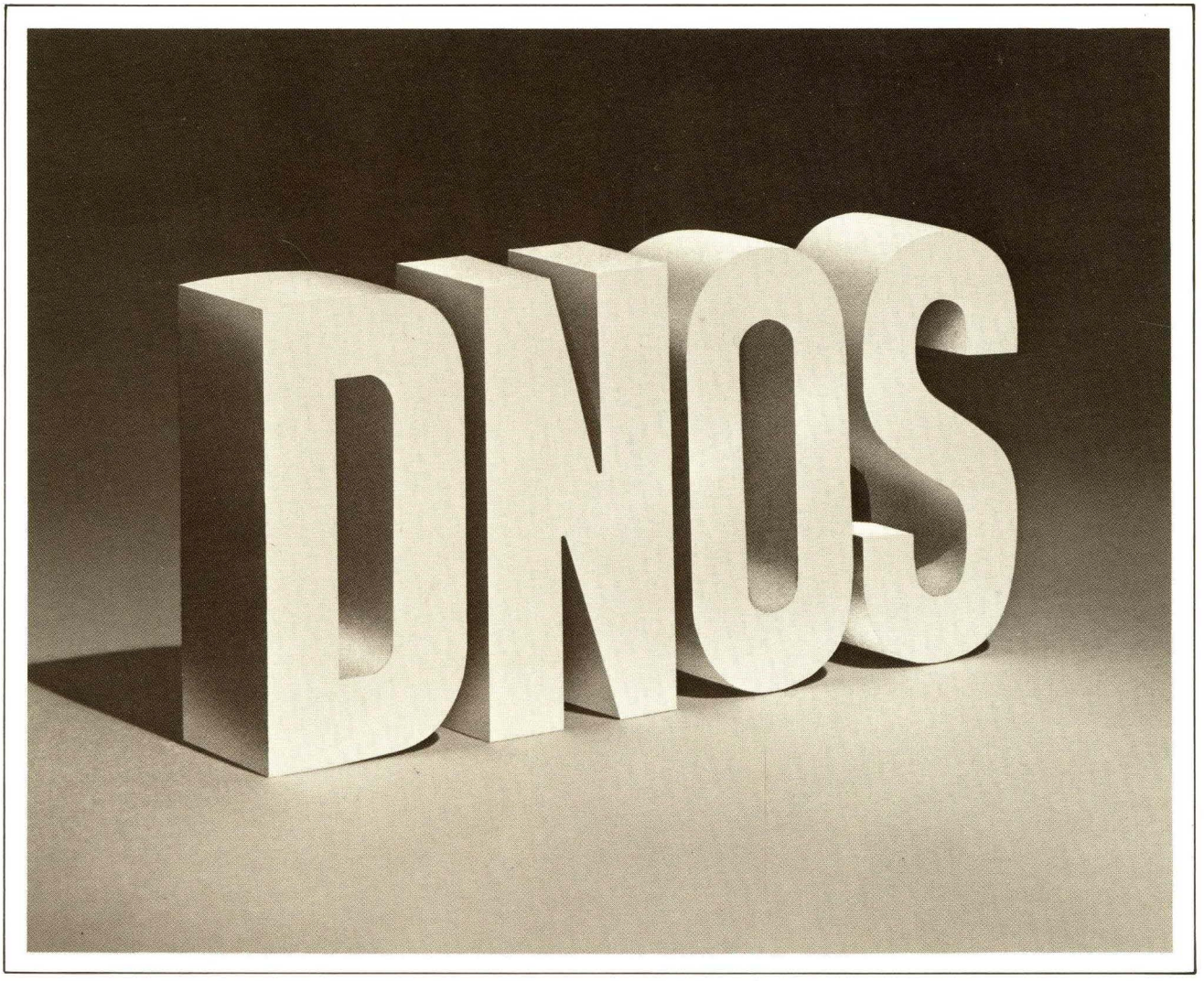
Model 990 Computer
DNOS Data Base Administrator
User's Guide



Part No. 2272059-9701 *A
15 July 1982

TEXAS INSTRUMENTS
INCORPORATED

MANUAL REVISION HISTORY

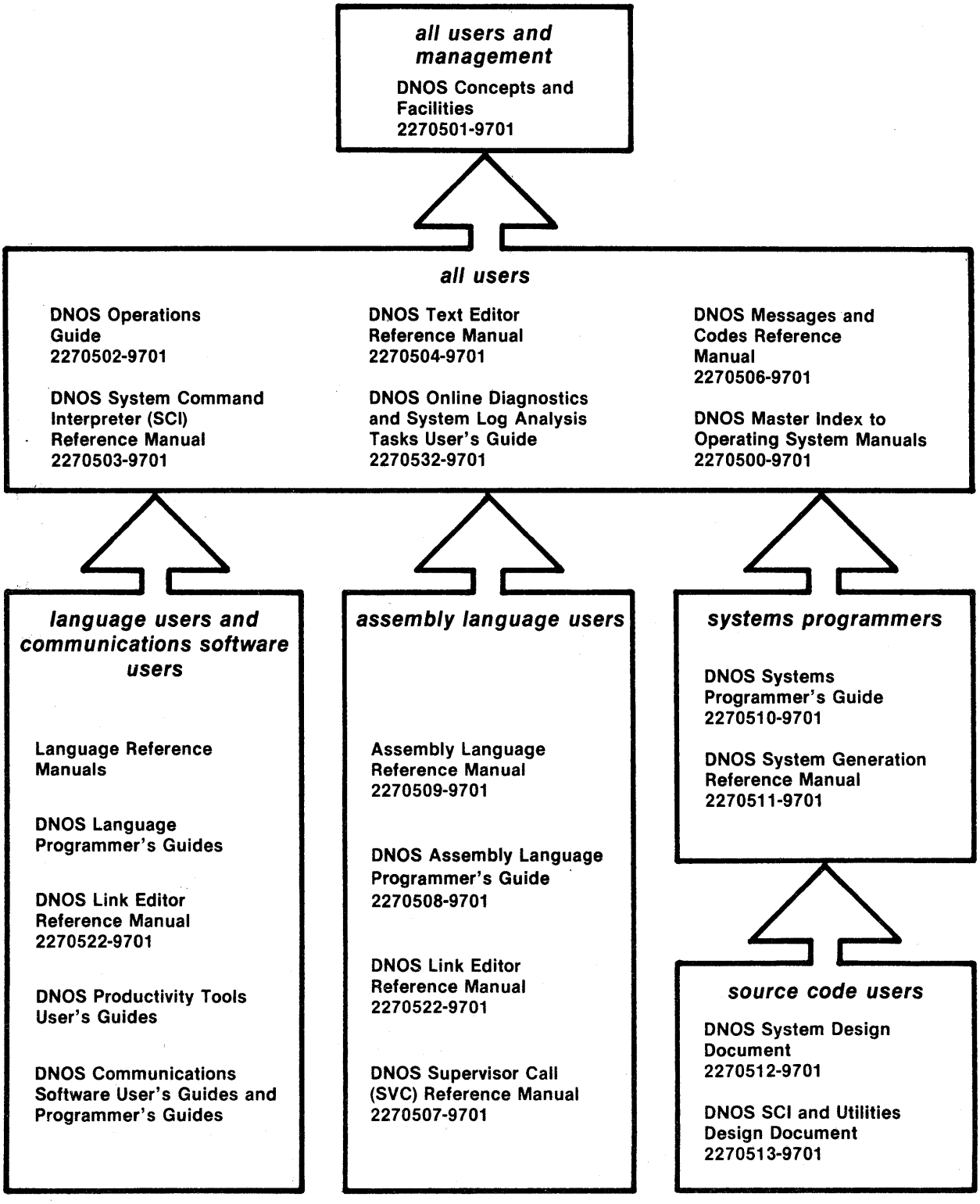
Model 990 Computer DNOS Data Base Administrator User's Guide
(2272059-9701)

Original Issue ..... 1 August 1981
Revision ..... 15 July 1982

The total number of pages in this publication is 126.

DNOS
Distributed Network Operating System Software Manuals

The manuals supporting DNOS are arranged in this diagram according to the type of user. The manuals most beneficial to your needs are those contained in the block identified as your user group and in all the blocks above that set.



2280078

DNOS
Distributed Network Operating System Software Manuals Summary

Concepts and Facilities
Presents an overview of DNOS with topics grouped into functions of the operating system. All new users (or evaluators) of DNOS should read this manual.

Operations Guide
Provides the information necessary to perform daily tasks at a TI 990 Computer installation using DNOS. Step-by-step procedures are presented for such tasks as operating peripherals, initializing the system, backing up the system, and manipulating disk files.

System Command Interpreter (SCI) Reference Manual
Describes how to use SCI in both interactive and batch jobs. Command procedures and primitives are described, followed by a detailed presentation of all SCI commands in alphabetical order for easy reference.

Text Editor Reference Manual
Shows how to use the Text Editor interactively on DNOS and includes a detailed description of each of the editing commands and function keys.

Messages and Codes Reference Manual
Lists the error messages, informative messages, and error codes reported by DNOS.

Online Diagnostics and System Log Analysis Tasks User's Guide
Provides the information necessary to execute the online diagnostic tasks and the system log analysis tasks and to interpret the results.

Master Index to Operating System Manuals
Contains a composite index to topics in the DNOS operating system manuals.

Programmer's Guides and Reference Manuals for Languages
Each programmer's guide describes one of the languages supported by DNOS (for example, assembly language, Pascal, COBOL). Each guide covers operating system information relevant to the use of that language in the DNOS environment. The details of the language itself, including language syntax and programming considerations, are in the language reference manual.

Link Editor Reference Manual
Describes how to use the Link Editor on DNOS to combine separately generated object modules to form a single linked output.

User's Guides for Productivity Tools
Each user's guide describes one of the productivity tools (for example, TIFORM, Query-990, DBMS-990, Sort/Merge) supported by DNOS. Each guide explains the function of the processor, its features, and its interface requirements.

User's Guides and Programmer's Guides for Communications Software
Describe the features, functions, and use of the communications software available for execution under DNOS. For example, there is a user's guide for the DNOS 3780/2780 Emulator software package.

Supervisor Call (SVC) Reference Manual
Presents detailed information about each DNOS supervisor call and general information about DNOS services.

Systems Programmer's Guide
Discusses the DNOS nucleus and subsystems at a conceptual and functional level and describes how to modify the system for a specific application environment.

System Generation Reference Manual
Contains the information needed to perform system generation, including pregeneration requirements, generation procedures, and information about postgeneration results.

System Design Document
Contains the information needed to understand the functioning of the system when using a source kit. This includes descriptions of the subsystems in detail, naming and coding conventions, module cross-references, data structure details, and information not found in other manuals.

SCI and Utilities Design Document
Presents design information about SCI and the DNOS utilities.

Preface

This manual describes the activities of the data base administrator (DBA). The DBA oversees the data base operation, maintains the system, manages the security assignments and the security system, and assists in the design and operation of the data base. Before reading this manual, read the *Model 990 Computer DNOS Data Base Management System Programmer's Guide*. It contains information necessary to define data elements and to write and test data base programs. Also, the DBMS Programmer's Guide provides a general overview of the system.

In this manual, angle brackets (< >) in instruction definitions indicate that the appropriate user-defined item is required.

This manual is organized into the following sections and appendixes:

Section

1 General Description — Describes the DBA's areas of responsibility.

2 Data Base Administrator (DBA) Considerations — Describes design criteria and efficient data base techniques.

3 Generation of DBMS-990 — Describes the necessary considerations for generating DBMS-990.

4 Security — Provides information necessary to maintain the security system.

5 Utilities — Provides information necessary to maintain DBMS-990.

6 General Operation Information — Provides information on the operation of DBMS-990.

7 Alias Feature — Provides information concerning substitute or alias names.

Appendix

A Key Retrieval Routines — Describes the available retrieval routines.

B Alternate Collating Sequences — Describes the procedures for internationalization.

C DBMS Error Messages and Codes — Lists and explains error messages and codes.

In addition to the DNOS manuals shown on the frontispiece, the following documents contain information relevant to this manual:

Title	Part Number
Model 990 Computer DNOS Data Base Management System Programmer's Guide	2272058-9701
Model 990 Computer DNOS Query-990 User's Guide	2276554-9701
Model 990 Computer DNOS DBMS-990 Object Installation Guide	2272092-9701

Contents

3 — Generation of DBMS-990

4 — Security

Appendixes

Index

Illustrations

Tables

1

General Description

1.1 INTRODUCTION

This section introduces the areas of responsibility for the data base administrator (DBA). Topics covered include DBA considerations, generation of DBMS-990, security, utilities, general operation information, the alias feature, and transaction-level integrity.

1.2 DBA CONSIDERATIONS

The DBA has a variety of responsibilities. The most important responsibility is to maintain the overall efficiency and organization of the data base operation. Related topics include files, data definition considerations, data manipulation and performance techniques, data base generation considerations, utility considerations, and the transaction-level integrity feature.

1.3 GENERATION OF DBMS-990

During DBMS-990 generation (DBGEN), you must make several key decisions, including whether to use alias names, whether to include file-access checking, and where security is needed. You must also choose certain linkage options, such as the key retrieval routines (random or sequential). In addition, you decide whether to set up backup logging and whether to include transaction-level integrity. You specify the number and size of interface buffers.

1.4 SECURITY

You have the option of including the security feature of DBMS-990 during the generation of DBMS-990. Passwords specify a certain access authorization for each file specified within a password. Various security commands are available to start security; change, add, and delete passwords; print the password file; and change the master password.

1.5 UTILITIES

Use the following DBMS utilities to maintain the DBMS-990 system:

Utility	Function
DBSTAT	Displays data base statistics
CDBL	Clears data base log
ILDFIL	Performs the initial load of a DBMS file
CPYFIL	Copies a DBMS file in proper format for reloading
RLDFIL	Reloads DBMS files
RECOVR	Recovers a data base
LSTDDL	Lists the data definitions for a file
SUMFIL	Provides summary statistics for a file
LADBF	Lists assigned data base files
NADB	Verifies DML calls

1.6 GENERAL OPERATION INFORMATION

General operation information includes starting and stopping DBMS-990, maintaining data integrity after a data base crash, assigning and releasing files, and estimating file sizes. The options installed determine the order for starting and stopping the various components of DBMS-990.

1.7 ALIAS FEATURE

Alias names provide substitute names for field, group, and file IDs. These longer substitute names are advantageous in that they are easier to remember than the short names they replace.

2

Data Base Administrator (DBA) Considerations

2.1 INTRODUCTION

Your duties as a DBA primarily involve controlling and standardizing the data base management system (DBMS) environment. Control refers to maintenance, and standardization involves the content and order of user data. The DBA controls not only the data structures defined in the data base but also the activities of the interactive user and the various application systems. This control requires a thorough knowledge of the various types of data structures, the appropriate procedure for defining these structures, and the best method of accessing data. You must also know how to use security, aliases, and the utilities.

The DBA imposes necessary standards on the use of DBMS-990 so that different applications can easily share data in the data base. These standards use common symbolic names for the elements defined in the data base. Standards may also involve the use of certain access techniques that produce data records that can be optimally read or updated. Develop these standards and the procedures for following them before using DBMS-990.

Be sure to consider the following DBMS-990 performance characteristics: the effects of definitions and accessing techniques on performance, and the most efficient configuration of DBMS-990 produced at data base generation (DBGEN) time. The DBGEN process involves specifying performance-oriented criteria such as linkage options, the maximum message size, whether security should be installed, whether violations are to be logged, whether file-access checking is needed, whether to install backup logging, and whether to include transaction-level integrity. Choose these features carefully, considering the trade-offs involved.

The following paragraphs discuss design criteria that cover the DBA considerations involved in operating DBMS-990. The intent of the design criteria is to establish guidelines for evaluating and solving situations peculiar to each installation. These criteria do not document or illustrate all of the possible environments.

2.2 FILES

It is advisable to give a unique file ID to each file created, as only one file with a duplicated file ID can be assigned at one time. If a user attempts to assign a file while another file with the same ID is already assigned, a duplicate file (DF) error occurs.

When estimating the size of a DBMS file, use DUMY as the file pathname for the data definition language (DDL) compiler output. The output listing file contains the results of the compile, including the file size information, even if the file does not yet exist. Use the Summarize DBMS File (SUMFIL) and Copy DBMS File (CPYFIL) utilities to obtain summary information about a file after it has been created.

