the overall effort spreads over **9 months** and can be estimated as follows:

30%	Planning
12%	Data Migration
8%	Application Conversion
50%	Test

1.3.1 Planning

For a conversion project to be finished successfully, major effort has to be spent on careful planning. Therefore, after the first phase of a study of typically around a week, a second phase has to follow making a pilot and a detailed analysis of the feasibility of the strategy. This phase can take several months. Only then can the third phase of the actual implementation be started.

Although the purpose of a conversion normally lies in the feasibility of extensions, changes and new applications, the implementation of any application change has to wait until the end of the conversion. It is important that **no changes at all** are intermixed with the conversion project; they have to be staged until the conversion project is finished. Otherwise testing of the conversion will not only become difficult and more expensive, but a complete test with high quality becomes impossible.

1.3.2 Data Migration

The effort that is involved to migrate the data can be split into the following areas:

1. Describe the current structure of the VSAM data.
2. Define the future data structure in DB2.
3. Set up the database and implement the structure.
4. Migrate the data from VSAM to DB2.
5. Repair the data.

Define the Current Structure

The first step can be very time consuming. If there is a detailed description of all the fields of a VSAM data set, including all the special meanings and multiple different formats the same columns can have depending on various conditions, it must be made sure that this description is correct and up to date, in synchronization with the applications accessing the one VSAM file to be migrated.

But often, such a description does not exist in a concise form so that the information must be extracted from all applications reading and writing the VSAM file. This can be a huge task.

The time spent on this is not at all lost because without this information no extension to any existing business data or application could be made anyway. Therefore, from a management point of view, this effort should not be rated as part of the conversion effort but rather as a necessary step in further developing the data structure and the applications. The conversion project is just a means to bring this step forward because it needs this description as a prerequisite.

Repair the Data

Also the last step can be very time consuming. Putting the data into a more structured, more logical form, for example in a “normal form” (normalization), very often leads to the detection of data errors. Fixing these can take a lot of time and is dependent on the quality and consistency of the data.

1.3.3 Application Conversion

To manually change all the data accessing modules at first glance might be considered the largest effort because it can most easily be imagined as complex. This can be assisted by tools so that, compared to the other tasks, this is the smallest effort. This fact shows the importance of good planning and testing.

1.3.4 Testing

Testing must be planned from the beginning. Before the first migration is done, all test tools must be available. The test tools must be run at least once before the actual migration is started. The following areas must be covered by the tests:

- Function
- Performance
- Concurrency

Function Test

The test tools must be able to prove the correct and complete functionality of the applications after the conversion. Alternatively, it can be tested that the application before and after the change do the same - if the applications before were in a good shape and had no problems.

In the most cases this can be considered the easiest part.

Performance Test

Since the relational database management system performs many tasks, it takes some processing capacity. Therefore, the applications have to be run and both response time and processor capacity have to be verified. On the other hand, applications can be relieved of many tasks that are performed by DB2.

Concurrency Test

More complex than the pure performance and capacity analysis is to test the behavior of the new applications and the relational database management system on concurrent access to data. DB2 can lock data access in a more granular way than VSAM. Therefore, generally concurrency can be improved when migrating to DB2.

1.4 Problem Area: Coexistence

During the conversion process, there will be data and programs that need to span both VSAM and DB2 concurrently. In some cases, this may be a short step towards full conversion, in other cases coexistence may be a way of life for a longer period of time. The various choices to overcome this include:

- Extract data from existing VSAM files on a regular basis, for example for query and decision support systems. No update to the DB2 data from the applications is allowed. The update from the VSAM copy to DB2 has to occur regularly.
- Update of one database management system with read-only of the other. Such applications can heavily increase the time and costs of a conversion project and should be avoided where possible.
- Updates to both database management systems.

1.5 Consultants

IBM and many consultants in many countries are experienced and willing to help you during a conversion from VSAM to DB2. There are also various tools to assist you in the one or the other step. For example, a vendor or consultant can have tools to test the function of an application, guaranteeing that all paths are run through. This can relieve you with the function test.

It is recommended to involve external help at least in planning for a conversion project. The savings due to the experience of a consultant having performed several conversions can by far outweigh the costs.

You can call IBM for consulting, or for closing the contact to another experienced consultant. Or you can contact a local user group or one of the worldwide user groups such as GUIDE, SHARE or WAVV to get such contacts. IBM is also willing to help you to get in touch with user groups in your area.

1.6 Positioning of DB2 VSAM Transparency for VSE/ESA

With DB2 VSAM Transparency for VSE/ESA there is a tool that can assist dramatically in a conversion. With a comparatively low effort you can migrate all your data and, leaving your applications unchanged, transparently access DB2. Because all your data is already in DB2, coexistence problems can be avoided.

The data structure can already be modified to a large extent during the above migration step. The data structure and the applications can later be modified step by step to further optimize performance and adapt to the individual customer′s needs. For example, it is possible to keep some VSAM accesses while other VSAM accesses are changed to DB2 accesses within the same application. In this case, the older VSAM accesses continue to be transparently intercepted, whereas the changed application statements can directly access DB2 data using SQL bypassing Transparency.

Using DB2 VSAM Transparency might therefore reduce the effort for an external consultant.

Chapter 2. Planning for Conversion

This chapter gives an overview of the planning necessary for a successful conversion. This includes aspects such as project phases, conversion methods, inventory needs, and coexistence strategies. It is strongly recommended to use the ITSO Redbook ″Planning for Conversion to the DB2 Family:Methodology and Practice″, GG24-4445 for that purpose. It gives a more detailed description than this overview. Although it was written for a VSAM to DB2 conversion on MVS, most of it is valid for a conversion from VSAM to DB2 on VSE/ESA. Some of that information is repeated here to help customers to:

- Get an overview of the conversion process.
- Learn the steps to be performed.
- Value the DB2 VSAM Transparency for VSE/ESA within that context.

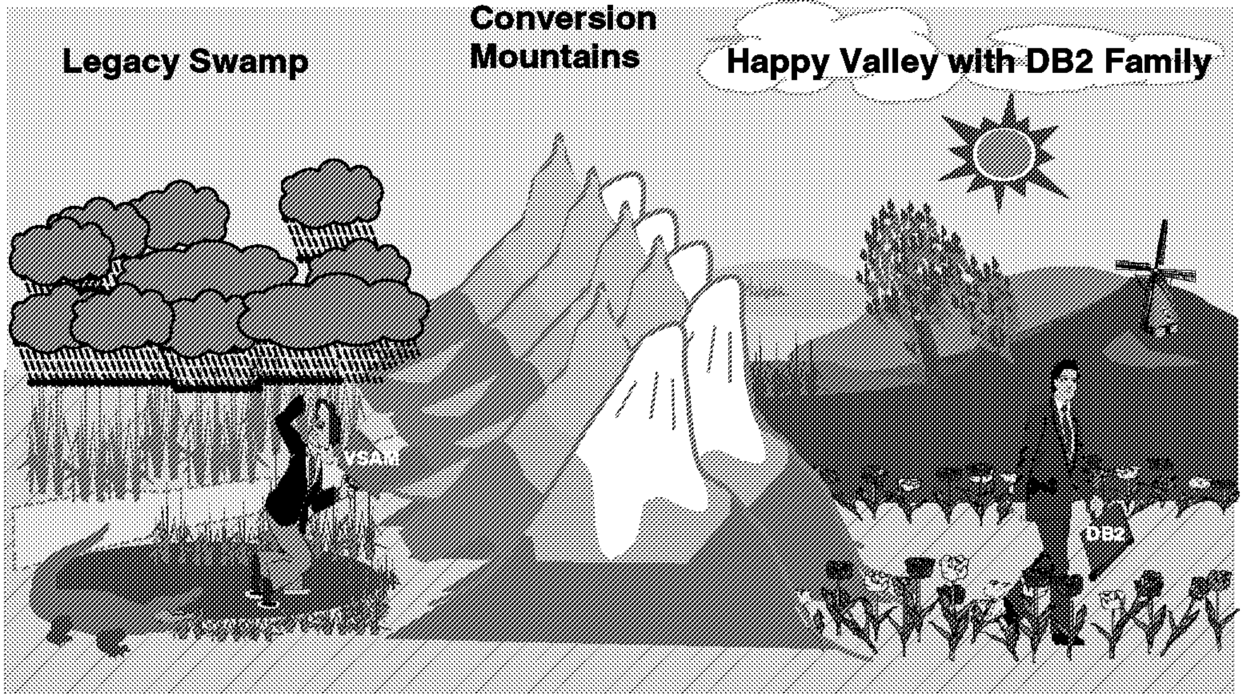


Figure 1. The Path through the Mountains

2.1 Why Relational

Conversion of an application from VSAM, a file management system, to DB2, a relational database management system, is not necessarily a productive effort. There are however, a number of relational advantages which justify the conversion of existing applications with little or no change in function. In the case of converting to DB2, these reasons are essentially the reasons that cause the choice of DB2 for new applications. Typically, the choice is to do new application development and major rewrites with DB2 and permit existing applications to atrophy. Additional factors come into play: data interchange and consistency between the two systems can also be costly and complex.

2.1.1 Value

Problems with VSAM

In many cases, the problems with heritage systems are that they are vital and form a huge investment, but lack flexibility to implement changes or to add new applications. Maintenance costs can take up to 80% of the development budget, but data processing is under pressure to cut costs.

Value of Relational

Generally, the relational model provides significant value for an application implemented in DB2. It can be shown to add value and reduce cost.

- Data in a relational database management system such as DB2 is understandable for end users and programmers.
- Data is extendable.
- Data in DB2 is globally accessible, from batch, online transactions, interactive end users and remote systems.
- Data replication to many platforms is possible, allowing for better performance and availability by remote systems.
- SQL language applications can easily be ported.
- DB2 offers good recovery management.
- DB2 offers the ability to protect various levels of data. You can keep all your data in a centralized table and still have the flexibility of restricting access to sensitive fields.

This leads to the following advantages:

- Reduced maintenance costs for applications.
- Reduced operational effort.
- The possibility to prepare for the year 2000.
- Additional applications are offered, such as decision support tools.
- The capability to build:
  1. Data warehouses
  2. Executive information systems
  3. Data marts
  4. Client/server applications
  5. Network computing applications
  6. Mobile computing applications.
- Higher productivity for the end users.

2.1.2 Inhibitors

Although DB2 is obviously a much better platform for data, various reasons might be given for not converting to DB2:

- Costs of conversion
- Lack of skill
- Uncertainty about costs and results
- Resistance from personnel and executives

- Lack of comprehensive help
- Impact on end users

But IBM and many consultants offer comprehensive help, tools can reduce the conversion costs dramatically, and the systematic analysis of the value of a conversion should help to overcome resistance. Good planning can size the costs, minimize the risks and reduce the impact on end users.

Costs

The additional data processing costs caused by a new database platform with additional applications can be split into the following areas:

1. One-time conversion costs. For an effort split in percentage, see 1.3, “Conversion Effort” on page 4. Depending on the individual customer situation, the one-time conversion costs may be caused by one or more of the following:
   - Additional people
   - Consultants to assist in planning or execution of parts of the conversion.
   - Additional disk space during the conversion time for tools and duplicate copies of application and data, see 2.7.4, “DASD Requirements” on page 22.
   - Additional processor capacity (including follow-on costs mentioned below) to perform the conversion of data and applications, see 2.7.3, “Processor Requirements” on page 22.
2. Costs for additional software: DB2, related features, tools and applications.
3. Possibly a larger processor, depending on the current processor size and load. If this happens, increased software licence costs for existing software will follow.
4. Additional disk space for the data. A relational database management system creates control space, redundancy and requires some spare disk space. Additionally, the new software will take extra disk space.
5. Other. This list is just to give you some ideas what areas should be considered. It is not necessarily complete and can vary to a great degree according to the individual customer environment and situation.

2.2 Conversion Principles

The following is a recommended strategy for a conversion:

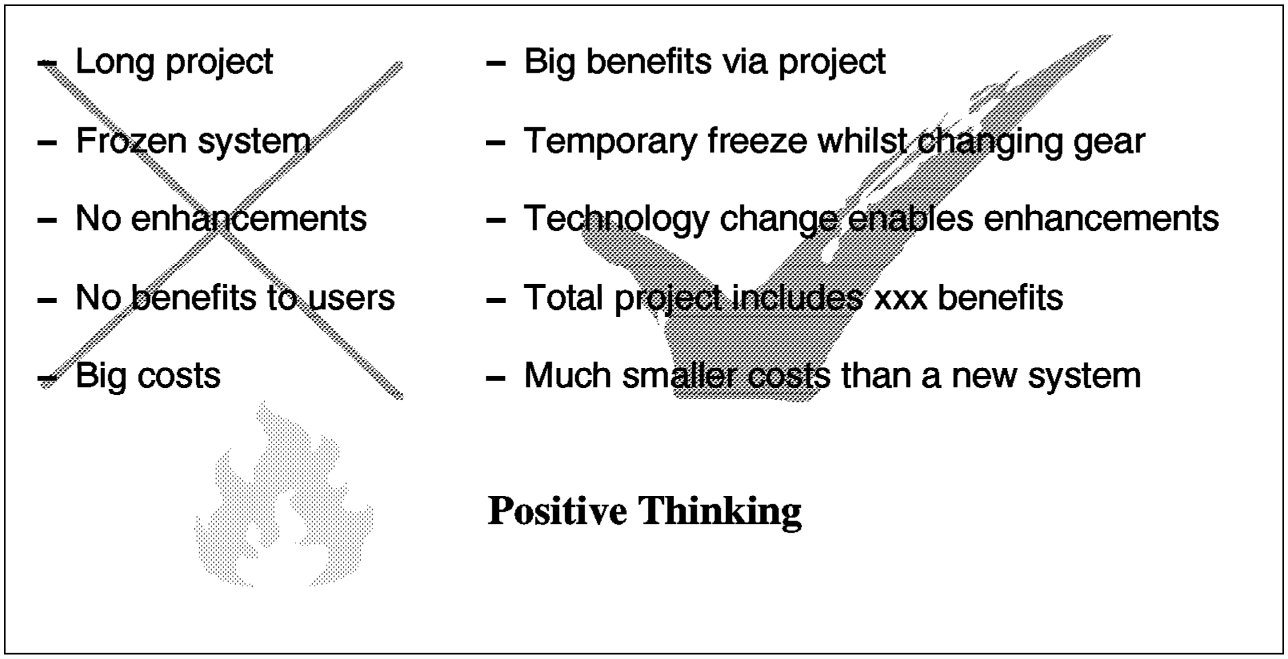
- Decide why you want to move.
  – Use business needs to drive a strategy rather than technology.
- Decide where you are now.
- Decide where you want to be.
- Evaluate the alternatives.
- Decide the best route, taking everything into account.
  – Use a phased approach rather than a giant leap.

For each of the phases, clearly identify:

- Objectives
- Inputs
- Deliverables

- Checkpoints

This means especially, that the end of the conversion process needs to be clearly defined.



- ~~Long project~~
- ~~Frozen system~~
- ~~No enhancements~~
- ~~No benefits to users~~
- ~~Big costs~~

- Big benefits via project
- Temporary freeze whilst changing gear
- Technology change enables enhancements
- Total project includes xxx benefits
- Much smaller costs than a new system

Positive Thinking

Figure 2. Positive Thinking

2.2.1 Conversion Phases

The steps in Figure 3 on page 13 have proven useful in many conversion projects:

Phase 1 "Conversion Assessment Study" means a portfolio analysis of the system to ensure that the start point is defined, the scope is understood, the client's objectives can be met and the appropriate strategy is set.

The pilot in Phase 2 is to provide proof of the concept by validating the technical findings of the assessment study. The pilot completes the analysis of Phase 1 and involves setting up a plan.

Phase 3 is the main conversion. If the project is well-managed, it can be a smooth and efficient process.

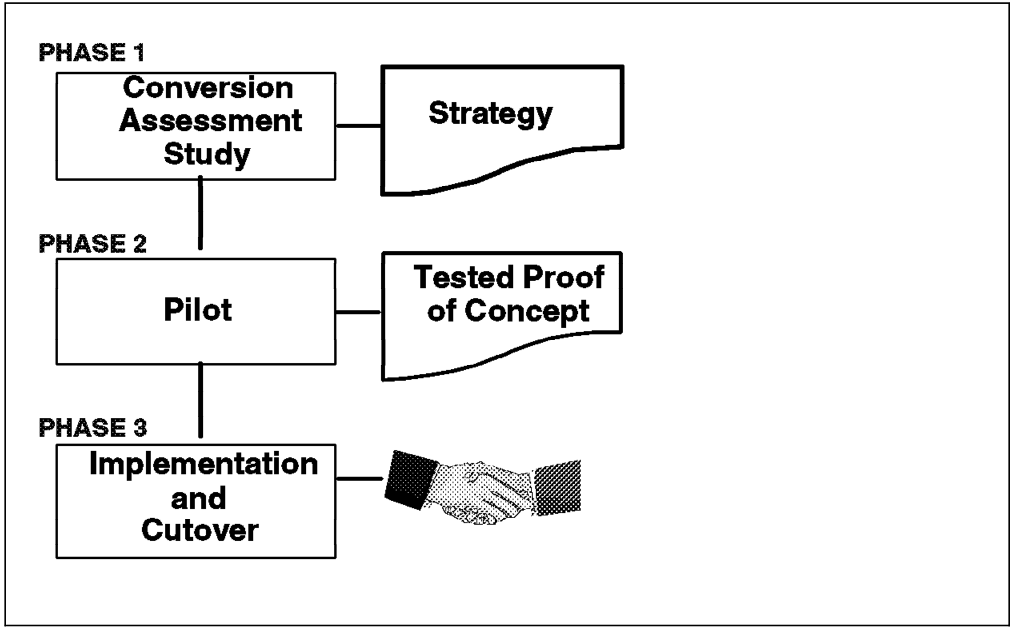


Figure 3. Conversion Phases

2.2.2 Switchover Strategies

During the study, various strategies are examined in order to find the best one for a particular situation.

- Management Information System

  It may be questioned if report writers are to be converted, or to be replaced by reporting systems such as QMF.

- Big-Bang

  When you switch over from the old to the new system from Saturday to Sunday, you take a high risk. Therefore, a fallback plan is required. On the other hand, coexistence problems do not appear, which can save a lot of extra work.

- Piece at a time

  One application or group of applications is moved at a time. What if a program needs access to data under control of both technologies? This kind of coexistence problem can require a high investment in code which will be dropped afterwards.

2.2.3 Functional Changes

It is understood that one of the reasons for the conversion is to more readily accommodate changes to the database.

However, testing the conversion is a significant part of the overall effort. The effect of the addition of new functions with changes to data, programs, input and output dramatically complicates the effort of verifying the conversion. The easiest means of testing the converted system is to use identical input to both the former VSAM system and the new DB2 system and then compare the data and output of both systems. It is desirable to automate this as much as possible.

We strongly recommend, therefore, that no changes be permitted which will affect these variables during the conversion. As needs are recognized during the conversion, they should be logged for later implementation. Failure to follow this recommendation will extend the period of the conversion, increase the cost and might eventually cause the conversion project to fail.

2.3 Conversion Methods

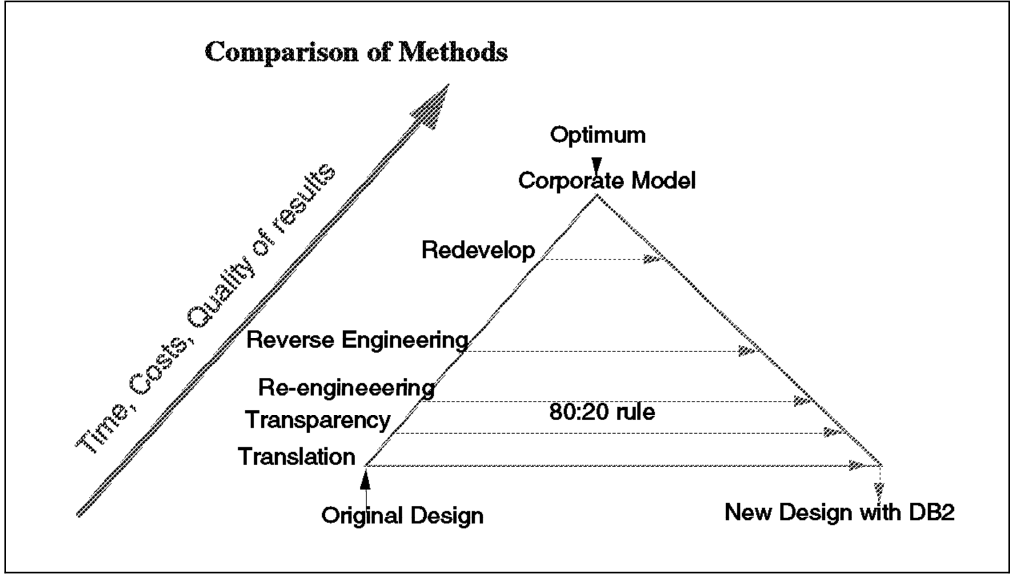


Figure 4. Conversion Methods in Perspective

To decide upon the right conversion method, it is important to understand the different approaches and their results.

1. Translation is the easiest to understand. It takes the VSAM calls and translates them to DB2. The data structure is left completely unchanged, this means, the VSAM records are stored as long character strings into DB2 tables.

   This is the easiest method, but does offer few of the advantages of DB2.

2. Transparency intercepts VSAM calls and re-routes them to database calls.

   The data can be remodeled to suit a relational structure; using Transparency, the application can function as before. Performance can be an issue because applications now contain much used but unneeded code. Applications can later on be modified step by step without coexistence issues.

   - Data propagation is a form of Transparency. It keeps two copies of the data, and updates are propagated (automatically or at regular times manually) to the new platform.

3. Re-engineering means minimum change consistent with DB2. This means changing enough to remove dependencies on the old structures and become compatible with good DB2 design, but not rethinking the whole data design.

   It offers the main advantages of DB2 and allows limited redesign to take advantage of special situations, but takes more time than translation. The data design still reflects old data structures.

4. Reverse engineering means to capture the old designs into models, modify them, and generate new programs and new database designs.

   This method requires that really effective, good tools are available. There are some, but they must be compatible with the programming languages used. This method needs more development time, but future maintenance will be easier. Testing for identical results is unlikely to be useful.

5. Redevelopment of the applications can be needed if old code is in a poor state. Data models are re-thought from scratch.

   This means a huge investment with longer development times, but with the chance to use a 4th generation language and new tools. Testing for identical results is unlikely to be useful.

6. The Corporate Model means that not only the applications are redeveloped but, from the modeling of the business and data, a completely new structure of the data and the applications is created.

Some intermixtures between methods are possible, for example **application** re-engineering combined with reverse re-engineering of the **data**. For more details, refer to the redbook ″Planning for Conversion to the DB2 Family:Methodology and Practice″, GG24-4445.

The “80:20 rule” should be a guideline to help decide: As in many other areas, you can achieve 80% of the benefits with 20% of the costs.

2.4 Conversion Personnel

It is obvious that the project manager is a key person; many conversion projects failed because the project manager was not suited for the job.

Less obvious is the fact that for a successful conversion it is crucial to have an executive sponsor who wants and needs the project to succeed. The management must understand the need for the conversion and back the efforts and the costs involved. If the upper management does not support the project, a conversion cannot be successful since in the short term, there are high costs involved and no big advantages visible - in the first phase no advantages at all.

Naturally, technically capable people are needed with various skills; since special skills are involved with a conversion process which will not be needed before or afterwards, it might be wise to use external people assisting the in-house team for the conversion.

2.5 VSAM Application Systems Inventory

In order to prepare accurate estimates of workloads, costs and schedules, it is necessary to set up a thorough inventory of each application to be converted. This section provides direction and guidelines for conducting an inventory of the VSAM applications, analyzing the results of the inventory and ranking each application in the order of conversion difficulty. The data that will be collected and the criteria that apply to that data will help make the final ranking and selection as objective as possible.

If you are a data dictionary user, its reports can be used for some of this inventory. It should be noted that if the data dictionary is current and complete, it can be a major tool during the database design and data conversion phases from VSAM to DB2. The conversion team should consider bringing the data dictionary up-to-date. The two primary advantages of doing so are:

- The capability to obtain quick answers to ″What uses″ type questions, and
- The assurance that all data entities are available for inventory.

The best source of information for the application inventory and resulting analysis may be a VSAM programmer who is knowledgeable in the VSAM data management system and the applications.

We cannot over emphasize the importance of this inventory. ***You should not attempt to do any steps of the conversion without completing this inventory***.

The specific skills needed by members of the conversion team responsible for the inventory are:

- Strong VSAM knowledge, especially of the various data organizations.
- Ability to use VSAM utilities.
- Host language (for example, COBOL) programming knowledge.
- Experience with the VSAM interfaces used in the installation (for example, CSP).
- Ability to retrieve data from the data dictionary, VSAM catalog, CSP Member Specification Library (MSL) and Application Load File (ALF), or other sources as appropriate.

2.5.1 Application Inventory

Minimum documentation points for each application are:

- The actual application name that is to be used for migration.
- A description which is a brief paragraph stating the primary business requirements and functions of the application.

  Other information that should be recorded here is how critical and visible each application is to the business and users.
- Document the number of files for each application. You should be alert to two conditions that could impact the conversion time or complexity. These are:

  1. A very high number of files
  2. Inter-relationships between files.
- File isolation is important when considering an application for conversion. Isolation refers to the degree of sharing of files between applications.
- Document the number of programs for the application by language.

  If COBOL programs are registered in the data dictionary, they can be listed from there. This approach assumes the data contained in the dictionary is current and complete. If not, a physical count against the production library is needed along with details about which applications use which COBOL programs.

  A list of the CSP programs may be generated through the Application Load File Utility (ALFUTIL) or the where-used capability of the List Processor Facility.
- The application should have few outstanding change requests. Users tend to expect changes to be made during conversion. Applying changes during the

conversion increases programming time and can dramatically increase the complexity and the cost of the testing effort.

A better strategy is first to convert the application to relational and later make the needed changes, taking advantage of the productivity and flexibility that the relational database provides.

- Often business issues dictate the timing of the conversion effort. For example, conversion of the budget application should be completed before the fiscal year ends.

  It would be convenient if a simple application could be selected for the first conversion. This gives the staff experience and confidence with minimum risk. However, business priorities may not permit this.

2.5.2 VSAM File Inventory

Difference Between File and Table

When creating the VSAM file inventory, it is important to recognize the difference between a VSAM file and a DB2 table. VSAM files often contain several types of records in the same file. These may be control records, or redefined records containing different information about the same entity. In the relational model, all rows in a table must contain the same columns. Because no variation is permitted, one or more tables are normally defined for each VSAM record type.

Another important difference is the level of data definition. In DB2 each column is defined in the DB2 catalog. In VSAM, data is defined at the record level, and field definitions are left to the individual program. To create an inventory of VSAM fields, each field must have a unique and consistent name. Therefore, special attention must be given to homonyms (different fields having the same names) and to synonyms (a field being addressed by different names). If consistent field (column) names have not been established through a data dictionary or other means, they will have to be established for the purposes of the inventory. Since it may be appropriate to use those same names consistently in the converted programs and as DB2 column names, their selection should be done with care. See 3.3.3, “Data Naming Considerations” on page 29.

Possible sources for these names are data dictionaries, COBOL copy library definitions, other source library definitions, CSP definitions, and COBOL program definitions.

File Inventory Creation

For each application, the files, record types, fields, and indexes need to be identified. VSAM catalog listings can provide a complete list of VSAM files and indexes if more application oriented information is not available or reliable. They also show the location of prime and alternate key fields, which must also be identified, but not the names of those fields. Record types may be derived from COBOL copy library entries, other source libraries, CSP data, or program listings.

For CSP applications, the Application Load File Utility (ALFUTIL) or the where-used capability of the List Processor Facility can greatly facilitate the scanning of applications and the documentation of application to file relationships, as well as record and field definitions.

Caution: In some cases a record in a file may be defined as one long field. In other cases there may be different record types within one file with a flag or indicator that the program must read and understand. If either case exists, investigation of the record layouts in each program accessing this file may be necessary to achieve accurate record descriptions.

The following information is considered important and should be shown for each file as a spreadsheet or table form for easy reading and understanding:

- File name
- File identifier
- Catalog name
- Type of VSAM file
- Record key and alternate index
- Number of record types
- Names of the different record structures
- Names of the source programs accessing the file
- Name or number of the library in which the source programs are stored (name for CSP in VSE library, number of ICCF library)
- For each record structure, list of fields with start positions and length
- For each record structure, number of fields
- Number of records in the file (to calculate dbspace).

2.5.3 Program Inventory

All programs that make up a selected application must be inventoried to ensure complete conversion. This section describes the data that is needed and how to collect it.

All CSP programs can be identified with the Application Load File Utility (ALFUTIL) or the where-used capability of the List File Processor. A complete inventory of the COBOL programs may or may not be available through a data dictionary. If the COBOL programs are not registered in the data dictionary, it may be necessary to list all COBOL programs in the source library, in order to locate the programs that apply to the selected application and record their name.

Hints:

- A pragmatic way to determine which programs are used, is through the accounting information. All important programs will be listed there. Every program that has not been run, for example during the past two years, is not worth converting.
- There are also some vendor tools available that will assist in performing this task.

Information that should be listed for **each application** is as follows:

- Inventory of all batch and online programs by program name.
- JCL of all batch programs.

- CICS transaction id for all online programs and maps.
- Library name or number where source programs are stored.
- For each of the programs within each application, the names of all VSAM files accessed.

2.6 Coexistence Strategies

During the conversion process, there will be data and programs that need to span both VSAM and DB2, as shown in Figure 5.

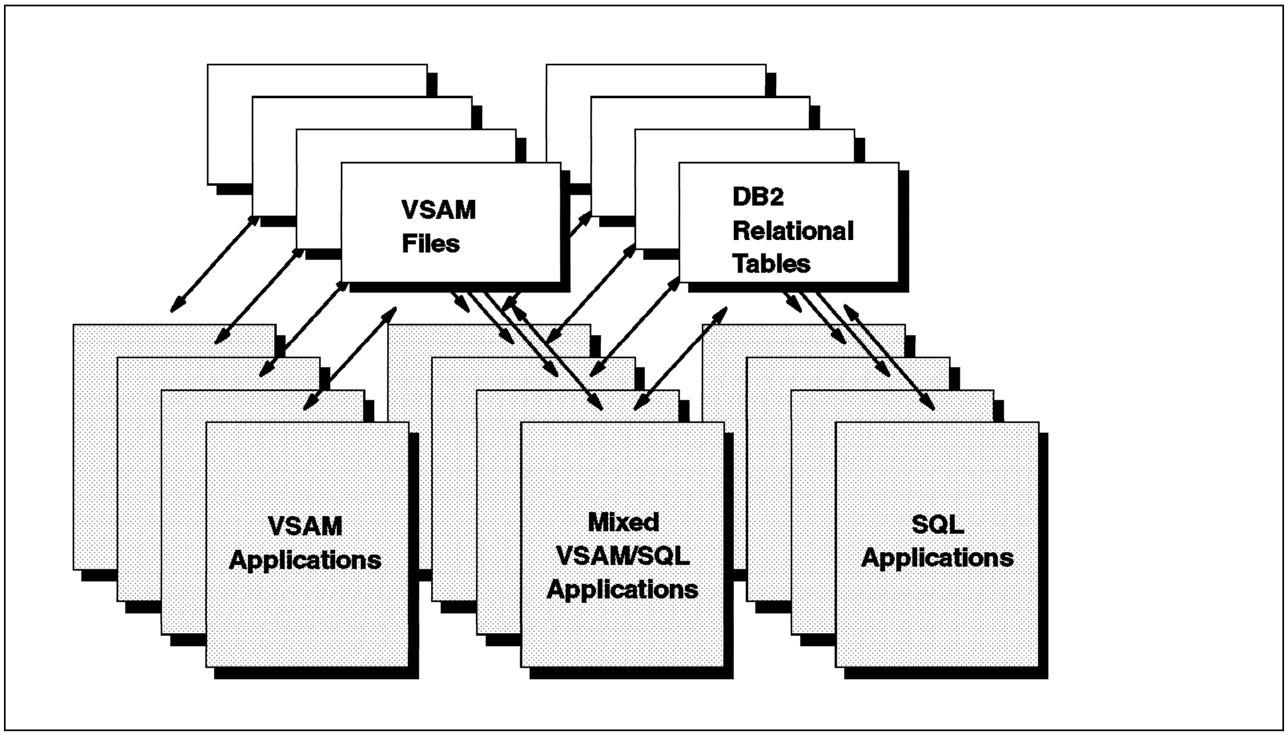


Figure 5. Coexistence Situation

In some instances, coexistence will merely be a stepping stone to full migration but in other cases, coexistence may be a way of life for a longer period of time. Therefore, a plan for dealing with data coexistence issues is outlined here.

There are a number of different approaches or strategies that can be defined as coexistence. The level of complexity and applicability of these strategies varies depending on the installation′s requirement. It is of lower importance whether data is replicated, this means actually duplicate data, or just files of which some have already been migrated and others not.

This section will cover various flavors of coexistence, beginning with the simpler ones and moving to the more complex ones. Most installations will find that as they move through the conversion cycle, they will be implementing several variations of coexistence, depending upon business situations, application types and use of the data.

2.6.1 Coexistence Scenarios

An initial activity for any form of coexistence is the mapping of VSAM file structures into the DB2 tables. This task is described in Chapter 3, “Database Design” on page 27. After data mapping has been decided, the remaining tasks for managing coexistence can be automated to some degree, depending upon the tools and aids in the installation.

Extract Data for Query and Decision Support Systems

The simplest approach is to extract data from existing VSAM files and load it into DB2 tables. The tables could then be accessed by a number of tools for query and decision support applications. This approach might be applied to data that is maintained in programs that are not to be converted immediately, yet the ability to get to the data with end user tools is a requirement.

Update of One Data Management System With Read-only of the Other

These applications (often called “Bridge Applications”) contain data access and logic for both DB2 and VSAM. This scenario may be the most pervasive during the conversion effort because of the inter-relationship of data and the length of the conversion effort. However, the conversion effort will be most cost-effective if the need for these applications can be kept at a minimum because they will require at least some rework when all data has been moved to DB2. Consideration should be given to locking algorithms, commit strategies and recovery methods when mixing the two data management technologies within the same program.

Updates to Both Data Management Systems

Updates to both copies of data occur when data is replicated in VSAM and DB2. The approach is dependent upon the timing of the updates. The first determination that must be made is: must the updates be real time (synchronous), or can a time lag be tolerated. Both approaches are discussed below.

Time Delayed Updates

Time delayed updates of DB2 data can be handled in one of three ways. All methods require a DB2 update program to be written.

1. The first method requires that the VSAM source program be altered and a transaction file be written which reflects the updates. This transaction file would then be the input to a DB2 batch or online updating program. Such a program might be initiated by JCL, time initiated, or be a long running program waiting for data to appear in a queue.
2. A different approach can be used by having the VSAM program spawn an online transaction that will immediately schedule a DB2 updating transaction to cause the corresponding update at transaction commit.
3. If the requirement exists to propagate updates of DB2 data back to VSAM data, the solution is simpler. Since the DB2 programs will be in the process of being written, logic to write a transaction file can be included from the beginning. As the conversion process unfolds and the requirement to update VSAM is discontinued, the code can then be removed or the output file dummied.

Synchronous Updates

Another situation would be transactions that update both VSAM and DB2 synchronously with full integrity. This approach may be a requirement if related production data resides in both DB2 and VSAM and a single transaction must update both.

Programs will have to be changed multiple times. If a program updates only replicated data, once the old VSAM data is no longer required, the logic for updating that data may be removed. If the program updates related data, it must be modified to update using SQL once the data is converted to DB2.

In either case, online programs can be written to update both VSAM and DB2 data concurrently. CICS two-phase commit can be used so that integrity between VSAM and DB2 data can be guaranteed.

Batch programs that update both VSAM and DB2 data must be given special attention. The two-phase commit and dynamic backout of VSAM data provided by CICS (which are required to guarantee data integrity) are not available in batch. If such a batch program abends, the DB2 databases will automatically be restored to the beginning of the program (or the last COMMIT point), but the VSAM file will not. Recovery procedures must be developed and executed very carefully in order to maintain data integrity.

2.7 Conversion Considerations

Many more areas have to be planned for up front to make the conversion successful. Here we will briefly mention the areas of tools, performance and hardware requirements. For more details, and for other areas that are not covered here such as

- education
- end point of the conversion
- recovery concept changes

refer to the redbook ″Planning for Conversion to the DB2 Family:Methodology and Practice″, GG24-4445.

2.7.1 Tools

A number of tools have emerged over the past few years, each of which can help to reduce the effort by automating parts of the process. The areas include:

- Database design tools
- Transparency tools can speed the data move during the conversion process. Applications can be changed later.
- Data movement tools
- Data propagation tools, synchronous and asynchronous
- Testing
  - Online identical function

    Ensure that the new code still provides the same function as the old one.
  - Full testing analysis
    - Ensure that all parts of a program are tested.
    - Ensure that the test data used will test every case possible.

  - Managing multiple tests
  - Batch output comparisons
  - Stress testing of systems

- Real-time performance monitoring

2.7.2 Performance

Performance is often considered to be critical. But compared with the overall question of operating system and database platform, performance is a problem that can be solved - through faster hardware if not otherwise. DB2 has many choices to influence and optimize performance, especially in the database design. Often a small change, even invisible to the end user, can make a large difference in performance. For more details refer to 3.9, “Estimating Performance” on page 47 and the redbook ″SQL/DS Version 3 Release 4 Performance Guide″, GG24-4047.

There is a performance estimator tool for MVS DB2. The tool runs on a workstation. You can fetch it from the download library on the World Wide Web under the IBM software home page at:

`http://www.software.ibm.com`

Support comes from an Internet ID of:

`estimate@vnet.ibm.com`

There is also the IBM VNET id “ESTIMATE at STLVM14.” The DB2 Estimator Version 5 runs with either OS/2 Warp, Windows 3.1, or Win95 and requires 4MB of memory and 5MB of hard disk space, 10MB while installing.

Although performance numbers cannot easily be correlated between MVS and VSE, the relative performance behavior can be estimated with this tool. You can have a first run with a primitive design of a one-to-one relationship of VSAM contents to DB2 table format, for example one large column with characters only, and then with your proposed database design alternatives. You will then be able to predict the relative performance of your designs compared to the primitive design and to one another.

2.7.3 Processor Requirements

In general the processor requirements for conversions are not significantly different from normal development efforts. Actual requirements depend on the specific application.

The final testing and production cutover with both VSAM and DB2 running similar work loads will likely be the peak CPU load. This peak load may be estimated by summing the time for applications running concurrently on both systems.

2.7.4 DASD Requirements

Both VSAM files and DB2 will exist in the system during the conversion. Some portion of the data must be replicated for the purpose of development, test and production cutover. Most development and test may be done with minimal size test data. During the initial phases the amount of test data may remain small. At the point of production cutover, the data used by an application is loaded into DB2 tables. After cutover verification, the VSAM version of the data can be archived, thus minimizing the time when duplicate data is online. Planning to convert data and applications in phases rather than all at once may reduce DASD requirements.

DB2 VSAM Transparency is independent of DASD device type; any DASD device can be used that is supported by the VSE/ESA operating system. In addition, DB2 VSAM Transparency has no specific hardware prerequisites and will function in any environment that supports DB2.

2.8 Conversion With DB2 VSAM Transparency for VSE/ESA

Conversion with DB2 VSAM Transparency principally is done as a conversion without it, but with considerably less effort and risk because you migrate the VSAM files rather than the application programs, as shown in Figure 6.

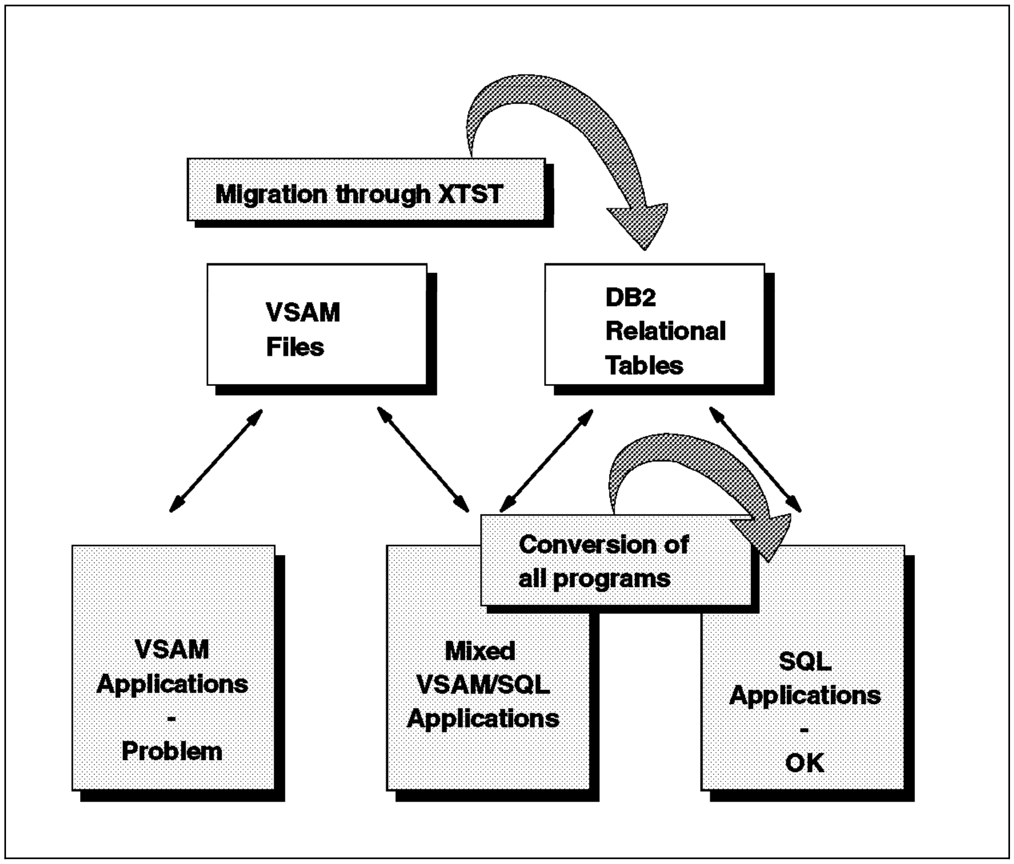

Migration through XTST
VSAM Files
DB2 Relational Tables
Conversion of all programs
VSAM Applications - Problem
Mixed VSAM/SQL Applications
SQL Applications - OK

Figure 6. Migration with DB2 VSAM Transparency

The same project phases apply, and the same aspects have to be considered. It must be checked whether the principle of Transparency fits into the overall project situation, purpose and targets. Especially, the limitations of DB2 VSAM Transparency in database design must be considered, as described in 7.1.3, "Limitations" on page 76. If the limitations apply, you have four choices:

1. Retreat from the requirements.

   This is a choice if the requirement is either not really strong, or it can be fulfilled later.

2. Consider other methods and tools rather than DB2 VSAM Transparency.

   This might be chosen if many applications and data are hit by this limitation and no circumvention is possible. Avoid the duplicate effort of testing.

3. Use DB2 VSAM Transparency and move the remaining design changes into a follow-on project.

This might be useful if you do not have a real alternative in tools available, and many applications and data are hit by this limitation. This might lead you relatively quickly to an intermediate state, but be aware that you duplicate the huge testing effort.

4. Circumvent and change

   Use DB2 VSAM Transparency and include the remaining design changes into in-between and follow-on steps within the same project, because the large testing effort should not be duplicated. For examples, see Chapter 8, "Beyond Transparency" on page 103.

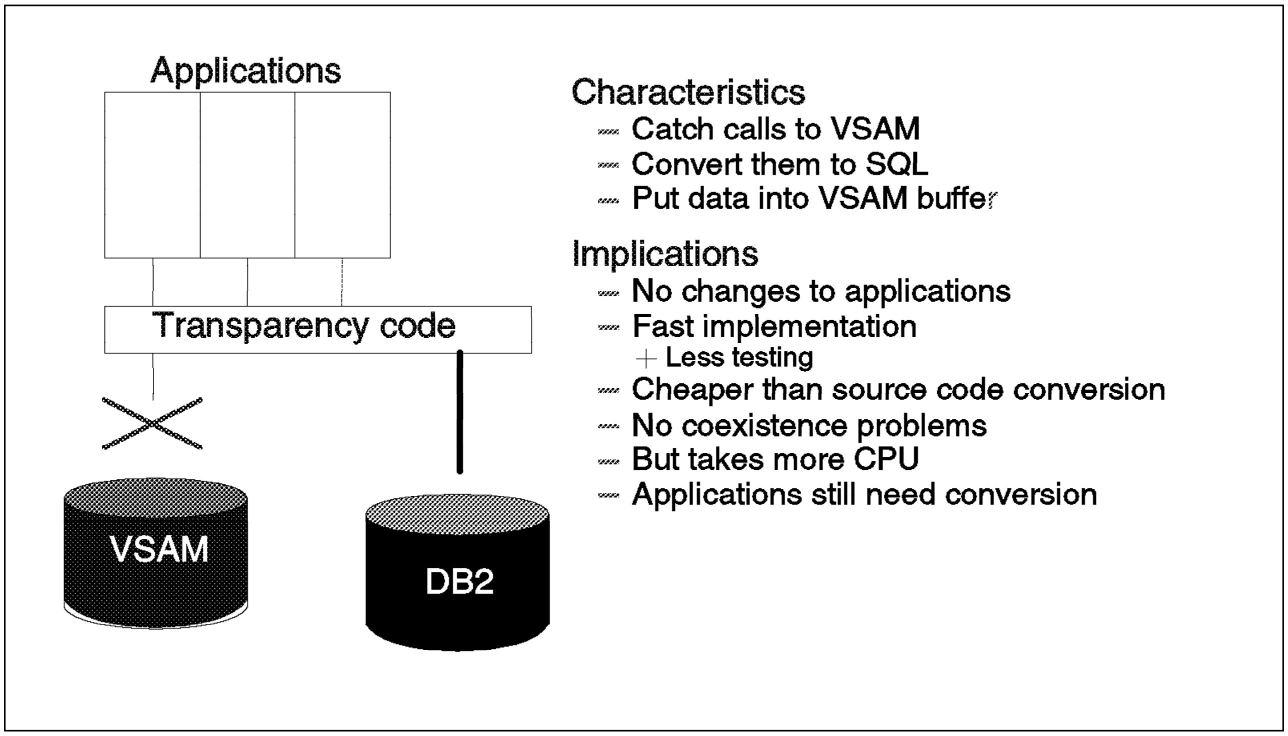


Figure 7. How Transparency Works

The advantages of DB2 VSAM Transparency are:

- The data may be remodeled to suit relational structures.
- Tools may be used for database design, refer to 2.7.1, "Tools" on page 21.
- Once one application suite has been successfully migrated, others can easily follow.
- New applications can be written taking advantage of the new data structures in DB2.
- After Transparency has gone live, each program can be reworked separately to access DB2 directly.

  Hint: In any single application program you can mix:
  - VSAM accesses (to files not enabled for Transparency)
  - Intercepted VSAM accesses (to files enabled for Transparency)
  - SQL accesses to DB2

  These different access modes do not interfere with each other, as shown in Figure 10 on page 60. This means, after migration using Transparency you can change an application even partially, step by step.

- Transparency is suited especially for very large databases, with high availability requirements.
- Transparency offers a lower-risk path because less is changed in any one step and fallback is easy.

The disadvantages of Transparency are:

- Performance is more likely to become an issue because applications now contain much used but unneeded code.
  - But performance issues can be solved, see 2.7.2, “Performance” on page 22.
- The conversion might become longer because it is now in two stages: Taking subsequent steps and achieving good quality and complete conversion may be hard to ensure.
  - But you are on the new platform more quickly and with less risk.

Using DB2 VSAM Transparency lets you take advantage of all DB2 usability improvements, without any reprogramming, recompiling, or relinking. When VSAM programs run under DB2 VSAM Transparency, VSAM access requests are intercepted and redirected to access DB2 data tables. In fact, an application program can access any mixture of:

- Unconverted VSAM data
- Converted VSAM data through DB2 VSAM Transparency
- DB2 tables through native SQL.

This flexibility removes the need for replicated data and for coexistence efforts.

Where coexistence issues may be a problem, Transparency should seriously be considered as a migration aid.

Chapter 3. Database Design

This chapter describes the task of converting VSAM files to a relational design. It covers the translation of both the logical data and the physical structures to DB2 tables with the same meaning. This document will discuss only those techniques that are unique to converting an existing VSAM file to DB2 tables. For general DB2 database design techniques, refer to existing DB2 design guidelines.

3.1 Types of Database Conversion

Conversions may be done in different degrees.

1. **Translation**. Conversion may be of a ″translation″ nature where the VSAM records are mapped directly to DB2 tables.
   - If the data has been designed and maintained as normalized data, this approach will be appropriate.
   - In those cases where data is not normalized, this is unlikely to provide optimum performance characteristics and the resulting system loses the productivity advantages of the relational model. Thus, while the migration is simplified, it is quite possible that the end result will be unsatisfactory for either current performance or future additions.
2. **Compatibility**. As a next level, it is possible to do a simple conversion with minimal analysis of the data. This provides essentially a relational result from the VSAM data but may still not optimize performance or application design.
3. **Redesign**. The best, but most expensive conversion is an intelligent analysis of the data, its logical structure, business rules and usages in existing and planned applications. While such a complete analysis is the most expensive, it should be understood that the cost of this conversion will still be less than the cost of a new database design. Most of the required data fields have been defined and the data exists in a processable form.
4. **New design**. A new database design according to newly defined business requirements offers the largest flexibility, but will also have the largest costs.

It should be remembered that in terms of the overall cost of the conversion, the database conversion is typically a small part.

A major difference exists in the resources required between doing a ″compatibility″ type conversion (2) and a true redesign (3). In the case of a true database redesign, the availability of one or more persons fully familiar with the data and its usages is required. This knowledge need not be provided full time to the conversion team, but must be available on a demand basis. Documentation rarely provides all the information necessary to do an intelligent conversion.

Compared to the conversion methods described in 2.3, “Conversion Methods” on page 14, the “Translation” above maps to “Translation” there, “Compatibility” here maps to “Re-engineering” there, “Redesign” here maps to “Reverse Engineering” there, and “New Design” here is used in “Redevelop” and “Corporate Model” there. “Transparency” can be seen somewhere between “Compatiblitlity” and “Redesign” above, because it allows limited changes in the data structure, as described in 7.1.2, “Capabilities” on page 76, 7.1.3, “Limitations” on page 76, and Chapter 8, “Beyond Transparency” on page 103.

3.2 Objectives of Conversion

The primary objective is to optimize both the performance and the application design for further usage and expandability. The approach is to do a limited redesign of the database and reprogram the parts of the applications that are product sensitive. Again, note that no attempt is made to change the application data requirements. The assumption is that the application will continue to need and use the data that is currently stored. ***It is axiomatic that there should be no application changes or extensions during the conversion except what may be mandatory to tolerate the changes in the database.*** There will be cases where the designer recognizes that better ways exist to store the data or that some other change should be made to improve the format or structure of the stored data. Despite the temptation to make the changes that appear minor, the designer must recognize that making such changes can seriously impact the application conversion and the testing procedures. Assuming that the objective is to minimize the time and cost to make the conversion, such changes should be delayed until the application enters the maintenance cycle. Such opportunities for improvements should be noted in a journal of future potential enhancements. This ensures that the knowledge of potential improvements is not lost and the changes can be incorporated when appropriate.

3.3 Database Design Philosophy

At first glance it seems possible to treat each VSAM file as a relational table. In fact it quickly becomes apparent that this is not completely possible. Conditions that are not acceptable in a relational table are acceptable in a VSAM file. Particularly, many VSAM file designs gather as much data into a single record as possible; the opposite is true of relational normalization philosophy. Relational capabilities relate minimally redundant tables to associate data. This major difference in the models means that in many cases it will be necessary to split VSAM files into several tables in order to get the full benefit of the relational model. Further, a relational system allows a user to interrogate any column by name, based on its data value. This prohibits ambiguous data formats which might be acceptable in VSAM such as COBOL REDEFINES, repeating groups of fields, or multiple occurring fields.

The VSAM files, then, become a starting point in a relational design. The intent is to make a table from each file, but each file will be analyzed as to normal form, redundancy, and conformance with relational standards.

3.3.1 Design Information Sources

Data dictionaries are potentially a primary source of information. If installation standards have made the use of a data dictionary a requirement, information stored here can normally be relied upon to reflect the data that is presently in the VSAM files.

If an inventory of the data has been done, most of the information required will already have been organized in a format that makes it readily available for the design process. When it is complete, the inventory will replace, in most cases, the need for interrogating the dictionary. As the design progresses, decisions will be recorded in the inventory tracking tables so that they become the record of the conversion.

3.3.2 Design Sequence

The following are the **basic** steps required to convert VSAM data to DB2 data:

1. Verify that each VSAM record type is in third normal form. It should be normalized if it is not already.
2. For each resulting record type, create a DB2 table.
3. Translate each VSAM field to a DB2 column.
4. Identify a primary key for all tables where it is desirable.
5. If referential integrity is to be implemented, a foreign key must be defined for each dependent table identical in format to the primary key of the parent table.

While the above may be adequate to accomplish the design in some installations, many will have exceptions and specific conditions that must be addressed. A large part of this chapter is intended to provide the details necessary for those installations whose VSAM files do not map completely and directly to DB2 tables.

3.3.3 Data Naming Considerations

Unlike VSAM fields, all DB2 columns must be assigned names in the CREATE TABLE statements. This provides the design team the opportunity of creating a naming convention for DB2 columns if one has not already been established. A consistent and understandable naming convention can speed coding time, make programs more understandable, and generally improve communication among team members. It is also important to remember that, once the conversion is completed, end users will be making direct use of the DB2 tables. Meaningful names improve the usability and may lead to an increased usage of the data. This will make your database more valuable.

If a single set of COBOL names has been used consistently with a VSAM file, or if a single set of names can be agreed upon, this can be used as a basis for establishing the DB2 column names. The translations should be done in concert between the program conversion and the database conversion personnel to ensure that all parties are aware of the new terms.

The technique to be used will vary greatly among installations due to differences in internal standards. Below are some techniques that have proven reasonable and amenable to automation (if the inventory has been placed in a processable form).

- Any COBOL names which are 18 characters or less in length need only be modified to conform with DB2 column name requirements (usually replacement of hyphens with underscores). The name must begin with a letter and may contain no special character other than an underscore.

For names longer than 18 characters, the following steps may be applied:

- Installation standards will often provide for prefixes such as the application name which are not specifically part of the field name. Such prefixes may be removed from the column′s base name since DB2 provides for a high level qualifier which may be used in its place.

- Suffixes can often be found in field names such as ″-REC″. Such suffixes exist for programmer clarity and may be either deleted or shortened.
- If there are a number of standard abbreviations in the installations, attempt to abbreviate them further by the removal of vowels and truncation until data names can be translated to 18 characters or less. This technique will ensure preservation of uniqueness if the abbreviations are kept unique.
- If the name is still longer than 18 characters, perform the following steps until they are 18 characters:
  1. Remove hyphens from the name
  2. Remove vowels
  3. Truncate from the right.

  This may leave duplicates which will have to be handled on an individual basis.

It may be useful to use the COBOL REPLACE statement, which provides string substitution throughout a COBOL program, to propagate the new naming convention throughout the program population.

CSP Data Names

In the event that CSP is used with a VSAM file, those names may serve as a good basis for generating DB2 column names. They may, of course, be used as is, but it is probably advisable to make use of the additional length allowed by DB2 to create more meaningful names. A view may then be created with the original CSP names to facilitate defining the columns to CSP with those names.

3.4 VSAM to Relational Considerations

It is useful to understand a few of the similarities and differences between VSAM data storage techniques and the relational database model. While the relational model can perform all of the functions of VSAM, the two are quite different in their concepts. In particular, the relational model is much more rigorous in its definition and conformance requirements.

VSAM permits the compilation of related data in the same record to allow single retrieval of all related data. The relational model uses the concept of normalization to reduce a table to only the primary key and its direct attributes. This generates a larger number of tables while reducing redundancy and processing anomalies. The relational operations overcome the complexity of the many tables by supporting a full range of multi-table operations.

Relationships

Relationships in VSAM are primarily implemented through application programming with some assistance from indexes and RBAs.

A relational database permits the direct association and retrieval of data from multiple tables (relational joins) based upon any data values present in the tables. Normally such associations are based upon keys. The relational model defines two specific types of keys, the primary key which identifies the individual row and the foreign key which provides a reference to a primary key of another row, which may be in the same or another table.

Referential Integrity

VSAM maintains inter-file data consistency through application programming. The relational implementation of referential integrity controls data consistency based on data values (primary and foreign keys) in related tables. As foreign key relationships are defined, data consistency rules may be specified which replace application programming.

Ordering

VSAM files imply ordering through prime and alternate indexes. This may be relied upon by the application program. This is foreign to the relational concept. Where ordering is relied upon in VSAM, the relational request must explicitly call for it with an ORDER BY clause.

File and Table Format

VSAM file formats are less restricted than relational table formats. A relational table must preserve the capability of a user to search based on *any* column value. This restricts the format of the table to ensure that the search column can be identified by the database manager without ambiguity. This causes the prohibition in a relational table of the REDEFINES clause that can be used in VSAM. For the same reason and also to enforce normalization to the degree possible, repeating fields and groups are prohibited. In general, VSAM will tolerate unstructured record content except for indexed fields. Where the structure violates the rules of DB2, a redesign or circumvention must be developed.

Record-at-a-Time

VSAM operates record-at-a-time as opposed to the set processing capabilities of the relational model. In some cases, such as data scrolling, the application depends upon this single record mode. In these instances, relational cursor processing will usually provide the equivalent function.

Physical constraints

The physical limit on VSAM record size is extremely large due to its ability to span control intervals (CIs). DB2 row size is limited to 4080 bytes (not including long field columns), with a maximum of 255 columns; problems with views may occur with 140 or more columns. Extremely large VSAM records (in excess of 4K or 255 fields) may have to be split among multiple tables to accommodate their size.

3.5 Logical Database Design

This section addresses the problem of generating a design which maps all of the data and capabilities of the VSAM file into a set of relational tables. The intent is to identify all known types of problems and to describe possible techniques for their solution. In many cases there will not be a single best solution. It will depend upon the use of the data by the application. Multiple techniques will be given and it will be left to the implementer to determine the best one for the application.

The designer will find differences in the ease of redesigning different files due to a number of factors, not the least of which is the idiosyncrasies of the original designer. An independent factor will be the ″age″ of the file. Files which have existed for a number of years are subject to a variety of conditions that make them more difficult to redesign. The older the design of a file, the greater the probability that design compromises have been made with the addition of new applications. These sorts of conditions will tend to complicate the file redesign. At the same time, these are the files that will gain the greatest benefit from a redesign.

3.5.1 Designing Tables

Designing the tables involves the mapping of VSAM files to DB2 tables taking into account the considerations mentioned above. Areas this guide will provide assistance with are:

1. Normalization
   - Repeating groups
   - Redefines
   - Redundancy
   - Combining files
2. Referential integrity
   - Primary key identification
   - Foreign key identification
3. Inconsistent data formats
4. Indexing

Much of what follows may appear to some to be unnecessarily complicated; a set of structured procedures that can be replaced by ″common sense″. Common sense, however, is really the sum of experience and differs greatly by individual. For a data administrator who is intimately familiar with the data and applications, there will be many shortcuts. For the consultant assisting in a shop without experience with their applications, it will be necessary to do a great deal more analysis. For the designer attempting to build tools to assist in the conversion, it can be very difficult to build ″common sense″ into a tool.

The conversion team must have available a database designer with skills in DB2 performance issues. These must be addressed with respect to the specific application. Both IBM education and documentation address DB2 performance planning and tuning. For that reason, general performance and application issues relevant to the database management system, but not relevant to the conversion, will not be addressed.

Undefined Fields

Some VSAM users have documented all of their data in a data dictionary, but there are many others who have no dictionary or have only documented portions of the necessary data. In other cases, the data documented may reflect only a ″default″ case and other fields or subfields may be redefined within the program. It is necessary to have a team member or have access to someone who can verify that the defined fields constitute all of the fields used by all of the applications. This should be done while the applications are being inventoried, see 2.5, “VSAM Application Systems Inventory” on page 15. The inventory will then contain the information required to include the fields in the DB2 definition.

Normalization

It is not our purpose here to discuss techniques to normalize the data nor the value of doing so. The redbook ″SQL/DS Version 3 Release 4 Performance Guide″, GG24-4047 has a section on this topic. *Introduction to Database* by C. J. Date contains an excellent discussion of all of the normal forms and their value in data processing. Suffice it to say that normalization attempts to isolate attributes with their unique primary key. The result is a set of non-redundant tables with high consistency and flexibility.

Older files may have a number of design compromises that have been made to accommodate the addition of new applications or for performance reasons. For instance, common data may appear in multiple files to avoid the addition of complex retrieval logic. Some VSAM file designs collect all data associated with an entity into a single file, this results in records unsuitable to the relational model. Such compromises, while appropriate for VSAM files, may be counter productive for relational tables. Normalization will uncover the compromises for examination.

The relational model is based upon normalization. Therefore, to achieve the benefits it is best to attempt to normalize the data as far as practical. A simple conversion may be done without normalization but the result will not be as effective as when normalization is done. The limitations of unnormalized data will result in reduced flexibility, especially for the ad hoc end users.

Repeating Groups (Fields)

Repeating groups or repeating fields of a record discovered in data definitions may be handled in multiple ways.

Fields 1:1

A possible but not recommended method that will conserve storage and minimize program impact is to take a field originally defined to VSAM as something like:

```
05 MONTH-ACTIVITY OCCURS 0 TO 12 TIMES . . .
     PIC X(04)
```

and define it to DB2 as:

```
MONTH_ACTIVITY VARCHAR (48)
```

By modifying the program to begin with an initial move from the input area to another structure with the original definition, the program logic is essentially preserved. This technique naturally loses all the relational capabilities for data in this column and obscures the true nature of the contents. In particular, all capability to index the values is lost. This technique should be avoided where possible.

Records to Columns

In the case of a repeating field for example, periodic fields, an independent table consisting of one column may be too inefficient. Where the data is a fixed number of repeating fields (for example, monthly sales for the last twelve months), the fields can be translated to columns with unique names such as JAN, FEB, . . . and the program logic adjusted accordingly. Unlike the multiple table solution, DB2 does not have the capability of searching these with a single argument. A more complex statement such as

```
    SELECT * WHERE JAN=arg OR FEB=arg OR . . .
```

must be used referencing each column.

In this case, the search logic in the program may be eliminated if the search columns are not used selectively in further processing. If further processing of selected individual columns is required, they can be moved into a redefined area. There the fields may be referenced by both individual field names and the original subscripted name, and the subscripting logic can be kept.

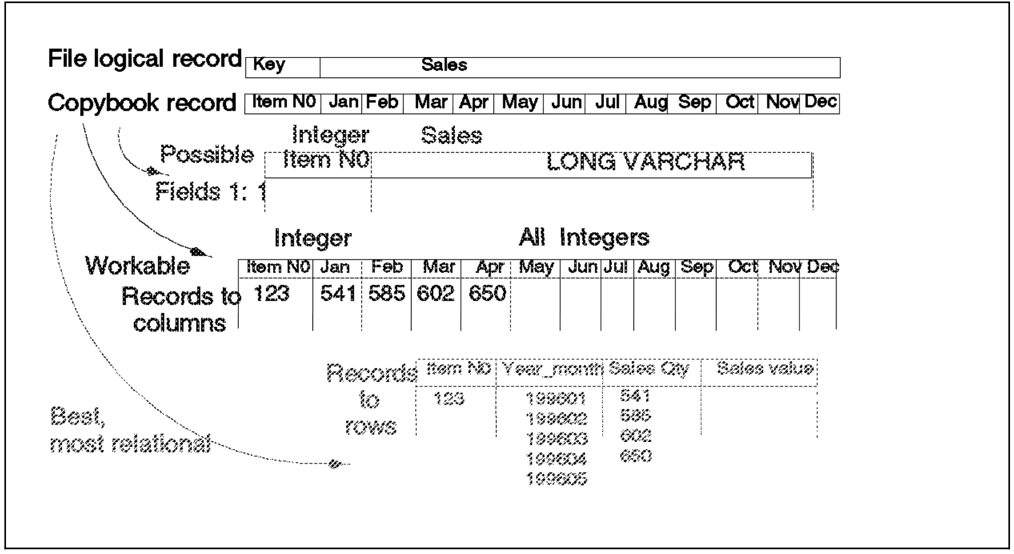


Figure 8. Options for Periodic Fields

A similar design may be used even with ″OCCURS DEPENDING ON″ type fields where the maximum number (and all applications have some maximum even if it is the maximum allowable record size) of fields is translated to columns with unique names and all unused columns set to NULL. There may be a considerable cost in storage for this option.