The page size of a file defines the number of bytes of data that DBMS-990 reads or writes during input/output (I/O) operations. Specify 256 or 288 bytes as the page size.

To modify the size of a file, first copy the file by using the Copy DBMS File (CPYFIL) utility. If a copy of the file's data definition does not exist, use the List DDL for DBMS File (LSTDDL) utility to obtain one. You can then delete the file by using the Delete File (DF) SCI command. If enough space is available, recreate the file with the DDL and load the copied data by using the Reload DBMS File (RLDFIL) utility. The SUMFIL utility prints a summary of the number of pages available in a file.

NOTE

When you replace the data definition and need to retain the data, use the CPYFIL utility to make a copy of the file. You can then reload any data copied from the old file into the new file by using the RLDFIL utility.

Paragraph 5.2.4 specifies the various conditions that allow the use of RLDFIL to reload a copy of the data. RLDFIL does not allow certain changes; however, you can include these changes by writing a user program that reads the data in the old file, makes the necessary changes, and adds the data to the new file. Using RLDFIL to reload a copy of a file made with the CPYFIL utility eliminates skew from the file's data records. Skew is a condition in which logically contiguous lines are physically scattered throughout the disk file.

The Create Data Base Multifile Set (CDBMF) command (described in paragraph 6.3.4) allows you to establish a set of files that is accessible by one logical name. The set can span disk volumes and is treated as one logical file.

2.3 DATA DEFINITION CONSIDERATIONS

Carefully consider the record that the DDL defines for a particular DBMS file. The definition of a data base record affects both the space overhead of the file and the performance of user programs.

2.3.1 Record Considerations

A file can contain only one record definition. Each data record contained in the file reflects the record definition with the exception of the primary key and line 01. Each record definition must contain a file ID and a primary key. Also, the definition for a file must contain at least one line, although it need not be line 01. A DBMS data record is created when the first data line for a new primary key value is added to the DBMS file. The primary key for the data record is created and valued at this time.

2.3.2 Line Type 01

The data line type 01 has special significance to DBMS-990. If line 01 is defined in a file, it must be the first data line added for a unique primary key value; it must also be the last data line deleted when the record is deleted from the file. Only one data line 01 can be added to a data record.

If line 01 is not defined, data lines can be added or deleted in any order. The first data line added need not be the same as the last data line deleted. Any number of data lines of a particular type can be added to a data record.

2.3.3 Key Retrieval Method

When the DDL defines primary and secondary keys, the key retrieval method must be selected for each key. The choices are random or sequential retrieval. To choose the retrieval methods, specify the appropriate routine when DBMS-990 is generated by the DBGEN utility. Some installations require both types of retrieval. The DBMS configuration includes only the specified routines; consequently, the fewer routines specified, the smaller the DBMS task area. Appendix A contains a description of each key retrieval routine.

2.3.4 Space Overhead

Space overhead is a normal condition in any DBMS. DBMS-990 uses space to maintain logical record integrity and to store linkage information concerning secondary references (within each line). Also, additional space is required to store the primary and secondary key values and the locations of associated data lines. This space is usually referred to as the primary or secondary key area, depending on the type of key value stored. To avoid excessive space overhead, justify the length of every item in a file definition.

2.3.4.1 Primary and Secondary Keys. The primary key value affects the length of every line. However, a secondary key value changes only the length of the line in which it is declared; to that line, the secondary key adds an eight-byte secondary linkage area.

An unnecessarily long primary or secondary key wastes space overhead by producing an excessively large primary or secondary key area. Also, since each data line contains information about its primary key value, an excessive primary key length causes unused space in the data lines. Similarly, since DBMS-990 must maintain the secondary key area and the linkages for the secondary key, a long secondary key or an unnecessary secondary key results in excessive overhead.

You cannot change a secondary key value by using a write function. To change a secondary key value, delete the data line containing the value and then reenter the line with the new value included. Since data line 01 cannot be deleted until all other data lines in the record have been deleted, do not define line 01 to contain a secondary key value if that key must be updated.

Choosing as a secondary reference a field or group that will not be valued can waste space, since most of the secondary key area will remain unused. All of the unvalued keys (that is, those secondary references left as spaces or zeros when a data line is added to a file) are chained together (an unnecessary system overhead). If, however, the item must be a secondary reference to facilitate access efficiency, the space overhead may be justified.

2.3.4.2 File Size. A large file size specification can result in unnecessary space overhead. Since each data record in a file has one unique primary key value, the number of data records defined is the number of unique primary key values that can occur in the file. If a file contains too many records, excessive unused space can occur in the primary and secondary key areas, as well as in the data area. If these records will not be used for a long time, specify a smaller number of records; later, you can expand the file if need be by using the appropriate utilities.

The rate of use for the keys and data lines, as well as the number of records and lines needed, varies for each DBMS. File size depends on how often you use the utilities to expand the files and

how much time is involved in each expansion. For each system, file expansion is necessary until guidelines on file size are established for that particular system. However, if a DBMS file is static, the file can be defined to contain a minimal amount of both DBMS space overhead and unused space.

2.3.4.3 Number of Lines. When a file is defined, you should specify that the file include only enough lines to last for a fixed period of time, expanding the file as needed. Also, determine how many data lines per data record are needed in a particular file. The number depends on the use of the specified file. Use the CPYFIL utility statistics to determine the average number of lines per record by dividing the number of data lines currently in use in the file by the number of primary key values. To determine the unused space, compare the number of actual occurrences of primary and secondary key values and lines with the number of records, secondary references, and lines that were specified when the file was defined. However, it will take some time to develop more extensive guidelines for determining how much expansion area to allow for each file.

2.3.4.4 Line Length. Since each data line in a file is of a uniform length (the length of the longest line rounded to an even value), unused space usually remains at the end of those lines that do not use the full line length for data storage. DBMS-990 tends to allocate an excessive amount of space when a file definition contains lines that vary extensively in length. As a result, you should ensure that all lines in a file are as close as possible to the same length (that is, that they contain approximately the same number of bytes for data storage). Some difference in line length is to be expected; consequently, some overhead in DBMS-990 is normal.

The summary information at the bottom of a DDL specifies the total bytes used for data and key information in each type of line defined in the file. This information also specifies the standard line length. Any difference between these two values is considered space overhead in that particular DBMS file. Careful analysis of the file definition's application and design can minimize this overhead. Placing related fields and groups on the same line limits the number of highly dissimilar line lengths. If, however, a particular line uses much less space than the other lines and is valued as a data line only a minimal number of times, space overhead stays at a minimum. Wasted space results only when the actual data lines are created. Justifying the lengths of fields or groups can also keep space overhead at a minimum by ensuring that, on the average, all bytes in a field or group are used.

2.4 DATA MANIPULATION AND PERFORMANCE TECHNIQUES

The DBA is responsible for determining the configuration-language run-time linkage when DBMS is generated (DBGEN utility). This linkage specification affects memory utilization. (Refer to Section 3.) Read the discussions involving these issues carefully before choosing the DBMS-990 configuration for your installation.

The manner in which a particular application program accesses the files affects the performance of DBMS-990. A well-designed access technique can improve overall performance, just as a poorly planned technique can degrade performance. Use the guidelines presented in the following discussion to evaluate and solve particular accessing situations.

2.4.1 Field Access

Do not modify the field or group IDs used in the field list area of the line list for every DBMS call. When a call occurs, DBMS checks the names in the field list to see whether they have been encoded; if not, DBMS encodes them to its internal format. The next time the call is performed,

DBMS ignores any names already encoded in the field list, and encodes any names that must be changed. Establishing a standard set of line lists keeps overhead at minimum and improves performance. In this set, modify the field names only when necessary. Similarly, do not unnecessarily modify the password, filename, and function for each call to the DBMS; if security is installed, DBMS encodes these fields.

NOTE

You can use one function control block and line list to access different files. However, DBMS must load the encoded fields with each new file's user-defined names, whether or not the names used are the same from file to file.

2.4.2 Data Format Checking

Take care in choosing the data format of a field in the application programs. DBMS-990 does not check added or updated data in the data base to determine whether it is of the same format as that defined in the file. You must edit the data and ensure that it is of the correct format. The Primitive Query (PQUERY) command of DBMS-990 uses the data formats, displaying the data contained in the data base according to the data format defined for the field requested. If the data in a field is not of the defined format, the displayed data is meaningless.

2.4.3 Key Value Storage and Retrieval

The primary or secondary key values stored in a DBMS file are kept in the order specified by their access method (sequential or random). The read key ascending (RA) and read key descending (RD) functions read key values in the order in which they appear in the key area. This is not the order in which the key values were added to the file. However, if a read forward (RF) is performed using a secondary key value and that value occurs more than once in the secondary key chain, the specified data lines are read in the order in which they were added. In effect, when identical secondary key values are added to a file, the new values are added to the end of the chain.

2.4.4 Number of Line Types

The variety of line types defined in a DBMS file affects application program execution. To access the data lines, a file with only a few line types requires less program area, fewer line lists, and fewer data manipulation language (DML) calls than a file with many line types. This holds true whether single or multiple line lists are used. The number of DML calls depends on both the number of data lines of each line type and the number of different line types.

2.5 DATA BASE GENERATION CONSIDERATIONS

Data base generation involves choosing the key retrieval routines to be used and justifying the system linkage. To justify the system linkage, determine the size of the interprocess communication (IPC) buffer, decide whether security and violation logging are required, and determine whether file-access checking, transaction-level integrity, or backup logging is needed. All of these factors affect the performance as well as the memory requirements of DBMS-990. Refer to the DBMS Programmer's Guide for further information. Check all DBGEN listings for possible errors, and save the successful DBGEN listing.

2.5.1 File-Access Checking

File-access checking is an optional feature of DBMS-990. The primary concern when installing this feature is that of performance. With file-access checking, DBMS-990 checks each DML call to ensure that the type of access requested is allowed. The open and close file DML functions must be used with this feature. Additional memory is required to store information specifying the type(s) of access allowed for each file. (See Section 3.)

2.5.2 Backup Logging

Backup logging, an optional feature of DBMS-990, records successful updates to the data base. Once backup logging is installed during DBGEN and activated with the Start Data Base (SDBMS) command or Open Log File (OPLOG) command, it records all successful updates, whether they are interactive or batch. To recover a data base, use the last backup to restore the data base. Use the Recover DBMS File (RECOVR) utility to apply the logged updates. The End DBMS (EDBMS) command closes the log file so that any updates still in the log buffer can be recorded on the log.

In regard to backup logging, carefully consider performance. Each call to the data base must be monitored, and the successful updates must be recorded on the log. Memory is required for executing the logging routines and for the log buffer. In a strictly batch environment, this overhead may not be necessary or justified since you can easily rerun the batch jobs to restore the DBMS. Refer to Sections 3 and 5 for further information.

If the system crashes, you may lose some of the last updates performed against the data base. Usually, these updates are those that were in the log buffer and could not be written to the log file when the system crashed. However, if a batch program is processing update transactions against the data base when a crash occurs, any transactions that the program has not yet processed are lost. Determine which updates were not written to the log and which update transactions were not processed. Guidelines for identifying the lost updates depend on the individual user applications.

2.5.3 Security

Although no security system can completely prevent unauthorized access, such a system can significantly limit unauthorized access to the secured data. Security is an optional feature of DBMS-990, a feature that the DBA should carefully justify before installing it in the system at DBGEN time. You can keep the performance overhead that results from including the security system to a minimum by carefully considering the level of security necessary for each file. For example, consider a data base file that requires a password. First, you must obtain access to the security file to retrieve the authorization information; then, you must execute instructions to determine whether the type of function specified in the call is allowed for the designated password. DBMS-990 security limits the required overhead by not retrieving security information that is already available from a previous call.

After security is initially selected, installed, and created at DBGEN time, the master password holder must enter the appropriate user passwords and authorizations before the DBMS can be accessed. File access can occur after the passwords and authorizations are assigned.

At DBINS time, carefully consider the responses to the prompts MAXIMUM ENTRIES and MAXIMUM PASSWORDS. If security has already been created by DBINS and either the maximum number of entries or the maximum number of passwords must be changed, reexecute DBINS. If only these two parameters are changed from the original DBINS, do not relink programs that access the data base. To ensure that the security data is not lost, use the CPYFIL utility to copy the existing security and alias files. The security file IDS are $SC0 and $SC1. After reexecuting DBINS, use the RLDFIL utility to reload the old security data into the appropriate files.

When security is installed, a valid password is normally assigned a set of files. The files are then assigned authorizations that are valid for that password. Although a given password can be assigned to only one file, this requires extra space and results in additional demands on performance.

The level of security chosen should not be below that which is absolutely necessary. Security implemented at the file level is preferred; at any lower level (line, field, or group), the cost of security is high (in terms of performance) when retrieving authorization information. To maintain simplicity in the security checking process, groups and fields are treated alike. Resolve any conflicts in the authorizations assigned to a group and one of its fields. (See Section 4.) Do not assign unnecessary password entries and authorizations below the file level.

You can specify that an optional security-violation logging feature be installed at DBGEN time. This feature causes an additional overhead in the system because of the time involved in formatting and writing the log message. Such violation logging is necessary only in highly secure applications. In an interactive system where users are constantly logging in and out, security violations usually occur when users accidentally pass invalid passwords to the application programs. The messages that report security violations are intentionally vague. These messages state only that a security violation has occurred or that an invalid identifier was used. Additional information might inadvertently supply security information to users. Section 4 shows the format of security log messages.

Since a storage file is required for security information, the installation of security in a DBMS results in a space overhead. The number of passwords and entries to be contained determines the size of the security file. Therefore, these numbers should be calculated carefully. Passwords in the security file are coded so that they cannot be identified when the security file is displayed.

The DBA is responsible for controlling passwords. For example, the DBA assigns user passwords and authorizations and determines the number of passwords needed. The security system is not effective until you supply this information. Usually, you are also responsible for maintaining password information.

If possible, wait until all users are halted to change the security file. If all users cannot be halted, halt those using the passwords and files whose security information is to be changed. Unpredictable results can occur when security information for a password and file is updated while the information is being used. When installing security in a DBMS, give each data base file a unique name. This prevents assigning two files with the same name to one password.

2.5.4 Alias Feature

The alias feature allows substitute DBMS names of from 1 to 20 characters to be established and used in conjunction with QUERY-990. These alias names do not replace the standard DDL IDs but can be used instead of the standard IDs for the files, primary keys, fields, or groups. (Refer to Section 7.)

The alias data is stored in the file $AL1. Create this file during DBGEN by entering YES in response to the prompt ALIAS USAGE. The response to the DBINS prompt MAX # OF ALIASES determines the file's size. To change the size, reexecute DBINS. If you change no other prompt response from the original DBINS, you need not relink programs that access the data base. To save the data in the alias file, use the CPYFIL utility to copy the file. After creating the new alias file, you can reload the old data by using the RLDFIL utility and add new data by using the appropriate alias commands. (See Section 7.)

When using aliases, be sure that none are duplicated and that no alias is the same as a field or group ID.

2.5.5 Transaction-Level Integrity

Transaction-level integrity is an optional feature that may be selected by the DBA at DBGEN time. It allows the programmer to define a series of operations as a transaction. By utilizing the transaction-level integrity feature, a programmer is able to require that all operations within the defined boundaries of the transaction be performed successfully or, if one operation cannot be performed, that the data base be restored to its pretransaction state.

In the event of a system crash, all transactions in progress are rolled back, restoring the data base to its pretransaction state. This relieves you of the need to manually examine the contents of the file records in order to verify where processing was interrupted. Automatic rollback of all transactions in progress at the time of a system crash is a feature of transaction-level integrity.

2.5.5.1 How Transaction-Level Integrity Protects Your Data Base. The advantage of transaction-level integrity in preventing erroneous changes to the data base is illustrated in the following example.

A bank customer has a savings account and a checking account and wishes to make a transfer of $1000 from savings to checking. Two data base operations must be performed to complete the transfer, as follows:

1. Subtract $1000 from savings.
2. Add $1000 to checking.

In this sequence, steps 1 and 2 comprise a transaction. That is, unless both operations are completed, you do not want either operation applied to the data base.