The selected technique can be easily entered for program conversion use in a field to column intersection table which provides a cross reference between the VSAM fields that require conversion and the corresponding DB2 column names.

Records to Rows

The purest way is to normalize the data and generate multiple tables. This will probably be the preferred method for repeating groups. In most cases, this will result in an additional table as some attributes will occur outside the repeating group area and require their own table and unique key. Multiple tables will complicate the conversion of the application programs and will increase the volume of data. Where the data is a variable number of repeating groups, this may be the only reasonable solution.

Note that the multiple table solution will provide the capability of framing a single argument request that will search all occurrences. Previously, programs had to provide logic to scan them.

Whatever technique is used, it is necessary to reflect the design decision in the conversion documentation such as a field to column intersection table. This documentation will be required by the program conversion group.

Overlapping Fields (REDEFINES)

Inspection of VSAM record definitions may uncover overlapping or COBOL REDEFINES types of fields. Typically this condition is the result of ″subtypes″ of records. It is a common condition for there to be multiple types of an entity. For instance, an application may address multiple types of products that a company sells; manufactured at this location, manufactured elsewhere (plant name and shipping information required), and purchased (vendor and price required). A single record may well redefine the content for the three subtypes. In cases of redefined record content it will be necessary to analyze the data and determine whether multiple tables should be defined or whether all types of fields properly belong in the same table. This will require normalization of the data to determine the correct association. It will also require the assistance of someone familiar with the application and its associated data. The first test that can be applied is to determine if a part of the prime key is redefined. If so, the data is not normalized and should be divided into multiple tables with unique primary keys. In some cases data may be unnormalized but still have a single prime key for the attributes.

If the record is in normal form (third normal), then the redefined data must be mutually exclusive (fourth normal form would generate multiple table types for this condition). This is the case with ″subtype″ information as described above. It may or may not be appropriate to go to fourth normal form. In a relational database, each field must be defined as a separate column in the table. The program will have to be modified to cause one to be ″null″ while the other has a value. Which fields should be expected by the program will normally be determined by some code field in the record which the program interrogates. It will be necessary to understand how the programmer determines which of the fields exist in the VSAM file. Decisions on how the programs are to be modified to address two tables instead of the single file will determine whether an identifier is carried into the new table.

If the programs in question will process under DB2 VSAM Transparency, the data may be normalized into one or more tables without concern for immediate impact on the programs not yet converted. DB2 VSAM Transparency will reproduce the appropriate form of the record using criteria defined to it (the same criteria used by the program to determine which fields exist).

In any case, a REDEFINES condition will require planning between the database conversion and the program conversion groups to ensure consistency in the solution. A field to column intersection table will show the results of the decisions with the relationships of the record fields to table columns. This is a good demonstration of the need for such an inventory to assist the program converter in understanding the new format of the database.

Redundant Data

For performance, the same data may be located in multiple VSAM files. These cases may be difficult to identify due to the lack of common names. If a dictionary exists, it may contain information identifying where this occurs.

Without a dictionary it will require someone with extensive knowledge of the application to identify all of the cases.

Where redundant occurrences of the same field exist, it is necessary to determine whether the data should be normalized to a single table or left in independent tables. Two factors will apply:

- Application impact

  If the application will require considerable program change to accommodate the single copy form of the data, it may not be economical to eliminate the redundancy. Trade-off between the cost of maintaining the multiple copies and the cost of application change must be identified. The risk of inconsistency of the data values must also be taken into account.

- Performance impact

  In some cases the same performance advantages of separating the data will apply to the DB2 implementation. This will have to be balanced against the cost of multiple copies of the data such as duplicate maintenance. In those cases where the data is typically very stable (for example, customer name), duplicate copies of the data are probably acceptable. This is a typical consideration in assessing the desirable degree of normalization.

Where a decision is made to maintain redundant data, every attempt should be made to keep the data type definitions of the redundant columns identical. This can have a positive impact on performance, particularly where the data is used as the basis of a JOIN.

Another concern will arise in consolidating these fields into a single version. It is almost axiomatic that there will be some inconsistencies between copies. It is necessary to have an agreed upon technique for handling these conditions. This may be as simple as ignoring all but one copy or as complex as the analysis by a committee of users.

Table Combination

Normalization of the data may bring to light instances where data in two or more files should be combined into a single table. For instance, an accounts receivable file and an order entry file may each have as one of their resultant tables, a table of customer information. While not necessarily redundant, these tables may properly be normalized to a single table. Most programs can be protected from the change through the use of views. Programs responsible for the addition or deletion of rows will be made more complicated.

A problem will undoubtedly arise in combining records due to an inconsistency of the data. This can take two forms:

- Unmatched records

  There will be cases where not all of the record occurrences required to make up the combined record exist. This may be due to an application error or due to the nature of the data. It will be necessary to make some decisions as to how the condition is to be handled. It may be adequate to simply create null columns for the missing data.

- Inconsistent data

  Errors in input data or application logic can result in data in two logically associated records being inconsistent such as CURRENT-PURCHASES in one record exceeding YTD-PURCHASES in another record. A method of resolution will have to be determined for such cases. It may be possible to ignore the problem (at year end, the YTD-PURCHASES will be reset) or it may be necessary to reconstruct correct values where consistency is important to the integrity of the application. In many cases there will already be methods in place to discover and correct such inconsistencies as they come to light in various reports. In these cases, the inconsistencies may be ignored as the existing techniques will reveal and correct them.

Referential Integrity

A major capability of the relational model is referential integrity. This capability permits the database administrator to specify consistency rules which will be maintained between tables. For instance, an order may not exist without the corresponding customer. The deletion of an order would mandate the deletion of all associated order items. In VSAM, such rules between files are maintained by programming. In making the conversion, there is a choice as to whether referential integrity should be implemented to replace the existing programming. In general, this is probably not a good choice. Replacing existing programming will complicate the reprogramming and the testing functions, thus slowing the conversion. Further, it may well be found that the rules implemented do not exactly map with the DB2 referential integrity. This will mean results from the new system will be inconsistent with results from the old system to at least some degree. In addition, there may be inconsistent rules in the VSAM system due to different programmers or different conditions at the time of the implementation of a given program. It would be better to withhold the redesign of these programs until after the conversion.

There are some cases where referential integrity may be inserted without increased complexity of the conversion. Where files were converted to multiple tables, referential integrity may be defined between the resultant tables to supply the same consistency that was implicit in their being in a single record. That is, if data which is now an order item table was previously repeating groups in an order file, then referential rules could be implemented to ensure that no item row is created unless the order existed and that deletion of an order would result in the deletion of the items of that order. Such an implementation will reduce the coding necessary to support the multiple table design and will reflect the same conditions that existed in the single file.

In addition, if one of the incentives for the conversion of the application to DB2 was the desire to make major extensions to the application or use of the data, it may be worthwhile to insert the referential integrity during the conversion. This will reduce the development effort for the new programs. It should be understood, however, that the conversion may be lengthened.

To make the following discussion as understandable as possible, let us establish some terminology for clarity:

- Prime key, prime index. Every VSAM key-sequenced data set (KSDS) must have a prime index on one or more contiguous fields. The values in these fields must be unique. They are the prime keys.
- Primary key. The DB2 columns which serve as the unique identifier of a relational row.
- Foreign key. DB2 column containing the value of a primary key of a row in the same or another table for the purpose of identifying a relationship.

- Dependent table. The DB2 table which contains the foreign key in question.
- Parent table. The related DB2 table referenced by the dependent table.

Primary Key Determination

The relational model requires the existence of a primary key to ensure entity integrity, that is, the ability to uniquely identify any row. DB2, however, does not enforce this requirement except where referential constraints are defined. If it has been decided that no referential constraints will be defined during conversion, then primary keys are not required but will probably be desirable. Therefore, as the primary key does not imply referential integrity, it may be wise to assign a primary key wherever possible to allow for ready addition of referential integrity in the future.

Generally, the prime key of a KSDS is a good candidate for a primary key. For an RRDS, the relative record number (RRN) may be a good candidate. Note that an additional column will have to be added to the table to hold the relative record number.

For an ESDS, the RBA, though an obvious possibility, should probably be avoided. Since DB2 has nothing equivalent to an RBA, in addition to adding a column, conversion of maintenance programs is complicated by the need to simulate new RBAs for new records. Where practical, other fields should be selected as candidate keys for the new DB2 table. If there are no practical choices currently in the record, consider the addition of a new field, such as timestamp or an artificial identifier.

If this table is not to serve as a parent in referential integrity (the ″one″ side of a ″one to many″ relationship) and there is not sufficient data to create the logical primary key, it is preferable to not define a primary key. In this case, definition of the key would require the addition of more data to the table complicating the program conversions and increasing data volumes with no immediate payback.

An example of a case where a primary key is not necessary and may not be desirable is that of the order item example used above. If there is no need to maintain items in a given sequence, then there is no need to identify an item beyond the customer and order to which it pertains.

In some cases, it may be the existence of foreign keys which determines the primary key. That is, there will be other tables which reference this table with a foreign key, the values available for the foreign key may determine the most appropriate of alternate primary keys.

Once a primary key has been determined, the column inventory should be updated to show which columns participate in the primary key and their order. The definition of the primary key will sometimes also result in the definition of additional columns for the table. These must be documented fully in the inventory in order for application logic to reflect their existence.

Foreign Key Determination

The discussion under “Referential Integrity” on page 37 discusses whether or not referential integrity should be implemented during the conversion. Even if it has been decided that referential integrity is not to be implemented to replace application code providing inter-file consistency, referential integrity should still be used to provide integrity for data that has been split into multiple tables from a single file. Conversion effort can be reduced by using referential integrity in the database management system to provide the same kind of implied rules as a single file solution. In such a case, determination of foreign keys is usually simple.

In breaking up a file into multiple tables, there will normally be a natural parent/dependant relationship. The unique (not repeating, not redefined) part of the file will become the parent table, the non-unique part will become the dependent table. If the newly created dependent table has a primary key specified, it will probably be an extension of the primary key of the parent table. Therefore, the foreign key which references the parent is a subset of the primary key of the dependent table. The rule for such a relationship should be specified as CASCADE to simulate the original file condition.

In some cases it will be determined that the immediate advantages of referential integrity outweigh any additional cost in the conversion. In this case, there is a high probability that the interrelated files contain sufficient information to locate each other. Typically this information will be the same data that becomes the primary key of the related table, and is an appropriate foreign key. There are two exceptions to this condition:

- The link information rather than being the primary key of the other table, is an alternate key. In this case, if referential integrity is to be implemented, the primary key must be defined into the table and applications adjusted to place it there. If the alternate key is as likely a candidate for a primary key, it may be reasonable to change the choice of the primary key.
- The link information is an RBA (Relative Byte Address). Since there is no corresponding capability in a relational database, the RBA must be replaced with a primary key of the other table.

It is not wise to attempt to establish foreign keys for any relationship which does not currently have a reference to a unique row of the related table. A non-unique relationship would require the construction of an additional table to relate individual rows and would require more changes to the application design than should be tolerated for a conversion.

Finally, note that column formats must be identical between the related primary and foreign keys, and, for performance reasons, should be identical wherever columns may be used as the basis of a JOIN.

VSAM Groups

VSAM provides the user with the ability to specify groups of fields as a unit. DB2 does not provide an equivalent capability. This issue is discussed in greater detail in "Group Level Items" on page 42.

Sequential Processing

VSAM programs often require that data be processed in a particular order to support the program logic or create output in a given sequence. Where a VSAM index, either primary or alternate, was used to order the data, SQL can create the same sequence by including an ORDER BY clause on the columns which correspond to the VSAM key fields. To skip sequential processing can be accomplished in basically the same way, with the addition of a WHERE clause in the SQL statement to begin processing from a given value in those columns.

Entry sequenced processing of an ESDS may require the addition of a new column to the DB2 table. SQL guarantees no sequence unless an ORDER BY clause is specified, and, in the case of an ESDS, often there is no field in the VSAM record which can provide the proper ordering. In these cases, a new column such as a timestamp must be added to the table and the appropriate programs modified to add this data to the new rows.

When converting an existing file of this type, it would be prudent to ensure that the new values are all unique.

3.5.2 Field to Column Mapping

There are several generic differences between VSAM and DB2 data definitions that must be reflected in the design. While most field definitions will have a direct translation, some will require special consideration.

Nulls

There is no equivalent in VSAM of the DB2 concept of NULLS, that is, an indicator that no value has yet been provided for a field. In VSAM, because data is created at the record level, every field has some value. It is derived from the work area used to create the record and may contain a value provided by the program, a default value, or an unpredictable value left over from an earlier use of the work area. If the program conversion is done correctly, the same condition will be true after the conversion. Since the handling of nulls would require additional program logic, it is recommended that all fields be specified as NOT NULL. Some exceptions have already been noted: “Repeating Groups (Fields)” on page 33; “Overlapping Fields (REDEFINES)” on page 35; and “Table Combination” on page 36.

Nulls have significant value in preventing misleading calculations and counts. For example, if an average salary is calculated for a department of twenty people and one of those people has a zero salary because it has not yet been entered on the file, the calculation of the average will progress assuming the zero value is a legitimate amount. If a null value is used instead, DB2 will recognize that no value has yet been entered and calculate the average for the remaining nineteen people in the department, yielding a very different and more accurate value. Inclusion of null values should be logged for later implementation during the maintenance phase.

Alphanumeric Fields

In general, VSAM alphanumeric data may be converted to DB2 character (CHAR) data. For exceptions, see “Date, Time and Timestamp Fields” on page 42.

Binary Integer Fields

Binary integer (COMP) fields may be converted to DB2 INTEGER or SMALLINT data types. Two-byte binary integers, defined as S9(4) or smaller, correlate to SMALLINT, while four-byte binary integers, S9(5) to S9(9), correlate to INTEGER.

Floating-Point Fields

Floating-point fields, both single precision (COMP-2) and double precision (COMP-1), may be converted to DB2 FLOAT data types. Precision of the FLOAT column is determined by the integer value associated with the column (1 - 21 provides single precision, 22 - 53 provides double precision).

Packed Fields

Packed (COMP-3) fields may be converted directly to DECIMAL. These data types are a direct mapping except where the packed fields exceed 15 digits. In this case, the data should be analyzed to determine whether the full length is really required. If the length is simply to ensure precision (for example, five decimal places for a monetary field), these fields can be redefined with normal precision for storage while using program work areas of greater precision for computation.

Otherwise, two options are possible:

- If the data is not arithmetic, that is, no computations will be done on it, it may be specified as character (or variable character).
- If the data is arithmetic and computations will be made on it, it is necessary to specify it as FLOAT. Double precision floating point will permit the handling of numbers of this magnitude. In this case, arithmetic results may not be exactly consistent with those of the decimal operations. The application programs and application logic must be analyzed to determine the impact.

Whatever method is selected, it needs to be recorded in the inventory and tracking information.

Unpacked Fields

There is no direct translation of unpacked numeric (external decimal) data to DB2. Such data may have two possible interpretations. Where it is used as a means of storing an identifying number, such as an employee serial, it may be translated to either character or decimal. The former will ensure that no attempt can be made to do arithmetic operations on the column while the latter will take less space for values longer than two digits.

In the case where the field is true numeric and the data is subject to arithmetic operations, DECIMAL is the correct specification. Programs will have to be inspected as to their use of the data to ensure that it is consistent with the re-specification.

One exception exists in converting unpacked format to DECIMAL. VSAM permits unpacked fields up to 18 digits. Where a field has been defined with this extreme length, it cannot be converted to DB2 decimal. If arithmetic operations are to be performed on this field, it will be necessary to specify it as FLOAT. Due to differences in results of computations, every attempt should be made to reduce it to a size where it can be directly converted to DB2 DECIMAL (15 digits). If no computations are to be made on the field, it should be specified as character (or variable character).

The decision must be recorded in the field to column intersection table for the program conversion personnel.

Graphic Strings

Graphic string (DISPLAY-1) fields are two-byte representations of single characters. They may be converted directly to DB2 GRAPHIC or VARGRAPHIC data types.

Mixed fields, combinations of graphic strings and normal data in a single field, should be defined as character (CHAR) columns. If the graphic strings are delimited by the proper start and stop characters, and if the MIXED DATA parameter is set to YES, DB2 will properly handle the mixed data.

CSP users should note that, since there is no equivalent designation in DB2 of the MIXED data type, fields designated as MIXED should be moved to a character (CHA) field prior to conversion. If the corresponding DB2 column is defined as CHAR, and the conditions mentioned above are met, DB2 will handle the mixed data properly.

Group Level Items

VSAM supports the concept of fields which are made up of a number of subfields. DB2 does not support the concept of subfields.

For group level items that are dates, times, or timestamps it is advisable to use the specific DB2 data type designed for those items. See “Date, Time and Timestamp Fields” below.

There are two fundamental ways to provide support for group items:

- The most common will be to define the subfields as columns to DB2. This approach is the most general and permits indexing each of the subfields independently and allows the use of the subfields in different or overlapping foreign keys. Where it is necessary to address the group level item as well as the subfields, after the FETCH of the data, the individual columns may be moved with a COBOL move to a structure where they can be treated as an entity.
- Alternatively, it is possible to define the group level field as a column to DB2. After FETCHing the data from the table, a COBOL move can be performed to a structured work area where the data can be perceived as subfields. This will, however, prohibit the use of the subfields directly by DB2.

Again, once a technique is selected, it is necessary to reflect the design decision in conversion documentation such as the field to column intersection table.

Date, Time and Timestamp Fields

Representations of dates, times, and timestamps, whether defined as alphanumeric or some form of numeric, should be defined to DB2 with the specific data type: DATE, TIME, or TIMESTAMP. There are two advantages:

- SQL supports arithmetic calculations on dates. For example, it can calculate the number of years, months, and days between two DATE columns. Although it is inappropriate to make use of this facility in the conversion process where it impacts the program logic, defining columns in this way now will enable future programs to take advantage of this capability.

- In SQL statements, portions of these fields may be referenced by name: YEAR, MONTH, DAY, HOUR, MINUTE, SECOND, and MICROSECOND. If the VSAM field is defined as a group level and portions of it are used to determine if an action is to affect this particular record, this ability combined with the arithmetic capability described above may allow the SQL SELECT statement to replace program logic. Since some action is needed to handle group levels anyway, replacement of the selection logic with a more specific SQL statement may be a reasonable alternative.

Note that the format of these SQL data types may differ from the VSAM fields, especially the four digit year. If so, the differences should be noted and appropriate modifications made during data migration.

3.6 Data Security

Data security definition begins with an investigation of VSAM security. Once the restrictions established for the VSAM users are fully understood, they can readily be translated to DB2 through the use of view authorizations.

In installations where very little has been done with respect to security in VSAM, it may well be adequate to define all data as PUBLIC (accessible to all users) and use CICS security for online applications and existing security for batch. Additional security will be necessary as ad hoc users come online.

While additional security options are available within the DB2 system, it is not recommended that you extend the security during the conversion. This will have a larger impact on operations and potentially on the execution of programs. Following the conversion, security options may be added gradually.

3.7 Designing for Data Related Between VSAM and DB2

While every attempt should be made to avoid a conversion where files which maintain relationships are split between VSAM and DB2, there will be cases, especially if DB2 VSAM Transparency is not used, where it is unavoidable. This coexistence will considerably complicate both the database design effort and the application programming effort.

There are several ways how files may be interrelated under VSAM.

- The simplest one is for one file to contain a key field of another file. Such references can be maintained across mixed data storage technologies, as long as the VSAM file and the DB2 data remain on the same physical system.
- A second kind of interrelationship is that a file carries the RBA of a record in another file. Two conditions can thus be generated:
  - The file **containing** the RBA is moved to DB2. In this situation, little needs to be done. The RBA can still be used to reference data under VSAM.
  - The file **referenced** by the RBA is moved to DB2. In this case, the RBA must be replaced by the primary key of the DB2 row. This will result in a change in the VSAM file with attendant changes in application programs. In cases where many programs access the file with the RBA but do not use it directly, it may prove easier to add a column to the DB2 table

which contains the former VSAM RBA. Additions of new rows to the DB2 table would require the assignment of dummy RBAs and their placement in the corresponding VSAM records, see "Primary Key Determination" on page 38.

Another alternative is to duplicate the data on both systems. This produces problems as discussed under 2.6, "Coexistence Strategies" on page 19.

3.8 Physical Database Design

After finishing the logical design, the next step is physical design. The following describes how to complete the conversion from the VSAM physical design to the DB2 physical design.

Storage Pools and Dbspaces

Use of DB2 physical constructs such as storage pool and dbspace are not a conversion issue. Nothing in the VSAM design will constrain or influence the choices made for these objects. All normal DB2 design considerations apply. For example:

- Place one table in a dbspace.
- Consider defining only one dbspace in a storage pool when using Guest Sharing with VM Data Space Support or if a high degree of control is required for the placement and distribution of data on the available disk space. This enables I/O balancing of the available DASD devices.

Indexes

There are a number of points to be considered in determining indexes when migrating VSAM files. Each VSAM index implies an index in DB2. Some considerations exist that may result in not converting a VSAM index to a DB2 index.

- **DB2 does not require an index** to retrieve a record based on a search argument. In some cases (such as tables of less than 16K bytes) performance will be improved through not converting the VSAM index to a DB2 index. Analysis must be done to determine whether there is an access advantage using the index in DB2 that exceeds the cost of index maintenance. The EXPLAIN capability of DB2 will assist in this, or an ad hoc query can be made of the catalog to determine which indexes are never used by static SQL.
- Some VSAM indexes exist **to provide the data in a specific sequence**. DB2 has an imbedded sort capability. If the data is stored in a different sequence, DB2 can often produce the same sequence more quickly through an internal sort invoked by an ORDER BY clause.

There are several other considerations regarding indexes which may be important in specific situations:

- It may be more efficient in some instances to specify a DB2 index as **clustering index**, which causes the records to be physically stored in the approximate sequence of the index. Where the data will often be accessed in that sequence or in clusters of records having sequential keys, as in the case of tables generated as a result of normalizing repeating groups, the number of physical I/Os can be significantly reduced by specifying a clustering index. Typically, the prime index of a KSDS will convert to a clustered, unique primary index in DB2. (Only one clustering index may be specified per table.)

- The key fields of VSAM indexes must be contiguous. There are instances where a VSAM key has been made artificially long to satisfy this requirement. Where these cases can be identified, the **extraneous columns should be eliminated** from DB2 indexing.
- VSAM provides the ability to read an index as if it were an ordinary VSAM file. Few programs use this facility and most of those that do are dealing only with key values. The same ability can be provided by **selecting only the columns of the table** that equate to the target of the index. In the rare instance where the program is also using the index pointers (RBAs or keys of the prime index) some program changes will have to be made. A program can select the primary key of the table and the columns that equate to the VSAM alternate index key, but RBAs may have to be replaced or simulated.

Probably the most important aspect of DB2 indexing to be considered is its flexibility. Unlike VSAM DML (Data Manipulation Language), SQL code does not depend on the presence or absence of an index. Indexes can be added to improve performance or deleted to reduce overhead without regard to SQL code. This greatly reduces the importance of the initial implementation of DB2 indexing. Certainly, a reasonable effort should be made to determine the proper level of indexing at conversion time, taking into consideration the points made above, but changes to DB2 indexing are much easier than with VSAM, and can be viewed as another tuning aid.

Key Fields

VSAM key fields must be one contiguous area with a maximum length of 255 bytes. In the most extreme case, the key could consist of 255 fields, each one byte in length. DB2 limits an index to 16 columns and up to 254 bytes of data. If the VSAM key maps into more than 16 DB2 columns, a compromise will have to be made.

- As DB2 does not require an index for all accesses, it is possible to reduce the number of columns in the index without affecting the program logic. The impact of the reduced index on performance will have to be analyzed and the indexing columns carefully chosen.
- Another possibility is combining columns to reduce the number in the key. This, of course, eliminates the possibility of accessing the DB2 tables based on individual field values and could introduce additional program complexity. See "Overlapping Fields (REDEFINES)" on page 35.

If the VSAM key is 255 bytes long, a compromise will have to be made.

- Either an entire column must be eliminated from the key (with the same effect as above)
- or one of the key columns must be reduced in size by one byte.

In either case, if performance is a concern, the ability of DB2 to utilize multiple indexes in a search can be useful. By carefully defining two or more indexes which cover the entire key, DB2 will make optimum use of the indexes prior to searching the database and probably provide better performance than a partially indexed search.

Display Scrolling

Scrolling or any other form of browsing which permits backing up in a set is easier with record-at-a-time processing than set processing. In particular, VSAM may use RBAs to effect repositioning across screen interactions. If the set is typically smaller than a single screen of output, this can be readily addressed with CLOSE/OPEN of the SQL cursor. If the set is large, however, such processing can be long and alternative techniques may be desirable. One way of handling the condition is to provide an index over the table to be browsed with such keys as are necessary to allow repositioning at any point in the set.

If it is possible for the number of duplicates of a key to exceed the size of one scroll page, it will be necessary to define a key with a more discriminating index. Without such a key, attempts to page forward might result in the user either skipping the last of the duplicates or failing to be able to proceed beyond the first page.

A special case of this is the provision of a sequence number in the rows. Where rows are referenced only in the sequence in which they were inserted, a sequence number column will allow scrolling forward or backward ″n″ rows or pages. If inserts are done within small sets of rows (for example, account transactions) on an infrequent basis and numbers are assigned within the set, the numbers may be reordered with an SQL command from the point of insert forward.

If columns are added in support of this type of function, they must be recorded in an inventory. Make sure that the program conversion group is aware of the new columns.

Another technique which may be employed in some instances is to make use of CICS temporary storage. Write the entire set of rows which may be scrolled to temporary storage, then perform all screen building from there.

Concurrency

In general, DB2 will improve concurrency. Multiple updates, both batch and online, can run concurrently with guaranteed data integrity. There are other specific instances where the differences between VSAM and DB2 may have an impact.

Issues of contention between ad hoc inquiry and production update are the same for converted VSAM installations as for any other DB2 installation.

Views of Data

Data access can be defined with DB2 views to limit access or to simplify the coding requirements.

Where files have been split into multiple tables, views can assist in preserving the logic of the application program. Files split due to redefines can be reassembled by a view with a join of the tables. Files split by OCCURS clauses cannot be fully reconstituted, but can be at least partially. This will preserve some of the logic of the application. Note that not all cases can use this facility due to the DB2 restriction that does not allow an update to a view that contains a join. For the complete list of limitations, see “Circumvent Limitations” on page 104.

Access Authorization

Whether access is requested through DB2 views or the actual base tables, the necessary authorization is defined by the DB2 privileges in effect for the associated user. Note that these privileges are granted independently of any access request, and the resultant privilege set is what is used to authorize each request as it occurs. DB2 views as authorization vehicles can assure full support of the VSAM security options.

Locking

There is a variety of locking options available with DB2. It is not appropriate to attempt to exactly reproduce those of VSAM, but rather the desirable options should be found for the individual program/table. Since this effort is no different with a new application, it will not be discussed here for online programs.

For batch VSAM programs, there is no locking because there is no concurrency or sharing between VSAM batch programs or VSAM batch and VSAM online programs. Choose the locking level dependent on the concurrency needed and the performance overhead tolerated.

3.9 Estimating Performance

One of the tasks that needs to be done as part of the initial planning is to estimate what the effect of using DB2 VSAM Transparency will have on performance with regards to run time for batch applications and response time for online applications. It needs to be established that there are adequate machine resources available to perform the following:

- Run the batch jobs in the time available. For example, the batch processing required to be performed during the period when the online system is not available. This is referred to as the ′batch window′. Do not forget the batch processing that needs to be performed at month and year end as well as quarterly. Also take into consideration backups for recovery and restart purposes.
- Give acceptable response time to the online transactions.

The current run time of each batch job must be recorded and if a performance monitor is available, it can be used to record the current load on the system for each batch job. For the online system, the CICS statistics may be useful but ideally an online monitor is required to record the resource utilization by transaction as well as frequency of use.

When these statistics have been collected, a factor needs to be applied to calculate the estimated run times and resource utilization:

- One way is to get a reading from the pilot conversion and use that as a factor. Refer also to Appendix B, “Performance Statistics” on page 109 to get an indication of the wide spread of performance numbers that was recorded during the sample conversion project for this book.

- Another possibility is to make use of the performance estimator for MVS DB2, see 2.7.2, “Performance” on page 22 and use that to calculate a factor to apply to the recorded run times.
- The best will be to combine both methods described above and get a factor that is verified by the pilot conversion project.

This may be a lot of work to do up front, but will ensure that the project is properly planned and that adequate machine resources are available for the successful completion of the conversion to DB2.

3.10 Design Review

At the completion of all of the database design steps, the entire design should be reviewed with application analysts familiar with the current implementation of the application. It is essential that these reviewers have this kind of background; they need not be conversion team members.

It is necessary to ensure that the database designers have not overlooked some quirk of the application programmers′ use of the data in their design. Only those well familiar with the application implementation can provide that.

The database designers must trace the data from the VSAM files and describe its implementation in DB2. It is the responsibility of the application analysts to make sure that they fully understand how the new format of the data can be used to provide the same functions as were provided previously. They must challenge any part of the design that seems inconsistent with the program use of the data. This information becomes input to the database design for any final revisions.

Chapter 4. Testing

This chapter details testing methodology and some procedures with a concentration on system integration testing. It does not address specifics of the testing methodology if DB2 VSAM Transparency is employed. It is assumed that applications are moved into production in the DB2 environment while the VSAM system continues to run.

Testing may be one half of the entire conversion project time and effort. Therefore, a comprehensive test plan is important. In this test plan,

- Measurable tests,
- Acceptance criteria,
- Test exit criteria,
- Schedule dependencies, and
- Task relationships

need to be planned for and agreed to. The test plan also needs to cover fallback and recovery synchronization options.

4.1 Testing Methodology

Testing should be done on as stable an environment as possible. That is, minimize version changes to the operating system, compilers, database products, and other programs which may cause the results to differ from the original application system.

Even though the current applications seem correct, inconsistencies may be introduced by changes in the environment. For example, a COBOL program may compile into different object code by using a new compiler level. The resulting new executable code may produce different results. Thus debugging may include recompiling and executing the original application to isolate the source of inconsistencies.

Review any changes to the application made during the conversion to see if new test conditions are needed. All design or function changes should have associated documentation. This documentation is used to assist with the test definition.

Adoption of the philosophy “no changes to the design during conversion,” pays off during testing. The more consistent the input and environments are between the original and converted systems, the less effort will be required for testing. Testing covers the following areas:

- Function - refer to “Function Test” on page 5.
- Performance - refer to “Performance Test” on page 5.
- Concurrency - refer to “Concurrency Test” on page 6.

The function test can be split into:

- Unit test - running on a small unit of one or more programs only
- Integration test - running on a whole application system.

Each of these tests should cover the aspects of:

- Accuracy

- Error recovery
- Messages

These aspects should consider the behavior of each of the following:

- Graphical representations
- Batch jobs
- Online screens

Some of these themes are outlined in the following with more details.

4.2 Performance

An application may actually do more work with fewer lines of code due to the built-in functions provided by DB2. Testing, then, may assume more of a performance perspective beyond just checking for equivalency.

DB2 EXPLAIN provides information about the access paths available and which path was chosen by the optimizer for the SQL call. Look for cases where indexes may be defined to speed access to the data. However, there are costs associated with maintaining indexes. As a general rule: Often queried, seldom updated, performance critical data is the best candidate for indexing.

For other performance aspects, refer to Chapter 3, "Database Design" on page 27 and the redbook ″SQL/DS Version 3 Release 4 Performance Guide″, GG24-4047.

4.3 Data Sampling

During unit test, the individual program will probably be initially tested against a minimum set of data. For an integration test, it is necessary to operate against more comprehensive sets of data. If a small but complete test file exists under VSAM the conversion procedures that will be used for the production files will also apply for the test data. In fact, such a procedure will provide useful validation of the conversion process.

The data volume issues are threefold:

- Is there a comprehensive test file for integrated test?
- Can the entire database be used during the testing phases?
- How long will it take to convert the desired data?

Depending on the answers to these questions it may be necessary to develop a migration technique not only for the complete database but also for a test subset. Such a technique is presented here.

Where a VSAM test file does not exist and the regular file is large, it will probably be expedient to extract a sample for development and testing. Such an extraction must also be carefully thought out and the migration program must be adapted to accept it. Data extracted for testing must be consistent. That is, foreign keys must refer to existing primary keys after the extraction. One technique for generating such a consistent set would be to expand the migration programs to incorporate the following:

1. Select one file as primary. This file should be as independent as possible from other files. That is, changes to other files should seldom, ideally never, result in changes to the primary file.
2. Determine the sampling technique. This may be based upon such simple methods as:
   - every 100th record
   - use specific identified key values
   - may involve logic such as the selection of the first ″n″ records with significant values in fields ″x, y and z″, in order to ensure that there is a representative range of data complexity.
3. Modify all data migration programs to write to work files all those primary keys that are migrated.
4. Execute the data migration program for the primary file.
5. Execute the data migration program for the second file using the work file of the primary file as the argument to determine which records to omit and which records to retrieve. All records with foreign keys referring to primary keys of the primary file which are not in the work file (that is, not migrated) should be discarded. Write all migrated primary keys of the second file into another work file.
6. Continue with running all file migration programs. Note that the sequence of data extraction will be critical and therefore the determination of the sequence must be based on the nature of the data reference, beginning with the primary file and ending with a file on which no other file depends.

   In the worst case that even the primary file is dependent, you must iterate this process. Stop iteration when no more changes occur.

4.4 Integration Test Procedures

Testing may be done in the following sequence:

1. Define and identify tests that establish the converted application results are consistent with the original applications.
2. Document the test plan and the criteria to be used for acceptance of the converted application.
3. Create tests according to the plan. This includes tests which verify that cutover is complete through full scale production.
4. Create test VSAM data sets. Hopefully, this will be a small subset of the current production data sets. The value of a small amount of data is to minimize the computing costs and reduce time during testing.
5. Procedures should be created for unloading the VSAM test data sets and loading sorted DB2 data.
6. Create a test DB2 database which represents data equivalent to the test VSAM data sets. This will test the data conversion procedure which will subsequently be used to convert the entire VSAM application to DB2.
7. Commence testing using the test DB2 tables.

The test effort may be divided between batch and online applications. Ad hoc queries must be included in the testing procedures as well as predefined applications. Formal testing may be managed step by step through unit test, function test, and cutover test in much the same way as any new application development.

Cutover procedures include back-out steps. Back-out allows for cutover to be tested, yet retain the ability to go back to the cutover point. It is important to test trial runs of the cutover.

QMF and ISQL can be integral parts of testing ad hoc queries and can supplement other planned testing. Allowing users to interrogate the data will accomplish testing and education at the same time.

As the tested applications belong to the “frozen” state, the actual production state needs to be re-implemented once the test is finished. When the original application changes in between, these modifications must be implemented into the tested application, and a second test is needed. Updates that have occurred can be stored in control files, or all updates to the production applications have carefully to be compared with updates that might have happened to the “frozen” copy of them during the test.

4.4.1 Compare Application Results

The output of each application before and after conversion should be scanned for equivalence. Output may be in the form of screens, reports and external files. Identify and reconcile the differences. Where the differences are not acceptable, change the application and retest.

4.4.2 Compare Database Contents

A set of VSAM files is migrated to DB2. A common set of test transactions can be run through both the VSAM and DB2 systems. By running the VSAM unload that was used in the initial load of the test DB2 tables, other DB2 tables may be created directly from the updated VSAM files. Now, both DB2 tables may be compared by queries. Alternatively, the tables may be unloaded and then compared with the unloaded VSAM data.

4.4.3 Identical Function Testing

A thorough way of testing both application results and database contents, is the one shown in Figure 9 on page 53. A set of VSAM files is migrated to DB2. Then, using applications to be tested, the same updates are applied three times to both the VSAM and the DB2 systems. Depending on the type of updates, this should result in an error message or inability to perform the function (for example, for a deletion of a row), or in a repeated execution (for example, for an insertion of a row if duplicates are allowed).

During each of the three steps, the screens, the data, and the error messages are compared.

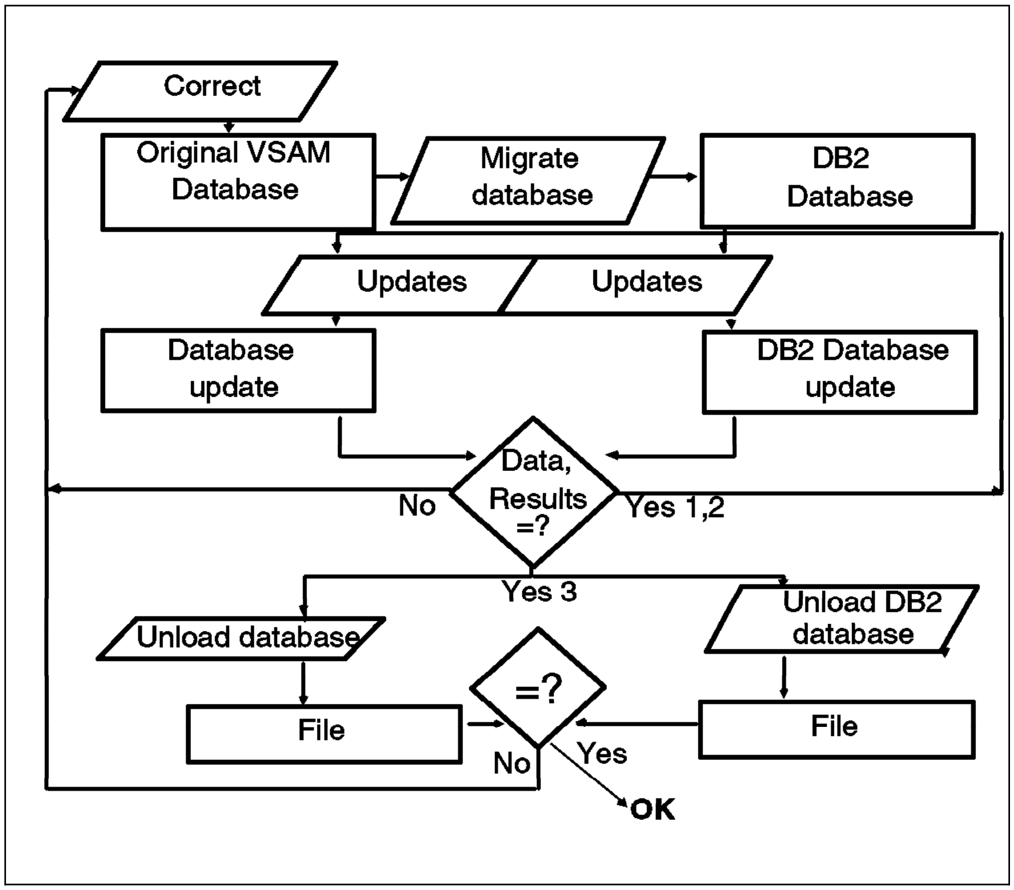


Figure 9. Equivalent Testing Flow

1. Freeze the test VSAM files and applications to have a reliable, undisputed source of testing.
2. Unload the test VSAM file and load to a new test DB2.
3. Verify that the initial iteration of the data conversion to DB2 is correct before proceeding.
4. Run a selected set of updates against the test VSAM files creating an updated set of VSAM files.
5. Run the same set of updates against the test DB2 tables creating an updated set of DB2 tables.
6. Compare the results, screens and error messages.
7. Run the same set of updates a second time against the test VSAM file, creating a twice updated set of VSAM files.
8. Run the same set of updates a second time against the test DB2 tables, creating a twice updated set of DB2 tables.
9. Compare the results, screens and error messages.
10. Run the same set of updates a third time against the test VSAM file, creating a three times updated set of VSAM files.
11. Run the same set of updates a third time against the test DB2 tables, creating a three times updated set of DB2 tables.
12. Compare the results, screens and error messages.

13. Unload the three times updated VSAM files and DB2 tables and compare them. If discrepancies occur, they should be accounted for.

This technique will show any differences in both, the application behavior and the data. Some differences may be expected. For example, time of day stamps may not be appropriate to compare for equivalence, though the relative value may be of interest to determine elapsed time performance. Use a tool with the capability to take time of day from each database, calculate the difference and chart it for inspection. This may be valuable for performance tuning of elapsed time.

4.5 Preparing for Cutover to Production

Production cutover encompasses migrating converted applications from test to production and loading DB2 tables from the appropriate VSAM files.

All cutover activity may proceed in parallel with VSAM production. In preparation, a trial run is valuable to exercise both the data migration and the program conversion. If time constraints prevent loading the entire production database, ensure that a representative sample is loaded for volume and function testing.

A sequence of events preparing for cutover may include the following:

1. Obtain unused DASD space to meet the requirements of the cutover. This may be a good time to delete unneeded files. A complete backup of the VSAM application should be taken.
2. CREATE DB2 tables. Be sure to include the production security definitions.
3. ALTER any columns as necessary if you are adding to an existing DB2 table. Populate any ALTERed columns as appropriate.
4. Load the DB2 tables - this should ensure the data conversion programs handle all conditions existing in production data. Verification that all the data was converted may be done as follows:
    a. Check record counts and condition codes from the VSAM unload and the DB2 load phases. If VSAM records with repeating groups are converted to multiple DB2 rows, the number of records will not necessarily equal the number of rows.
    b. Check control totals and hash values as available.
    c. The VSAM unload should complete with condition code 0.
    d. The DB2 DATALOAD should complete with condition code 0.
5. Verify database contents.
6. Run UPDATE STATISTICS for tables and their related indexes. This will create statistics to be used to determine the optimum access path to the data.
7. Precompile and link-edit the applications. Optimization is based on the current data so it is important to precompile and link-edit after the DB2 tables are loaded with production data.
8. Update run sheets, flowcharts, and related operational documentation for production.

9. Ready all converted applications and related files for production. All testing of converted applications is complete at this point. This may include copying them into production libraries.

Applications which do not change the contents of the files may continue to run with the VSAM production data through the time the equivalent application is running in production using DB2.

Even applications being cutover which do updates may continue to run although portions of the data files may be limited to read-only access during cutover. Since the VSAM files may be migrated in pieces, stopping the VSAM production system and waiting for the equivalent DB2 system may not be necessary. It may be possible to defer any required VSAM updates and apply them to the production DB2 systems after the cutover.

4.5.1 Online System Verification

This is a basic verification that the migrated system is accessible and the screens are functional including queries and updates.

1. Test user access to the converted production system. This involves actually logging on to the system.
2. Call each screen in the converted system.
3. Run the tests selected to verify that the system meets requirements. This may be thought of as an acceptance test set.

Online verification may be assisted by tools which allow a playback or replay of the CICS terminal activity. The online verification may be tested once on the original system. Then to ensure the results are consistent between the original and converted systems, the terminal interaction may be replayed in exactly the same sequence as necessary to produce the desired consistency on the new system.

4.5.2 Batch System Verification

Compare all reports and control totals between the original and converted systems. Similar to the online verification test, batch may also have a test set defined.

4.6 Cutover to Production

Once all verification is complete, converted applications must be put in production mode. The sequence of events may be as follows:

1. Limit updates against the data being migrated.
2. Use the same procedures as in earlier stages of testing to ensure the unload of VSAM data and the loading of DB2 data occurs in a logical order.
3. Load the DB2 tables - this should establish a point of consistency between the VSAM and DB2 data. Just as in preparing for cutover, verification of this may be done as described above. For performance reasons, it is recommended that single user mode with LOGMODE=N be used while loading the DB2 tables.
4. Verify the database contents.

5. Run UPDATE STATISTICS for tables and their related indexes. This will create statistics to be used to determine the optimum access path to the data. This should be done after every load.

6. Make an archive.

7. Precompile and link-edit applications.

8. Move all converted applications and related files into the DB2 production environment.

9. Run verification programs for the converted applications.

4.6.1 Post Cutover

As applications and data are converted to DB2, space allocated for VSAM data may be reduced.

1. Delete VSAM files as they are moved to DB2.

2. Remove the VSAM files from backup procedures after the file has been removed from the database.

3. Archive migration libraries.

4. Archive cutover data sets.

Now is the time to start on all the maintenance ideas which were logged during the conversion phases.

Part 2. Conversion Using DB2 VSAM Transparency for VSE/ESA

Part 2, “Conversion Using DB2 VSAM Transparency for VSE/ESA” describes the conversion process with DB2 VSAM Transparency involved.

- Chapter 5, “Overview of DB2 VSAM Transparency for VSE/ESA”

  This chapter describes how DB2 VSAM Transparency for VSE/ESA works, lists its components, and shows the steps needed to run it.

- Chapter 6, “Installation of DB2 VSAM Transparency for VSE/ESA”

  This chapter describes the installation of DB2 VSAM Transparency for VSE/ESA step by step.

- Chapter 7, “Using DB2 VSAM Transparency for VSE/ESA”

  This chapter explains the steps required to use DB2 VSAM Transparency for VSE/ESA, and its capabilities and limitations.

- Chapter 8, “Beyond Transparency”

  This chapter explains the logical next steps of a conversion after Transparency has been set up. Additionally, it describes how to expand the scope of its usage by circumventing some of its limitations.

Chapter 5. Overview of DB2 VSAM Transparency for VSE/ESA

This chapter describes how DB2 VSAM Transparency for VSE/ESA works, lists its components, and shows the steps needed to run it.

5.1 Introduction

DB2 VSAM Transparency intercepts the VSAM request from an application program and performs the requested action on a DB2 database. The following VSE/VSAM file types are supported by this product:

- Key Sequenced Data Sets (KSDS)
- VSAM Alternate Indexes
- Entry Sequenced Data Sets (ESDS)
- Relative Record Data Sets (RRDS)

DB2 VSAM Transparency for VSE/ESA does **not** support Variable Length Relative Record Data Sets (VRDS).

Typically, one VSAM file is represented in one DB2 table. It is supported to have more than one table for a VSAM file and to use different record types based on field contents.

To enable Transparency, no changes are required to the application program.

Hint: In any single application program you can mix:

- VSAM accesses (to files not enabled for Transparency)
- Intercepted VSAM accesses (to files enabled for Transparency)
- SQL accesses to DB2

These different access modes do not interfere with each other, as shown in Figure 10 on page 60. This means, after migration using Transparency you can change an application even partially, step by step.

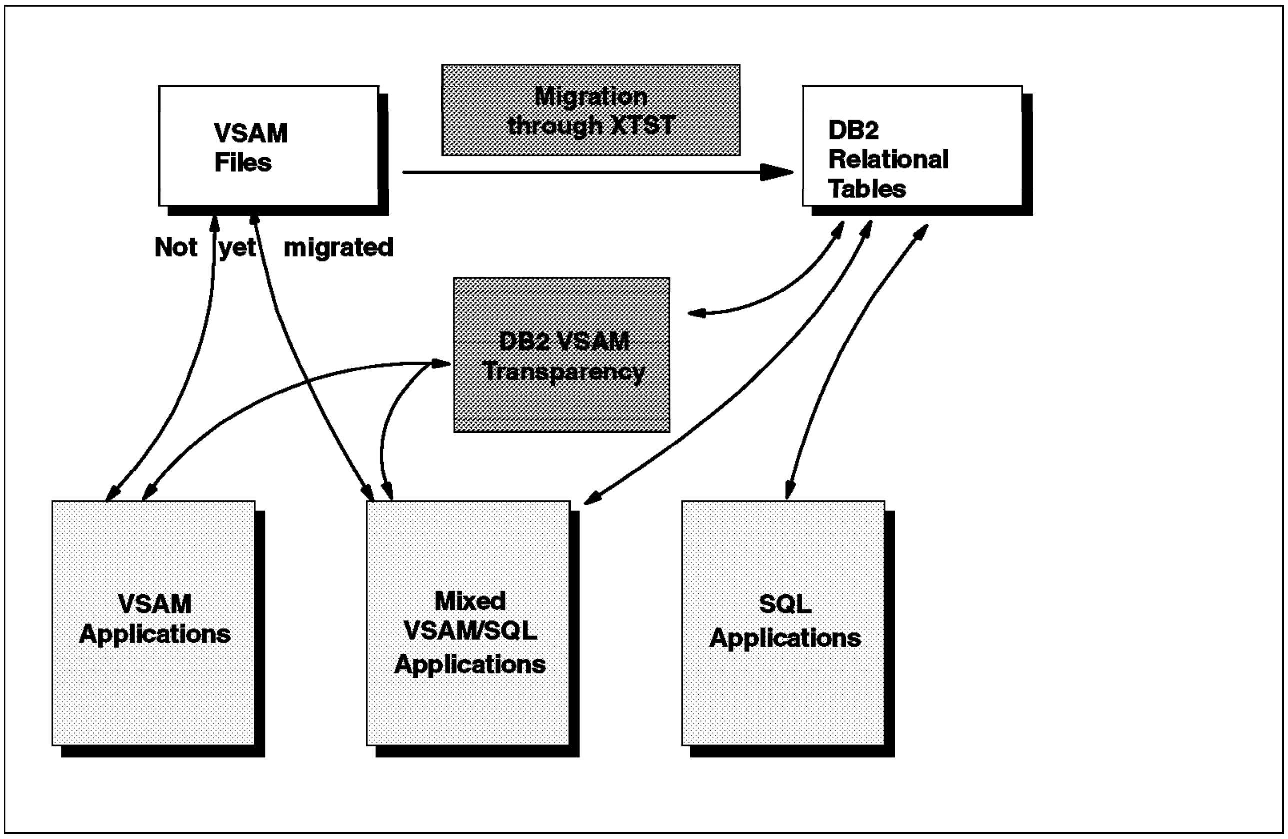


Figure 10. Data Access with DB2 VSAM Transparency

When DB2 VSAM Transparency for VSE/ESA is installed, parts of it go into the DB2 database which is to be supported.

With the help of an online tool the characteristics of each VSAM file are described as well as of the corresponding DB2 table(s). Batch programs are supplied to automatically migrate the data. For each VSAM file,

- a data move program,
- a batch access program and
- an online access program

are created. A list describes which VSAM files are migrated and which not. SVA phases intercept the VSAM accesses.

DB2 VSAM Transparency for VSE/ESA is activated by the XTSTSTRT job in batch and the XTON transaction in CICS. To de-activate DB2 VSAM Transparency for batch and online usage, apply the batch job XTSTSTOP and the CICS transaction XTOF, respectively. This also means that you can switch between using VSAM files and DB2 tables. For test purposes, DB2 VSAM Transparency for VSE/ESA can be limited to a single batch partition if required.

Migration can be performed for one VSAM file after another.

For more details, refer to the sections 7.1.1, “File Types Supported” on page 75, 7.1.2, “Capabilities” on page 76 and 7.1.3, “Limitations” on page 76.

5.2 Components of DB2 VSAM Transparency

The DB2 VSAM Transparency for VSE/ESA product consists of the following components:

- **Batch Programs**

  XTST1050 - This batch utility performs the following functions depending on the JCL SYSIN input:

  - DEFINE - this function generates the data move program, which contains all the characteristics of a VSAM file. For dependent files (ALT files, NEXT files, or SUBS belonging to a MULT file), additional programs are generated for each kind of record (SUBS files).
  - MIGRATE - this function creates the DB2 table that corresponds to the VSAM file defined by using the online tool (XTST transaction under CICS), loads the data from the VSAM file into the table, creates the primary index on the table and generates both the batch and online access modules which are punched to the internal reader for assembly and link-edit. MIGRATE does the CREATE, LOAD and GENERATE functions in one step.
  - CREATE - this function creates the DB2 table that corresponds to the VSAM file defined by using the online tool (XTST transaction under CICS).
  - LOAD - this function transfers the records from the VSAM file to the DB2 table.
  - GENERATE - this function creates the primary index on the table and generates both the batch and online access modules which are punched to the internal reader for assembly and link-edit. In the case of an alternate index (ALT), this function creates an alternate index instead of loading data into a table.

    This function is especially needed if an alternate index is added after MIGRATE has been run.
  - INIT - this function calculates the maximum Relative Byte Address (RBA) and Relative Record Number (RRN) for ESDS and RRDS files. These values are stored in the XTSTPRM phase that is loaded into the SVA.

  XTSTSTRT - this phase activates DB2 VSAM Transparency for batch partitions.

  XTSTSTOP - this phase deactivates DB2 VSAM Transparency for batch partitions.

- **CICS Programs**

  XTSP0001 - this program is used during installation to supply the CPU identification and password. Invoked by transaction code XTSP. The authors circumvented this installation step by using ISQL to insert the necessary data into the DB2 table XTSOFT.XTSTCNTL.

  XTST0001 - this is the online tool used for defining the VSAM file characteristics to DB2 VSAM Transparency as well as the column descriptions. Invoked by transaction code XTST.

  XTST0010 - this program is used to open or close VSAM files defined to DB2 VSAM Transparency in the XTSTPRM phase. Invoked by transaction code XEMT. Can only be executed when DB2 VSAM Transparency is active in the CICS partition.

XTSTON - this program activates and deactivates DB2 VSAM Transparency in the CICS partition. Invoked by transaction codes XTON and XTOF.

XPLORE - this program is used for problem determination. Invoked by transaction code XPLO.

- **SVA Phases**

  XTSTPRM - this phase contains a list of all VSAM files on which DB2 VSAM Transparency may be activated and is required for both online and batch.

  XTSTVTMS - required for batch Transparency.

  $$BCXTST - required for batch Transparency.

  $$BXTST - required for batch Transparency.

  $$BXTMSG - required for batch Transparency.

- **DB2 Tables**

  XTSOFT.XTSTCNTL - contains the CPU identifier and password.

  XTST.XTST_FILE - contains the VSAM file description and corresponding DB2 table information for all the VSAM files defined by the online tool.

  XTST.XTST_COLUMN - contains the column information for all the VSAM files defined by the online tool.

  XTST.XTST_MULT - contains the column information to identify the VSAM record layout for data sets that contain more than one record type for all the VSAM files defined by the online tool.

  XTST.XTST_DATATYPE - contains the different data types supported by VSAM.

  XTST.XTST_SQLTYPE - contains the different data types supported by DB2.

  XTST.XTST_JCL - contains the job control language (JCL) to assemble the program created during the DEFINE process. Customized from XTST.MOD_JCL by using the XTST transaction General Parameters option.

  XTST.XTST_JCL2 - contains the job control language (JCL) to assemble and link-edit the batch program created during the MIGRATE or GENERATE process. Customized from XTST.MOD_JCL2 by using the XTST transaction General Parameters option.

  XTST.XTST_JCL3 - contains the job control language (JCL) to assemble and link-edit the online program created during the MIGRATE or GENERATE process. Customized from XTST.MOD_JCL3 by using the XTST transaction General Parameters option.

  XTST.MOD_JCL - contains the model job control language (JCL) to assemble the program created during the DEFINE process.

  XTST.MOD_JCL2 - contains the model job control language (JCL) to assemble and link-edit the batch program created during the MIGRATE or GENERATE process.

XTST.MOD_JCL3 - contains the model job control language (JCL) to assemble and link-edit the online program created during the MIGRATE or GENERATE process.

5.3 Steps to Enable DB2 VSAM Transparency for a VSAM File

After the installation of the product described in Chapter 6, "Installation of DB2 VSAM Transparency for VSE/ESA" on page 65, the ′transparent′ access to a DB2 table from a simple VSAM KSDS file in an application program is achieved by performing the following steps:

1. Define the VSAM file characteristics, such as file type (KSDS), record length, key length and key position, using the XTST online tool.
2. Use the XTST online tool to define each field in the VSAM record and the corresponding DB2 column.
3. Execute the batch DEFINE function to create the data move program.
4. If not yet done, acquire the dbspace.
5. Execute the batch MIGRATE function which creates the DB2 table and loads the data into the table. If this was successful, the batch access and online access programs are created, assembled and link-edited.
6. Modify the list of files migrated and submit XTSTPRM.
7. If Transparency is active, perform XTSTSTOP (batch) and XTOF (online).
8. Execute XTSTSDL2 to load XTSTPRM into the Shared Virtual Area (SVA).
9. Activate the batch transparency by executing the XTSTSTRT job.
10. For the CICS system define two entries in the Processing Program Table (PPT) for each VSAM file and start the CICS partition, or use RDO to update CICS.
11. Activate the transparency in the CICS partition by executing the transaction XTON.

All accesses to the VSAM file will now be directed to the DB2 table.

Refer to Chapter 7, "Using DB2 VSAM Transparency for VSE/ESA" on page 75 for a detailed description of each of the steps described above. 7.13, "Summary of DB2 VSAM Transparency Step by Step" on page 99 shows the above list in more detail, also valid for other cases than just a KSDS file migrated to one table.

Chapter 6. Installation of DB2 VSAM Transparency for VSE/ESA

This chapter describes the installation of DB2 VSAM Transparency for VSE/ESA step by step. It is assumed that the reader is already familiar with VSE systems, VSE job control language, VSAM and the CICS system, and has some basic knowledge of DB2.

The following is a summary of the installation steps. They are described in the following sections in more detail.

1. Prepare installation, determining the names and values for a number of parameters.
2. Install DB2 VSAM Transparency.
3. Acquire dbspaces in the database.
4. Create tables in the database.
5. Reload the DB2 VSAM Transparency packages into the database.
6. Change the CICS configuration tables.
7. Perform post-installation initialization.
8. Customize the execution JCL set.

6.1 Prepare Installation

Obtain the latest Preventive Service Planning (PSP) information, which contains the list of required actions and service, and apply any relevant service.

If not yet done, install DB2 Server for VSE & VM and create a database according to your needs.

Before installing DB2 VSAM Transparency, determine the parameters that are needed during the installation steps and which also will affect the customization of the sample JCL provided with DB2 VSAM Transparency. Write down the appropriate values as you work through this section. Check your current system environment and standards, and refer to the VSE manuals where needed.

- Determine the sublibrary into which you are going to install DB2 VSAM Transparency. It needs one or two cylinders of DASD. We chose **PRD2.DB2VSAM**.
- Determine the storage pool for the public dbspace XTST. 1024 pages are needed. We chose storage pool **01**.
- Determine the DASD volume for temporary files. We chose **SYSWK1**.
- If you are running SQL/DS Version 3 Release 4 or later, determine the database name. We chose **SQLVSE01**.
- Determine in your DB2 database a user ID with DBA authority. It is required to have **SQLDBA** with password **SQLDBAPW**.
- A tape drive is required for loading DB2 VSAM Transparency. We chose unit **180**.

6.2 Install DB2 VSAM Transparency

The installation process for DB2 VSAM Transparency uses the standard VSE/ESA software installation procedure.

6.2.1 Define New Sublibrary

This is an optional step. However, for the purpose of program isolation and easy software maintenance, we recommend that you define a new sublibrary for DB2 VSAM Transparency. Figure 11 is a sample JCL to create a new sublibrary "PRD2.DB2VSAM." The default library name is PRD2.XTV510.

```
// JOB DEFLIB
// EXEC LIBR
DEFINE SUBLIB=PRD2.DB2VSAM
/*
/&
```

Figure 11. Sample JCL to Define a SUBLIB

6.2.2 Install from Product Tape

1. Attach a tape drive.
2. Log on as the VSE Administrator (for example, SYSA)
3. Load DB2 VSAM Transparency from the VSE/ESA Function Selection Panel
   a. select 1 Installation
   b. select 1 Install Program V2 Format
   c. select 2 Install Program(s) from Tape
4. or use Fast-Path 112
5. The following panel will be displayed on which you enter the tape identifier and select library and sublibrary where the product will be installed.

```
LIST OF PROGRAMS TO BE INSTALLED

OPTIONS:  1 = INSTALL    2 = SKIP INSTALLATION    5 = DELETE ENTRY

                                LIBRARY   SUBLIBR.
  OPT       IDENTIFIER          NAME      NAME      SEQ.NO.  TAPE NO.

   1        DB2VSE.X  TV.5.1.0  PRD2      XTV510       1        1
   1        ________  ________  PRD2      PROD         2        1
   1        ________  ________  PRD2      PROD         3        1
   1        ________  ________  PRD2      PROD         4        1
   1        ________  ________  PRD2      PROD         5        1
   1        ________  ________  PRD2      PROD         6        1

F1=HELP      2=REDISPLAY  3=END                    5=PROCESS
             8=FORWARD
ENTER YOUR PRODUCTS IN AN EMPTY LINE.
```

Figure 12. Install Panel for DB2 VSAM Transparency

6. Then press PF5 to create the installation jobstream.

7. Submit the job to start the installation.

8. At this point in time you might want to punch the JCL files needed later. Refer to Chapter 7, "Using DB2 VSAM Transparency for VSE/ESA" on page 75.

6.3 Acquire Dbspace

To acquire the dbspace needed by DB2 VSAM Transparency, punch member XTVACQ.Z to your private ICCF library (for example, using LIBRP command). Modify the job as shown in the example to your needs.

```
 (1)--->.* $$ JOB JNM=XTVACQ,CLASS=0,DISP=D
         // JOB XTVACQ
 (2)--->.// LIBDEF *,SEARCH=PRD2.DB2510
 (3)--->.// EXEC ARIDBS,SIZE=AUTO,PARM=¢DBNAME(DBNAME)¢
         SET ERRORMODE CONTINUE;
 (4)--->.CONNECT SQLDBA IDENTIFIED BY SQLDBAPW;
 (5)--->.ACQUIRE PUBLIC DBSPACE NAMED XTST(PAGES=1024);
 (6)--->.GRANT DBA TO XTST IDENTIFIED BY XTSTPW;
         /*
         /&
        ..* $$ EOJ
```

Figure 13. Sample for DBSPACE DB2 VSAM Transparency

- Statement 1 : Alter POWER statement to run the job in an appropriate class.
- Statement 2 : Specify the library/sublibrary for DB2 V5.1.0.
- Statement 3 : Replace DBNAME with your database name.
- Statement 4 : Specify the connect password for user ID SQLDBA.
- Statement 5 : Acquires the required public dbspace. You can modify this to add a storpool parameter.
- Statement 6 : GRANT DBA authority to XTST user.

You can also use ISQL to acquire the dbspace.

Start ISQL and execute the following command sequence:

```
   CONNECT SQLDBA IDENTIFIED BY sqldbapw
   GRANT DBA TO XTST IDENTIFIED BY XTSTPW
   ACQUIRE PUBLIC DBSPACE NAMED XTST(PAGES=1024,STORPOOL=01)
```

Figure 14. Sample ISQL Command Sequence for Acquire Dbspace

sqldbapw is the password for the database administrator (SQLDBA). The dbspace acquired must be in a recoverable storage pool.

6.4 Create Tables

To have DB2 VSAM Transparency create the control tables it needs you have to run job XTVCRE, shown below. Punch the job into your private ICCF library and modify it to suit your needs.

```
(1)--->..* $$ JOB JNM=XTVCRE,CLASS=0,DISP=D
        // JOB XTVCRE
(2)--->// LIBDEF *,SEARCH=PRD2.DB2510
(3)--->// EXEC ARIDBS,SIZE=AUTO,PARM=¢DBNAME(DBNAME)¢
       SET ERRORMODE CONTINUE;
(4)--->CONNECT XTST IDENTIFIED BY XTSTPW;
       GRANT CONNECT TO XTSOFT  IDENTIFIED BY XTSOFT;
       DROP TABLE XTSOFT.XTSTCNTL;
       SET ERRORMODE OFF;
       CREATE TABLE
         XTSOFT.XTSTCNTL(
         PWD        CHAR(11) NOT NULL,
         CPU        CHAR(05) NOT NULL) IN XTST;
       CREATE UNIQUE INDEX IXTSTCNTL ON XTSOFT.XTSTCNTL(CPU);
       COMMIT WORK;
       SET ERRORMODE CONTINUE;
       ...
       /*
       /&
       * $$ EOJ
```

Figure 15. Sample for CREATE TABLES DB2 VSAM Transparency

- Statement 1 : Alter POWER statement to run the job in an appropriate class.
- Statement 2 : Specify the library/sublibrary for DB2 V5.1.0.
- Statement 3 : Replace DBNAME with your database name.
- Statement 4 : Specify the connect password for user ID SQLDBA.