Without transaction-level integrity, each operation stands alone. If a system crash were to occur after step 1 and before step 2, the data base would contain an inaccurate entry. In this simple example, you could possibly recover by examining the contents of the data base and making the necessary changes. However, in more complicated transactions, the recovery may be considerably more complicated.

With the transaction-level integrity option, the fact that the two operations are defined as a transaction ensures that neither operation is applied or both are applied. This ensures that the data base never contains erroneous data.

2.5.5.2 Locking Protocol. In order to prevent two users from accessing the same data base line during the course of two separate transactions, transaction-level integrity employs a system of locks. A line becomes locked whenever a read or write operation involving that line occurs in a transaction. Whenever an add or delete is performed, the entire record is locked.

There are two levels of locking: shared lock and exclusive lock. Under shared lock, the data is available for reading but cannot be modified. Under exclusive lock, the data can neither be read nor modified by transactions other than the transaction with the exclusive lock. Table 2-1 diagrams the locking protocol.

Table 2-1. Locking Protocol

Type of Lock	First User Access	Other User Access
Shared	Read Only	Read Only
Exclusive	Read/Write	Delayed

When a lock is changed from a shared lock to an exclusive lock, it is said to be upgraded. The function for upgrading is hold line (HL), as shown in Figure 2-1.

Upgrading of a user's lock (from shared to exclusive) can only occur when no other user has a shared lock on that line. If a transaction cannot be upgraded because another user also has a shared lock on the line, a delay occurs in the transaction requesting the upgrade until one of the following conditions occurs:

- The competing transaction releases its shared lock on the line, at which point the delayed transaction is allowed to proceed.
- The second transaction also requests an exclusive lock, and a deadlock occurs.

In the second case, the system identifies the deadlock and gives the exclusive lock to the transaction that first requested the exclusive lock. The second transaction is denied access and rolled back. This automatic resolution of deadlock prevents two transactions from waiting on each other indefinitely.

2.5.5.3 Deadlock. The importance of the deadlock resolution feature is illustrated in the following example of its use in a multiuser environment. Table 2-2 summarizes the steps in the example.

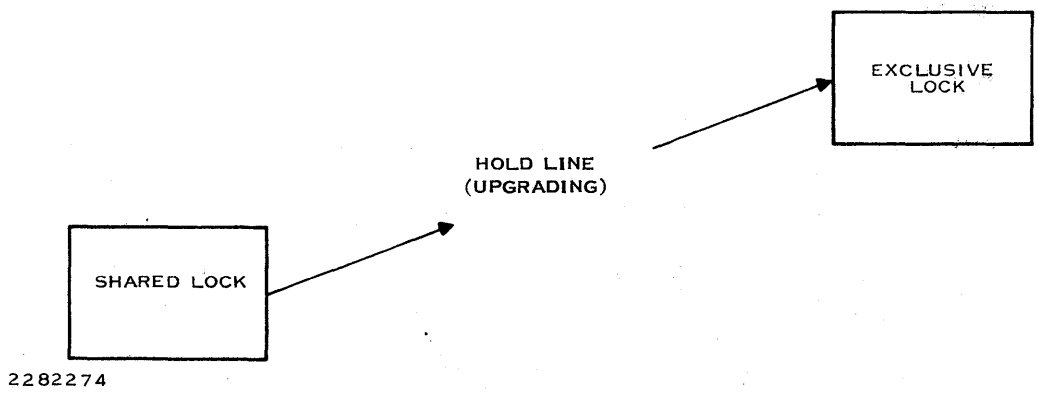


Figure 2-1. Upgrading a Lock

Table 2-2. Illustration of Deadlock

	Agent A		Agent B		
	User Action	**Program Function**	**User Action**	**Program Function**	**System Status**
1.	Makes inquiry	Start transaction (TS) Read with release	Makes inquiry	Start transaction (TS) Read with release	Shared lock on record
2.	Requests 3 seats	Hold line (HL)			Conflict (A is waiting on B; system cannot grant exclusive lock)
3.	Waits	Task delayed			
4.			Requests 3 seats	Hold line (HL)	Deadlock (B is waiting on A; A is waiting on B)
5.				Rollback transaction (TR) DL status returned	System: a. Rolls back B's transaction b. Returns DL status to B's transaction c. Grants A's transaction exclusive lock
6.	Receives 3 seats	Commit transaction (TC)			

In the example, two travel agents are each servicing a customer. Both customers desire three seats on the same airline flight. They initially want to know if three seats are available. The procedures that each agent executes are as follows:

1. Each agent starts a transaction by making an inquiry to the data base. The application program would perform a start transaction (TS) and a read with release on the same record for each agent. Both agents have access to the same record. The record, therefore, is in a shared lock state. Neither agent can write to the record while the record is in this state.

2. The customer working with Agent A decides that he wants the three seats that are available. Agent A issues a reservation request. The application program attempts to perform a hold line function to upgrade the record lock from shared to exclusive.

3. The system delays Agent A's transaction due to the conflict with Agent B's shared lock on the record. Agent A's transaction is waiting on Agent B's transaction.

4. The customer working with Agent B decides that he wants the three seats also. Agent B issues a reservation request. The DML program attempts to perform a hold line function to upgrade the record lock from shared to exclusive. The two transactions are now waiting on each other. A's transaction is delayed waiting for B's transaction to release the shared lock. At the same time, B is waiting for A's transaction to release its lock. This condition is called deadlock.

5. The system identifies the deadlock and resolves it as follows:

   a. Rolls back Agent B's transaction, thereby releasing Agent B's shared lock

   b. Returns a deadlock status message to Agent B's transaction

   c. Grants Agent A's transaction the exclusive lock it needs

6. Agent A's transaction is committed to the data base.

The deadlock message returned to Agent B's transaction can be used by the programmer to restart Agent B's transaction or output a message to Agent B's terminal informing him of the conflict and instructing him to restart the transaction.

2.5.5.4 Advantages of Transaction-Level Integrity.
Transaction-level integrity allows the programmer to define a series of data base operations as a transaction. In doing so, the programmer can require that the system execute all operations within the transaction successfully or, if one or more operations cannot be performed, roll back any data base modifications made as part of the transaction.

DBMS-990 allows transaction nesting up to a maximum level of 10. The level of nesting is specified at DBGEN.

The transaction-level integrity feature simplifies the programmer's work in that the system performs the following functions without programmer effort:

- Ensures that all conditions necessary for the success of a defined transaction are met before making a permanent change to the data base

- Returns an indication of a deadlock that the programmer can use to cause a restart of the transaction or abort the transaction and display a message to the user indicating a problem with the transaction

- Locks all data lines involved in a transaction until the transaction is either committed to the data base or rolled back

- Ensures that the outcome of any transaction operating in a concurrent environment is identical to that obtained by running the transaction by itself

Remember that there is a performance cost associated with the use of the transaction-level integrity option. System response time is sacrificed and additional memory space is required in return for a more secure data base. It is recommended that you specify small parameter values when first using the transaction-level integrity feature and increase the values if experience demonstrates that deadlock occurs too frequently.

2.5.5.5 Troubleshooting Deadlock. Any one of the following conditions can cause a deadlock:

- Two or more transactions request a lock on the same record.

- The system lock table is full. Too many locks are in use for the lock table to accommodate them all.

- A transaction contains too many updates.
- The primary keys of one or more users are longer than the maximum length specified at DBGEN time.
- More users are on the system than were specified at DBGEN time.

If frequent occurrence of deadlock is a problem, you should determine the principle reason for deadlock. If the data base statistics option is enabled, you may do this by executing the DBSTAT command and examining the statistics. Refer to paragraph 5.2.8 for information on this utility.

2.6 UTILITY CONSIDERATIONS

The utilities provided with DBMS-990 enable you to maintain the DBMS installation. You should be thoroughly familiar with the functions and characteristics of the DBMS-990 utilities.

2.6.1 Copy File and Reload File

Use these utilities to rearrange or change the DDL or to eliminate skew in DBMS files. You should use these utilities for backup and recovery only in an emergency when backup logging data cannot be used.

To rearrange a file, first make a copy of it with the CPYFIL utility. You can then use RLDFIL to redefine and load the file with the copied data. To change the DDL of a particular file, make a copy of the file then delete the original file. Create the file with the new DDL, and use RLDFIL to load the new file with the copied data.

The RLDFIL utility does not preserve the order in a secondary key chain. Keep this in mind when adding lines to a file in a specific sequence; secondary key lines are added to the end of the chain in the order in which they are added to the file. The order of the lines in any secondary key chain will probably change after a reload, regardless of the order in which the lines were added.

2.6.2 Backup, Restore, and Recover

The backup and restore functions are provided by the operating system utilities. Use the Disk Copy (DCOPY) utility to backup and restore a disk that contains a DBMS-990 data base. Use the Backup Directory (BD) and Restore Directory (RD) utilities to backup and restore directories on a disk. After making an initial backup copy of the disk, you can restore files wherever necessary. If the backup logging option is not installed and the data base must be restored, you must reenter all of the updates made since the last backup.

If the backup logging option is installed, you can apply the logged updates to the data base to recover lost updates. Use the log only if you used DCOPY or BD and RD to backup and restore the data base.

2.6.3 List DDL

The List DDL (LSTDDL) utility converts the internal DDL object code of a DBMS file into standard DDL source syntax. The converted source is written to the sequential file specified and can be used to make changes to the DDL. You can use the resulting definition as input to the DDL processor. This utility allows you to copy the data definition of a file whenever necessary, eliminating the need to retain source copies of the data definition.

2.6.4 Summarize File

The Summarize DBMS File (SUMFIL) utility allows you to obtain statistics concerning the current utilization of the space in a particular DBMS file. Summary information is written to a file concerning the current number of primary and secondary key values, along with the number of valued lines and the number of total lines in the specified DBMS file. The number of pages currently available for use are always calculated for the specified file. You can use this information to determine how space is being utilized in the data base and to make adjustments if necessary.

2.6.5 Initial Load of a Data Base File

The Initial Load DBMS File (ILDFIL) utility enables DBMS users to load data initially into a DBMS file without writing a program. (You still need programs that add data or update existing data in the file.) This utility facilitates loading test (or live) data into a file. Create the load file according to the format specified in Section 5. If the DBMS file to be loaded already contains data, the new data is added to the file. However, ensure that the load file does not contain any duplicate information.

2.6.6 List Assigned Data Base Files

The List Assigned Data Base Files (LADBF) utility allows you to list all currently assigned data base files. The listing shows the four character ID of each data base file and the corresponding pathname.

The LADBF utility is especially useful after recovery from a system crash. LADBF enables you to quickly identify which files were assigned at the time of the crash.

2.6.7 Clear Data Base Log

The Clear Data Base Log (CDBL) utility should be used sparingly and with caution. CDBL erases all internal data base log data associated with the transaction-level integrity feature. The ability of the system to roll back any incomplete transactions is destroyed. CDBL does not affect user-defined files (such as the log file specified for LOGGING ACCESS NAME).

Use CDBL to erase the internal data base logs when you believe they contain invalid data. The internal log data could block the restarting of the data base if, for example, a file had been deleted after an EDBMS. Clearing the internal data base logs allows you to restart the data base.

CDBL can only be executed when the data base task is not operating.

2.6.8 Data Base Statistics

The Data Base Statistics (DBSTAT) utility can be used as a developmental tool to analyze the performance of the data base management functions. DBSTAT is useful in determining the causes of transaction deadlock, and the number of physical and logical I/O operations associated with each DML function.

3

Generation of DBMS-990

3.1 INTRODUCTION

DBMS-990 can be generated in different configurations to meet the specific processing and memory requirements of the installation. The generation of DBMS-990 involves two utilities: DBGEN and DBINS. In DBGEN, you choose the data base options and IPC buffer size. The specified DBMS configuration is assembled and linked, and batch streams are prepared. DBINS then installs the DBMS on the target disk. The software prepares the security/alias files, if requested, and generates the installation test program.

3.2 LINKAGE OPTIONS

The link between the application program and DBMS-990 is through IPC, using the module S$DBMS.SNDMSG and an interface module for the program language (either S$DBMS.COBINT or S$DBMS.FRGMY). Each DBMS-990 user task must include these modules in the link edit.

DBGEN parameters determine the size of the IPC buffer.

3.3 DBGEN

DBGEN generates the linked object and batch streams necessary to install the DBMS-990 system. (Refer to the DBMS-990 object installation guide for details about installation.) DBGEN is only a configuration processor and does not install DBMS-990. The following paragraphs explain the possible configurations and how to determine the best configuration for a particular system.

The DBGEN command has the following form, with the specified defaults:

```
DBMS GENERATION   <VERSION L.V.E.>
          DBMS990 PATHNAME:  DBMS990N
              TRANSACTIONS?:  YES
                STATISTICS?:  YES
               BUFFER SIZE:  512
                SECURITY?:  NO
     FILE ACCESS CHECKING?:  NO
          BACKUP LOGGING?:  YES
              ALIAS USAGE?:  NO
```

After you respond to each prompt, the following screen appears:

```
SELECT KEY RETRIEVAL ROUTINES
     RETRIEVAL ROUTINES:  R1,S1
```

If you select the sequential routine (S1), the following screen appears:

```
INTERNATIONALIZATION
 ALTERNATE COLLATING SEQUENCE:   NO
```

Refer to Appendix B for further discussion of internationalization.

If you enter YES in response to this prompt, the following screen appears:

```
SPECIFY ALTERNATE COLLATING SEQUENCE
                        PATHNAME:
```

If you requested security, the following screen also appears:

```
DBMS SECURITY
  LOG SECURITY VIOLATIONS?:   NO
```

Since DBGEN modifies the DBMS-990 installation disk, copy this disk before proceeding. The installation disk retains all listings from DBGEN. Also, DBGEN creates a file to contain the configuration listing. The name of this file is DBMS990N.CONFIG.

3.3.1 DBMS990 Pathname

The response to this prompt is the name of the DBMS-990 installation disk DBMS990N, or the DBMS-990 directory pathname.

3.3.2 Transactions

A YES response to the TRANSACTIONS? prompt requires that you also respond YES to the BACKUP LOGGING? prompt. In subsequent steps of DBMS generation (SDBMS), the system specifies the log file blocking factor of 1 as a result of selecting transaction-level integrity.

If you select transaction-level integrity, the following screen appears:

```
TRANSACTION-LEVEL INTEGRITY
              CONCURRENT USERS:  4
                LEVEL OF NESTING:  5
                 MAX LINE IMAGES:  20
                     MAX RECORDS:  10
                  MAX DATA LINES:  50
                     MAX KEY SIZE:  10
```

The prompts and appropriate responses are as follows:

CONCURRENT USERS — Enter the number of users that can have concurrent tasks pending.

LEVEL OF NESTING — Enter the maximum level of transaction nesting. Up to 5 levels are permitted.

MAX LINE IMAGES — Enter the maximum number of updates that can occur within a transaction. This value should be at least as large as the largest record, taking into account that each data base line equals one record on file.

MAX RECORDS — Enter the maximum number of records that all active transactions can access at any one time. The value entered affects the memory allocated for the transaction-level integrity feature.

MAX DATA LINES — Enter a value that is the sum of the maximum number of line images accessed per transaction and the maximum number of read operations per transaction. This value determines the size of the memory allocated for the transaction-level integrity feature.

MAX KEY SIZE — Enter the maximum number of characters used in any primary key of your data base file.

Remember that entering unnecessarily large values in response to these prompts results in a performance degradation that you may find unacceptable. You must balance this concern for limiting unnecessary overhead with the opposing concern for providing enough memory workspace for the feature to operate without a frequent occurrence of deadlock.

3.3.3 Statistics

The response to the STATISTICS? prompt determines whether the system maintains statistics on the frequency of use of the DBMS functions. The information provided by this option is especially useful during the initial stages of implementing applications using transaction-level integrity. The statistics are used to adjust the DBGEN parameters to their optimum size for your configuration and the nature of your application programs.