Note: The job should end with a return code of 0.

6.5 Reload DB2 VSAM Transparency Packages into the Database

```
(1)--->..* $$ JOB JNM=XTVPREP,CLASS=0
        // JOB XTVPREP
        * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
        * XTVPREP: VSAM TRANSPARENCY
        *        : RELOAD PACKAGES
        * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
// LIBDEF *,SEARCH=prd2.db2510
(2)--->..// EXEC ARIDBS,SIZE=AUTO,PARM=¢DBNAME(dbname)¢
(3)--->  CONNECT SQLDBA IDENTIFIED BY sqldbapw;
(4)--->  RELOAD PROGRAM(XTST.XTSTP011) INFILE(SYSIPT)REPLACE;
(5)--->  !!Ø!XTST    !!XTSTP011!!!!!!!!!!!!!ENGLISH           !!!!
         !@ !C0                NY!;!!!}•!!!u!!Î•!ÎyCOMMIT   NY!!!;!!!!
         !Î{EXECUTE           !Y!!!!!!!!!!!!!!!!!!!!
         !À!!!;!!!D!!:!íSELECT DDNAME,ACCMETH FROM XTST.XTST_FILE WHERE
         PRIMENAME=:FIRSTNAME ORDER BY ACCMETH;
         !!!!!!!!:!!           COMMIT WORK;
         /*
         ...
         /*
         /&
         * $$ EOJ
```

Figure 16. Sample for Reloading DB2 VSAM Transparency into Database

- Statement 1 : Alter POWER statement to run the job in an appropriate class.
- Statement 2 : Replace DBNAME with your database name.
- Statement 3 : Specify the connect password for user ID SQLDBA.
- Statement 4 : Reload the programs
- Statement 5 : The package (object code) to be reloaded.

6.6 Change the CICS Configuration Tables

To use DB2 VSAM Transparency in a CICS environment, you have to modify your CICS PPT and PCT tables. There are two possibilities to update such definitions:

- Use RDO for online definition
- Modify the tables and compile in batch

The following sections show some examples. Refer to the CICS manuals for detailed information.

6.6.1 CICS Installation Using RDO

To update your CICS tables using RDO, execute the job in member XTVRDO shown partly in the following figure. Punch the member.

```
 (1)--->..* $$ JOB JNM=XTVRDO,CLASS=0,DISP=D
           // JOB XTVRDO
 (2)--->  // LIBDEF *,SEARCH=PRD1.BASE
 (3)--->  // DLBL DFHCSD,¢CICS.CSD¢,,VSAM,CAT=VSESPUC
 (4)--->  // EXEC DFHCSDUP,SIZE=AUTO
 (5)--->  DELETE GROUP(XTST)
          DEFINE PROGRAM(XPLORE)  LANGUAGE(ASSEMBLER) GROUP(XTST)
          DEFINE PROGRAM(XTST0001)  LANGUAGE(ASSEMBLER) GROUP(XTST)
          DEFINE PROGRAM(XTST0002)  LANGUAGE(ASSEMBLER) GROUP(XTST)
          DEFINE PROGRAM(XTSH0063)  LANGUAGE(ASSEMBLER) GROUP(XTST)
          DEFINE TRANSACTION(XTST)  PROGRAM(XTST0001)  GROUP(XTST)
          DEFINE TRANSACTION(XEMT)  PROGRAM(XTST0010)  GROUP(XTST)
          DEFINE TRANSACTION(XTON)  PROGRAM(XTSTON)  GROUP(XTST)
          DEFINE TRANSACTION(XTOF)  PROGRAM(XTSTON)  GROUP(XTST)
          DEFINE TRANSACTION(XTSP)  PROGRAM(XTSP0001)  GROUP(XTST)
          DEFINE TRANSACTION(XPLO)  PROGRAM(XPLORE) GROUP(XTST)
          ...
         /*
         /&
       ..* $$ EOJ
```

Figure 17. Sample to Update CICS Tables via RDO DB2 VSAM Transparency into Database

- Statement 1 : Alter POWER statement to run the job in an appropriate class.
- Statement 2 : Specify your CICS library/sublibrary.
- Statement 3 : Specify your CSD file label.
- Statement 4 : Runs program DFHCSDUP.
- Statement 5 : Executes the RDO command to define VSAM Transparency objects. (Only a part is shown here.)

6.6.2 Compiling CICS PPT and PCT Tables

Since copy books have been loaded into the DB2 VSAM Transparency library during installation of the product, you only need to add these copy books into the PPT and PCT tables. For more details, refer to the CICS manuals.

DFHPCT Modifications

Add the following copy books into your PCT table:

```
      copy xtstpct
      copy xtsppct
```

Figure 18. DFHPCT Modifications to Install Transparency

Add the DB2 VSAM Transparency library to the LIBDEF statement, and submit the job to assemble.

DFHPPT Modifications

Add the following copy books into your PPT table:

```
    copy xtstppt
    copy xtspppt
```

Figure 19. DFHPPT Modifications to Install Transparency

Add the DB2 VSAM Transparency library to the LIBDEF statement, and submit the job to assemble.

DFHSIT Verification

Verify in your SIT that

```
BFP=YES
```

Figure 20. DFHSIT Modifications to Install Transparency

is specified otherwise modify it.

Note: Verify that your DFHEIP is not loaded into the SVA.

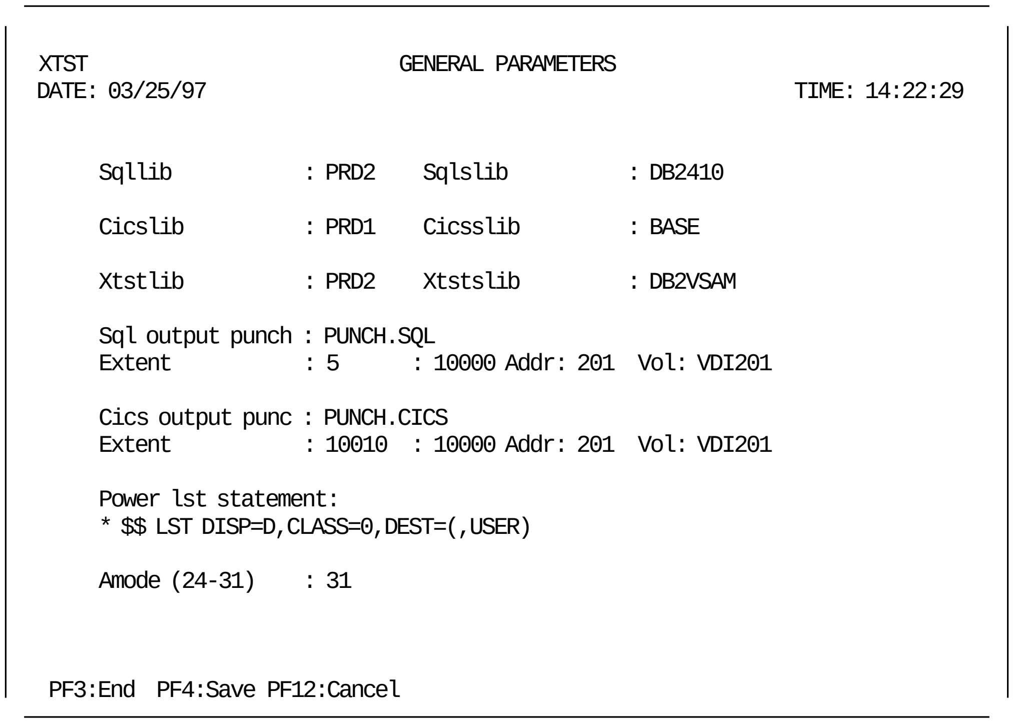
6.6.3 Customize CICS Operating Environment

Add the DB2 VSAM Transparency library to the LIBDEF statement of the CICS STARTUP JCL. Shut down CICS and start it again with this new JCL.

6.7 Customize the Execution JCL Set

This step is supported by a CICS tool. In PRODCICS activate CICS Transaction XTST to get the DB2 VSAM Transparency Online panel. Then use PF8 for "General Parameters." Figure 21 on page 72 is an example how we customized it. In this example we are using a Virtual Disk for temporary files.

Note: It is sufficient to specify either address or volume.



```
XTST                             GENERAL PARAMETERS
DATE: 03/25/97                                                      TIME: 14:22:29


     Sqllib           : PRD2    Sqlslib          : DB2410

     Cicslib          : PRD1    Cicsslib         : BASE

     Xtstlib          : PRD2    Xtstslib         : DB2VSAM

     Sql output punch : PUNCH.SQL
     Extent           : 5      : 10000 Addr: 201  Vol: VDI201

     Cics output punc : PUNCH.CICS
     Extent           : 10010  : 10000 Addr: 201  Vol: VDI201

     Power lst statement:
     * $$ LST DISP=D,CLASS=0,DEST=(,USER)

     Amode (24-31)    : 31



 PF3:End  PF4:Save PF12:Cancel
```

Figure 21. Screen to Customize JCL

6.8 Guest Sharing

Set up the DBNAME directory to point to the VM database as the default - refer to the DB2 Installation and the DB2 System Administration manuals for information on how to do this. This is very important to do as there is no way of specifying which database to access when using DB2 VSAM Transparency for batch execution of application programs.

In **ALL** the batch job streams that access the VM database using Guest Sharing, insert

```
// SETPFIX LIMIT=1M
```

before the EXEC statement. For example, during installation of DB2 VSAM Transparency in:

- XTVACQ - acquire dbspace
- XTVCRE - create and load tables

  Insert it also in quotes into the MOD_JCL2 and MOD_JCL3 tables within XTVCRE.
- XTVPREP - load DB2 VSAM Transparency packages

and in running the following batch jobs:

- XTSTDBA
- XTSTDEF
- XTSTMIGR
- XTSTCREA
- XTSTLOAD
- XTSTGEN

- XTSTINIT

When using the MIGRATE, GENERATE, CREATE and LOAD processes, specify the database name on the EXEC statement, for example:

```
// EXEC XTST1050,SIZE=AUTO,PARM=¢DBNAME(SQLVMDB1)¢
```

In the CICS environment, use the CIRB transaction to point to the VM database as the default when using Guest Sharing.

Chapter 7. Using DB2 VSAM Transparency for VSE/ESA

This chapter explains the steps required to use DB2 VSAM Transparency for VSE/ESA, and its capabilities and limitations. An overview of the steps is given in 5.3, “Steps to Enable DB2 VSAM Transparency for a VSAM File” on page 63, with more details in 7.13, “Summary of DB2 VSAM Transparency Step by Step” on page 99.

You should not attempt any steps of the DB2 VSAM Transparency without completing the VSAM file and program inventory.

Note: When calling XTST1050, it is recommended to add the parameter ″PARM=′DBNAME(SQLVSE01)′″, even if this is the default database. SQLVSE01 is the name of the database we used when writing this redbook. If this parameter is missing, Transparency tries to take the default database, which is indicated by ARISDIRD. Remember that when DB2 VSAM Transparency for VSE/ESA is installed, parts of it go into the DB2 database which is to be supported.

7.1.1 File Types Supported

VSAM file types

KSDS A VSAM KSDS file having only one kind of record.

ESDS A VSAM ESDS file.

RRDS A VSAM RRDS file.

KSDS Transparency file types

ALT A VSAM alternate index file to a KSDS file. In the KSDS definition screen there is no hint that an ALT exists.

MULT A VSAM KSDS file having different kinds of records (for example HEADER and DETAIL records).

SUBS A subset of a VSAM KSDS file having different kinds of records (MULT). Each specific kind of record is defined as a SUBS file, for example, HEADER as one SUBS file and DETAIL as a second SUBS file.

Example: In Figure 23 on page 78 the KSDS file SQTVVW2 is a MULT file with the SUBS files SQTV2W2 and SQTV3W2 (and the virtual file SQTV4W2, see below).

Continuation

NEXT A “virtual” file for the continuation of a KSDS, RRDS, ESDS, NEXT, or SUBS file. A VSAM KSDS file can contain very long records. When migrating the file into DB2 tables, only 255 columns can be defined per table, or you may want to split the VSAM file into several tables for design reasons. A NEXT file is processed immediately after the KSDS, RRDS, ESDS, SUBS or NEXT file to which it belongs.

Example: In Figure 23 on page 78 the SQTV3W2 file is a SUBS file and has a NEXT file SQTV4W2. SQTV4W2 is a virtual file and belongs to SQTV3W2. Because this is a SUBS file, finally SQTV4W2 belongs to SQTVVW2.

A NEXT file can also have a NEXT file.

7.1.2 Capabilities

1. Create and load tables from VSAM files.
2. Run transparent access to the DB2 tables without changing the application, neither source nor object.
3. In any single application program you can mix
   - VSAM accesses (to files not enabled for Transparency)
   - Intercepted VSAM accesses (to files enabled for Transparency)
   - SQL accesses to DB2

   as shown in Figure 10 on page 60.
4. One VSAM file can be converted to more than one table.
5. Redefines (different record types) are supported (MULT/SUBS).
6. Alternate index files are supported (ALT, GENERATE).
7. Convert column formats.
   - Convert 2 digit year CHAR field to 4 digit year DATE format.
8. DB2 VSAM Transparency allows for repeating groups to be migrated into different DB2 tables and columns (not rows, see below).
9. Additional columns can be created in DB2 that did not exist in the VSAM file.
10. Fields can be mapped into more than one column. Loading from DB2 into the VSAM buffer is performed according to the sequence number given (see *SEQ* in Figure 27 on page 82.) Transparency performs synchronous updates.
11. Default values for DB2 and VSAM can be given.
12. User exits can be specified on the file description screen. Each time a DB2 access is then processed, these user exits are executed either before and/or after the request.
13. User exits can be specified on the columns format description screen. Each time the column is processed (from DB2 to VSAM or from VSAM to DB2), this user exit is executed.
14. A file can be mapped into a DB2 table already defined by another VSAM file by creating a view for each file.

7.1.3 Limitations

1. **Only one database** is supported by DB2 VSAM Transparency; into this database DB2 VSAM Transparency has to be installed.
2. VRDS VSAM type is not supported.
3. A VSAM file allows only one prime key. Therefore, DB2 VSAM Transparency allows only **one column** to be the **primary key** for a DB2 table, although DB2 allows a primary key to be made up of more than one column.
4. DB2 VSAM Transparency allows for repeating groups to be migrated into different DB2 tables and columns, but does **not** allow periodic **fields** to be moved into **rows**, see “Repeating Groups (Fields)” on page 33.
5. For MULT files, DB2 VSAM Transparency allows **only one level of decision to branch** into the SUBS. No logical AND and OR operations are allowed. But since the decision is made sequentially you can circumvent this.

6. For **ALT files** (alternate keys), the **NOUPGRADE** option **cannot** be handled. DB2 VSAM Transparency treats it like the UPGRADE option. Since this option was mainly for VSAM performance, the only consequence is that newer information will be provided.
7. **DFHEIP** must not be loaded in the SVA; if it has been, it must be detached and loaded elsewhere in the CICS partition.
8. XEMT instead of **CEMT** is to be used as an operator command to open and close Transparency files.
9. Only "normal" VSAM file accesses can be converted; **user-written I/O and your own CCWs cannot be handled**.
10. No ALT files on a MULT file can be processed.
11. No update is allowed on an ALT file.
12. If the capability of converting column formats is used upon the key column, the deletion of a row might result in an error.

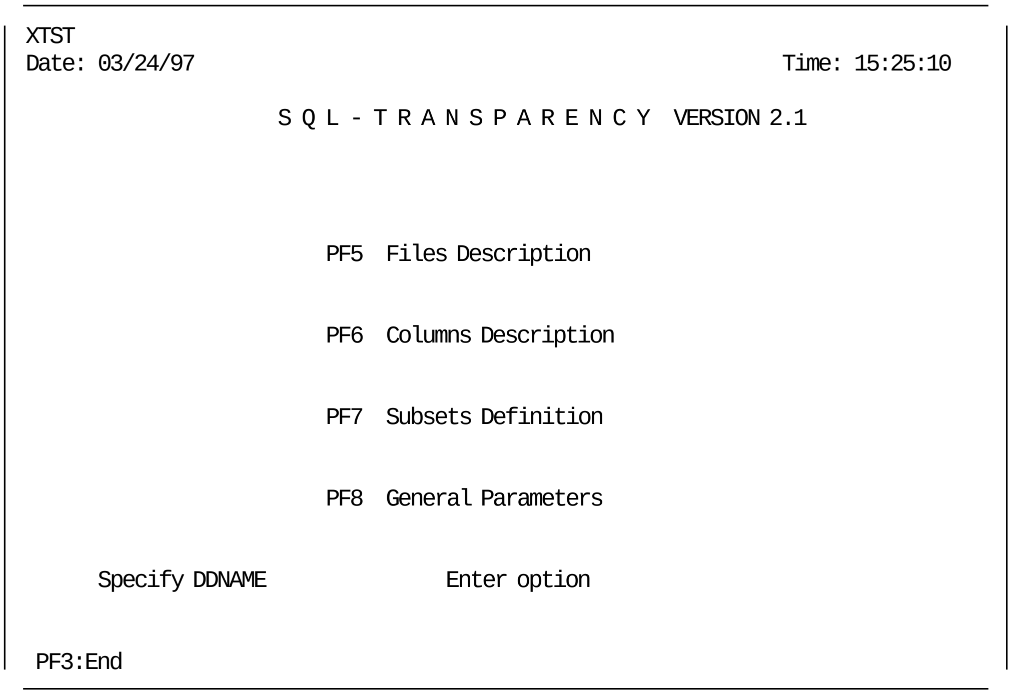
7.2 Defining VSAM Files

To enable Transparency for a VSAM file, the first step is to define all the characteristics of the file, such as file type (for example KSDS, ESDS, ALT), record length, key length and so on. The second step is to define the field characteristics: For each VSAM field in a record, the starting position, the length and type of data and the associated DB2 column name, type and length are defined. For invalid VSAM data, a default value for the DB2 column can be specified.

An online tool is provided under CICS to specify all that information for DB2 VSAM Transparency. This section describes how to use this online tool to define VSAM file characteristics.

7.2.1 Main Screen

To start the online component of DB2 VSAM Transparency for VSE/ESA, execute the XTST transaction. Figure 22 on page 78 shows the main screen.



```
XTST
Date: 03/24/97                                           Time: 15:25:10
                    S Q L - T R A N S P A R E N C Y  VERSION 2.1



                          PF5  Files Description


                          PF6  Columns Description


                          PF7  Subsets Definition


                          PF8  General Parameters


      Specify DDNAME             Enter option


 PF3:End
```

Figure 22. Main Screen of DB2 VSAM Transparency for VSE/ESA

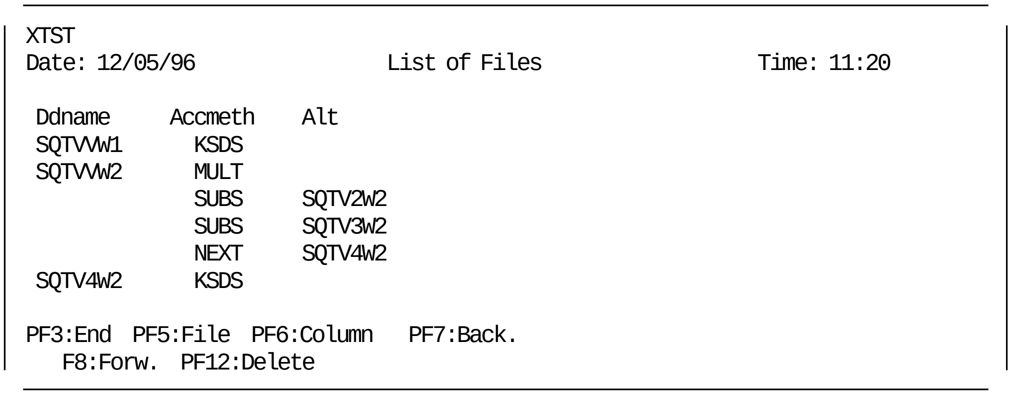
Specify the *ddname* of the VSAM file to edit, and press PF5.

If the VSAM file is unknown for XTST an empty file description screen will appear for creation, otherwise the information specified previously will be displayed.

To list all the defined file names, the user must specify a '?' in the first character of the *ddname* field and press the ENTER key.

7.2.2 List of Files

Figure 23 shows a sample list of files screen.



```
XTST
Date: 12/05/96              List of Files              Time: 11:20

 Ddname     Accmeth    Alt
 SQTVVW1      KSDS
 SQTVVW2      MULT
              SUBS     SQTV2W2
              SUBS     SQTV3W2
              NEXT     SQTV4W2
 SQTV4W2      KSDS

PF3:End  PF5:File  PF6:Column   PF7:Back.
   F8:Forw.  PF12:Delete
```

Figure 23. Sample List of Files Screen of DB2 VSAM Transparency

To choose one file, the user positions the cursor on the *ddname* to edit and presses a function key: ENTER to select *ddname* and return with it to the main screen, PF5 to edit the file description, or PF6 to edit the columns description.

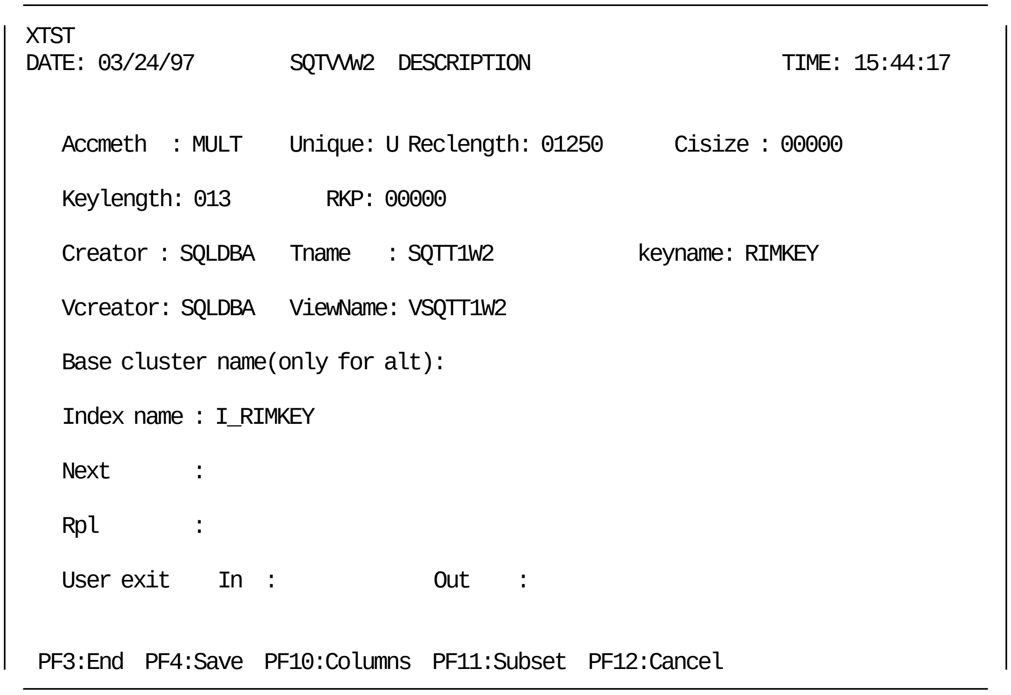
To delete a file definition, position the cursor on the *ddname* to delete and press PF12. Confirm by pressing PF12 once more. Only the DB2 VSAM Transparency definition will be deleted, but neither the DB2 table if already created (refer to 7.5, “MIGRATE” on page 87 and 7.6, “CREATE” on page 88) nor the data move and access programs (see 7.3, “DEFINE” on page 85 and 7.8, “GENERATE” on page 90).

7.2.3 File Description

To edit the file description, specify on the main screen the *ddname* of the file to edit and press PF5 or select the *ddname* from the list of files screen.

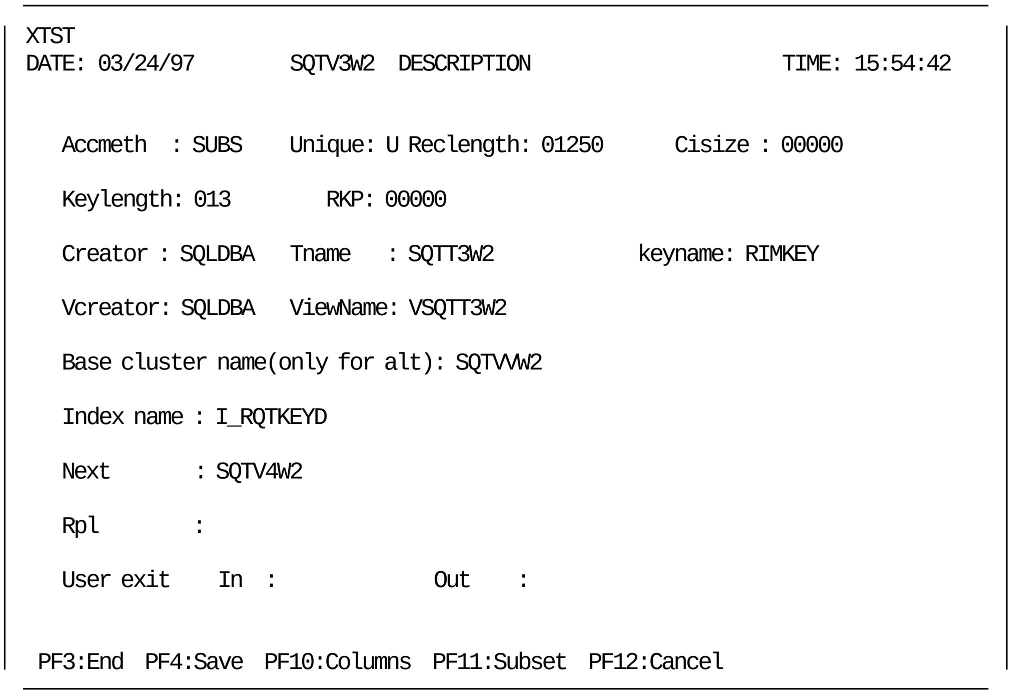
If the *ddname* specified is new for XTST the user must fill in all fields. Otherwise, the current values are displayed and can be modified. Figure 24 shows a sample file description screen for MULT. Figure 25 on page 80 shows a sample file description screen for SUBS.

Note: Don't forget to enter the base cluster name on the screen.



```
XTST
DATE: 03/24/97       SQTVW2  DESCRIPTION                  TIME: 15:44:17

   Accmeth  : MULT    Unique: U Reclength: 01250      Cisize : 00000

   Keylength: 013        RKP: 00000

   Creator : SQLDBA   Tname   : SQTT1W2           keyname: RIMKEY

   Vcreator: SQLDBA   ViewName: VSQTT1W2

   Base cluster name(only for alt):

   Index name : I_RIMKEY

   Next       :

   Rpl        :

   User exit    In  :             Out    :

 PF3:End  PF4:Save  PF10:Columns  PF11:Subset  PF12:Cancel
```

Figure 24. Sample File Description Screen for MULT



```
XTST
DATE: 03/24/97       SQTV3W2  DESCRIPTION                  TIME: 15:54:42


   Accmeth  : SUBS    Unique: U Reclength: 01250      Cisize : 00000

   Keylength: 013       RKP: 00000

   Creator : SQLDBA   Tname   : SQTT3W2           keyname: RIMKEY

   Vcreator: SQLDBA   ViewName: VSQTT3W2

   Base cluster name(only for alt): SQTVVW2

   Index name : I_RQTKEYD

   Next       : SQTV4W2

   Rpl        :

   User exit    In  :            Out    :


 PF3:End  PF4:Save  PF10:Columns  PF11:Subset  PF12:Cancel
```

Figure 25. Sample File Description Screen for SUBS with a NEXT File

Specify the VSAM file characteristics and the corresponding DB2 table name and the view name. If the view name is identical to the table name, no view will be generated.

Hint: The *RKP* (relative key position) count starts from 0, as VSAM does - as opposed to the start column in the column description, which follows the standard of all programming languages and starts with 1. See 7.2.5, “Columns Description” on page 81.

ALT: For alternate files *(Accmeth=ALT)* enter the base cluster name.

MULT: If a VSAM KSDS file contains different record types, specify MULT in the *(Accmeth=MULT)* field, and press PF11 to obtain the Subset Definition Screen to define which record format to choose depending on which condition.

For the MULT file, a table will be created containing the key column only; the SUBS files will be represented in one or more tables as specified in SUBS.

SUBS: Each record format is defined as a separate file using again the file description screen with *Accmeth=SUBS*. Enter the base cluster name there (name of the MULT file).

Each record type must be defined in a SUBS file, and the subset definition of the MULT file determines the condition under which a record type is applied. Each record type (SUBS file) will later generate a DB2 table (or more than one using NEXT).

NEXT: If the VSAM file (*Accmeth=KSDS, RRDS, ESDS, SUBS* or *NEXT*) is to be split into more than one table, you can define a DB2 table for any part of the record. To specify the fact that there will be a *NEXT* file definition, fill in a "virtual" file name in the *NEXT* field.

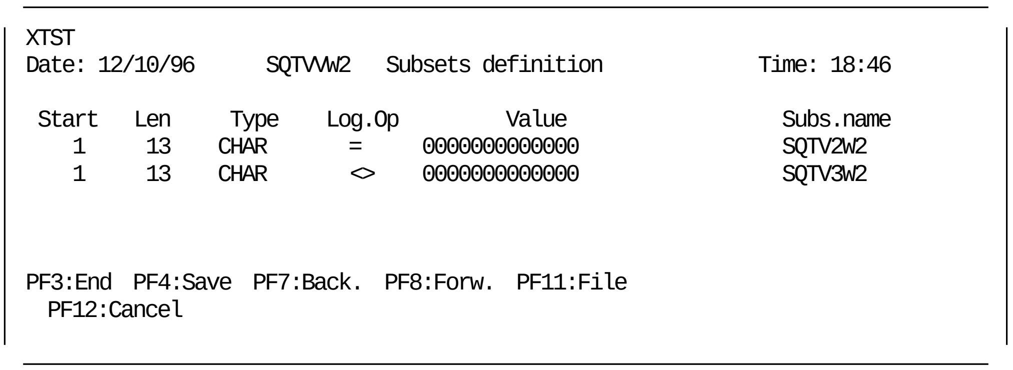
Later on, you must fill in a new file description for the virtual file using again the file description screen with *Accmeth=KSDS, RRDS or ESDS*, respectively. Again, a continuation is allowed by entering another virtual file name in the *NEXT* field. Otherwise, leave it blank.

Hint: A MULT file cannot have a NEXT file, but a SUBS file can.

7.2.4 Subset Definition

This function specifies for a file with multiple record types (MULT files), under which conditions which subset (VSAM record types) is used for a record.

To define the subsets, specify the *ddname* on the main screen and press PF7, or on the file definition screen of the MULT file press PF11. Figure 26 shows a sample subset definition screen.



```
XTST
Date: 12/10/96      SQTVVW2   Subsets definition           Time: 18:46

 Start   Len     Type    Log.Op        Value                  Subs.name
    1     13    CHAR       =     0000000000000                SQTV2W2
    1     13    CHAR       <>    0000000000000                SQTV3W2



PF3:End  PF4:Save  PF7:Back.  PF8:Forw.  PF11:File
  PF12:Cancel
```

Figure 26. Sample Subset Definition Screen

The conditions are analyzed sequentially. This means, if the first condition is encountered for a VSAM record, other conditions are not evaluated.

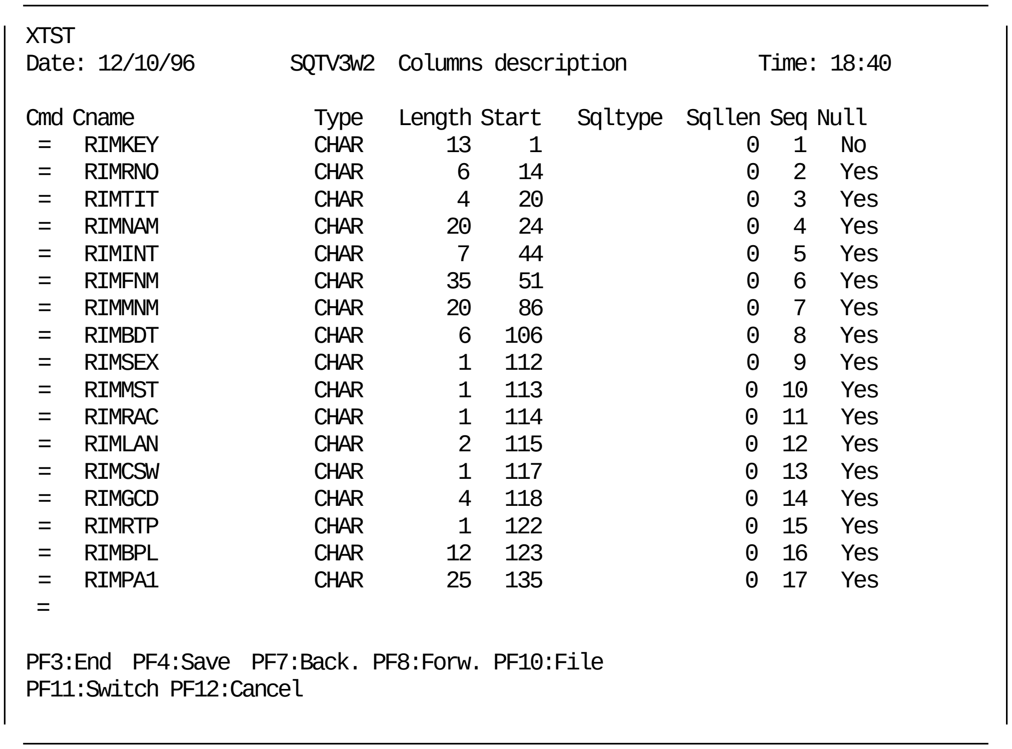
If the SUBS name contains *NONE*, then the record will be dropped during conversion.

7.2.5 Columns Description

This function specifies the VSAM field characteristics and the corresponding columns in the DB2 table.

To edit the columns description, specify the *ddname* of the file on the main screen and press PF6; or place the cursor on the *ddname* in the list of files screen and press PF6; or from the files description screen, press PF10.

A columns description is not available for alternate files. If the function is executed for such files, the associated base cluster columns descriptions will be shown. Figure 27 on page 82 shows a sample columns description screen.



```
XTST
Date: 12/10/96       SQTV3W2  Columns description          Time: 18:40

Cmd Cname               Type   Length Start   Sqltype  Sqllen Seq Null
 =   RIMKEY             CHAR       13     1                 0   1   No
 =   RIMRNO             CHAR        6    14                 0   2   Yes
 =   RIMTIT             CHAR        4    20                 0   3   Yes
 =   RIMNAM             CHAR       20    24                 0   4   Yes
 =   RIMINT             CHAR        7    44                 0   5   Yes
 =   RIMFNM             CHAR       35    51                 0   6   Yes
 =   RIMMNM             CHAR       20    86                 0   7   Yes
 =   RIMBDT             CHAR        6   106                 0   8   Yes
 =   RIMSEX             CHAR        1   112                 0   9   Yes
 =   RIMMST             CHAR        1   113                 0  10   Yes
 =   RIMRAC             CHAR        1   114                 0  11   Yes
 =   RIMLAN             CHAR        2   115                 0  12   Yes
 =   RIMCSW             CHAR        1   117                 0  13   Yes
 =   RIMGCD             CHAR        4   118                 0  14   Yes
 =   RIMRTP             CHAR        1   122                 0  15   Yes
 =   RIMBPL             CHAR       12   123                 0  16   Yes
 =   RIMPA1             CHAR       25   135                 0  17   Yes
 =

PF3:End  PF4:Save  PF7:Back. PF8:Forw. PF10:File
PF11:Switch PF12:Cancel
```

Figure 27. Sample of a Columns Description Screen

Specify all the VSAM fields to be created in the DB2 table. For existing fields in the VSAM file, type must be specified with length and start positions.

If the column allows nulls, specify *Yes* in the rightmost field *NULL*.

To delete a column, enter “D” in the *CMD* field and press the ENTER key.

To quit the screen, save the modifications by pressing PF4 or PF12 to cancel the modifications.

Hints: Do not allow nulls for the key column because a primary key cannot be NULL. In the case that an alternate file (ALT) exists, this might lead to processing errors.

Fields can be mapped into more than one column. The *SEQ* field (you can change it) determines the sequence of loading the columns into the VSAM buffer. This is important to know if updates have occurred to some but not all of the DB2 data (not-synchronous updates). Transparency performs synchronous updates.

Similarly, a MULT key column (represented as the only column in the table of the MULT file) can again be defined within a SUBS file.

Additional columns can be created in DB2 that did not exist in the VSAM file. Specify SQL in the *Type* field and specify *Sqltype* and *Sqllen* in the appropriate fields. An example where this is useful is given in Chapter 8, “Beyond Transparency” on page 103.

If you want to use a different *Sqltype* than the default mapping, enter that type in this column and specify a default value on the default value description screen. To go there, press PF11. If you choose the default mapping, omit the *Sqltype* and *Sqllen* entries, which leads to “0” in *Sqllen*.

The *Column Start* follows the standard of all programming languages, starting with 1 - as opposed to the *RKP* (relative key position) count, which starts from 0, as VSAM does. See 7.2.3, “File Description” on page 79.

7.2.6 Columns Default Values Description

This screen is obtained by pressing PF11 on the columns description screen.

SQL Forced Value

For new DB2 columns that do not exist in the VSAM file (*TYPE SQL* on columns description screen), or for numeric fields (PACKED or ZONED) in the VSAM file if the record to be loaded contains an invalid value, you can load a default value instead of canceling the process (for example, the LOAD or MIGRATE process, see 7.5, “MIGRATE” on page 87 and 7.7, “LOAD” on page 89).

If the column allows NULLs (*NULL YES*) you can specify '?' in the *SQL FORCED* value field to initialize the column with NULL.

DB2 column types DATE, TIME and TIMESTAMP can be initialized with the current value by specifying CUR in the default value field.

VSAM Forced Value

The *VSAM FORCED VALUE* will be used if a column contains a NULL (if allowed) or the *SQL FORCED* value. Figure 28 shows a sample columns default value screen.

```
XTST
Date: 12/02/96       REGFTRN  Columns description        Time: 18:05:47

     Cname         SQL forced (invalid-unknown)   VSAM forced value
RITPCD
RITDCS
RITWSW
RITWDT
RITWTP
RITWRE
RITWAN
RITCRS
RITTRC
RITSTP
RITRNO
RITDVC
RITDVN
RITRSW
RITRDT             CURRENT DATE
RITCVL
RITSDT             CURRENT DATE

PF3:End  PF7:Back.  PF8:Forw.  PF11:Switch  PF12:C
ancel
```

Figure 28. Sample Columns Default Values Description Screen

To save the modifications, use PF3 (or press PF11 twice) to display the column description screen and then press PF4 to save. It is advisable to return and save before scrolling.

7.2.7 Columns Format Description

This screen is obtained by pressing PF11 on the columns default values description screen.

If you want to modify the format of a column between the VSAM file and the DB2 column, you must specify the format in the *DATA FORMAT* field. Each character &n represents the nth position in the VSAM field. Figure 29 shows a sample columns format description screen changing the columns within the date in the VSAM file from YYMMDD format into 19YY-MM-DD format in DB2.

Another example: if you want to specify a timestamp format for a

```
YY MM DD
```

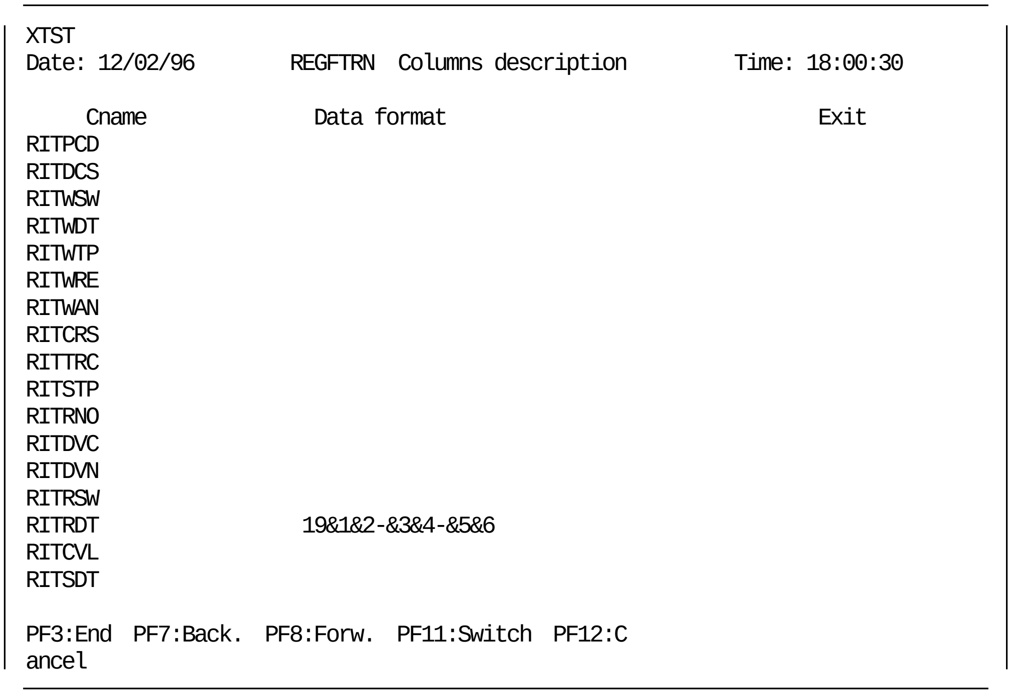
VSAM field you must specify

```
19&1&2-&3&4-&5&6-01.01.01.000001
```

For some specific cases, it might be necessary to perform a special action when data is loaded or removed. If an *EXIT* is specified, the *EXIT* program will be called after the column (specified in the field *CNAME*) has been loaded into or removed from the VSAM buffer.

Please note that DB2 VSAM Transparency for VSE/ESA **only** uses the ISO format for date and time.

For DB2 data types of CHARACTER, DATE, TIME and TIMESTAMP use the VSAM data type of CHARACTER. For the DB2 numeric data types, the VSAM data types must also be numeric.



```
XTST
Date: 12/02/96       REGFTRN  Columns description        Time: 18:00:30

     Cname              Data format                              Exit
RITPCD
RITDCS
RITWSW
RITWDT
RITWTP
RITWRE
RITWAN
RITCRS
RITTRC
RITSTP
RITRNO
RITDVC
RITDVN
RITRSW
RITRDT                 19&1&2-&3&4-&5&6
RITCVL
RITSDT

PF3:End  PF7:Back.  PF8:Forw.  PF11:Switch  PF12:C
ancel
```

Figure 29. Sample Columns Format Description Screen

7.2.8 User Exit Definition

A user *EXIT* name can be specified on the columns format description screen (see Figure 29 on page 84). Each time the column is processed (from DB2 to the VSAM buffer or from the VSAM buffer to DB2) the specified *EXIT* is called. An *EXIT* can be required to modify data after insertion into the DB2 table or after moving it into the VSAM buffer.

Punch member XTSTEXIT.A from the source library to obtain a JCL example for the EXIT function.

Hint: In addition there are also the Installation Exits available from DB2. Refer to Chapter "Creating Installation Exits" of the manual *DB2 for VSE System Administration*, GC09-2406 for further information.

7.3 DEFINE

The XTST1050 program with the DEFINE function generates the data move program for the file *ddname* on the "*FILE=*" parameter. The assembler program generated is named as the *ddname* (be careful that there is no other program with that name).

When executing DEFINE on a file that contains different kinds of records (a MULT file with SUBS), a data move program is generated for each SUBS file. Therefore, do not execute DEFINE on a SUBS or NEXT file.

Similarly, when running DEFINE on a file to which an alternate file belongs, a program is generated for the ALT file. Therefore, do not execute DEFINE on an ALT file.

The same applies when running DEFINE on a file with a NEXT file; that is, a program is generated for the NEXT file. Therefore, do not execute DEFINE on a NEXT file.

Each punched file is assembled and link-edited. To make sure that this will be successful, verify the results of this job as well.

Punch the member XTSTDEF.A from the source library to obtain a JCL example for the DEFINE function. Figure 30 on page 86 shows a sample XTSTDEF JCL.

```
* $$ JOB JNM=XTSTDEF,CLASS=A,DISP=D
* $$ LST CLASS=Q,COPY=1
* $$ PUN CLASS=A,DISP=D,PRI=3
// JOB XTSTDEF
*
*                     SQL TRANSPARENCY
*
*   DEFINE ENVIRONMENT FOR VSAM FILE FROM CICS DESCRIPTION
*
*   REPLACE PRD2.DB2VSAM BY YOUR SQL-TRANSPARENCY LIBRARY
*
// OPTION NODUMP
// LIBDEF *,SEARCH=(PRD2.DB2410,PRD2.DB2VSAM)
// EXEC XTST1050,SIZE=AUTO,PARM=¢DBNAME(SQLVSE01)¢
DEFINE  FILE=SQTVVW2
/*
/&
* $$ EOJ
```

Figure 30. Sample XTSTDEF JCL

7.4 Acquire Dbspace

Acquire the dbspace for the VSAM file to be migrated. You can use either ISQL or DBSU to define the dbspace. You should read the System Administration Manual for more information about acquiring dbspaces.

Figure 31 shows a sample acquire dbspace JCL.

```
* $$ JOB JNM=XTSTDBA,DISP=D,CLASS=A,NTFY=YES,USER=VJAY
* $$ LST CLASS=Q,COPY=1
// JOB XTSTDBA
* *-----------------------------------------------------* *
* |   ACQUIRE DBSPACE FOR TABLE USING DBSU               | *
* *-----------------------------------------------------* *
// LIBDEF *,SEARCH=PRD2.DB2410
// EXEC ARIDBS,SIZE=AUTO
 CONNECT SQLDBA IDENTIFIED BY SQLDBAPW;
 SET ERRORMODE CONTINUE;
 ACQUIRE PUBLIC DBSPACE
 NAMED DBS_SQTVVW2
 (PAGES=40960,PCTFREE=25,STORPOOL=2);
/*
/&
* $$ EOJ
```

Figure 31. Sample Acquire Dbspace JCL

7.5 MIGRATE

Now you can use either different steps:

- CREATE for table creation, refer to 7.6, “CREATE” on page 88.
- LOAD for data loading, refer to 7.7, “LOAD” on page 89.
- GENERATE for access programs generation for the base cluster and the dependent files, refer to 7.8, “GENERATE” on page 90.

or perform all three steps in one, called MIGRATE.

Punch member XTSTMIGR.A from the source library to obtain a JCL example for the MIGRATE function. Then specify:

- the *ddname* in the “*FILE=*” parameter
- the dbspace in which the table is to be created in the “ *DBSPACE=*”parameter.

Figure 32 on page 88 shows a sample XTSTMIGR JCL.

The names of the access programs have eight characters:

- *ddname* with $ as the trailing character (for example, SQTVVW2$) for batch processing
- *ddname* with @ as the trailing character (for example, SQTVVW2@) for online processing. @ is equivalent to x′7c′. Depending on your code page, it can be a different character.

Each punched file is assembled and link-edited. Each punched file containing the SQL statements is assembled and link-edited. To successfully execute these steps, verify the results of these jobs as well.

Hints: If you run MIGRATE (or CREATE) a second time, the table is deleted first (within the supplied JCL). For performance reasons, the deletion of a dbspace is recommended over the deletion of a table; this requires that you again execute the acquire dbspace job. **You can delete the dbspace only if you follow the recommendation to have just one table in a dbspace.** For more details, see ″SQL/DS Version 3 Release 4 Performance Guide″, GG24-4047.

If the MIGRATE results in a program check interruption, verify that the dbspace is large enough for the amount of data to be loaded.

```
* $$ JOB JNM=XTSTMIGR,CLASS=A,DISP=D
* $$ LST CLASS=Q,COPY=1
* $$ PUN CLASS=A,DISP=D,PRI=3
// JOB XTSTMIGR
*
*                    SQL TRANSPARENCY
*
*  MIGRATE VSAM FILE INTO SQL TABLE (STRUCTURE AND DATA)
*
*
*
// LIBDEF *,SEARCH=(PRD2.DB2410,PRD2.DB2VSAM)
* INCLUDE THESE STATEMENTS IF YOU ARE RUNNING MIGRATE FOR THE 2ND TIME
*
// EXEC ARIDBS,SIZE=AUTO
CONNECT SQLDBA IDENTIFIED BY SQLDBAPW;
DROP TABLE SQLDBA.SQTT1W2;
/*
// DLBL SQTVVW2,¢WORKSHOP.TWO.FILE¢,,VSAM,CAT=VSESPUC
// EXEC XTST1050,SIZE=AUTO,PARM=¢DBNAME(SQLVSE01)¢
MIGRATE  FILE=SQTVVW2 DBSPACE=DBS_SQTVVW2
/*
/&
* $$ EOJ
```

Figure 32. Sample XTSTMIGR JCL

7.6 CREATE

After you have run the XTST transaction to define the characteristics of the VSAM file and DB2 table(s), and after running the DEFINE function of XTST1050, the function CREATE creates the table in the database. Additionally, if the view name given is not identical to the table name, the view is created in DB2. For more information on how to use views, refer to Chapter 8, “Beyond Transparency” on page 103.

The CREATE function of XTST1050 should only be used if the MIGRATE function was not used because CREATE is implicitly processed during MIGRATE.

Punch member XTSTCREA.A from the source library to obtain a JCL example for the CREATE function and specify:

- the *ddname* in the “*FILE=*” parameter
- the DBSPACE in which the table is to be created in the “*DBSPACE=*”parameter.

Figure 33 on page 89 shows a sample XTSTCREA JCL.

```
* $$ JOB JNM=XTSTCREA,CLASS=A,DISP=D
* $$ LST CLASS=Q,COPY=1
* $$ PUN CLASS=A,DISP=D,PRI=3
// JOB XTSTCREA
*
*                      SQL TRANSPARENCY
*
*    FUNCTION CREATE TO CREATE SQL TABLE FOR SPECIFIED FILE
*
*
*   REPLACE PRD2.DB2VSAM BY YOUR SQL-TRANSPARENCY LIBRARY
*
// OPTION NODUMP
// LIBDEF *,SEARCH=(PRD2.DB2410,PRD2.DB2VSAM)
// EXEC XTST1050,SIZE=AUTO,PARM=¢DBNAME(SQLVSE01)¢
CREATE   FILE=SQTVVW2 DBSPACE=DBS_SQTVVW2
/*
/&
* $$ EOJ
```

Figure 33. Sample XTSTCREA JCL

Hint: If you run CREATE (or MIGRATE) a second time, the table must be dropped first. This is contained within the MIGRATE, but not within the CREATE job. For performance reasons, the deletion of a dbspace is recommended over the deletion of a table; this requires that you again execute the acquire dbspace job. **You can delete the dbspace only if you do not share a table between more than one files via views, and if you follow the recommendation to have just one table in a dbspace.** For more details, see ″SQL/DS Version 3 Release 4 Performance Guide″, GG24-4047.

7.7 LOAD

Before executing DB2 VSAM Transparency between the VSAM file and the DB2 table, it is necessary to migrate the contents of the VSAM file into the DB2 table. The LOAD function executes this migration.

The LOAD function of XTST1050 should only be used if the MIGRATE function was not used because LOAD is implicitly processed during MIGRATE.

Punch member XTSTLOAD.A from the source library to obtain a JCL example for the LOAD function and specify the *ddname* in the “*FILE=*” parameter.

Figure 34 on page 90 shows a sample XTSTLOAD JCL.

```
* $$ JOB JNM=XTSTLOAD,CLASS=A,DISP=D
* $$ LST CLASS=Q,COPY=1
* $$ PUN CLASS=A,DISP=D,PRI=3
// JOB XTSTLOAD
*
*                    SQL TRANSPARENCY
*
*   FUNCTION LOAD TO TRANFER VSAM RECORD INTO SQL TABLE
*
*
*  REPLACE PRD2.DB2VSAM BY YOUR SQL-TRANSPARENCY LIBRARY
*
// OPTION NODUMP
// LIBDEF *,SEARCH=(PRD2.DB2410,PRD2.DB2VSAM)
// DLBL SQTVVW2,¢WORKSHOP.TWO.FILE¢,,VSAM,CAT=VSESPUC
// EXEC XTST1050,SIZE=AUTO,PARM=¢DBNAME(SQLVSE01)¢
LOAD    FILE=SQTVVW2
/*
/&
* $$ EOJ
```

Figure 34. Sample XTSTLOAD JCL

Hints: The DLBL for the file must be specified.

If the LOAD results in a program check interruption, verify that the dbspace is large enough for the amount of data to be loaded.

7.8 GENERATE

The GENERATE function punches, assembles and link-edits the batch and online access programs for the base cluster and for the dependent files (ALT, NEXT, SUB). For ALT files, no DB2 table is created but an index. No data has to be loaded.

The GENERATE function of XTST1050 should only be used if the MIGRATE function was not used on this file because GENERATE is implicitly processed during MIGRATE.

Do not execute GENERATE on a SUBS or NEXT file. Likewise, do not execute GENERATE on an ALT file except if the ALT file has been defined after the base cluster has been generated with MIGRATE or GENERATE.

Punch the member XTSTGEN.A from the source library to obtain a JCL example for the GENERATE function and specify the *ddname* of the dependent (ALT, SUBS or NEXT) file in the "*FILE=*" parameter.

To make sure that the assemble and link-edit will be successful, verify the results of these jobs as well. Figure 35 on page 91 shows a sample XTSTGEN JCL.

```
* $$ JOB JNM=XTSTGEN,CLASS=A,DISP=D
* $$ LST CLASS=Q,COPY=1
* $$ PUN CLASS=A,DISP=D,PRI=3
// JOB XTSTGEN
*
*
*                       SQL TRANSPARENCY
*
*  GENERATE PROGRAM TO PREPROCESS AND CREATE INDEX ON SQL TABLE
*  (ALTERNATE FILES MUST BE GENERATED)
*
*  REPLACE PRD2.DB2VSAM BY YOUR SQL-TRANSPARENCY LIBRARY
*
*
// LIBDEF *,SEARCH=(PRD2.DB2410,PRD2.DB2VSAM)
// DLBL SQTV2W2,¢WORKSHOP.TWO.FILE¢,,VSAM,CAT=VSESPUC
// EXEC XTST1050,SIZE=AUTO,PARM=¢DBNAME(SQLVSE01)¢
GENERATE FILE=SQTV2W2
/*
/&
* $$ EOJ
```

Figure 35. Sample XTSTGEN JCL

7.9 Define the List of VSAM Files to Process

DB2 VSAM Transparency lets you migrate your VSAM files to DB2 tables file by file. You can mix VSAM accesses and Transparency accesses in the same program. You have to specify which files have been migrated and need to be processed by Transparency.

Punch member XTSTPRM.A from the source library to obtain the JCL example to assemble this table. Specify the *ddname, dataset name and catalog name* for all files, this means for base clusters as well as for ALT, SUBS and NEXT file, in the “*TYPE=DATASET*” parameter.

The program to assemble and link-edit must contain

```
XTST TYPE=INITIAL
```

on the first line, and

```
XTST TYPE=FINAL
```

on the last line. In between, specify the list of files to be processed. Figure 36 on page 92 shows an example of the XTSTPRM JCL.

```
* $$ JOB JNM=XTSTPRM,CLASS=0,DISP=D
* $$ LST CLASS=A
*
// JOB XTSTPRM
*                    SQL TRANSPARENCY
*
*  SPECIFY ALL FILES TO BE PROCESSED BY SQL-TRANSPARENCY
*
*  REPLACE XXX.YYYY BY YOUR SQL-TRANSPARENCY LIBRARY
*
// OPTION CATAL
// LIBDEF *,SEARCH=PRD2.DB2VSAM
// LIBDEF PHASE,CATALOG=PRD2.DB2VSAM
// EXEC ASSEMBLY
         XTST      TYPE=INITIAL
*
VSESPUC  XTST      TYPE=CATALOG,DSN=¢VSESP.USER.CATALOG¢
IJSYSCT  XTST      TYPE=CATALOG,DSN=¢VSAM.MASTER.CATALOG¢
*
CLIENT   XTST      TYPE=DATASET,CAT=VSESPUC,DSN=¢CLIENT.TEST¢
INVOICE  XTST      TYPE=DATASET,DSN=¢INVOICE.TEST¢
EMPLOYE  XTST      TYPE=DATASET,CAT=IJSYSCT,DSN=¢EMPLOYE¢
SQTVVW1  XTST      TYPE=DATASET,CAT=VSESPUC,DSN=¢WORKSHOP.ONE.FILE¢
SQTVVW2  XTST      TYPE=DATASET,CAT=VSESPUC,DSN=¢WORKSHOP.TWO.FILE¢
SQTVVY   XTST      TYPE=DATASET,CAT=VSESPUC,DSN=¢WORKSHOP.SQTVVY.FILE¢
SQTVVY2  XTST      TYPE=DATASET,CAT=VSESPUC,DSN=¢WORKSHOP.SQTVVY2.FILE¢
*
         XTST      TYPE=FINAL
         END
/*
// EXEC LNKEDT
/*
/&
* $$ EOJ
```

Figure 36. Sample XTSTPRM JCL

Hints: This phase must be loaded in the SVA. Each time the XTSTPRM job is run to specify the list of VSAM data sets to be processed by DB2 VSAM Transparency for VSE/ESA, the XTSTSDL2 job must be run to reload it into the SVA.

Do not forget that also ALT files must be specified in the XTSTPRM table. NEXT and SUBS files need no entry.

To enable the new version of the XTSTPRM phase, stop Transparency both for online (XTOF) and batch (XTSTSTOP). Set a new version of the program under CICS with ″CEMT SET PROG(XTSTPRM) NEW″ and execute the XTON transaction.

7.10 INIT

For ESDS and RRDS files, when a new record is inserted into the VSAM file, a new RBA or RRN is attributed to the record. DB2 VSAM Transparency computes the RBA or RRN from the DB2 table. However to optimize this computing, a table initialization is required using the INIT function.

The INIT function is processed to compute the maximum RBA and RRN for ESDS and RRDS files from the corresponding DB2 table. The computed values are stored into the XTSTPRM phase. Therefore, execute INIT after loading XTSTPRM into the SVA.

No parameters are required.

Punch member XTSTINIT.A from the source library to obtain a JCL example for the INIT function.

Figure 37 shows a sample XTSTINIT JCL.

```
* $$ JOB JNM=XTSTINIT,CLASS=A,DISP=D
* $$ LST CLASS=Q,COPY=1
* $$ PUN CLASS=A,DISP=D,PRI=3
// JOB XTSTINIT
*
*                         SQL TRANSPARENCY
*
*   INITIALIZE PARAMETER TABLE TO ASSIGN RBA/RRN TO ESDS/RRDS FILES
*
*
*   REPLACE PRD2.DB2VSAM BY YOUR SQL-TRANSPARENCY LIBRARY
*
// LIBDEF *,SEARCH=(PRD2.DB2410,PRD2.DB2VSAM)
// EXEC XTST1050,SIZE=AUTO,PARM=¢DBNAME(SQLVSE01)¢
INIT
/*
/&
* $$ EOJ
```

Figure 37. Sample XTSTINIT JCL

7.11 Implementing Transparency for Batch Applications

7.11.1 Loading Programs into SVA

Before starting Transparency it is necessary to load phases into the Shared Virtual Area (SVA). For batch, five programs must be defined in the System Directory List (SDL). For online only, XTSTPRM is sufficient.

These five phases can be loaded into the SVA at IPL time. Except for XTSTPRM, they must not be loaded a second time; additional loading may partially remove the link to DB2 VSAM Transparency for VSE/ESA. The one phase, XTSTPRM must be reloaded after each modification. Repeated loading of XTSTPRM takes up additional space in the SVA. The space is freed at the next IPL.

Punch member XTSTSDL2.A from the source library to obtain the sample JCL to load XTSTPRM, and XTSTDSL.A to load the other four phases.

Figure 38 shows a sample XTSTSDL2 JCL.

```
* $$ JOB JNM=XTSTSDL2,CLASS=A,DISP=D
* $$ LST CLASS=Q,COPY=1
* $$ PUN CLASS=A,DISP=D,PRI=3
*
// JOB XTSTSDL2
*
*                    SQL TRANSPARENCY
*
*  UPDATE SDL FOT XTSTPRM PROGRAM (LIST OF FILES TO PROCESS)
*
*  REPLACE XXX.YYYY BY YOUR SQL-TRANSPARENCY LIBRARY
*
*  Note : This job must be executed in the BG partition
*
// LIBDEF *,SEARCH=PRD2.DB2VSAM
 SET SDL
XTSTPRM,SVA
/*
/&
* $$ EOJ
```

Figure 38. Sample XTSTSDL2 JCL for Batch and Online Transparency

Punch member XTSTSDL.A from the source library to obtain a JCL example for files SDL definition.

Figure 39 shows a sample XTSTSDL JCL.

```
* $$ JOB JNM=XTSTSDL,CLASS=A,DISP=D
* $$ LST CLASS=Q,COPY=1
* $$ PUN CLASS=A,DISP=D,PRI=3
*
// JOB XTSTSDL
*                    SQL TRANSPARENCY
*
*  SDL DEFINITION FOR SQL TRANSPARENCY
*
*
*  REPLACE PRD2.DB2VSAM BY YOUR SQL-TRANSPARENCY LIBRARY
*
// LIBDEF *,SEARCH=(USER.PROD,PRD2.DB2VSAM)
 SET SDL
$$BCXTST,MOVE
$$BXTST,MOVE
$$BXTMSG,MOVE
XTSTVTMS,SVA
/*
/&
* $$ EOJ
```

Figure 39. Sample XTSTSDL JCL for Batch Transparency

To permanently use Transparency, we modified the startup procedures to include loading the SVA phases within the JCL02 job. Figure 40 on page 95 shows the job to set the LIBDEF environment correctly, and Figure 41 on page 95 shows extracts of the JCL02 we used during startup.

Hints: We are using the job names LIBSDL2 rather than LIBSDL and JCL02 rather than JCL0 to keep the capability to boot the unchanged VSE in case of errors.

This job must execute in the BG partition.

```
* $$ JOB JNM=LIBSDL2,DISP=D,CLASS=0
// JOB LIBSDL2
// EXEC LIBR,PARM=¢MSHP¢
ACCESS S=IJSYSRS.SYSLIB
CATALOG LIBSDL2.PROC             DATA=YES REPLACE=YES
// LIBDEF PHASE,SEARCH=(PRD1.BASE,PRD2.DB2VSAM)
/+
/*
/&
* $$ EOJ
```

Figure 40. Sample LIBSDL2 Job Including Transparency LIBDEF for SVA Load

```
// EXEC PROC=LIBSDL2   PROVIDE CORRECT LIBDEF FOR SET SDL
SET SDL
LIST=$SVAVTAM
LIST=$SVACICS
LIST=$SVAREXX
LIST=$SVAASMA
$$BCXTST,MOVE
$$BXTST,MOVE
$$BXTMSG,MOVE
XTSTVTMS,SVA
XTSTPRM,SVA
/*
// LIBDROP PHASE
```

Figure 41. Extract from Sample Startup Job JCL02 Including SVA Load

7.11.2 Start Batch Transparency

To start batch Transparency processing, execute the XTSTSTRT batch program.