3.3.4 Buffer Size

The buffer size defines the largest DBMS call block that can go through the IPC channel. A call block consists of the function control block, line list, and data area specified as parameters for a call to DBMSYS. The maximum buffer size is 512 bytes. To calculate the buffer size, use the following formula:

BS = 10 + CB + LL + DA

where:

CB is the length in bytes of the function control block, which contains the key value as the last item. Note that the length of the control block is 24 bytes plus the key length area. If the key area has an odd length, round it to the next even integer; CB must be even. In the following control block, CB is 28:

```
01  ITEM-FUNCTION.
    03  ITEM-PASS   PIC X(4).
    03  ITEM-FUNC   PIC XX.
    03  ITEM-STAT   PIC XX.
    03  ITEM-FILE   PIC X(4).      VALUE "ITEM".
    03  ITEM-LOC1   PIC X(4).
    03  ITEM-LOC2   PIC X(4).
    03  ITEM-KEY    PIC X(4)       VALUE "ITMN".
    03  ITEM-NUM    PIC X(3).
```

LL is the length in bytes of the line list of the maximum call block. In the following line list, LL is 32:

```
01  ITEM-LINE.
    03  FILLER      PIC X(7)      VALUE "LINE=01".
    03  ITEM-TST    PIC X         VALUE ",".
    03  FILLER      PIC X(20)     VALUE
                    "DESCUPRCQTYOQTYH****".
    03  ITEM-HOLD   PIC X(4)      VALUE "RLSE".
```

DA is the length in bytes of the data area in the longest call block. If the length is an odd number, round it to the next even integer; DA must be even. In the following data area, DA is 34:

```
01  ITEM-INFO.
    03  ITM-DESC    PIC X(20).
    03  ITM-PRIC    PIC 9(3)V99.
    03  ITM-QTYO    PIC 9(4).
    03  ITM-QTYH    PIC 9(4).
```

BS is the length in bytes of the IPC buffer. In this example, total buffer length is as follows:

10 + 28 + 32 + 34 = 104 bytes

You need not specify the exact buffer size, but the size must be large enough to accommodate the largest DBMS call parameters.

3.3.5 Security

Whether to include security depends on the environment in which DBMS-990 is running. For example, if only one terminal is attached to the system and only one person knows how to operate the terminal, security is probably unnecessary. Security increases performance overhead by checking the buffer during each request to DBMS-990. The amount of this overhead depends on the level of security defined. Security does not add additional memory overhead to the application task, but it does make the DBMS task larger. Also, additional storage is required for the security file.

3.3.6 Log Security Violations

If the DBMS system includes security, you have the option of logging security violations on the system log. The overhead involved is minimal.

3.3.7 File Access Checking

File-access checking allows you to assign shared, exclusive, or read only exclusive access to a file. In comparison the Open File and Close File commands allow you to designate file access but not the type of access. File-access checking is important in systems where the user needs exclusive access to certain data base files.

If file-access checking is not specified, all files are considered to be shared and available for any task. The user can run programs that include Open File and Close File command calls to DBMS-990 even if file-access checking is not specified. In this case, the Open File and Close File commands are ignored, and the file is considered to be shared.

File-access checking does not affect DBMS interface size, although it affects DBMS performance and total memory usage.

3.3.8 Backup Logging

Backup logging keeps track of adds, updates, and deletes. This feature allows you to restore a data base from a previous copy of the data base files. Backup logging is required if you include transaction-level integrity.

You first make a copy of all files that DBMS will use. To enable logging, the user enters either the Start Data Base Manager Command or the Open Log command, specifying the file to which the log is placed. Then, each call that changes data in the data base causes an entry in the log file. If a severe error is encountered in the data base, the user can refer to the previous copy of the files and the log file to recover the data base.

Although logging does not add memory to the application task, it does affect the performance of DBMS-990. The time required to monitor the updates and write the successful ones to the log can delay the execution of adds, updates, and deletes.

3.3.9 Alias Usage

If Query-990 is available to the DBMS, the user can request the DBMS-990 alias feature. This feature allows the user to specify alternate names for fields in a data base in Query-990. Refer to the *Model 990 Computer DNOS Query-990 User's Guide* for further details.

The alias feature affects neither the size of the DBMS interface nor the memory size of DBMS. Alias substitution affects only Query-990 performance and disk storage space (for the alias file).

3.4 DBINS

To install the DBMS, execute the DBINS process. This process deletes the old version of DBMS-990, installs the new tasks and commands, creates the installation test environment, and optionally creates new security/alias files. For further details on installation procedures, consult the DBMS-990 object installation document.

The DBINS command has the following form:

```
INSTALL DBMS-990
        DBMS990 PATHNAME:
           TARGET VOLUME:
```

If the security or alias features were configured at DBGEN time, the following prompt also appears:

```
NEW SECURITY/ALIAS FILE?:  YES
```

If you request new security/alias files and the security option was configured, the following screen appears:

```
DBMS SECURITY
        MAXIMUM ENTRIES:
    MAXIMUM PASSWORDS:
      MASTER PASSWORD:
```

If you request new security/alias files and the alias option was configured, the following screen appears:

```
DBMS ALIAS USAGE
        MAXIMUM ALIASES:
```

The following paragraphs discuss each of these prompts. The DBMS-990 installation disk retains all listings from DBINS.

3.4.1 DBMS990 Pathname

The response to this prompt is the name of the DBMS-990 installation disk DBMS990N, or the value of the DBMS-990 directory pathname.

3.4.2 Target Volume

The response to this prompt is the volume name of the disk to which the DBMS is to be installed. Although this disk must be a system disk, it need not be the system disk of the current system.

3.4.3 New Security/Alias File

If either the security or alias feature is requested, a special file stores password and/or alias information. If this file has not yet been created, create one by responding YES to this prompt.

3.4.4 DBMS-990 Security

If the security feature is requested at DBGEN time and a new security file is to be created, you must respond to the prompts MAXIMUM ENTRIES, MAXIMUM PASSWORDS, and MASTER PASSWORD.

MAXIMUM ENTRIES specifies the maximum number of passwords, files, lines, and field/groups to be assigned.

MAXIMUM PASSWORDS specifies the maximum number of passwords to be assigned.

MASTER PASSWORD is the four-character alphanumeric value to be used as the master password of the installed system.

If the security file becomes full, execute a DBINS to increase the size. Programs that use the data base need not be relinked as long as the same DBGEN configuration is used. After the DBINS executes, reassign the passwords and authorizations.

3.4.5 DBMS-990 Alias Usage

If the alias feature is requested at DBGEN time and a new alias file is to be created, the prompt MAXIMUM ALIASES appears during the DBINS process. This value is the maximum number of alias names that can be assigned for the data base. The alias information is stored in the alias file.

To change this value at a later time, execute another DBINS. Then, reassign the aliases.

4

Security

4.1 INTRODUCTION

Password security is an optional feature that can be enabled during DBMS-990 generation. Although this security system cannot prevent all unauthorized use of the data base, it does prevent easy and accidental access to the DBMS data. In deciding whether to install security in DBMS-990, keep in mind that security increases access time for DBMS requests and memory requirements for the DBMS processor. Refer to Section 2 for further details.

4.2 PASSWORDS

When security is installed in DBMS-990, every DBMS call except open and close commands must contain a valid password. The two types of passwords are master and user. The master password is known only to a small number of people who are responsible for assigning user passwords and maintaining data base stability (by means of the utilities). The master password allows all accesses to the data base. A user password is assigned to any person or group that requires access to only a portion of the data in the data base.

4.3 ACCESS AUTHORIZATION

User access may be limited to the file, line, group, or field level, or to a combination of these levels. For example, a data base user who requires access to data in the personnel file might be denied access to the salary field. Also, you can restrict the functions (such as read, write, and delete) available to a specific user. For example, a user might be allowed to read a customer's identification number and address without being allowed to change that data.

The Add Password (ADDPSW) command assigns each user password to a set of files. Any file that is not in this set is inaccessible when this password is in use. Thus, accessing a file requires either a password that is assigned authorization to the file or the master password.

Each file in the set is also assigned an access authorization that restricts the type of function that can be performed on that file. The following access types are available:

- Read
- Write (Update)
- Add
- Delete
- No access

These access types are combined to form the user's access authorization for a particular DBMS file. For example, one password may specify an authorization with only read access to a particular file, while another password specifies an authorization with read, write, and add access to that file. A user who is assigned the first password has only read access to that file.

You can use almost any combination of these access types to form a permissible access authorization. For a no access authorization, do not specify any of the access types. For example, if a DBMS file allows all authorizations, you can assign no access to specific lines in the file by giving them no authorizations when executing the Add Password Entry (ADDPE) command.

You can assign an authorization code to all levels of data. In the absence of an assigned authorization code, data elements assume the authorization of the next highest elements. For example, lines assume the authorization code of the file. If necessary you can avoid this by assigning to a line only a subset of its file's authorization, including no access. If a file has read and write authorization, a line in that file might be assigned only read authorization. Similarly, if a user password includes access only at the file and line levels, all fields in the line have the same authorization as the line. To avoid this, you can assign to a field only a subset of the line's authorization, including no access.

To facilitate security checking, groups are treated as fields. The DBA (or whoever assigns the passwords) must resolve any conflicts in authorization access between a field and its group.

NOTE

Assigning authorization to lines and fields requires more time and space for security checking. Because of this additional system overhead, you should assign authorization at the file level only, whenever possible. Thoroughly justify authorizations assigned to a line, group, or field.

To make a line, group, or field more secure than a file, assign a subset of the file's authorization to the line, group, or field. The following priorities apply when assigning authorization:

Data Level	Priority
Field/Group/Line	Highest
File	Lowest

The following restrictions apply to authorization assignments:

- Any authorization that includes delete or write access must also have read access. This restriction is necessary because a read with hold must be performed before a delete or write.

- All lower-level authorizations must be a subset of their associated higher-level authorizations. A subset can be an identical authorization. For example, if an authorization is made at the field level, its associated line(s) must be assigned an authorization that contains the field's authorization. If the line needs no special restrictions, the line should be assigned the same authorization as the associated file.

- If you assign delete authorization at the line level, all fields and groups in the line must also have delete authorization. This restriction is necessary because a delete is performed on a line rather than on a field.

Security does not check the key specified in a command on the field level. If the file passes the security check, the user can access the key. If the user includes a secondary key in the command, an unexpected response may occur. (A data base user can access information by using a secondary key whether or not the key element is available to him at the field level.) Also, the user can execute the data manipulation language (DML) functions read key ascending (RA) and read key descending (RD) whenever the file authorization is read access.

4.4 PASSWORD PROCEDURES

To add and change password information, the user must enter a master password. The master password is necessary for computer operations in which one person or group controls all password authorizations.

All passwords are stored in security files of the data base in a coded format. Changes to the password information can occur only when DBMS-990 is running. Security files are stored on the system disk under the directory .S$DBMS.

The system checks security whenever a user issues a DML call to DBMS-990. However, the security program need not access security files for each call. Previous call information remains in the user control block in order to minimize this overhead.

CAUTION

Any change in the password entries immediately affects the DBMS security checking. Therefore, use caution when changing password entires while DBMS-990 is running. If an error occurs during password assignment, an error message is generated.

4.4.1 Change Master Password (CMPSW)

To change the master password, enter the SCI command Change Master Password (CMPSW). The following prompts appear:

```
CHANGE MASTER PASSWORD
    OLD MASTER PASSWORD:
    NEW MASTER PASSWORD:
```

The prompt responses are as follows:

OLD MASTER PASSWORD — Enter the current master password.

NEW MASTER PASSWORD — Enter four alphanumeric characters that will become the new master password.

4.4.2 Change Password (CPSW)

To change a password, enter the SCI command Change Password (CPSW). The following prompts appear:

```
CHANGE PASSWORD
        MASTER PASSWORD:
            OLD PASSWORD:
            NEW PASSWORD:
```

The prompt responses are as follows:

MASTER PASSWORD — Enter the current master password.

OLD PASSWORD — Enter the password to be changed.

NEW PASSWORD — Enter the four alphanumeric characters that will replace the old password value.

Although this procedure requires a number of data base operations and may take a considerable amount of time, it is more efficient than recreating all of the password information.

4.4.3 Add Password (ADDPSW)

To add a password, enter the SCI command Add Password (ADDPSW). The following prompts appear:

```
ADD PASSWORD
        MASTER PASSWORD:
                PASSWORD:
```

The prompt responses are as follows:

MASTER PASSWORD — Enter the current master password.

PASSWORD — Enter four alphanumeric characters that will be assigned as a new user password.

This command enters the specified password into the security file. Then, use the Add Password Entry (ADDPE) command to enter specific authorizations for the password.

4.4.4 Add Password Entry (ADDPE)

To add an entry to a password, use the SCI command Add Password Entry (ADDPE). This command includes two prompting screens. The user can repeat these screens to make any number of entries to the password in a given ADDPE command. The prompt menus appear as follows:

```
ADD PASSWORD ENTRY
                MASTER PASSWORD:
                       PASSWORD:
          TYPE (FILE, LINE, ITEM):
                           FILE:
                           LINE:
        ITEM (FIELD OR GROUP):

AUTHORIZATION
                    READ ACCESS?:  NO
                   WRITE ACCESS?:  NO
                     ADD ACCESS?:  NO
                  DELETE ACCESS?:  NO
                   MORE ENTRIES?:  NO
```

The prompt responses are as follows:

MASTER PASSWORD — Enter the current master password.

PASSWORD — Enter a previously assigned user password.

TYPE(FILE, LINE, ITEM) — Enter the type of addition: F for file, L for line, or I for item (which refers to a field or group).

FILE — If the response to the type prompt is F (file), enter the file ID to be added. Otherwise, enter the file ID of the entry to be added. (A four-character value is expected.)

LINE — If the response to the type prompt is F (file), this entry is ignored (enter a carriage return). If the type is L (line), enter the line to be added. If the type is I (item), enter the line number of the entry to be added. (A two-character value is expected.)

ITEM (FIELD or GROUP) — If the response to the type prompt is F (file) or L (line), this entry is ignored (enter a carriage return). If the type is I (item), enter the field or group ID to be added. (A four-character value is expected.)

READ ACCESS? — A YES response assigns read authorization to the entry. A NO response prohibits read access to the entry when using this password.

WRITE ACCESS? — A YES response assigns write authorization to the entry. A NO response prohibits write access to the entry when using this password. Note that if you enter a YES response, read access must also be assigned.

ADD ACCESS? — A YES response assigns add authorization to the entry. A NO response prohibits add access to the entry when using this password.

DELETE ACCESS? — A YES response assigns delete authorization to the entry. A NO response prohibits delete access to the entry when using this password. Note that if you enter a YES response, read access must also be assigned.

You can repeat the ADDPE screens any number of times until you enter NO in response to the MORE ENTRIES? prompt. Add only one element to the password for each password entry request. For example, if line 02 needs read authorization and the file containing it needs read and write authorization, the required entries are as follows:

```
ADD PASSWORD ENTRY
          MASTER PASSWORD:  XXXX
                PASSWORD:  YYYY
  TYPE (FILE, LINE, ITEM):  FILE
                    FILE:  FILX
                    LINE:
  ITEM (FIELD OR GROUP):

AUTHORIZATION
            READ ACCESS?:  YES
           WRITE ACCESS?:  YES
             ADD ACCESS?:  NO
          DELETE ACCESS?:  NO
            MORE ENTRIES?:  YES

ADD PASSWORD ENTRY
          MASTER PASSWORD:  XXXX
                PASSWORD:  YYYY
  TYPE (FILE, LINE, ITEM):  LINE
                    FILE:  FILX
                    LINE:  02
  ITEM (FIELD OR GROUP):

AUTHORIZATIONS
            READ ACCESS?:  YES
           WRITE ACCESS?:  NO
             ADD ACCESS?:  NO
          DELETE ACCESS?:  NO
            MORE ENTRIES?:  NO
```

The assignment of file authorization must precede the assignment of line authorization in the file. Similarly, line authorization must precede field/group authorization. The line authorization must be a subset of the file authorization, and the field/group authorization must be a subset of the line authorization.

4.4.5 Delete Password (DELPSW)

To delete a password and all associated information, enter the SCI command Delete Password (DELPSW). The following prompts appear:

```
DELETE PASSWORD
          MASTER PASSWORD:
                PASSWORD:
```

The prompt responses are as follows:

MASTER PASSWORD — Enter the current master password.

PASSWORD — Enter the user password to be deleted.

4.4.6 Delete Password Entry (DELPE)

To delete an entry in the password, use the SCI command Delete Password Entry (DELPE). The following prompts appear:

```
DELETE PASSWORD ENTRY
        MASTER PASSWORD:
              PASSWORD:
TYPE (FILE, LINE, ITEM):
                  FILE:
                  LINE:
ITEM (FIELD OR GROUP):
```

The prompt responses are as follows:

MASTER PASSWORD — Enter the current master password.

PASSWORD — Enter the user password of the entry to be deleted.

TYPE(FILE, LINE, ITEM) — Indicate the type of entry to be deleted: F for file ID, L for line ID, or I for field or group ID.

FILE — If the response to the type prompt is F (file), enter the identifier (ID) of the file to be deleted from the password security information. Otherwise, enter the file ID of the entry to be deleted.

LINE — If the response to the type prompt is F (file), this prompt is ignored (enter a carriage return). If the type is L (line), enter the ID of the line to be deleted. If the type is I (item), enter the line ID of the entry to be deleted.

ITEM (FIELD OR GROUP) — If the response to the type prompt is F (file) or L (line), this prompt is ignored (enter a carriage return). If the type is I (item), enter the ID of the group or field to be deleted from the password security information.

Only one element can be specified per DELPE command. If a line is deleted, all field and group entries associated with it are also deleted. If a file entry is deleted, all lines associated with it are deleted.

4.4.7 Map Password File (MPSWF)

Use the SCI command Map Password File (MPSWF) to obtain information about the contents of the password file. The following prompts appear:

```
MAP DBMS-990 PASSWORD FILE
        MASTER PASSWORD:
    LISTING ACCESS NAME:
```

The prompt responses are as follows:

MASTER PASSWORD — Enter the current master password.

LISTING ACCESS NAME — Enter a standard pathname or device to which the output is written.

A sample output file is as follows:

```
MAP OF DBMS-990 PASSWORDS                AUTHORIZATION SYMBOLS

CREATED:02/11/78                           R  READ ACCESS
PASSWORDS ASSIGNED:2                       W  WRITE ACCESS
PASSWORDS AVAILABLE:28                     A  ADD ACCESS
                                           D  DELETE ACCESS
                                           N  NO ACCESS


PASSWORD     FILE     LINE     GROUP/FIELD     AUTHORIZATON

  XXXX
               FIL1                                RWAD
                        02                          R
                                    ITM1            N
                        04                          RW
  YYYY
               FIL2                                RWA
               FIL3                                RD
```

The authorization column contains a one-character value for each access authorization assigned. The values appear in the upper right-hand corner of the listing.

4.5 USING SECURITY

If security checking is requested when DBMS-990 is generated, each DML call (except open and close file functions) requires a valid password. The DML call control block parameter has the following format, shown in COBOL. (Refer to the *Model 990 Computer DNOS Data Base Management System Programmer's Guide* for further details.)

```
01 COMMAND.
    02  PASSWORD   PIC X(4).
    02  FUNCTION   PIC XX.
    02  STATUS     PIC XX VALUE IS "**".
    02  FILENAME   PIC X(4).
    02  LOC1       PIC X(4) VALUE IS "****".
    02  LOC2       PIC X(4) VALUE IS "****".
    02  KEYNAME    PIC X(4).
    02  KEYVALUE   PIC X(6).
```

If the password used in the DML call is invalid, security returns a status code of SV (security violation) and the DML call is not performed. If the password does not have authorization for the function specified in the DML call, security returns a status code of II (invalid identifier), and the DML call is not performed.

If you did not specify security checking at installation time, DBMS-990 ignores the password area of the control block parameter. The function still begins at the fifth character of the control block.

4.6 ERROR MESSAGES

The error messages displayed during password procedures are listed and explained in Appendix C. If you request logging of security violations, a message is printed on the system log whenever a DBMS call returns a security violation (SV status code). The format is as follows:

```
JJJ:HHMM DBMS SV T (task) = II (RR):SS
```

where:

JJJ is the Julian date.
HHMM is the time of error (24-hour clock).
II is the task-installed ID;
RR is the task session ID;
SS is the station ID of the terminal associated with the task.

5

Utilities

5.1 INTRODUCTION

The following DBMS utilities help you in maintaining a DBMS-990 installation:

ILDFIL — Initial Load DBMS File
LSTDDL — List DDL for DBMS File
CPYFIL — Copy DBMS File
RLDFIL — Reload DBMS File
SUMFIL — Summarize DBMS File
LADBF — List Assigned Data Base Files
CDBL — Clear Data Base Log
DBSTAT — Display Data Base Statistics
RECOVR — Recover DBMS File
NADB — Navigate Data Base

5.2 UTILITY FUNCTIONS

If security has been installed, you must enter the master password to perform any utility function. If security has not been installed, the prompt MASTER PASSWORD does not appear in the screen display. The following examples assume that security has been installed.

5.2.1 Initial Load of a File (ILDFIL)

The ILDFIL command uses the following prompts:

```
INITIAL LOAD DBMS FILE
          MASTER PASSWORD:
          DB FILE PATHNAME:
                DB FILE ID:
        LOAD FILE PATHNAME:
```

The responses to the prompts are as follows:

MASTER PASSWORD — Enter the DBMS master password (required only if security is installed in DBMS-990).

DB FILE PATHNAME — Enter a valid pathname of a data base file.

DB FILE ID — Enter the four-character ID for the data base file.

LOAD DB FILE PATHNAME — Enter the name of a sequential file or device that contains the data to be loaded; signifies the load file.

This utility allows you to load data from a sequential file into a previously created DBMS file. You can use ILDFIL to add data to an existing DBMS file. Since the DBMS file may already contain data, take care not to load duplicate data. All lines to be loaded are added; no updates are performed. The load file data must have the following format:

Columns	Data
1 and 2	Line type
3 through N	Primary key (where N equals the length of the key plus 2)
N + 1 to end	Data for the line in the exact format as defined in the DDL.

DBMS-990 must be running and the file assigned to operate this utility. The data specified in columns N + 1 to end must be in the same order and of the same length as specified in the DDL. If the data is not in the correct format, the entire line of data may be affected. Data is added in the order given in the load file (by performing an add after operation). This utility does not convert the data.

If an error occurs during loading, an error message is printed and the loading aborts.

5.2.2 List DDL (LSTDDL)

The LSTDDL command uses the following prompts:

```
LIST DDL FOR A FILE
          MASTER PASSWORD:
        DB FILE PATHNAME:
                DB FILE ID:
    LISTING ACCESS NAME:
```

The responses to the prompts are as follows:

MASTER PASSWORD — Enter the DBMS master password (required only if security is installed in DBMS-990).

DB FILE PATHNAME — Enter the pathname for the file.

DB FILE ID — Enter the four character ID of the DBMS file.

LISTING ACCESS NAME — Enter the name of a sequential file or device to which the source of the data definition is written.

The LSTDDL command converts the DBMS internal DDL structure of the specified file into the standard external DDL format; also, it writes the DDL to the specified output file. If the output access name indicates a file that does not currently exist, a sequential file with the designated access name is created. The DDL written to the output file is in standard DDL source format. The user can edit the file and can use it as an input file to the DDL processor.

The line and volume sizes are computed from the internal structure; therefore, they represent the actual maximum size of the file. These sizes may be greater than specified in the original DDL of the file; this is because physical disk space is allocated in multiples of DBMS page size, and the amount of disk space required for the line and volume sizes specified in the original DDL might not be a multiple of the DBMS page size.

5.2.3 Copy File (CPYFIL)

The CPYFIL command uses the following prompts:

```
COPY DBMS FILE TO COPY FILE
          MASTER PASSWORD:
          DB FILE PATHNAME:
                DB FILE ID:
       COPY FILE PATHNAME:
     LISTING ACCESS NAME:
```

The responses to the prompts are as follows:

MASTER PASSWORD — Enter the DBMS master password (required only if security is installed in DBMS-990).

DB FILE PATHNAME — Enter the pathname for the file.

DB FILE ID — Enter the four character ID of the DBMS file to be copied.

COPY FILE PATHNAME — Enter the name of a sequential file or device to which the data is to be copied.

LISTING ACCESS NAME — Enter the name of a sequential file or device.

This function is one of a pair of utilities that reorganize DBMS files. CPYFIL creates a copy of a DBMS file in a sequential file. The companion to CPYFIL is RLDFIL, which reloads the file onto a disk. This set of utilities can reformat the DDL and can be used, if necessary, as a backup. The user can copy only one DBMS file at a time. If the sequential file does not exist, it is created.

DBMS-990 must be running to operate CPYFIL. To copy a DBMS file, DBMS-990 reads all of the data lines associated with the primary key entry and reads forward through all of the entries in the file. All of the data lines for each data record are gathered together physically while retaining the original logic order of the data records (that is, primary keys and their associated data lines); secondary key order is not necessarily preserved if random retrieval is specified for that key.

CPYFIL also provides a listing of statistics generated during the copy, as shown in Figure 5-1. These statistics provide a picture of the physical skew of the data area of the data base file being copied and include the following:

- The maximum number of lines in the file and the number of lines available
- The greatest number of lines per record and the average number of lines per record
- The most efficient number of lines per page and the most efficient number of pages per average record

```
DBMS-990     <L. V. R>          DBMS-990 CPYFIL     MM/DD/YY  HH:MM:SS

FILE SOFL SUMMARY
   PAGE SIZE:                    256      SHORTEST RECORD:          2
   LINE LENGTH:                   32      LONGEST RECORD:           8
   MAX # OF LINES:               304      AVERAGE RECORD:           5
   # OF LINES AVAILABLE:         278      # OF RECORDS:              5

   BEST POSSIBLE PAGES/AVG RECORD:          1


   % OF TOTAL    EXTRA PAGE READS
    RECORDS       FOR AVG RECORD
        0            9  OR MORE
        0            4  TO    9
        0            3  TO    4
       20            2  TO    3
       60            1  TO    2
       20            0  TO    1


   LINE TYPE        # OF OCCURRENCES
      02                      9
      03                     17
      BL                      0
     TOTAL                   26
```

Figure 5-1. CPYFIL Output Example

- A breakdown, by the amount of skew, of the records in the data area as a percentage of the total
- A breakdown, by line type, of the total number of lines

Reloading the data records (using RLDFIL) retains the logical structures and eliminates physical skew.

5.2.4 Reload the File (RLDFIL)

The RLDFIL command has the following prompts:

```
RELOAD DBMS FILE FROM COPY FILE
                MASTER PASSWORD:
                DB FILE PATHNAME:
                      DB FILE ID:
              COPY FILE PATHNAME:
             LISTING ACCESS NAME:
```

The responses are the same as for the CPYFIL.

The DBMS must be running to operate the RLDFIL utility. This utility is the companion to CPYFIL and can either extend files or reformat the DDL. You can reload only one DBMS file at a time. RLDFIL can be used, if necessary, as a restore function.

NOTE

If RLDFIL has been performed, unexpected errors may result from using RECOVR. Since RLDFIL changes the structure of the file, the structure no longer conforms to that of the backup version. If logging is installed, back up the data base and open a new log file immediately after executing the RLDFIL.

The file that CPYFIL copies is reloaded onto disk. This function permits changing field lengths and positions within lines, adding fields to lines, making fields into secondary keys, enlarging the file, and making limited field format changes. Primary key order is preserved. However, RLDFIL will probably change any user-imposed ordering of data lines with respect to secondary keys. RLDFIL aborts if the specified file is not of the proper format.

The DDL to which the copy file is to be restored (new DDL) is compared to the DDL retained in the copy file (old DDL). If these are equal (except for file size), the file is immediately reloaded. If the DDLs are different, the following mapping of data fields occurs:

- All field values in the new DDL receive values from the fields of the same name in the old DDL. Any field not present in the old DDL receives binary zeros. Any field in the old DDL that is not in the new DDL is dropped.
- New fields can be defined as secondary keys. However, these fields are initialized to binary zeros; thus, they are all linked together in the same secondary key chain. To value these secondary key fields, the specific data lines must be deleted and readded to the file.
- When field lengths are changed, the data values in the old file must be mapped into the new fields. The rules for fitting old values into new fields are as follows:
  - The types of changes allowed:

Format	Type of Change
AN	The precision of the field. Truncation on high- or low-order digits may occur.
AS	The precision of the field. Truncation on high- or low-order digits may occur.
CH	The length of the field. The field will be left justified; extra character positions will be padded with blanks.
CS	The precision of the field. Truncation on high- or low-order digits may occur.
CN	The precision of the field. Truncation on high- or low-order digits may occur.

IS	No changes to this field are allowed.

- Any value with decimal digits is justified in the new field by aligning the decimal point. If the new field is larger, zeros fill space on the left or right, as needed.

- If the new field is smaller, the value is truncated. The new value is aligned with the new decimal position. The sign of the value is retained. It is possible to specify a new field that requires padding on one side and truncation on the other.

The following warning messages might be written to the terminal local file after the user reloads the data (all DBMS-990 error messages are described in Appendix C):

FIELD DROPPED IN NEW DDL—XXXX
XXXX denotes a field ID. The field was found in the old DDL but was not listed in the new DDL.

POSSIBLE LOSS OF SIGNIFICANCE IN FIELD—XXXX
XXXX denotes a field ID. This message results if the field in the old DDL has more significant digits than the field declared in the new DDL.

XX LINE TYPE MISSING IN NEW DDL
XX denotes the line identifier missing in the new DDL. This message occurs if the old DDL contains a line type not in the new DDL.

The following error message might be written to the terminal local file:

DBMS FILE IS NOT EMPTY
The file into which the data is to be loaded is not empty. The reload does not occur. The system also returns the error FERROR (Appendix C).

5.2.5 Summarize File (SUMFIL)

The SUMFIL command uses the following prompts:

```
SUMMARIZE FILE STATUS
          MASTER PASSWORD:
        DB FILE PATHNAME:
                 DB FILE ID:
     LISTING ACCESS NAME:
```

The responses to the prompts are as follows:

MASTER PASSWORD — Enter the DBMS master password (required only if security is installed in DBMS-990).

DB FILE PATHNAME — Enter the pathname for the file.

DBMS FILE ID — Enter the four character name of the DBMS file.

LISTING ACCESS NAME — Enter the access name to which the summary will be written. If the file does not exist, it is created.

A summary of a file is obtained and printed. For each file, the following is provided:

- The maximum number of pages
- The maximum number of lines

For each key in the file, the following is provided:

- The maximum number of entries
- The actual number of entries
- The number of collisions (random key retrieval only)
- The number of key chains (random key retrieval only)
- The longest key chain (random key retrieval only)
- The average chain length (random key retrieval only)

Figure 5-2 shows an example of SUMFIL output when RANDOM/1 is in use.

5.2.6 List Assigned Data Base Files (LADBF)

The List Assigned Data Base Files (LADBF) utility allows you to list all currently assigned data base files. The listing shows the four character ID of each data base file and its corresponding pathname.

The LADBF utility is especially useful after recovery from a system crash. LADBF enables you to quickly identify which files were assigned at the time of the crash.

The LADBF command uses the following prompt:

```
LIST ASSIGNED DATA BASE FILES
        OUTPUT ACCESS NAME:
```

In response to the OUTPUT ACCESS NAME prompt, enter the pathname of the file in which the output is to be placed. Accepting the default causes the system to return the output to the terminal screen.

```
DBMS-990     <L. V. R>          DBMS-990 SUMFIL     MM/DD/YY  HH:MM:SS

SOFL SUMMARY

MAX # OF PAGES:      71                    MAX # OF LINES:         304

   KEY          ENTRIES          TOTAL        TOTAL       LONGEST      AVG KEY
 ID/TYPE      MAX     TOTAL   COLLISIONS     CHAINS     KEY CHAIN      CHAIN

SONM/R1        50         5            4          1             4          4
BILL/R1        50         0            0          0             0          0
SHIP/R1        50         9            4          3             2          1
ITEM/R1       200        16            0          0             0          0
```

Figure 5-2. SUMFIL Output Example with Random (R/1) Retrieval

5.2.7 Clear Data Base Log (CDBL)

The Clear Data Base Log (CDBL) utility should be used sparingly and with caution. CDBL erases all internal data base log data associated with the transaction-level integrity feature. The ability of the system to roll back any incomplete transactions is destroyed. CDBL does not affect user-defined files such as the after-image log.

Use CDBL to erase the internal data base logs when you believe they contain invalid data. The internal log data could block the restarting of the data base if, for example, a file had been deleted after an EDBMS. Clearing the internal data base logs allows you to restart the data base.