With a parameter, the user can specify to activate Transparency only in the current partition with

```
PARM=¢PART=*¢
```

where “*” means the current partition in which the job is running, or to activate Transparency in all partitions with

```
PARM=¢PART=ALL¢
```

Punch member XTSTSTRT.A from the source library to obtain a JCL example for the START function. Figure 42 on page 96 shows a sample XTSTSTRT JCL.

```
* $$ JOB JNM=XTSTSTRT,CLASS=A,DISP=D
* $$ LST CLASS=A,DISP=D,PRI=3
* $$ PUN CLASS=A,DISP=D,PRI=3
// JOB XTSTSTRT
*
*                    SQL TRANSPARENCY
*
*  START SQL TRANSPARENCY
*
*  REPLACE PRD2.DB2VSAM BY YOUR SQL-TRANSPARENCY LIBRARY
*
*
// LIBDEF *,SEARCH=PRD2.DB2VSAM
// EXEC XTSTSTRT,SIZE=AUTO,PARM=¢PART=ALL¢
/*
/&
* $$ EOJ
```

Figure 42. Sample XTSTSTRT JCL

7.11.3 Executing a Batch Program Using Transparency

To execute your batch program using Transparency:

- Modify your general environment JCL (or each application JCL) to add the library of DB2 VSAM Transparency in your LIBDEF statements.
- If ′PART=*′ was specified in XTSTSTRT, run your job in the same partition, where Transparency is active.
- Verify that the partition is large enough. Refer to program documentation or refer to the values we used, see Appendix A, “Environment Used” on page 107.
- Verify the results of your job.

7.11.4 Stop Batch Transparency

To stop the Transparency, the batch program XTSTSTOP must be executed.

Punch member XTSTSTOP.A from the source library to obtain a JCL example for the STOP DB2 VSAM Transparency.

Figure 43 on page 97 shows a sample XTSTSTOP JCL.

```
* $$ JOB JNM=XTSTSTOP,CLASS=A,DISP=D
* $$ LST CLASS=A,DISP=D,PRI=3
* $$ PUN CLASS=A,DISP=D,PRI=3
// JOB XTSTSTOP
*
*                      SQL TRANSPARENCY
*
*  STOP  SQL TRANSPARENCY
*
*
*  REPLACE PRD2.DB2VSAM BY YOUR SQL-TRANSPARENCY LIBRARY
*
*
// LIBDEF *,SEARCH=PRD2.DB2VSAM
// EXEC XTSTSTOP,SIZE=AUTO
/*
/&
* $$ EOJ
```

Figure 43. Sample XTSTSTOP JCL

7.12 Implementing Transparency for CICS Applications

7.12.1 Modify PPT Table

To prepare Transparency online processing, phases generated during the DEFINE and MIGRATE or GENERATE functions will be loaded. They must be defined in the PPT table with two entries:

1. One entry is for the phase generated during the DEFINE function. This means, that also for each ALT, SUBS and NEXT file such an entry is required.
2. The other entry is for the DB2 program generated during the GENERATE or MIGRATE function. This means, that for each ALT, SUBS and NEXT file such an entry is required.

This definition can be done using the CEDA transaction or by editing and re-compiling the CICS PPT table. Figure 44 shows the PPT entries for the list of files shown in Figure 23 on page 78.

```
DFHPPT TYPE=ENTRY,PROGRAM=SQTVWW1,PGMLANG=ASSEMBLER,RSL=PUBLIC
DFHPPT TYPE=ENTRY,PROGRAM=SQTVWW1@,PGMLANG=ASSEMBLER,RSL=PUBLIC
DFHPPT TYPE=ENTRY,PROGRAM=SQTVWW2,PGMLANG=ASSEMBLER,RSL=PUBLIC
DFHPPT TYPE=ENTRY,PROGRAM=SQTVWW2@,PGMLANG=ASSEMBLER,RSL=PUBLIC
DFHPPT TYPE=ENTRY,PROGRAM=SQTV2W2,PGMLANG=ASSEMBLER,RSL=PUBLIC
DFHPPT TYPE=ENTRY,PROGRAM=SQTV2W2@,PGMLANG=ASSEMBLER,RSL=PUBLIC
DFHPPT TYPE=ENTRY,PROGRAM=SQTV3W2,PGMLANG=ASSEMBLER,RSL=PUBLIC
DFHPPT TYPE=ENTRY,PROGRAM=SQTV3W2@,PGMLANG=ASSEMBLER,RSL=PUBLIC
DFHPPT TYPE=ENTRY,PROGRAM=SQTV4W2,PGMLANG=ASSEMBLER,RSL=PUBLIC
DFHPPT TYPE=ENTRY,PROGRAM=SQTV4W2@,PGMLANG=ASSEMBLER,RSL=PUBLIC
```

Figure 44. Sample DFHPPT Changes

7.12.2 Loading Program into SVA

Before starting the online Transparency, it is necessary to load the latest XTSTPRM phase into the SVA. Since this is one of the five phases needed for batch Transparency, you don't need to reload the phase into the SVA if you already have loaded it for batch Transparency.

Punch member XTSTSDL2.A from the source library to obtain a JCL example to load XTSTPRM into the SVA.

Figure 45 shows a sample XTSTSDL2 JCL.

Note: This job must be executed in the BG partition.

```
* $$ JOB JNM=XTSTSDL2,CLASS=A,DISP=D
* $$ LST CLASS=Q,COPY=1
* $$ PUN CLASS=A,DISP=D,PRI=3
*
// JOB XTSTSDL2
*
*                    SQL TRANSPARENCY
*
*  UPDATE SDL FOT XTSTPRM PROGRAM (LIST OF FILES TO PROCESS)
*
*  REPLACE XXX.YYYY BY YOUR SQL-TRANSPARENCY LIBRARY
*
*  Note : This job must be executed in the BG partition
*
// LIBDEF *,SEARCH=PRD2.DB2VSAM
 SET SDL
XTSTPRM,SVA
/*
/&
* $$ EOJ
```

Figure 45. Sample XTSTSDL2 JCL for Online Transparency Only

7.12.3 Start Transparency for CICS Processing

Use the XTON transaction to activate Transparency.

To start Transparency processing regularly together with CICS, just add the program *XTSTON* into the DFHPLTPI table as follows.

```
DFHPLT TYPE=ENTRY,PROGRAM=XTSTON
```

A sample part of the DFHPLTPI table with this statement is shown in Figure 46.

```
*---------------------------------------------------------------------*
*          LOCAL ENTRIES SHOULD BE MADE AFTER THIS POINT             *
*---------------------------------------------------------------------*
        SPACE 3
        DFHPLT TYPE=ENTRY,PROGRAM=XTSTON
        DFHPLT TYPE=FINAL
```

Figure 46. Sample DFHPLTPI Entry

7.12.4 Execute a Transaction using Transparency

To verify if the online Transparency is successful:

- Close the VSAM file (using CEMT transaction).
- Execute the transaction accessing the closed file.
- Verify the correct execution of the transaction.

7.12.5 Stopping Transparency for CICS Processing

To stop Transparency under CICS, execute the XTOF transaction.

7.12.6 Disable Transparency for Online Processing

It might sometimes be necessary to close a file for CICS to avoid concurrent updates from batch and CICS.

DB2 VSAM Transparency provides a CICS transaction to close migrated files for CICS. This transaction only opens or closes files defined for Transparency in the XTSTPRM phase.

To display the list of migrated files defined in Transparency online, execute the XEMT transaction. There you can disable and enable migrated files, similar to the CEMT transaction for real, existing VSAM files.

7.13 Summary of DB2 VSAM Transparency Step by Step

The following steps are required:

1. Define the VSAM file characteristics - such as file type (KSDS, ESDS, RRDS , MULT, SUBS or ALT), record length, key length and key position - using the XTST online tool.
2. Use the XTST online tool to define each field in the VSAM record and the corresponding DB2 column.
3. Execute the batch DEFINE function. The information specified during the online definition is used to create the data move program which has the same name as the VSAM file name specified on the DLBL job control statement.
4. If not yet done, acquire the dbspace for the VSAM file to be migrated.
5. Execute the batch MIGRATE function which creates the DB2 table and the DB2 view and loads the data into the table. If this was successful, the batch access and online access programs for the base cluster and the dependent files (ALT, SUBS and NEXT) are created, assembled and link-edited. The batch access program becomes the name "*ddname$$*" (where *ddname* is the base or dependent VSAM file name, followed by "*$*" to a length of 8) and the online program becomes the name "*ddname@@*" (where *ddname* is the base or dependent VSAM file name, followed by "*@*" to a length of 8).

   Alternatively, you can use the CREATE, LOAD and GENERATE functions to do the same.

   In case of ALT, the GENERATE function will also create the alternate index in DB2.
6. Modify the list of files migrated in XTSTPRM; add an entry for each new VSAM file and for each ALT. Not required for SUBS and NEXT files.

Submit XTSTPRM for assembly and link-edit.

7. If Transparency is active, perform XTSTSTOP (batch) and XTOF (online).

8. Execute XTSTSDL2 to load the XTSTPRM SVA phase into the Shared Virtual Area (SVA). If the other four phases have not yet been loaded and batch Transparency is needed additionally, use XTSTSDL to load the four phases. If only online Transparency is needed, the phase XTSTPRM is enough.

   Execute the INIT function to initialize parameters for ESDS and RRDS files within XTSTPRM in the SVA.

9. Activate batch Transparency by executing the XTSTSTRT job, specifying whether it will be active for all partitions or just for any single one.

10. For the CICS system define two entries in the Processing Program Table (PPT) for each VSAM file - one for the phase created during the DEFINE function (*ddname*) and one for the phase created during the MIGRATE or GENERATE function ("*ddname@@*"), and additionally one for each of the online access phases for the ALT, SUBS and NEXT. Assemble the PPT.
    If the CICS partition is running shut it down.
    Start the CICS partition.

    Alternatively, use RDO to update the CICS definitions.

    If not programmed to be performed during CICS startup, enable the link to the DB2 database (CIRB).

11. Activate Transparency in the CICS partition by executing the transaction XTON.

These steps are summarized in Figure 47 on page 101.

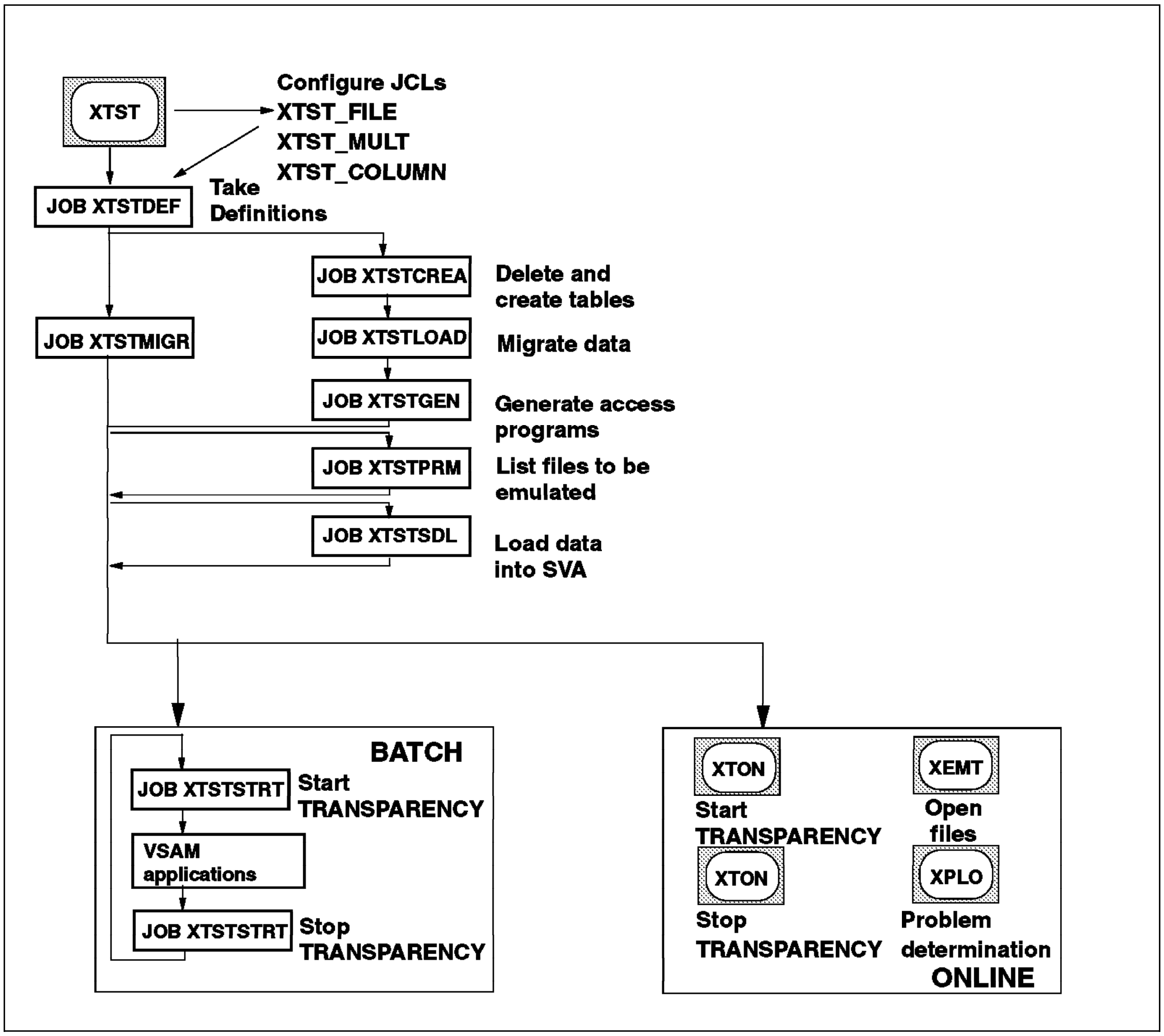


Figure 47. Steps for Using DB2 VSAM Transparency

Chapter 8. Beyond Transparency

This chapter explains the next logical steps of a conversion after Transparency has been set up. Additionally, it describes how to expand the scope of its usage by circumventing some of its limitations.

Generally, DB2 VSAM Transparency offers a way to quickly migrate data from VSAM to DB2. After establishing Transparency, a test should be made. Even if other conversion changes follow, a thorough test should be made here anyway. Since a lot of the test work is preparing the test environment, the duplicate testing does not represent a duplicate amount of effort.

Further Conversion Changes

Establishing Transparency and testing should by far not be considered the only effort; application changes and redesigns will have to follow to take advantage of DB2. Remember, that in 2.8, "Conversion With DB2 VSAM Transparency for VSE/ESA" on page 23 it was mentioned as a disadvantage of DB2 VSAM Transparency that in the application there will be much unnecessary code, which might lead to performance problems. Additionally, as listed in 7.1.3, "Limitations" on page 76 there are limitations in DB2 VSAM Transparency that might prevent you from achieving your ideal database design using DB2 VSAM Transparency. Therefore, you might begin to modify the application programs to directly access DB2 data using SQL.

Hint

In any single application program you can mix:

- VSAM accesses (to files not enabled for Transparency)
- Intercepted VSAM accesses (to files enabled for Transparency)
- SQL accesses to DB2

These different access modes do not interfere with each other. This means, after migration using Transparency you can change an application even partially, step by step.

Such changes can be within the same conversion project rather than during a follow-on project to avoid duplication of the test effort.

An example is to introduce foreign key relationships. This requires that the sequence of your applications respects that relationship. This can mean changes to your applications, but does not disturb running the Transparency.

Normally you should define your database structure according to your requirements and not retreat from them, as described in 2.8, "Conversion With DB2 VSAM Transparency for VSE/ESA" on page 23. For the exceptional case that you still have to slightly modify the data structure, now is the time to do that. But this only makes sense if very few applications are affected.

Circumvent Limitations

In many cases, you will be able to circumvent the limitations of DB2 VSAM Transparency for VSE/ESA so that you can implement your database design or come very close to it.

DB2 VSAM Transparency can use views, and the underlying data structure can be different.

View Handling

If you have a simple structure, for example you want to join the VSAM files FILE1 and FILE2 into table TABLEA, with the views VIEW1 and VIEW2, respectively, you can perform the following steps:

1. Define FILE1 with table TABLEA and view VIEW1 using the XTST transaction.
2. Define FILE2 with table TABLEA and view VIEW2 using the XTST transaction.
3. Defining the columns, add a dummy SQL type column with different default values from FILE1 and from FILE2, to be able to differentiate the rows via views. For example, add an “SM” (small integer) column “DUMMY” with the default values “1” from FILE1, and “2” from FILE2.
4. Run the DEFINE jobs on both VSAM files.
5. Run MIGRATE on FILE1. This creates and loads TABLEA. Additionally, this creates VIEW1, but this is not yet usable.
6. Run LOAD on FILE2.
7. Run GENERATE on FILE2.
8. Manually update VIEW1 to differentiate based upon the dummy column. In ISQL or in a DBSU job stream, enter

   ```
   DROP VIEW VIEW1
   CREATE VIEW VIEW1 AS SELECT col1,col2,... FROM TABLEA WHERE DUMMY=1
   ```

   with col1,col2,... listing all columns except the dummy column.
9. Manually create VIEW2 differentiating based upon the dummy column.

   ```
   CREATE VIEW VIEW2 AS SELECT col1,col2,... FROM TABLEA WHERE DUMMY=2
   ```

   with col1,col2,... listing all columns except the dummy column.
10. Perform the other steps described in 7.13, “Summary of DB2 VSAM Transparency Step by Step” on page 99 to enable Transparency and test it.

View Limitations

But there are limitations when applying the functions INSERT, UPDATE and DELETE on views:

- The view must span only one table for UPDATE and INSERT.
- The columns omitted from the view must allow nulls for INSERT.
- The column to be updated must not be using a column function for UPDATE and INSERT.
- No scalar functions must be used for UPDATE and INSERT.
- The column updated must not be a constant or an expression for UPDATE and INSERT.

- Two view columns, derived from the same table columns, cannot be updated within the same UPDATE or INSERT statement.
- For DELETE of a line with a primary key, having a foreign key related to it, the behavior depends on the setup of the referential constraint (refer to the DB2 manuals):
  - Either the deletion is prevented.
  - Or the foreign key in the dependent table is set to NULL.
  - Or the row with the foreign key in the dependent table is deleted.

Circumvention Example

A customer has - historically grown - many different VSAM files with different column formats for principally similar kinds of data. Now he wants to redesign the database and convert to DB2 Server for VSE & VM. Additionally, he wants to design completely new applications, based on tasks to be fulfilled rather than on data to be accessed.

A major problem lies in the fact that much of the data is kept more than once, and that the data cannot generally be relied upon. The limitations of DB2 VSAM Transparency are even welcome because otherwise one tended to believe that repairing the data could be automated, which actually cannot be done. Therefore, the following steps are planned:

1. Get a thorough overview of the data structure and quality.
2. Design the final structure of the database according to the vision.
3. Design a structure of the data that can be realized using Transparency. This comprises many more tables of the same format to be merged later.
4. Design a structure of the data that can be achieved using Transparency and the circumvention steps below to come much closer to the vision.
5. Where the effort is justified, change the VSAM applications to remove problems that might cause more costs if solved or circumvented later.
6. Establish Transparency and define as many tables as you want from each VSAM file. This leads to actually many more tables than wanted, because you cannot join tables from other VSAM files. But prepare their later merge by:
   - Giving the tables of the different VSAM files the same structure.
   - Adding dummy DB2 columns to the tables to allow you to distinguish them in a view.
7. With these table definitions, run the DEFINE and MIGRATE jobs.
8. Test that Transparency works.
9. Merge the different tables of the same format, sourcing from different VSAM files, into a single table and clean the data reducing duplications. This step might need manual intervention and semi-automated steps.
10. Create different views of the table, simulating the original tables. Each view must comprise only one table and fulfill the limitations listed above to enable updates. The view has the purpose to distinguish which rows are presented to which application; using the contents of the dummy column this can be achieved easily.

11. Perform the online definition step again for the VSAM files and the merged DB2 tables including the dummy columns, but this time use the name of the views.
12. Run the DEFINE and GENERATE steps again to create the batch and online access programs, so that Transparency will access the views. Do not acquire the dbspace or perform any of the steps CREATE, MIGRATE or LOAD.
13. Do a thorough test.
14. If this data structure is not yet close enough to the envisioned one, establish the envisioned data structure by creating a new set of views.
15. Begin to design the new applications using the new set of views.
16. Once the new applications run successfully and the old applications can be dropped, convert the data from the current tables into tables of the format of the new set of views.

A pilot has to be made to ensure that the above steps can be made in the individual customer environment, and to adapt the plan where necessary. After the pilot has been successful, the actual implementation begins.

Testing

Having run Transparency, a thorough test must follow according to the rules in Chapter 4, “Testing” on page 49. Depending on the amount of manual changes you made after establishing Transparency, the effort might be reduced slightly.

Future Use of DB2 VSAM Transparency for VSE/ESA

DB2 VSAM Transparency can be an excellent help for a migration. Beyond this, **you can run DB2 VSAM Transparency and continue to use your existing applications for years** - until you decide *for business reasons* to replace these applications.

Therefore eventually, be it sooner or later, DB2 VSAM Transparency will no longer be used - either when new applications have been created to replace the old ones, or when all applications have been changed from accessing VSAM to directly access DB2 via SQL. Any Transparency will ultimately be discarded - either by obsoletion or by migration.

Functional Changes

Other changes, such as adaptation to the year 2000, extension of the database, joining several tables of the database, or extensions of applications, have to be implemented during follow-on projects and have to be strictly separated from the conversion project. The reason is that testing is a major part of the conversion effort; and testing is much more efficient when you can expect the same results in both VSAM and DB2 environments and have them compared automatically.

Although such changes often are the real reason for the conversion, to be patient here and do the steps one after the other will lead you to the intended goal more quickly than when complicating the processes by intermixing steps.

Therefore, now that the conversion project is considered to be complete, the time has come to implement these functional changes. Doing so, you will experience the strengths of the relational database DB2 Server for VSE & VM.

Appendix A. Environment Used

This appendix describes the environment used during the sample conversion project. This was the base for the performance statistics in Appendix B, “Performance Statistics” on page 109.

Operating system

- VM/ESA Version 1 Release 2 Modification 2 (1.2.2)
- VSE/ESA Version 2 Release 1 Modification 2 (2.1.2)
- The size of the BG partition we used to run our batch programs was: V-SIZE=3072K and GETVIS=2048K.

VSE is running on VM.

Database Server

- SQL/DS Version 3 Release 5 Modification 0 (3.5.0)
- Preliminary version of DB2 VSAM Transparency (producing COBOL access programs)
- Preliminary version of DB2 VSAM Transparency (producing Assembler access programs)
- The database name in VSE/ESA is “SQLVSE01” in partition F8
- The database name in VM/ESA is “S35VMDB1”

Database Layout

Figure 48 on page 108 shows the layout of the database in VSE/ESA.

```
// PROC CAT=¢VSESPUC¢
// DLBL IJSYSUC,¢VSESP.USER.CATALOG¢,,VSAM
* ********************************************************
* SQLVSE01: SQL/DS DATABASE IDENTIFICATION
*
* ********************************************************
// DLBL BDISK,¢SQLVSE01.BDISK.SQLDIR40¢,,VSAM,CAT=&CAT
// DLBL LOGDSK1,¢SQLVSE01.LOGDSK1.SQLLOG¢,,VSAM,CAT=&CAT
// DLBL DDSK1,¢SQLVSE01.DDSK1.POOL1¢,,VSAM,CAT=&CAT
// DLBL DDSK2,¢SQLVSE01.DDSK2.POOL1¢,,VSAM,CAT=&CAT
// DLBL DDSK3,¢SQLVSE01.DDSK3.POOL2¢,,VSAM,CAT=&CAT
// DLBL DDSK4,¢SQLVSE01.DDSK4.POOL2¢,,VSAM,CAT=&CAT
// DLBL DDSK5,¢SQLVSE01.DDSK5.POOL2¢,,VSAM,CAT=&CAT
// DLBL DDSK6,¢SQLVSE01.DDSK6.POOL3¢,,VSAM,CAT=&CAT
// DLBL DDSK7,¢SQLVSE01.DDSK7.POOL3¢,,VSAM,CAT=&CAT
// DLBL DDSK8,¢SQLVSE01.DDSK8.POOL3¢,,VSAM,CAT=&CAT
// DLBL DDSK9,¢SQLVSE01.DDSK9.POOL4¢,,VSAM,CAT=&CAT
// DLBL DDSK10,¢SQLVSE01.DDSK10.POOL4¢,,VSAM,CAT=&CAT
// DLBL DDSK11,¢SQLVSE01.DDSK11.POOL4¢,,VSAM,CAT=&CAT
// DLBL DDSK12,¢SQLVSE01.DDSK12.POOL5¢,,VSAM,CAT=&CAT
// DLBL DDSK13,¢SQLVSE01.DDSK13.POOL5¢,,VSAM,CAT=&CAT
// DLBL DDSK14,¢SQLVSE01.DDSK14.POOL5¢,,VSAM,CAT=&CAT
```

Figure 48. Database Layout

A.1 Installation

The installation examples were created on a different environment:

Operating system

- VM/ESA Version 2 Release 1 Modification 0 (2.1.0)
- VSE/ESA Version 2 Release 2 Modification 0 (2.2.0)
- The size of the BG partition we used to run our batch programs was: V-SIZE=3072K and GETVIS=2048K.

VSE is running on VM.

Database Server

- Beta level of DB2 Server for VSE & VM Version 5 Release 1
- Beta level of DB2 VSAM Transparency Version 5 Release 1
- The database name in VSE/ESA is “SQLVSE01” in partition F8

Appendix B. Performance Statistics

This appendix classifies the applications and data used and gives some indication about the performance experiences made during our sample conversion project.

Applications and Data Used

The data and applications used are from a VSE/ESA customer, a South African Technikon (comparable to a University). The VSAM data was in pretty good shape and nearly in a normalized form already. The applications were not changed, but continued to run as DOS PL/I batch applications and CSP CICS transactions.

New data and new applications were created for demonstration purposes in PL/I for VSE and COBOL for VSE using LE for VSE. These are not listed here.

Figure 49 shows the list of VSAM files and the corresponding tables. Note that the files "d" and "e" are represented in more than one table each. Legend:

- VSAM is the reference to the VSAM file mentioned in Figure 50 on page 110.
- TNAME is the name of the table in DB2.
- INAME is the name of the index.
- ROWCOUNT is the number of rows.
- AVGROWLEN is the length of the row.
- FREEPCT is the percentage of free space in the dbspace.
- NPAGES is the number of pages occupied by either data or index.
- NCOLS is the number of table columns.
- KEYLEN is the number of bytes of the primary key.
- IPCTFREE is the amount of free space in the index.

```
VSAM TNAME    INAME              ROWCOUNT AVGROWLEN FREEPCT      NPAGES  NCOLS KEYLEN IPCTFREE
---- -------- ------------------ -------- --------- ------- ----------- ------ ------ --------
 f   RITACD   PKEYCEZTQVG95NSC      31699       201      25        2882     25     28       10
 c   RITCLF   PKEYCEYNW1RHYFAG       2194        95      25         100      3     15       10
 a   RITCRS   PKEYCE16JL9D75UA        659       178      25          55     21     16       10
 b   RITEMP   PKEYCEYNND3C2XTA       3112       171      15         164     10      5       10
 e   RITMAS   PKEYCE2HBUSMGHF4       2933        29      25         419      2     13       10
 e   RITMASD  I_RIMKEYD              2932      1283      25        2932     26     13       10
 e   RITMASH  I_RIMKEYH                 1      1267      25           1     14     13       10
 e   RITSSUB  PKEYCE2HB2BQH2LM       2932        85      25         419      9     13       10
 g   RITSUB   PKEYCEYZQUBJSNCU       7125       104      25         492      8     20       10
 d   RITTRN   PKEYCE1X2GGQU3UM       9274       418      25        2319     35     18       10
 d   RITPSE   PKEYCE1X2L73NDED       9274       169      25        2319     11     18       10
 i   RXTEXR   PKEYCE1VTM437G4W      16670       378      25        2779     16     24       10
```

Figure 49. VSAM Files and DB2 Tables Used

Performance measured

Generally it is known from various customer conversions, that - besides the use of DB2 VSAM Transparency - as long as the applications have not been re-written to exploit the capabilities of DB2, the resource consumption can increase by the factor of 4 and 40 (300% to 3900%).

Figure 50 gives some information about the applications accessing the VSAM files, and the tests that have been performed during the residency creating this redbook. Figure 51 on page 111 shows similar information, but while using a different level of DB2 VSAM Transparency for VSE/ESA using COBOL access modules for Transparency. Each figure shows on the leftmost columns the application program name and the VSAM files used. The next columns show the times measured without and with Transparency active, and the difference among these in percent. The last two columns show the number of columns that have been read sequentially (from file A, B, C, or D), and the number of records accessed in each file (except for the first three lines, this means: from all files - except file D - this number of records has been selected via direct access). Therefore the number is to be multiplied by the number of files.

Notes: Please don't take these figures as representative performance numbers. From the numbers below you must not conclude that DB2 VSAM Transparency can perform miracles. As mentioned above, the customer data is pretty clean and already in a normalized state in VSAM, which most often will not be the case. You also see slight differences in the numbers and percentages, indicating that variations can be expected between releases of DB2 VSAM Transparency for VSE/ESA.

```
                        XTST Stop  XTST Start  Diffe- records records
Program   VSAM File      hh/mm/ss  hh/mm/ss    rence   read   selected
========  ==========    =========  ==========  ====== ======= ========
sr0053pj  a              00/00/13  00/00/13    00.00%   0626   008
sr0061pj  b              00/00/12  00/00/23    91.66%   3000   118
sr0080pj  c              00/00/14  00/00/18    28.57%   2161  1983
sr0302pj  d/e/f          00/01/07  00/01/42    52.23%   9275   019*2
sr0303pj  d/e            00/01/02  00/01/17    64.51%   9275   023*1
sr0318pj  d/e/f/g        00/01/07  00/01/19    17.91%   9275   019*3
sr0319pj  d/e            00/01/06  00/01/18    18.18%   9275   018*1
sr0348pj  d/e/b/c/h/i/j  00/01/27  00/01/35    09.19%   9275   122*6

  Note: i and j are VSAM only, not migrated.
        VSAM accesses are 122*2.
```

Figure 50. Application Performance With Access Modules in COBOL

```
                            XTST Stop  XTST Start  Differ records records
Program   VSAM File          hh/mm/ss  hh/mm/ss            read   selected
=======   =========         ========   =========   =====  ======  =======
sr0053pj  a                  00/00/12  00/00/12     0%      0626   008
sr0061pj  b                  00/00/12  00/00/20    66%      3000   118
sr0080pj  c                  00/00/14  00/00/17    21%      2161  1983
sr0302pj  d/e/f              00/01/07  00/01/45    56%      9275   019*2
sr0303pj  d/e                00/01/02  00/01/20    29%      9275   023*1
sr0318pj  d/e/f/g            00/01/07  00/01/27    29%      9275   019*3
sr0319pj  d/e                00/01/06  00/01/15    13%      9275   018*1
sr0348pj  d/e/b/c/h/i/j      00/01/26  00/01/27     1%      9275   122*6

  Note: i and j are VSAM only, not migrated.
        VSAM accesses are 122*2.
```

Figure 51. Application Performance With Access Modules in Assembler

Appendix C. Special Notices

This publication is intended to help system programmers, application programmers, database administrators, data processing managers and other people involved with a database conversion to get an overview of the conversion efforts and how to use DB2 VSAM Transparency for VSE/ESA.

The information in this publication is not intended as the specification of any programming interfaces that are provided by DB2 VSAM Transparency for VSE/ESA.

See the PUBLICATIONS section of the IBM Programming Announcement for DB2 Server for VSE & VM for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM′s product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM′s intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, NY 10594 USA.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The information about non-IBM (″vendor″) products in this manual has been supplied by the vendor and IBM assumes no responsibility for its accuracy or completeness. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer′s ability to evaluate and integrate them into the customer′s operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Any performance data contained in this document was determined in a controlled environment, and therefore, the results that may be obtained in other

Appendix D. Related Publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

D.1 International Technical Support Organization Publications

For information on ordering these ITSO publications see "How to Get ITSO Redbooks" on page 117.

D.2 Redbooks on CD-ROMs

Redbooks are also available on CD-ROMs. **Order a subscription** and receive updates 2-4 times a year at significant savings.