CDBL can only be executed when the data base task is not operating. There are no prompts associated with this utility.

5.2.8 Data Base Statistics (DBSTAT)

The Data Base Statistics (DBSTAT) utility can be used to analyze the performance of the data base management functions. DBSTAT is useful in determining the causes of transaction deadlock and the number of physical and logical I/O operations associated with each DML function. An EDBMS or SDBMS resets all the statistics to zero.

The DBSTAT command uses the following prompts:

```
DBMS STATISTICS
                    FUNCTIONS:  ALL
        LISTING ACCESS NAME:
```

The responses to the prompts are as follows:

FUNCTIONS — Enter the two character function codes (for example, RB, RF, or RS) denoting the functions for which you want statistics displayed. Use commas to separate the entries. Accepting the initial value ALL causes the statistics for all functions to be displayed.

LISTING ACCESS NAME — Enter the pathname of a file if you wish to save the output. Specifying a pathname for the output allows you to retain the statistics for reference. Accepting the default causes the statistics to be displayed at your terminal.

The DBSTAT command displays various statistics concerning the operation of the data base. DBSTAT generates a report in four parts, as follows:

NUMBER OF CALLS
This part displays the number of times a function has been called by an application program.

LOGICAL I/O
This part displays information concerning the number of logical I/O operations associated with each DML function. The total number of logical I/O operations together with the minimum, maximum, and average I/O operations per call are displayed. If the average number of calls to a function is less than one, the symbol <1 appears instead of the actual value.

PHYSICAL I/O

This part displays information concerning the number of physical I/O operations associated with each DML function. The total number of physical I/O operations together with the minimum, maximum and average I/O operations per call are displayed. If the average number of calls to a function is less than one, the symbol <1 appears instead of the actual value. Note that there may not be a physical I/O associated with a logical I/O.

DEADLOCK

This part displays information concerning the reasons for deadlock. The total number of deadlocks due to conflicts in lock requests and lock table overflows are displayed.

5.2.9 Navigate Data Base (NADB)

The Navigate Data Base (NADB) utility is an interactive tool designed to allow you to verify the effect of a sequence of DML calls without coding an application program. NADB prompts for the various parameters needed by each of the possible DML functions. The unedited parameters are submitted to DBMS. After the specified DML function is performed, the NADB utility returns the unedited results to the terminal. If a DML function operates in NADB, it will also work when coded into the logic of an application program. If the function fails in NADB, it will also fail in an application program.

The NADB command uses the following prompts:

```
Navigate a Data Base

                PASSWORD:

              FUNCTION:          STATUS CODE RETURNED
                FILE ID:
                KEY ID:      VALUE IN HEX?     CONTINUE?
             KEY VALUE:

              LINE TYPE:

              FIELD IDS:

                  LOC1 :              LOC2 :
```

The responses to the prompts are as follows:

PASSWORD — Enter a four character password.

FUNCTION — Enter the two character abbreviation for the DML function required.

FILE ID — Enter the four character file ID for the data base file. This is the same ID assigned to the file by the DDL utility.

ACCESS TYPE — Enter SHRD (Shared), EXCL (Exclusive), or ROEX (Read Only Exclusive).

KEY ID — Enter the four character ID that corresponds to a primary or secondary key defined by the DDL utility.

VALUE IN HEX? — Enter Y if the KEY VALUE prompt will be answered in hexadecimal digits rather than character format. Enter N if the KEY VALUE prompt will be answered in character format.

KEY VALUE — Enter up to 80 hexadecimal digits (two lines) if Y was entered for the VALUE IN HEX? prompt. Enter up to 40 ASCII characters if N was entered for the VALUE IN HEX? prompt.

LINE TYPE — Enter a two character ID for the line type required. Multiple line types are allowed and must be separated by a single comma. (No blanks are allowed between IDs.)

FIELD IDS — Enter the four character ID for a field in a particular line. Multiple field IDs are allowed and must be separated by a single comma. (No blanks are allowed between IDs.)

LOC1 — Enter four asterisks or eight hexadecimal digits for the location pointer.

LOC2 — Enter four asterisks or eight hexadecimal digits for the location pointer in an update.

NOTE

Inappropriate modification of the location pointers can cause serious damage to the data base file. It is recommended that you set the location pointer to asterisks so that the system will determine the valid location pointer for the record to be updated.

After information is entered for the LOC2 prompt, the following screen appears:

```
                DATA (FROM/FOR) DBMS:                    CONTINUE?

Function Keys:  F1,F2                                    DBMS STATUS:

0000  4130  3030  0000  0000  0000  0000    A0  00  ..  ..  ..  ..  ..  ..  ..
0000  4130  3030  0000  0000  0000  0000    A0  00  ..  ..  ..  ..  ..  ..  ..
0000  4130  3030  0000  0000  0000  0000    A0  00  ..  ..  ..  ..  ..  ..  ..
0000  4130  3030  0000  0000  0000  0000    A0  00  ..  ..  ..  ..  ..  ..  ..
0000  4130  3030  0000  0000  0000  0000    A0  00  ..  ..  ..  ..  ..  ..  ..

LINE TYPE:

FIELD IDS:
```

The function keys provide the following capabilities:

F1 — The F1 function key allows you to access both the main screen and the subsequent screen alternately. Viewing both screens alternately allows other portions of the control block to be viewed without losing the data area.

F2 — The F2 function key allows you to access the internal formatter options. This screen is discussed later in this section. The responses to the prompts are as follows:

COMMAND — The COMMAND key aborts the current session of NADB.

BACK FIELD — The BACK FIELD key returns to the last prompt during the NADB session.

The screen lists the memory and the translation for the line type and field IDs previously entered. The LINE TYPE and FIELD IDS prompts contain the values entered in the previous screen.

The internal formatter is a secondary function of the NADB utility. This secondary function allows you to enter a data type, decimal count, and an external format for a field value. NADB returns the data base internal format displayed in hexadecimal digits. The internal format screen is as follows:

INTERNAL FORMATTER

DATA TYPE:

DECIMAL COUNT:

LENGTH:

EXTERNAL FORM:

INTERNAL FORM:

The responses to the prompts are as follows:

DATA TYPE — Enter a two character ID for the data type required.

DECIMAL COUNT — Enter a two character number.

LENGTH — Enter a two character number.

EXTERNAL FORM — Enter the ASCII form of the data.

INTERNAL FORM — The system supplies the hexadecimal format of the data entered for the EXTERNAL FORM prompt.

5.2.10 Utilities for DBMS Logging

Backup logging is an optional feature of DBMS-990; you can install it with the system at DBGEN time. If logging is installed, all changes to the data base remain in a log file (on disk or tape). The following DBMS functions are logged: AA, AB, DL, DR, and WT.

5.2.10.1 Backup of the Data Base. To use logging, keep a backup of the DBMS, saving all files. To facilitate this, use the operating system utilities Disk Copy/Restore (DCOPY), Backup Directory (BD), Copy Directory (CD), and Copy/Concatenate (CC).

5.2.10.2 Restore a Data Base. Before beginning recovery, restore the DBMS to the state it was in when logging began. Use the utilities DCOPY, Restore Directory (RD), CD, and CC.

5.2.10.3 Recover a Data Base (RECOVR). The recover command uses the following prompts:

```
RECOVER DATA BASE FROM LOGGING FILE
                MASTER PASSWORD:
 LOGGING DATA ACCESS NAME:
                BAD FILE NAME:
            BAD FILE PATHNAME:
     MAXIMUM ASSIGNED FILES:
          MAXIMUM OPEN FILES:
```

The prompt responses are as follows:

MASTER PASSWORD — Enter the DBMS master password (required only if security is installed in DBMS-990).

LOGGING DATA ACCESS NAME — Enter the access name of the device or file that contains the logging data.

BAD FILE NAME — Enter ALL if you wish to recover all files, or enter the four character ID assigned for the DBMS file to be recovered.

BAD FILE PATHNAME — Enter the pathname that contains the bad file name. This prompt can be left blank if ALL was specified for the BAD FILE NAME.

MAXIMUM ASSIGNED FILES — Enter the maximum number of assigned files.

MAXIMUM OPEN FILES — Enter the maximum number of open files.

To apply the log to the data base, use the following steps:

1. Enter the End DBMS (EDBMS) command if the system is still operational. This command closes the log file.

2. Restore the data base file(s) to the state it was in when the DBMS log was opened by copying your backup file onto the bad file pathname.

3. Enter the RECOVR command.

4. Create a backup of the DBMS files, by using the Copy/Concatenate (CC) command.

5. Enter the Start DBMS (SDBMS) command and specify a new log file. This causes all transactions in progress at the time of the crash to be rolled back.

RECOVR reads all entries in the log file and applies the DML commands to the DBMS. You can use RECOVR only on a data base that has been restored.

NOTE

If RLDFIL has recreated a file, RECOVR does not work properly on that file. See the discussion on RLDFIL for further details.

5.3 UTILITY RESPONSES

The utilities attempt to perform the requested function. The status of the specified utility open completion is reported in the SCI message line; this is the bottom line of the video display terminal (VDT) screen or the last output line produced by the SCI procedure on hard-copy devices. Three messages can occur:

- DBMS UTILITY COMPLETED NORMALLY

- ILLEGAL FUNCTION:<function>
  In this message, <function> is the response made to the prompt FUNCTION: and is not a legal utility function (Appendix C).

- DBMS UTILITY ERROR:<error>
  In this message, <error> is a six-character error code (Appendix C).

Utility errors are listed and described in Appendix C.

5.3.1 Operating System Error Codes

Occasionally, an operating system file I/O error that is not covered by either the utility or DBMS error codes occurs. Such an error is reported in the <error> portion of the message line with O.S. in the first four character positions and the hexadecimal error code in the last two positions. Refer to the *Model 990 Computer DNOS Messages and Codes Reference Manual* for descriptions of these error codes.

5.3.2 Abnormal Termination

During DBMS utility execution, if a severe error occurs the normal error processing cannot accomplish recovery. The following message appears:

END ACTION TAKEN: ERROR CODE = XX WP = XXXX PC = XXXX ST = XXXX

where:

ERROR CODE is the termination code. (Consult the *Model 990 Computer DNOS Messages and Codes Reference Manual.*)

WP is the contents of the workspace pointer

PC is the contents of the program counter

ST is the contents of the status register

6

General Operation Information

6.1 INTRODUCTION

This section discusses starting and stopping DBMS-990, data integrity after a data base crash, and assigning of DBMS-990 files; it also provides formulas (along with examples) for estimating the size, in pages, of a file. The actual size of a file is determined by compiling the DDL definitions.

6.2 STARTING AND STOPPING THE DATA BASE

Use the Start DBMS (SDBMS) SCI command to start DBMS-990. If the security or alias feature is included, the security/alias files are automatically opened. If the backup logging feature is included, the SDBMS command prompts the user for the log file access name and then opens the log file.

The End DBMS (EDBMS) command stops DBMS-990 and closes the log file. Ensure that all application tasks using DBMS-990 have completed before issuing the EDBMS command.

6.2.1 Start DBMS (SDBMS)

The SDBMS command starts the DBMS-990 task. The DBMS must be started before executing any DML functions. The SDBMS command opens the backup log if this feature is installed. The following prompts appear:

```
START DATA BASE MANAGER <VERSION: L.V.R.>
    MAXIMUM ASSIGNED FILES:
        MAXIMUM OPEN FILES:
  LOG FILE BLOCKING FACTOR:  1
```

where:

L.V.R. identifies the release number.

Respond to these prompts as follows:

MAXIMUM ASSIGNED FILES — Enter the maximum number of files that can be assigned at the same time. (Assigning a DB file associates a file pathname with the file ID.) If the security or alias features are installed, include the security/alias files in this number.

MAXIMUM OPEN FILES — Enter the maximum number of data base files simultaneously open. (This prompt appears if the file-access checking feature has been installed.)

LOG FILE BLOCKING FACTOR — Enter an integer in the range of 1 through 10. This prompt appears if the backup logging feature has been installed. However, this prompt does not appear when transaction-level integrity is enabled. The blocking factor specifies the number of log records that are to be stored in memory before writing them to the log file. A response of 1 specifies that each log record is to be written immediately. In this case the value 1 is specified by the system and cannot be modified.

Note that a response of 1 for the log file blocking factor guarantees that at most 1 log record will be lost if the system fails. However, a response of 1 requires an access to secondary storage for each record and therefore requires more overhead. A response of 10 requires less overhead but can cause the loss of up to 10 log records if the system fails. Accordingly, a response between 1 and 10 compromises between these two considerations.

If SDBMS cannot open a file because the physical integrity of the file is in question, the following Bad File message appears:

0132 UNABLE TO ACCESS FILE <file id>
0138 DBMS OPEN ERROR IS 0A
0133 PATHNAME OF FILE IS <file pathname>
0135 UNABLE TO RESTART DBMS, CANNOT REOPEN FILES
0136 RESTART FAILED

All responses to the SDBMS command affect the size of the DBMS-990; therefore, use conservative estimates whenever the exact number is not known.

Any errors encountered are reported upon completion of the command.

6.2.2 Open Log File (OPLOG)

The OPLOG command opens the backup logging file. When backup logging is installed, the log file is automatically opened when the DBMS-990 is started by using the SDBMS command. Before any access to the data base can occur, the log file must be opened. The following prompts appear:

OPEN DBMS-990 LOG FILE
    LOGGING ACCESS NAME:

In response to the prompt LOGGING ACCESS NAME, enter the access name of the logging device or file. Entering DUMY indicates that log records need not be kept for a specified run. If a file pathname is specified, the file must be precreated with the following characteristics:

- Sequential file
- Logical record length equal to [2 + (BF * 548)] bytes, where BF is the blocking factor specified during the SDBMS command
- Physical record length greater than [LRL + 8], where LRL is the logical record length. The physical record length should always be a multiple of the allocatable disk unit (ADU) size of the disk containing the log file.
- Initial allocation set to the desired number of records in the log file

- Nonexpandable
- Forced write

The log file is nonexpandable to prevent the user from inadvertently using the entire disk. When the log file becomes full, all tasks using the DBMS-990 receive an error status until the log is closed and reopened with a new log file. In applications where this action is not acceptable, create the log file as an expandable file.

6.2.3 Close Log File (CLLOG)

The Close Log File (CLLOG) command uses no prompts. The End DBMS (EDBMS) command executes CLLOG so that the last log buffer is written to the log file.

6.2.4 End DBMS (EDBMS)

Execute the EDBMS command to halt DBMS-990 in an orderly manner. It is the user's responsibility to ensure that all programs (tasks) using DBMS-990 are halted before executing EDBMS. If a user task is running, it halts immediately. The following prompt appears:

```
END DATA BASE MANAGER
        ARE YOUR SURE?:  NO
```

In response to the prompt ARE YOU SURE?, enter YES or NO. The prompt has a default response of NO to prevent accidentally terminating DBMS-990.

6.2.5 Data Integrity After a DBMS Crash

The integrity of the data after a DBMS crash is questionable only if the crash occurs during an update. Also, only one data base file can be affected. If a file was being updated when the crash occurred, a Bad File (BF) message is returned when you attempt to restart the data base. The Bad File message indicates that the file is locked from being reassigned. Use one of the following procedures to ensure data integrity within that file:

1. If your system includes back-up logging, perform the steps described in paragraph 5.2.10.3. The RECOVR procedure first restores the data base files, then applies the updates from the log file. The updates that were in the logging buffer as the system crashed are *not* applied during RECOVR. When RECOVR completes and you create a backup of the recovered file, then execute the SDBMS command and proceed.

2. For systems that do not include back-up logging, or if for some reason you do not want to perform the RECOVR procedure, use the Unlock Data Base File (UDBF) command to unlock the file. UDBF *does not* guarantee data integrity for the file. Therefore, at some time after executing the UDBF command, execute the Copy File and Reload File utilities (paragraphs 5.2.3 and 5.2.4) to restore data integrity.

3. You can restore the locked file by copying a backup file (use the CC command). This restores the file to the state it was in at the time the backup copy was made. If necessary, reapply the updates made between the time of the backup and the time of the DBMS crash.

6.3 DATA BASE FILE COMMANDS

Use the following commands to assign, release, and unlock data base files and to create a file that can span disk volumes.