CD-ROM Title	**Subscription Number**	**Collection Kit Number**
System/390 Redbooks Collection	SBOF-7201	SK2T-2177
Networking and Systems Management Redbooks Collection	SBOF-7370	SK2T-6022
Transaction Processing and Data Management Redbook	SBOF-7240	SK2T-8038
AS/400 Redbooks Collection	SBOF-7270	SK2T-2849
RS/6000 Redbooks Collection (HTML, BkMgr)	SBOF-7230	SK2T-8040
RS/6000 Redbooks Collection (PostScript)	SBOF-7205	SK2T-8041
Application Development Redbooks Collection	SBOF-7290	SK2T-8037
Personal Systems Redbooks Collection	SBOF-7250	SK2T-8042

D.3 Other Publications

- *Planning for Conversion to the DB2 Family: Methodology and Practice*, GG24-4445
- *SQL/DS Version 3 Release 4 Performance Guide*, GG24-4047
- *Introduction to Database* by C. J. Date

How to Get ITSO Redbooks

This section explains how both customers and IBM employees can find out about ITSO redbooks, CD-ROMs, workshops, and residencies. A form for ordering books and CD-ROMs is also provided.

This information was current at the time of publication, but is continually subject to change. The latest information may be found at URL `http://www.redbooks.ibm.com`.

How IBM Employees Can Get ITSO Redbooks

Employees may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **PUBORDER** — to order hardcopies in United States
- **GOPHER link to the Internet** - type `GOPHER.WTSCPOK.ITSO.IBM.COM`
- **Tools disks**

  To get LIST3820s of redbooks, type one of the following commands:

  ```
  TOOLS SENDTO EHONE4 TOOLS2 REDPRINT GET SG24xxxx PACKAGE
  TOOLS SENDTO CANVM2 TOOLS REDPRINT GET SG24xxxx PACKAGE (Canadian users only)
  ```

  To get BookManager BOOKs of redbooks, type the following command:

  ```
  TOOLCAT REDBOOKS
  ```

  To get lists of redbooks:

  ```
  TOOLS SENDTO USDIST MKTTOOLS MKTTOOLS GET ITSOCAT TXT
  TOOLS SENDTO USDIST MKTTOOLS MKTTOOLS GET LISTSERV PACKAGE
  ```

  To register for information on workshops, residencies, and redbooks:

  ```
  TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ITSOREGI 1996
  ```

  For a list of product area specialists in the ITSO:

  ```
  TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ORGCARD PACKAGE
  ```

- **Redbooks Home Page on the World Wide Web**

  `http://w3.itso.ibm.com/redbooks`

- **IBM Direct Publications Catalog on the World Wide Web**

  `http://www.elink.ibmlink.ibm.com/pbl/pbl`

  IBM employees may obtain LIST3820s of redbooks from this page.

- **REDBOOKS category on INEWS**
- **Online** — send orders to: USIB6FPL at IBMMAIL or DKIBMBSH at IBMMAIL
- **Internet Listserver**

  With an Internet e-mail address, anyone can subscribe to an IBM Announcement Listserver. To initiate the service, send an e-mail note to `announce@webster.ibmlink.ibm.com` with the keyword `subscribe` in the body of the note (leave the subject line blank). A category form and detailed instructions will be sent to you.

How Customers Can Get ITSO Redbooks

Customers may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **Online Orders** (Do not send credit card information over the Internet) — send orders to:

	IBMMAIL	**Internet**
In United States:	usib6fpl at ibmmail	usib6fpl@ibmmail.com
In Canada:	caibmbkz at ibmmail	lmannix@vnet.ibm.com
Outside North America:	dkibmbsh at ibmmail	bookshop@dk.ibm.com

- **Telephone orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	(long distance charges apply)
(+45) 4810-1320 - Danish	(+45) 4810-1020 - German
(+45) 4810-1420 - Dutch	(+45) 4810-1620 - Italian
(+45) 4810-1540 - English	(+45) 4810-1270 - Norwegian
(+45) 4810-1670 - Finnish	(+45) 4810-1120 - Spanish
(+45) 4810-1220 - French	(+45) 4810-1170 - Swedish

- **Mail Orders** — send orders to:

IBM Publications	IBM Publications	IBM Direct Services
Publications Customer Support	144-4th Avenue, S.W.	Sortemosevej 21
P.O. Box 29570	Calgary, Alberta T2P 3N5	DK-3450 Allerød
Raleigh, NC 27626-0570	Canada	Denmark
USA		

- **Fax** — send orders to:

United States (toll free)	1-800-445-9269
Canada	1-403-267-4455
(+45) 48 14 2207 (long distance charge)	Outside North America

- **1-800-IBM-4FAX (United States)** or **(+1)001-408-256-5422 (Outside USA)** — ask for:

  Index # 4421 Abstracts of new redbooks
  Index # 4422 IBM redbooks
  Index # 4420 Redbooks for last six months

- **Direct Services** - send note to softwareshop@vnet.ibm.com

- **On the World Wide Web**

Redbooks Home Page	http://www.redbooks.ibm.com
IBM Direct Publications Catalog	http://www.elink.ibmlink.ibm.com/pbl/pbl

- **Internet Listserver**

  With an Internet e-mail address, anyone can subscribe to an IBM Announcement Listserver. To initiate the service, send an e-mail note to announce@webster.ibmlink.ibm.com with the keyword subscribe in the body of the note (leave the subject line blank).

IBM Redbook Order Form

Please send me the following:

Title	**Order Number**	**Quantity**

First name Last name

Company

Address

City Postal code Country

Telephone number Telefax number VAT number

- Invoice to customer number

- Credit card number

Credit card expiration date Card issued to Signature

We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

DO NOT SEND CREDIT CARD INFORMATION OVER THE INTERNET.

Glossary

ALF. Application Load File, contains the object code of all CSP applications.

assembler. The lowest level programming language. Also, the tool to convert programs written in that language, into object code.

CASCADE. An option in defining referential integrity in DB2.

CI. Control Interval in VSAM.

cluster. A VSAM file in VSE. In VSE, a dbextent is implemented by a VSAM cluster.

clustering index. The first index created for a table. The DB2 database manager uses it to determine the placement of subsequent rows.

COBOL. A high level programming language. On VSE/ESA, the old programs are DOS/VS COBOL and COBOL II, and the current program is “COBOL for VSE,” running with LE.

commit. (1) The operation that terminates a unit of work by releasing locks so that the database changes made by that unit of work can be perceived by other processes. (2) The process that allows data changes to be made permanent. When a commit occurs, other applications can reference the just-committed data.

coexistence. When for some time data exist in both forms VSAM and DB2, it can happen that any application has to access both. That can cause problems.

conversion. In this manual, the whole project in moving applications and data from VSAM to DB2 is names conversion, as opposed to migration, see there.

CSP. Cross Systems Product, an IBM programming language of the 4th generation. Follow-on is VisualAge Generator, creating COBOL source as output.

concurrency. The shared use of resources by multiple interactive users or application processes at the same time.

DBSS. Database Storage Subsystem, a part of DB2.

DBSU. Database Services Utility, a part of DB2.

DDL. Data definition language.

DDR. DASD Dump Restore, a VM tool for disk backup.

deadlock. An impasse that occurs when a process is waiting for a resource that is being held by another process that is waiting for a resource currently being held by the first process.

DELETE. A DB2 command to drop a row from a table.

dependent table. A DB2 table which contains the foreign key relating to a primary key in another table. See *parent table*.

DML. Data manipulation language.

ESDS. Entry-Sequenced Data Set, a VSAM file format. The application controls the access to the records.

foreign key. DB2 column containing the value of a primary key of a row in the same or another table for the purpose of identifying a relationship. See *parent table* and *dependent table*.

IBM. International Business Machines Corporation.

IDCAMS. A utility program, which is a part of VSAM in VSE.

INSERT. A DB2 command to add a row to a table.

ITSO. International Technical Support Organization.

JCL. Job Control Language, for batch processing in VSE.

KB. Kilobyte.

key. See *foreign key* and *primary key*, *parent table* and *dependent table*.

KSDS. Key-Sequenced Data Set, a VSAM file format with variable length records. VSAM is visibly keeping an index table. Addressing uses either a key or a CI-number or an RBA.

LE. Language Environment, a runtime library for various languages.

List Processor Facility. file list facility in CSP, working against the MSL.

locking. Mechanism used by the database manager to ensure the integrity of data. Locking prevents concurrent users from accessing inconsistent data.

log. A collection of records maintained by the DB2 database manager to describe events that occurred during the operation of the database. This information is used for recovery if a failure occurs while the database manager is executing.

migration. In this manual migration means the actual moving of data from VSAM to DB2, as opposed to conversion, see there.

MSL. Member Specification Library, contains CSP source code for a user.

MUM. A mode of operating the DB2 database manager, in which one or more users or application programs can access the database at the same time. Contrast with single user mode (SUM).

NULL. NULL is the contents of an "empty" column in DB2. NULLs can expressly be allowed or forbidden for each DB2 column. A NULL is not a zero ("0"), but, for example, will be ignored when building an average value.

parent table. The related DB2 table with the primary key column referenced by the foreign key column in a dependent table.

PCT. Processing Control Table, one of many CICS control tables. Assembling these tables is the older method compared to RDO.

PL/I. A high level programming language. On &vse, the old program is DOS PL/I, and the current program is "PL/I for VSE," running with LE.

pool. See storage pool.

PPT. Processing Program Table, one of many CICS control tables. Assembling these tables is the older method compared to RDO.

primary key. The DB2 columns which together serve as the unique identifier of a relational row. See *foreign key*, *parent table* and *dependent table*.

PSP. Preventive Service Planning information offered for IBM programs.

primary key. A sum of DB2 columns which together can identify a unique row.

prime index. Every VSAM Key-Sequenced Data Set (KSDS) must have a prime index on one or more contiguous fields. The values in these fields must be unique. They are the prime keys.

prime key. See prime index.

QMF. Query Management Facility, an IBM application program running on VSE/ESA (and other platforms) to query DB2 data.

RDO. Resource Definition Online, a newer method to control CICS than assembling control tables.

restore. Generic term for restoring data. This can be performed either through starting the database with the parameter STARTUP=R or F, or with a user restore (like DDR) and then starting the database with the parameter STARTUP=U, or with either of the DB2 commands DATALOAD, RELOAD, etc.

RBA. Relative Byte Address from beginning of file; count begins with zero (0). Addressing mechanism within VSE.

RDO. Resource Definition Online, a method to configure CICS.

retention period. Specifies in VSE, how long a file is to be kept. 0 days means, it is not kept.

rollback. The process of restoring data changed by SQL statements to the state at its last commit point. All locks are freed. Contrast with commit.

RRN. Relative Record Number, an addressing mechanism for RRDS and VRDS VSAM files.

RRDS. Relative Record Data Set, a VSAM file format with records of the same length. No index is maintained. RRN is used to address the records.

SDL. System Directory List, indicating what is to be loaded into the SVA.

SIT. Systems Intitialization Table, one of many CICS control tables. Assembling these tables is the older method compared to RDO.

SQLDA. SQL descriptor area, used with dynamic SQL. A set of variables that is used to provide description information in the execution of certain SQL statements. It may be used to describe columns, input variables, or output variables.

storage pool. A storage pool is composed of one or more dbextents, and defines the physical space for one or more dbspaces.

SUM. A mode of operation, in which the DB2 database manager and one application run in the same virtual machine. No other application programs or users can access the database at the same time. Contrast with multiple user mode (MUM).

SVA. Shared Virtual Area, an address range common to all address spaces in VSE.

update. Generic term. In DB2, this might include the commands UPDATE, INSERT and DELETE.

UPDATE. A DB2 command to modify existing data.

VRDS. Variable-length Relative record Data Set, an RRDS with variable record length, accessed via RRN.

VSAM. Virtual Storage Access Method, a file access method in VSE, VM and MVS.

VSE. Any version and release of VSE/ESA, supported by DB2.

VSE/ESA. Virtual Storage Extended/Enterprise Systems Architecture, a System/390 operating system.

List of Abbreviations

CEDA	Resource Definition Online Transaction
CEMT	Master Terminal Transaction
CICS	Customer Information Control System
COBOL	COmmon Business Oriented Language
CPU	Central Processing Unit
CSD	CICS System Definition
CSP	Cross-System Product
DASD	Direct Access Storage Device
DB2	DATABASE 2
DBSU	Data Base Services Utility
DDNAME	Data Definition Name
DLBL	Disk Label
DML	Data Manipulation Language
DOS	Disk Operating System
ESDS	Entry Sequenced Data Set
HTML	HyperText Markup Language
I/O	Input/output
IBM	International Business Machines
ICCF	Interactive Computing and Control Facility
IPL	Initial Program Load
ISO	International Organization for Standardization
ISQL	Interactive Structured Query Language
ITSO	International Technical Support Organization
JCL	Job Control Language
KSDS	Key Sequenced Data Set
LE	Linkage Editing
MVS	Multiple Virtual Storage
OS/2	Operating System/2
PCT	Partition Control Table
PL/I	Programming language 1
POWER	Priority Output Writers, Execution processor, input Reader s
PPT	Processing Program Table
RBA	Relative Block Address
RDO	Resource Definition On-line
RRDS	Relative Record Data Set
RRN	Relative-Record Number
SDL	System Directory List
SIT	System Initialization Table
SQL	Structured Query Language
SQL/DS	Structured Query Language/Data System
SVA	Shared Virtual Area
VM	Virtual Machine
VNET	Virtual NETwork
VSAM	Virtual Storage Access Method
VSE	Virtual Storage Extended
VSE/ESA	Virtual Storage Extended/Enterprise Systems Architecture

Index

ITSO Redbook Evaluation

How to Get to DB2 from VSE/VSAM Using DB2 VSAM Transparency for VSE/ESA
SG24-4931-00

Your feedback is very important to help us maintain the quality of ITSO redbooks. **Please complete this questionnaire and return it using one of the following methods:**

- Use the online evaluation form found at http://www.redbooks.com
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to redeval@vnet.ibm.com

Please rate your overall satisfaction with this book using the scale:
(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)

Overall Satisfaction ____________

Please answer the following questions:

Was this redbook published in time for your needs? Yes____ No____

If no, please explain:

____

____

____

____

What other redbooks would you like to see published?

____

____

____

Comments/Suggestions: **(THANK YOU FOR YOUR FEEDBACK!)**

____

____

____

____

____

Printed in U.S.A.

SG24-4931-00

Artwork Definitions

id	File	Page	References	
ITLOGO	4931SU	i		i
ITLOGOS	4931SU	i		

Table Definitions

id	File	Page	References	
R1	REDB$EVA	135		135, 135
R2	REDB$EVA	135		135

Figures

id	File	Page	References	
SWAMP	4931CH02	9	1	
POSIT	4931CH02	12	2	
PHASE	4931CH02	13	3	12
METHO	4931CH02	14	4	
COEX	4931CH02	19	5	19
MIGR	4931CH02	23	6	23
CONWT	4931CH02	24	7	
PERIO	4931CH03	34	8	
3TEST	4931CH04	53	9	52
TRP	4931CH05	60	10	24, 59, 76
0601	4931CH06	66	11	66
0602	4931CH06	66	12	
0603	4931CH06	67	13	
0607	4931CH06	67	14	
0693	4931CH06	68	15	
0694	4931CH06	69	16	
0695	4931CH06	70	17	
PCTMOD	4931CH06	70	18	
PPTMOD	4931CH06	71	19	
SITMOD	4931CH06	71	20	
GENPARM	4931CH06	72	21	71
CH714	4931CH07	78	22	77
CH722	4931CH07	78	23	75, 75, 78, 97
CH716	4931CH07	79	24	79

CH740	4931CH07	80	25	79
CH721	4931CH07	81	26	81
CH717	4931CH07	82	27	76, 81
CH718	4931CH07	83	28	83
CH719	4931CH07	84	29	84, 85
CH701	4931CH07	86	30	85
CH702	4931CH07	86	31	86
CH703	4931CH07	88	32	87
CH704	4931CH07	89	33	88
CH705	4931CH07	90	34	89
CH706	4931CH07	91	35	90
CH707	4931CH07	92	36	91
CH709	4931CH07	93	37	93
CH78A	4931CH07	94	38	94
CH708	4931CH07	94	39	94
LIBSDL2	4931CH07	95	40	95
JCL02	4931CH07	95	41	95
CH710	4931CH07	96	42	95
CH711	4931CH07	97	43	96
7PPT	4931CH07	97	44	97
CH712	4931CH07	98	45	98
PLTPI	4931CH07	98	46	98
WAYS	4931CH07	101	47	100
DBFILES	4931AX01	108	48	107
FILES	4931AX02	109	49	109
APPLSA	4931AX02	110	50	109, 110
APPLSC	4931AX02	111	51	110

Headings

id	File	Page	References
CONVCON	4931IMBD	1	Part 1, General Conversion Considerations 1
INTCONV	4931CH01	3	Chapter 1, Introduction 1
DB2VAL	4931CH01	3	1.1, Value of DB2 over VSAM
MGMTPER	4931CH01	3	1.2, Management and Personnel
CONEFF	4931CH01	4	1.3, Conversion Effort 11
EFFPLAN	4931CH01	4	1.3.1, Planning
EFFMIGR	4931CH01	4	1.3.2, Data Migration
EFFAPPL	4931CH01	5	1.3.3, Application Conversion
EFFTEST	4931CH01	5	1.3.4, Testing
FUNTEST	4931CH01	5	Function Test 49
PERTEST	4931CH01	5	Performance Test 49
CONTEST	4931CH01	6	Concurrency Test 49
PROBL	4931CH01	6	1.4, Problem Area: Coexistence
POSTRP	4931CH01	6	1.6, Positioning of DB2 VSAM Transparency for VSE/ESA
PLNCONV	4931CH02	9	Chapter 2, Planning for Conversion 1
WHYREL	4931CH02	9	2.1, Why Relational
CONVVAL	4931CH02	10	2.1.1, Value
VPROBL	4931CH02	10	Problems with VSAM
RELVAL	4931CH02	10	Value of Relational
INHIBIT	4931CH02	10	2.1.2, Inhibitors
CONCOST	4931CH02	11	Costs
CONPRIN	4931CH02	11	2.2, Conversion Principles
CONPHAS	4931CH02	12	2.2.1, Conversion Phases
SWSTRAT	4931CH02	13	2.2.2, Switchover Strategies
FUNCCHG	4931CH02	13	2.2.3, Functional Changes
CONMETH	4931CH02	14	2.3, Conversion Methods 27
CONPERS	4931CH02	15	2.4, Conversion Personnel
SYSINV	4931CH02	15	2.5, VSAM Application Systems Inventory 32
DBINAN	4931CH02	17	2.5.2, VSAM File Inventory
PROGINV	4931CH02	18	2.5.3, Program Inventory
COEXIST	4931CH02	19	2.6, Coexistence Strategies 44
COXSCEN	4931CH02	20	2.6.1, Coexistence Scenarios
SYNCH	4931CH02	21	Synchronous Updates
CONCONS	4931CH02	21	2.7, Conversion Considerations
CONTOOL	4931CH02	21	2.7.1, Tools 24
CONPERF	4931CH02		

		22	2.7.2, Performance	25, 48
PROCREQ	4931CH02	22	2.7.3, Processor Requirements	11
DASDREQ	4931CH02	22	2.7.4, DASD Requirements	11
CONWTRP	4931CH02	23	2.8, Conversion With DB2 VSAM Transparency for VSE/ESA	103, 103
DBDSGN	4931CH03	27	Chapter 3, Database Design	1, 20, 50
DBNAMES	4931CH03	29	3.3.3, Data Naming Considerations	17
NET2REL	4931CH03	30	3.4, VSAM to Relational Considerations	
LDBD	4931CH03	31	3.5, Logical Database Design	
REPEAT	4931CH03	33	Repeating Groups (Fields)	40, 76
REPET1	4931CH03	33	Fields 1:1	
REPET2	4931CH03	33	Records to Columns	
REPET3	4931CH03	34	Records to Rows	
OLAP	4931CH03	35	Overlapping Fields (REDEFINES)	40, 45
TABCOM	4931CH03	36	Table Combination	40
REFINT	4931CH03	37	Referential Integrity	38
PKEYD	4931CH03	38	Primary Key Determination	44
FKDET	4931CH03	38	Foreign Key Determination	
FLD2COL	4931CH03	40	3.5.2, Field to Column Mapping	
NULLS	4931CH03	40	Nulls	
GRAPH	4931CH03	42	Graphic Strings	
GLEV	4931CH03	42	Group Level Items	39
DB2DATE	4931CH03	42	Date, Time and Timestamp Fields	40, 42
DREF	4931CH03	43	3.7, Designing for Data Related Between VSAM and DB2	
DBPD	4931CH03	44	3.8, Physical Database Design	
DISPSCR	4931CH03	46	Display Scrolling	
ESTPERF	4931CH03	47	3.9, Estimating Performance	22
TEST	4931CH04	49	Chapter 4, Testing	1, 106
METHODO	4931CH04	49	4.1, Testing Methodology	
PERFORM	4931CH04	50	4.2, Performance	
INTEGRA	4931CH04	51	4.4, Integration Test Procedures	
COMAPPL	4931CH04	52	4.4.1, Compare Application Results	
COMDATA	4931CH04	52	4.4.2, Compare Database Contents	
IDFTEST	4931CH04	52	4.4.3, Identical Function Testing	
PREPARE	4931CH04	54	4.5, Preparing for Cutover to Production	
ONLINE	4931CH04	55	4.5.1, Online System Verification	
BATCH	4931CH04	55	4.5.2, Batch System Verification	
CUTOVER	4931CH04			

		55	4.6, Cutover to Production	
POST	4931CH04	56	4.6.1, Post Cutover	
CONVTRP	4931IMBD	57	Part 2, Conversion Using DB2 VSAM Transparency for VSE/ESA	57
INTTRPF	4931CH05	59	Chapter 5, Overview of DB2 VSAM Transparency for VSE/ESA	57
TRPOVER	4931CH05	59	5.1, Introduction	
ICOMP	4931CH05	61	5.2, Components of DB2 VSAM Transparency	
IDATSTP	4931CH05	63	5.3, Steps to Enable DB2 VSAM Transparency for a VSAM File	75
INST	4931CH06	65	Chapter 6, Installation of DB2 VSAM Transparency for VSE/ESA	57, 63
LOAD	4931CH06	66	6.2.2, Install from Product Tape	
GUEST	4931CH06	72	6.8, Guest Sharing	
APPLTRP	4931CH07	75	Chapter 7, Using DB2 VSAM Transparency for VSE/ESA	57, 63, 67
FTSUPP	4931CH07	75	7.1.1, File Types Supported	60
CAPAB	4931CH07	76	7.1.2, Capabilities	28, 60
LIMITAT	4931CH07	76	7.1.3, Limitations	23, 28, 60, 103
FILDESC	4931CH07	79	7.2.3, File Description	83
SUBDEF	4931CH07	81	7.2.4, Subset Definition	
COLDESC	4931CH07	81	7.2.5, Columns Description	80
DEFI	4931CH07	85	7.3, DEFINE	79
HEAD74	4931CH07	87	7.5, MIGRATE	79, 83
HEAD75	4931CH07	88	7.6, CREATE	79, 87
HEAD76	4931CH07	89	7.7, LOAD	83, 87
HEAD77	4931CH07	90	7.8, GENERATE	79, 87
APTRSVA	4931CH07	93	7.11.1, Loading Programs into SVA	
STEPSUM	4931CH07	99	7.13, Summary of DB2 VSAM Transparency Step by Step	63, 75, 104
FOLL	4931CH08	103	Chapter 8, Beyond Transparency	24, 28, 57, 82, 88
CIRCLIM	4931CH08	104	Circumvent Limitations	47
FOLTEST	4931CH08	106	Testing	
FUNCHA	4931CH08	106	Functional Changes	
ENV	4931AX01	107	Appendix A, Environment Used	96
STATS	4931AX02	109	Appendix B, Performance Statistics	47, 107
NOTICES	SG244931 SCRIPT	113	Appendix C, Special Notices	ii
BIBL	4931BIBL			

		115	Appendix D, Related Publications
ORDER	REDB$ORD		
		117	How to Get ITSO Redbooks 115
EVAL	REDB$EVA		
		129	ITSO Redbook Evaluation x

Index Entries

id	File	Page	References
ADVANT	4931VARS	i	(1) advantages 3, 4, 24
APPLS	4931VARS	i	(1) applications 10, 10, 10, 10, 15, 15, 15, 16, 16, 16, 16, 16, 17, 17, 18
BUSIN	4931VARS	i	(1) business 3, 3, 3
COEXIST	4931VARS	i	(1) coexistence 6, 19, 20, 20, 20, 20, 20, 20, 20, 21, 25, 43
CONCURR	4931VARS	i	(1) concurrency 6, 49
CONVERS	4931VARS	i	(1) conversion 3, 3, 3, 4, 4, 4, 4, 4, 6, 9, 10, 11, 11, 12, 12, 13, 14, 14, 14, 14, 14, 14, 15, 15, 15, 15, 16, 21, 21, 21, 21, 21, 22, 23, 27, 27, 27, 32, 33, 33, 33, 34, 35, 35, 36, 36, 40, 40, 40, 41, 41, 41, 41, 41, 41, 42, 42, 42, 42, 42, 42, 42, 45, 46, 50
COSTS	4931VARS	i	(1) costs 4, 11, 11, 11
COLUMNS	4931VARS	i	(1) columns 17, 81, 83, 84
CSP	4931VARS	i	(1) CSP 16, 17, 18, 30
CUSTO	4931VARS	i	(1) customize 71
DATA	4931VARS	i	(1) data 3, 3, 5, 10, 10, 10, 14, 15, 17, 20, 20, 21, 27, 27, 28, 29, 29, 30, 30, 30, 30, 31, 31, 31, 31, 32, 32, 35, 36, 37, 38, 38, 43, 43, 44, 50, 50, 52
DB2	4931VARS	i	(1) DB2 3, 3, 62, 103, 103, 103, 103
DESIGN	4931VARS	i	(1) design 22, 23, 27, 27, 29, 32, 44, 48
FIELDS	4931VARS	i	(1) fields 32, 33, 35, 35, 40, 40, 40, 41, 41, 41, 41, 41, 42, 42, 42, 42, 42, 42, 42, 45
FILE	4931VARS	i	(1) file 4, 16, 16, 16, 17, 17, 77, 79, 99
INVEN	4931VARS	i	(1) inventory 4, 15, 16, 17, 17, 18, 18
JCL	4931VARS	i	(1) JCL 71, 85, 85, 86, 87, 88, 89, 90, 91, 93, 94, 94, 95, 96, 98
KEY	4931VARS	i	(1) key 30, 30, 32, 32, 37, 37, 37, 38, 38, 38, 38, 38, 45, 122, 122
LIMITA	4931VARS	i	(1) limitations 31, 76, 103, 104, 104
MIG	4931VARS	i	(1) migration 32, 33, 33, 34, 35, 35, 35, 36, 36, 40, 40, 40, 41, 41, 41, 41, 41, 41, 42, 42, 42, 42, 42, 42, 42, 45
ONLINE	4931VARS	i	(1) online 50, 78, 97, 98
PREPARE	4931VARS		

```
                              i          (1) prepare
                                               54, 65
PROBLEM        4931VARS
                              i          (1) problems
                                               6, 6, 10, 16, 19
RELATIO        4931VARS
                              i          (1) relational
                                               9, 10, 30
SCREEN         4931VARS
                              i          (1) screen
                                               77, 78, 79, 81, 81, 83, 84
SWITCH         4931VARS
                              i          (1) switchover
                                               13, 13, 13, 13, 13, 13, 13, 13
TABLE          4931VARS
                              i          (1) table
                                               17, 30, 30, 31, 32, 36, 37, 37, 62, 70, 103, 103, 121, 122
TEST           4931VARS
                              i          (1) test
                                               5, 5, 6, 21, 49, 49, 49, 49, 49, 49, 49, 49, 49, 50, 50, 50,
                                               50, 50, 50, 51, 52, 52, 52, 55, 55
TOOLS          4931VARS
                              i          (1) tools
                                               22
TRP            4931VARS
                              i          (1) Transparency
                                               6, 23, 23, 23, 24, 25, 25, 59, 59, 59, 61, 61, 61, 61, 61, 61,
                                               61, 61, 61, 61, 61, 61, 61, 61, 62, 62, 62, 62, 62, 62, 62,
                                               62, 62, 62, 62, 63, 65, 65, 65, 65, 65, 65, 65, 66, 66, 66,
                                               67, 68, 69, 69, 70, 70, 70, 71, 71, 71, 76, 76, 77, 77, 78,
                                               78, 78, 79, 81, 81, 83, 84, 85, 85, 85, 86, 87, 88, 89, 90,
                                               91, 93, 93, 93, 95, 96, 97, 97, 98, 98, 98, 99, 99, 99, 103,
                                               103, 103, 103, 103, 103, 103
VALUE          4931VARS
                              i          (1) value
                                               3, 4, 10
VSAM           4931VARS
                              i          (1) VSAM
                                               4, 10, 17, 17, 30, 31, 39, 42, 59, 59, 59, 59, 59, 59, 75, 75,
                                               75, 75, 75, 75, 75, 76, 77
```

Processing Options

```
Runtime values:
        Document fileid ........................................ SG244931 SCRIPT
        Document type .......................................... USERDOC
        Document style ......................................... REDBOOK
        Profile ................................................ EDFPRF30
        Service Level .......................................... 0029
        SCRIPT/VS Release ...................................... 4.0.0
        Date ................................................... 97.04.09
        Time ................................................... 05:34:15
        Device ................................................. 3820A
        Number of Passes ....................................... 4
        Index .................................................. YES
        SYSVAR D ............................................... YES
        SYSVAR G ............................................... INLINE
        SYSVAR S ............................................... OFFSET
        SYSVAR X ............................................... YES


Formatting values used:
        Annotation ............................................. NO
        Cross reference listing ................................ YES
        Cross reference head prefix only ....................... NO
        Dialog ................................................. LABEL
        Duplex ................................................. YES
        DVCF conditions file ................................... (none)
        DVCF value 1 ........................................... (none)
        DVCF value 2 ........................................... (none)
        DVCF value 3 ........................................... (none)
        DVCF value 4 ........................................... (none)
        DVCF value 5 ........................................... (none)
        DVCF value 6 ........................................... (none)
        DVCF value 7 ........................................... (none)
        DVCF value 8 ........................................... (none)
        DVCF value 9 ........................................... (none)
        Explode ................................................ NO
        Figure list on new page ................................ YES
        Figure/table number separation ......................... YES
        Folio-by-chapter ....................................... NO
        Head 0 body text ....................................... Part
        Head 1 body text ....................................... Chapter
```

Head 1 appendix text	Appendix
Hyphenation	NO
Justification	NO
Language	ENGL
Layout	OFF
Leader dots	YES
Master index	(none)
Partial TOC (maximum level)	4
Partial TOC (new page after)	INLINE
Print example id′s	NO
Print cross reference page numbers	YES
Process value	(none)
Punctuation move characters	.,
Read cross-reference file	(none)
Running heading/footing rule	NONE
Show index entries	NO
Table of Contents (maximum level)	3
Table list on new page	YES
Title page (draft) alignment	RIGHT
Write cross-reference file	(none)

Imbed Trace

```
Page 0                  4931SU
Page 0                    4931VARS
Page 0                      REDB$BOE
Page i                  REDB$ED1
Page i                  4931EDNO
Page i                  REDB$ED2
Page ix                 4931ABST
Page ix                 4931ACKS
Page x                  REDB$COM
Page x                  4931IMBD
Page 1                    4931PT1
Page 1                    4931CH01
Page 7                    4931CH02
Page 25                   4931CH03
Page 48                   4931CH04
Page 57                   4931PT2
Page 57                   4931CH05
Page 63                   4931CH06
Page 73                   4931CH07
Page 101                  4931CH08
Page 106                  4931AX01
Page 108                  4931AX02
Page 113                4931SPEC
Page 113                REDB$SPE
Page 113                4931TMKS
Page 114                4931BIBL
Page 115                  REDB$BIB
Page 116                REDB$ORD
Page 119                4931GLOS
Page 123                4931ABRV
Page 134                REDB$EVA
```