6.3.1 Assign Data Base File ID (ADBF)

The ADBF command associates the pathname of a file with a DBMS file ID. Files must be assigned before DBMS can use them. You must start DBMS-990 in order to assign a DBMS file.

The ADBF command has the following prompts:

```
ASSIGN DATA BASE FILE ID
                    FILE ID:
          DB FILE PATHNAME:
```

The prompt responses are as follows:

FILE ID — Enter a four-character alphanumeric value.

DB FILE PATHNAME — Enter the pathname of the file.

6.3.2 Release Data Base File ID (RDBF)

The RDBF command releases a file ID that the ADBF command assigned. Once a file ID has been released, DBMS-990 no longer recognizes the file.

The RDBF command has the following prompts:

```
RELEASE DATA BASE FILE ID
                    FILE ID:
```

In response to the FILE ID prompt, enter the four-character file ID to be released.

6.3.3 Unlock Data Base File (UDBF)

Read the discussion on data integrity (paragraph 6.2.5) before using this command. The UDBF command unlocks a data base file that was locked during a data base crash and is used in conjunction with the Copy File and Reload File utilities.

The UDBF command has the following prompts:

```
UNLOCK DATA BASE FILE
                    FILE ID:
          DB FILE PATHNAME:
              ARE YOU SURE?:
```

The prompt responses are as follows:

FILE ID — Enter a four-character alphanumeric value.

FILE PATHNAME — Enter the pathname of the file.

ARE YOU SURE? — Enter Y or N to confirm your selections.

A Good File (GF) status code is returned if the file has not been locked. A File Set (FS) status code is returned if the file has already been unlocked.

6.3.4 Create Data Base Multifile Set (CDBMF)

This procedure creates two or more data base files that are accessible by one logical name. Using the logical name, you can treat the set as if it were one file. The purpose of this procedure is to establish a data base file that spans disk volumes.

The first step in creating a multifile set is to determine the total file size (in pages) of the set. Determine this as you would any other data base file, by defining the DDL for the file, formatting the DDL using DUMY as the DB file pathname, and looking at the DDL listing for the total pages required. (Refer to paragraph 3.6 in the DNOS DBMS Programmer's Guide for details on defining the data base file.)

When you know the total pages required for the set, enter the CDBMF command. The following prompts appear:

```
[ ] CDBMF
    CREATE DB MULTIFILE SET

                    LOGICAL NAME:
                       PAGE SIZE:  256
          TOTAL MULTIFILE PAGES:
```

Enter a logical name for the set in response to the LOGICAL NAME prompt. For PAGE SIZE, enter 256 (the default) or 288. In response to the TOTAL MULTIPLE PAGES, enter the total pages required from the DDL listing. After you answer these prompts, you are prompted to define the pathname and the number of pages for the individual data base files (one at a time), as follows:

```
CREATE DB FILE #1
                       PATHNAME:
            PAGES FOR THIS FILE:
```

The number of pages for the first file should be a portion of the total number of multifile pages entered in the first screen. You might set this number so that the file uses the remainder of a disk volume. After you answer these prompts, a similar screen prompts you to define the next file in the set. After the first CREATE DB FILE screen, the initial value for the PAGES FOR THIS FILE prompt is the total multifile pages minus the pages for the file(s) already defined. CREATE DB FILE screens appear until the sum of the number of pages for the individual files equals the total multifile pages. The CDBMF command creates the individual files as they are defined. Consequently, if you abort the command, you may want to delete the files that have been defined.

The CDBMF command assigns the specified logical name to the set of files. The logical name is picked up when DBMS is started. Therefore, if you execute CDBMF while the data base is running, you must stop and restart in order for DBMS to pick up the logical name. Use the List Logical Name (LLN) command to list the components of a logical name.

Finally, format the multifile set by executing the DDL command and entering the logical name of the multifile set as the DB FILE PATHNAME.

6.4 ESTIMATING FILE SIZE

The following formulas provide an estimate of the number of pages required to hold a file. Each page is 256 or 288 bytes of data. Round all sizes with remainders to the next integer. The page size in all examples is 256 bytes. The variable PS is set to the page size; thus, in these examples PS equals 256.

Calculate the size of the data line as follows:

DLB = (10 + LPK + LD + (8 * NSK))

where:

DLB is the data line length in bytes.

LPK is the length of the primary key in bytes.

LD is the length of the longest data line in bytes. The longest data line must include eight bytes for each secondary key specified in that line. Thus, the number of bytes of actual data may not determine the longest data line, since eight bytes of length are added for each secondary reference. See Figure 6-3.

NSK is the number of secondary keys associated with the longest data line.

Calculate the total number of data pages as follows:

TDP = (DLB * NDL) /PS

where:

TDP is the total number of data pages.

DLB is the data line length in bytes.

NDL is the maximum number of data lines specified on the file statement under LINES=.

PS is the page size.

After calculating the key area for each key, calculate the total key area required by adding each of the individual key areas together as follows:

For RANDOM/1

KAP = ((16 + KL) * NK) /PS

For SEQUENTIAL/1

KAP = (NK/((PS - 10)/(12 + KL))) * 2 + 2

where:

KAP is the key area in pages.

KL is the key length in bytes.

NK is the maximum number of unique values allowed for the key.

PS is the page size.

After calculating the key area for each key, calculate the total key area required by adding each of the individual key areas together as follows:

TKP = KAP1 + KAP2 + KAP3 + ...+ KAPn

where:

TKP is the total key area size in pages.

Each file requires one set of description records. The description record contains all of the field and group IDs and the primary key ID specified in the DDL. Each page can contain 20 IDs. The result is the following formula:

DRP = NF/20

where:

DRP is the number of description pages.

NF is the number of field, group, and primary key names specified in the DDL.

Total size of the file is as follows:

TFP = 1 + TDP + TKP + DRP

where:

TFP is the total number of pages in the file.

TDP, TKP, and DRP have already been calculated.

Figures 6-1, 6-2, and 6-3 show these calculations applied to actual DDLs.

```
 DBMS-990     <L.V.R>              DDL TRANSLATOR      MM/DD/YY   HH:MM:SS

FILE=ITEM,LINES=50
ID=ITMN=CH/4,VOL=50,ACCESS=RANDOM/1
*
LINE=01
  FIELD=DESC=CH/20
  FIELD=UPRC=CN/6.3
  FIELD=QTYO=CN/4.0
  FIELD=QTYH=CN/4.0
  ENDL
END.

          TOTAL PAGES REQUIRED - 16
           LINE LENGTH (BYTES) - 48
       TOTAL DESCRIPTION PAGES - 1
                TOTAL KEY PAGES - 4

  LINE 01 --   BASE = 14 , DATA = 34 , LINKAGE = 0   , TOTAL = 48

0112  **  NEW DATA BASE FILE CREATED  **
```

Figure 6-1. ITEM File DDL Listing

For the file ITEM (Figure 6-1), the following calculations apply:

Size of data line:

LPK = 4, LD = 34, NSK = 0
DLB = (10 + 4 + 34) = 48

Number of data pages required:

NDL = 50
TDP = (48 * 50)/256 = 9.4 = 10

Primary key area in pages:

KL = 4, NK = 50
KAP = ((16 + 4) * 50)/256 = 3.9 = 4

Total key pages:

KAP1 = 4, KAP2, KAP3,... = 0
TKP = 4

Description pages required:

NF = 5
DRP = 5/20 = .25 = 1

Total size of file:

TFP = 1 + 10 + 4 + 1 = 16

```
 DBMS-990     <L. V. R>              DDL TRANSLATOR      MM/DD/YY   HH: MM: SS

FILE=SOFL, LINES=300
ID=SONM=CH/6, VOL=50, ACCESS=RANDOM/1
*
LINE=BL
  FIELD=BILL=CH/5
  FIELD=LOCK=CH/2
  ENDL
*
LINE=02
  FIELD=SHIP=CH/5
  ENDL
*
LINE=03
  FIELD=ITEM=CH/4
  FIELD=QUAN=CN/4. 0
  ENDL
*
SECONDARY-REFERENCES
BILL=VOL=50
SHIP=VOL=50
ITEM=VOL=200
END.

          TOTAL PAGES REQUIRED - 71
           LINE LENGTH (BYTES) - 32
       TOTAL DESCRIPTION PAGES - 1
                 TOTAL KEY PAGES - 31

  LINE BL --   BASE = 16 , DATA = 7  , LINKAGE = 8  , TOTAL = 31
  LINE 02 --   BASE = 16 , DATA = 5  , LINKAGE = 8  , TOTAL = 29
  LINE 03 --   BASE = 16 , DATA = 8  , LINKAGE = 8  , TOTAL = 32

0112  **  NEW DATA BASE FILE CREATED  **
```

Figure 6-2. Sales Order File DDL Listing

For the file SOFL (Figure 6-2), the following calculations apply:

Size of data line:

LPK = 6, LD = 8, NSK = 1

DLB = (10 + 6 + 8 + (8 * 1)) = 32

Total number of data pages:

NDL = 300

TDP = (32 * 300)/256 = 37.5 = 38

Primary key area in pages:

KL = 6, NK = 50

KAP1 = ((16 + 6) * 50)/256 = 4.3 = 5

Secondary key area in pages for BILL and SHIP:

KL = 5, NK = 50
KAP2 = KAP3 = ((16 + 5) * 50)/256 = 4.1 = 5

Secondary key area in pages for ITEM:

KL = 4, NK = 200
KAP4 = ((16 + 4) * 200)/256 = 15.6 = 16

Total key area in pages:

KAP1 = 5, KAP2 = 5, KAP3 = 5, KAP4 = 16
TKP = 5 + 5 + 5 + 16 = 31

Description pages required:

NF = 6
DRP = 6/20 = .3 = 1

Total size of file in pages:

TFP = 1 + 38 + 31 + 1 = 71

```
 DBMS-990      <L.V.R.>          DDL TRANSLATOR    MM/DD/YY  HH:MM:SS

FILE=SXXX,LINES=300
ID=SZZZ=CH/6,VOL=50,ACCESS=SEQUENTIAL/1
*
LINE=03
  FIELD-ITXX=CH/2
  FIELD=LOCK=CH/2
  FIELD=GRPX=CH/2
  ENDL
*
LINE=04
  FIELD-NAME=CH/20
  ENDL
*
SECONDARY-REFERENCES
  ITXX=VOL=50
  LOCK=VOL=50
  GRPX=VOL=50
  END.

        TOTAL PAGES REQUIRED - 78
         LINE LENGTH (BYTES) - 46
    TOTAL DESCRIPTION PAGES - 1
            TOTAL KEY PAGES - 22

LINE 03 --   BASE = 16 ,   DATA = 6  ,  LINKAGE = 24 ,  TOTAL = 46
LINE 04 --   BASE = 16 ,   DATA = 20 ,  LINKAGE = 0  ,  TOTAL = 36
```

Figure 6-3. SXXX File DDL Listing

For the file SXXX (Figure 6-3), the following calculations apply:

Size of data line:

LPK = 6, LD = 6, NSK = 3
DAB = (10 + 6 + 6 + (8 * 3)) = 46

Line 03 has less data than line 04, but the number of secondary keys makes line 03 the longest line.

Number of data pages required:

NDL = 300
TDP = (46 * 300)/256 = 53.9 = 54

Primary key area in pages:

KL = 6, NK = 50
KAP1 = (50/((256 - 10)/(12 + 6))) * 2 + 2 = 9.3 = 10

Secondary key area in pages for ITXX:

KL = 2, NK = 50
KAP2 = ((16 + 2) * 50)/256 = 3.5 = 4

Secondary keys LOCK and GRPX have the same length and number of keys as KAP2. Thus, total key pages:

KAP1 = 10, KAP2 = 4, KAP3 = 4, KAP4 = 4
TKP = 10 + 4 + 4 + 4 = 22

Total description pages:

NF = 5
DRP = 5/20 = .25 = 1

Total size of file:

TFP = 1 + 54 + 22 + 1 = 78

6.5 OPTIMUM ORGANIZATION OF DATA BASE FILES

Occasionally, you will need to reorganize a data base file to improve the performance of the DBMS-990. Reorganization restructures the data items in the data base file to minimize the number of times the DBMS interface must access secondary storage to process a single request. Reorganization consists primarily of copying the data base to a sequential file and reloading it back into the data base file.

Data base files have two areas that might become skewed and require reorganization: the data area and the key area. Skew is the condition in which logically contiguous lines are physically

scattered throughout the data base file. The data area becomes skewed in the process of deleting lines from and adding lines to the data base. The organization of the data area is considered optimum when, in traversing all of the lines linked to a primary key, DBMS reads the fewest number of pages. For example, if the average primary key has five lines linked to it, each line is 100 bytes in length, and the data base file has a page size of 256, the best possible size for the data area is two pages; the first two lines and part of the third are on one page, and the remainder of the third line and the last two lines are on the next page. Thus, the smallest number of page reads required will be two. However, if the data area is skewed, each data line might be on a separate page, requiring five page reads to retrieve the entire record; in the worst case, each line would span two pages and require 10 page reads to retrieve the entire record.

Reorganization concerns only the general case where the lines under a primary key are traversed. Usually, it is impossible to provide an optimum organization for more than one key; consequently, reorganization of the data area does not pertain to secondary keys.

6.6 HOW TO REORGANIZE A DATA BASE FILE

The steps for reorganizing a data base file are as follows:

1. Execute CPYFIL (to copy the data base file to a sequential file).
2. Execute RDBF (release the data base file).
3. Execute DF (delete the data base file).
4. Execute DDL (create a new DDL with the new file characteristics).
5. Execute ADBF (assign the data base file).
6. Execute RLDFIL (reload the data from the sequential file into the data base file).

Also, use these steps when expanding a data base file.

6.7 WHEN TO REORGANIZE A DATA BASE FILE

A batch stream can perform the steps of the reorganization, requiring no operator intervention. As a result, in some applications it may be advantageous to reorganize files on a regular basis by executing the batch stream whenever the system has little or no load. However, in applications where this is not practical, reorganize data base files whenever the expected improvement in the performance of DBMS-990 justifies the time required to reorganize the file.

Determining when this point has been reached depends on the application environment; thus, DBMS does not automatically perform the reorganization. Instead, two utilities are provided to generate the statistics necessary to determine when a file reorganization is appropriate. The SUMFIL utility generates statistics concerning the key area for a given file. Use CPYFIL, rather than SUMFIL, to compile statistics about the data area because CPYFIL can compile the statistics while copying the data area. Thus, if the space required for the copy file is available, perform a CPYFIL and obtain the statistics about the data area at the same time. If the results indicate that reorganization would be beneficial, another pass of the data base file is not necessary.

7

Alias Feature

7.1 INTRODUCTION

The alias feature allows the data base user to establish alternate names for the data base elements for use with Query-990. Typically, the alias names are longer and easier to remember. Alias names can be established for the file, primary key, field, or group IDs. Any secondary reference is automatically assigned the alias of its related field or group. Each alias must begin with an alphabetic character and be at least one character in length. The maximum length of an alias name is 20 alphanumeric characters. This optional feature is selectable at DBGEN time and is available for use by the Query user.

7.2 USING ALIASES WITH QUERY-990

Aliases are very useful with Query-990 since the user is not required to have extensive knowledge of the data base. Therefore, aliases serve as a means to easily recognize the data elements when access to the data is desired.

When the alias feature is used with security, access assignment must be made to the DBMS-990 file $AL1 for all passwords that will use the alias file. If the user password is not assigned to the alias file, a security violation results when Query-990 attempts to resolve the alias.

Certain utilities are available to help maintain the alias file. Error messages assist the user in working with the utilities.

7.3 ALIAS UTILITIES

Alias utilities consist of four commands that allow the user to add, modify, delete, and list alias information. When security is generated into the system, the password for the alias file is requested at the beginning of each procedure. When security is not a part of the system, the password prompt does not appear. The basic assumption for the following discussion is that security has been generated into the system and that the password prompt appears in the procedures.

7.3.1 Add Alias (ADDALIAS)

To add aliases, use the Add Alias (ADDALIAS) command. Not all prompts for the command require responses in all cases. You must specify only the alias name and the file ID to add a file alias. Do not specify a line type when adding primary key aliases. For line type aliases, answer all prompts except FIELD ID. For field or group aliases, answer all prompts. The ADDALIAS command has the following prompts:

```
ADD ALIAS ENTRY
                PASSWORD:
              ALIAS NAME:
                 FILE ID:
               LINE TYPE:
                FIELD ID:
```

The responses to the prompts are as follows:

PASSWORD — Enter a password with read or add access authorization to the alias file. (Section 4 describes password authorizations.)

ALIAS NAME — Enter the name associated with the file, primary key, field, or group. The length is from one to 20 alphanumeric characters. The first character must be alphabetic.

FILE ID — Enter the ID of the file for which an alias is assigned or the ID of the file containing the field, group, or primary key.

LINE TYPE — Enter the line type of the line containing the field or group. Leave blank for primary key and file aliases.

FIELD ID — Enter the four-character field, group, or primary key ID. Leave blank for file aliases.

7.3.2 Modify Alias (MODALIAS)

To change an alias entry, use the Modify Alias command (MODALIAS). The MODALIAS command uses the following prompts:

```
MODIFY ALIAS ENTRY
                PASSWORD:
              ALIAS NAME:
                 FILE ID:
               LINE TYPE:
                FIELD ID:
          NEW ALIAS NAME:
```

The responses to the prompts are as follows:

PASSWORD — Enter a password with read, delete, and add access authorization to the alias file.

ALIAS NAME — Enter the name associated with the file, primary key, field, or group. The length is from one to 20 alphanumeric characters. The first character must be alphabetic.

FILE ID — Enter the ID of the file for which an alias is assigned or the ID of the file containing the field, group, or primary key.

LINE TYPE — Enter the line type of the line containing the field or group. Leave blank for primary key and file aliases.

FIELD ID — Enter the four-character field, group, or primary key ID. Leave blank for file aliases.

NEW ALIAS NAME — Enter the new alias name associated with the file, primary key, field, or group.

With this command, the alias name is the only portion of the alias file entry that can be modified. When changing a file alias, do not specify line type or field ID. Do not specify line type when changing a primary key alias. When changing a field or group alias, enter the appropriate information in all prompts. If an alias is assigned to the wrong file, line type, or field/group ID, delete the alias and add it with the correct information.

7.3.3 Delete Alias (DLTALIAS)

The Delete Alias command (DLTALIAS) removes an alias from the alias file. The DLTALIAS command has the following prompts:

```
DELETE ALIAS ENTRY
                PASSWORD:
            ALIAS NAME:
                FILE ID:
            LINE TYPE:
                FIELD ID:
```

The responses to the prompts are as follows:

PASSWORD — Enter a password with read and delete access authorization to the alias file.

ALIAS NAME — Enter the name associated with the file, primary key, field, or group. The length is from one to 20 alphanumeric characters. The first character must be alphabetic.

FILE ID — Enter the ID of the file for which an alias is assigned or the ID of the file containing the field, group, or primary key.

LINE TYPE — Enter the line type of the line containing the field or group. Leave blank for primary key and file aliases.

FIELD ID — Enter the four-character field, group, or primary key ID. Leave blank for file aliases.

To delete a file alias, enter only the alias name and the file ID. When deleting a primary key alias, do not enter anything for line type. For field and group aliases, enter the appropriate information in all prompts.

7.3.4 List Alias (LSTALIAS)

The List Alias (LSTALIAS) command lists an alias to a listing file. The LSTALIAS command uses the following prompts:

```
LIST ALIAS ENTRIES
                    PASSWORD:
                        NAME:
      TYPE(ALIAS, FILE, FIELD):
       LISTING ACCESS NAME:
```

The responses to the prompts are as follows:

PASSWORD — Enter a password with read access authorization to the alias file.

NAME — Enter the alias name, file ID, or field ID desired to be listed.

TYPE(ALIAS,FILE,FIELD) — Enter the type of name specified.

LISTING ACCESS NAME — Enter the listing file pathname or a device name.

If you specify an alias name for the name prompt, enter ALIAS for the TYPE prompt. In this case, one listing contains all of the aliases of the same name that are associated with the various files, lines, and fields. Figure 7-1 shows an example of an alias listing for the alias ADDRESS with the corresponding command. Note that the distinguishing element of the aliases is the file ID.

```
LIST ALIAS ENTRIES
                    PASSWORD:
                        NAME:  ADDRESS
      TYPE(ALIAS,FILE,FIELD):  ALIAS
      LISTING ACCESS NAME:     LP01
```

```
                LISTING OF ALIAS INFORMATION

FILE        LINE        FIELD        ALIAS

CUST        01          ADDR         ADDRESS
MARK        01          ADDR         ADDRESS
ELIZ        01          ADDR         ADDRESS
RAND        01          ADDR         ADDRESS
RONA        01          ADDR         ADDRESS
```

Figure 7-1. Example Listing of Fields with the Same Alias

If you specify a file ID for the name prompt, enter FILE for the type prompt. In this case, the listing contains all of the aliases associated with the fields in the file. Figure 7-2 is an example of a command that produced the listing that contains all of the aliases for the file CUST.

```
LIST ALIAS ENTRIES
                  PASSWORD:
                      NAME:  CUST
TYPE(ALIAS,FILE,FIELD):  FILE
LISTING ACCESS NAME:  LP01

                LISTING OF ALIAS INFORMATION

FILE          LINE          FIELD          ALIAS

CUST          01            ADDR           ADDRESS
CUST          01            SSSS           SA
```

Figure 7-2. Example File Alias Listing

If you specify a field ID for the name prompt, the listing contains all aliases associated with the field. In this case, multiple files have one field. Figure 7-3 is an example of the prompts for listing the field ID ADDR and its associated aliases. The bottom half of Figure 7-3 is the corresponding listing generated from the command.

```
LIST ALIAS ENTRIES
                  PASSWORD:
                      NAME:  ADDR
TYPE(ALIAS,FILE,FIELD):  FIELD
LISTING ACCESS NAME:  LP01

                LISTING OF ALIAS INFORMATION

FILE          LINE          FIELD          ALIAS

CUST          01            ADDR           ADDRESS
MARK          01            ADDR           COLORADO
ELIZ          01            ADDR           ADDRESS
RAND          01            ADDR           ADDRESS
RONA          01            ADDR           ADDRESS
```

Figure 7-3. Example Field Alias Listing

7.4 ALIAS ERROR MESSAGES

Error messages for the alias procedures are listed and described in Appendix C.

Appendix A

Key Retrieval Routines

A.1 OVERVIEW

This appendix supplies some background information about the DBMS retrieval techniques. The two types of retrieval routines are random and sequential. For each key defined in a data base system, users can choose either routine. Only those routines that are to be used by an installation are included during data base generation. The routines selected are automatically linked into the DBMS task segment.

Evaluating algorithms for use in a storage/retrieval routine shows that no single routine is better than the others. In applications where key order is not important but speed in update and retrieval is, a hashing algorithm is probably best. However, the characteristics of hashing techniques vary, depending on the type and size of the data and the order in which data is entered. In applications where key order is important, factors such as the ratio of modifications to retrievals, and memory size and disk space affect which storage/retrieval method will function most efficiently.

Rather than arbitrarily picking one retrieval algorithm to implement key storage/retrieval, DBMS-990 is designed so that users can select random or sequential storage/retrieval routines.

A.2 SEQUENTIAL KEY ROUTINE: S1

For sequential access, a B tree (or variation of a B tree), is preferable because a large set of indexes can be searched and maintained using fairly simple algorithms. The S1 routine of DBMS-990 is a variation on the B tree described by Donald E. Knuth (in *The Art of Computer Programming*, Volume 3, Reading, Mass.: Addison-Wesley, 1973).

A.3 RANDOM KEY ROUTINE: R1

The random retrieval routine R1 of DBMS-990 uses a hashing algorithm that stores the keys for quick retrieval. This method affects the reorganization of data base files. The hashing algorithm converts the key value into an offset into the key area. Searching is not normally required to find a key. However, when two different key values hash to the same offset (known as a collision), the second key is chained to an overflow area. Thus, when a reference is made to this key, the DBMS-990 must compute the offset into the key area and then follow the chain until the key is found. Since this is unnecessary overhead, it is helpful to know how many chains exist, the length of the longest chain, and the average length of a chain. When the number of chains or the chain length reaches an unacceptable level, reorganize the data base file and increase (or possibly decrease) the key space. To assist in determining when the number or length of chains is unacceptable, the SUMFIL utility (see Section 5) provides statistics regarding the number of collisions, the number of key chains, the longest key chain, and the average chain length.

Appendix B

Alternate Collating Sequences

For installations using sequential data retrieval, DBMS-990 supports collating sequences other than the standard USASCII collating sequence. The current release of DBMS-990 includes two alternate collating sequence files. These are the collating sequences for the Germany/Austria and the Sweden/Finland character sets. These collating sequences are compatible with the operating system's internal support for key-indexed files (KIFs). You can define another collating sequence as follows:

1. During DBGEN, after selecting the sequential retrieval routine, you are prompted for the pathname of the file that defines the collating sequence (the default file contains the USASCII sequence). Enter the file pathname that you will use to define the alternate sequence. (You can install the file under the directory .S$DBMS.)

2. Perform that data base installation (DBINS) as usual after the DBGEN.DBINS creates the directory .S$DBMS.

3. Finally, use the format described below to create a file that defines the desired collating sequence, and install the file on .S$DBMS.

The format for the collating sequence file has been strictly defined. The file contains eight records, and each record contains eight pairs of characters, left justified. You must define alternate character pairs for all of the characters in the set (40 through 70).

The file .S$DBMS.ALTSEQGA contains the collating sequence for the Germany/Austria character set; the file .S$DBMS.ALTSEQSF contains the collating sequence for the Sweden/Finland character set. The following is an example of the collating sequence for the Germany/Austria character set:

```
404041415B4242434344444545464647        GERMAN/AUSTRIAN
4748484949A4A4B4B4C4C4D4D4E4E4F         TO U.S. ASCII
4F505C515052515352545355545655 7        ALTERNATE COLLATING
5D585659575A585B595C5A5D5E5E5F5F        SEQUENCE TABLE
606061617B6262636364646565666667
67686869696A6A6B6B6C6C6D6D6E6E6F
6F707C7170727173727473757E767477
75787D79767A777B787C797D7A7E7F7F
```

You can use the right half of each record for comments.

To implement an alternate collating sequence after your data base is generated, you can specify an alternate collating sequence to the SDBMS SCI command processor. Use the Text Editor to modify the synonym $DBMSA in the SDBMS command procedure, pathname S$CMDS.SDBMS. If you do not specify a value for this synonym, the USASCII collating sequence is used.

SDBMS issues an error if the specified file does not exist or is not in the proper format.

Appendix C

DBMS Error Messages and Codes

C.1 INTRODUCTION

DBMS-990 errors consist of the following:

- DML errors
- Utility errors
- DBMS errors

The status code of the control block parameter reports DML errors. Utility errors are those that DBMS-990 utilities encounter.

C.2 DML ERRORS

Check the status parameter of the DML call control block after each call. If the status parameter contains anything other than asterisks, an error condition has occurred.

Table C-1 lists and explains the error codes that appear in the status area.

C.3 DBMS ERRORS

DBMS error messages are in the following form:

nnnn <message>

The nnnn is the DBMS message number.

Table C-2 lists the utility error codes. Table C-3 lists the DBMS error messages and explanations. Table C-4 shows the internal error codes and the corresponding message numbers. This is useful on systems that do not support expanded message files. Use the internal code to find the message number. Then, look up the message number in Table C-3 to find the explanation.

Table C-1. Error Codes

Code	Type of Error	Probable Cause
AC	Access error	An attempt has been made to access a file using an improper access type (i.e., an update attempt to a file opened with ROEX access), to close a file that has not been opened, or to open a file with an undefined access type.
AE	Address error	The user has supplied an invalid address in loc1 or loc2. This error is possible on all functions if the user has accidentally altered loc1 or loc2. However, it is most likely to occur on a read forward (RF) function. Check the program logic for possible modifications to loc1 or loc2.
AS	Add error	The user is attempting to execute an add command on a line type 01 with loc1 not set to asterisks.
BF	Bad file	The data base file has a bad internal pointer or address. This can result if a system failure occurred while the file was being modified and the file was not recovered. Rebuild the file by using CPYFIL/RLDFIL.
BP	Bad pathname	The pathname supplied for the log file access name was too long.
DA	Delete asterisks	A delete record (DR) function has been specified and loc1 does not contain asterisks.
DB	Data base error	The data base is not running.
DF	Duplicate file	An attempt has been made to assign a file ID that has already been assigned.
DL	Deadlock	The current transaction has been rolled back.
DU	Duplicate	The user is attempting to add multiple type 01 lines to the same data record.
FA	Find asterisks	The asterisks at the end of the line list were not found. This can result if the asterisks do not start on a word boundary or if they are not the correct length.
FB	File buffers	No file buffers exist to open the file. If the error occurred during an ADBF command, the size specified in response to the prompt MAXIMUM ASSIGNED FILES in the SDBMS command was not large enough. If the error occurred during an OF function, the size specified in response to MAXIMUM OPEN FILES was not large enough. Wait until a file is closed, or restart DBMS-990 and specify a larger number of file buffers.

Table C-1. Error Codes (Continued)

Code	Type of Error	Probable Cause
FE	Field error	The user has specified an invalid field or group ID(s) in the parameter list. DBMS-990 cannot find the field or group ID. Verify the spelling in the file definition and calling program. Verify that the correct line type is specified.
FH	Full hold buffer	The internal buffer that DBMS-990 uses to register the held lines is full. This results when too many tasks are holding lines; it can also occur when a number of tasks terminate without releasing held lines, thus filling the table.
FL	Full	The area reserved for data lines is full. This error can occur during an add after (AA) or add before (AB) function. Delete any unnecessary records or create a larger file and copy the data to the new file by using CPYFIL/RLDFIL.
FN	Function error	An invalid function code was specified. The function passed to DBMS-990 in the control block is not defined.
FS	File reset	A UDBF was performed on a file that had previously been unlocked.
FU	File in use	Current file in use and not available at this time.
GF	Good file	A UDBF was performed on a file that was not locked.
HL	Hold line error	A hold line (HL) function has been attempted with loc1 set at asterisks.
IE	Invalid entry ID	The key ID specified is not the primary or secondary key. For an add or delete function, the key ID must be the primary key.
IG	Invalid group	The group ID specified for a query group (QG) function is not a group.
II	Invalid item	The user is not authorized to perform the function against a data item specified in the call.
IK	Invalid key	Either the data line to which loc1 points does not contain the same value as the key value given in the control block, or an attempt has been made to execute a read on a secondary key that does not exist in the line specified.
IL	Invalid line	The specified line type does not contain the specified field.
IO	I/O error	The operating system encounters an I/O error that occurred during a read or write to the disk.
IT	Invalid transaction	Transactions are not properly bracketed by TS, TC, and/or TR.

Table C-1. Error Codes (Continued)

Code	Type of Error	Probable Cause
KF	Key area full	Key area for specified key ID is full for an add after (AA) or add before (AB) function.
KU	Key update error	An attempt is being made to alter the value of a primary or secondary key with the write (WT) function. To alter a key value, delete the line and then reenter it with the new value included.
LA	Line asterisk	A multiple line type specification has been passed to DBMS-990 but no asterisk was found for a write (WT) or delete (DL) function.
LB	Lock tables full	Lock tables are full due to keys that are too large.
LE	Line error	DBMS-990 has received an invalid line list. The "LINE=" syntax cannot be located. Thus the field or group IDs cannot be found.
LF	Log full	The log file is full. Use CLLOG to close the file, define a new file, and then use OPLOG to open the new file.
LL	Lock line error	A lock line (LL) function has been attempted with loc1 set to asterisks.
LO	Log error	A log input/output error has occurred.
L1	Line=01 error	An attempt is being made to add a line type other than line 01 before a line type 01 has been added. Check the program logic. A line type 01 must exist before any other line type can be added to the record if a LINE=01 is specified in the DDL.
NB	No buffer	Not enough buffers are available to facilitate the required operation. This is a temporary condition. The size specified in response to the prompt MAXIMUM USERS in the SDBMS command was not large enough. Either wait until a buffer is free, or stop DBMS-990, increase the maximum number of users, and then restart DBMS-990.
NF	No file	DBMS-990 cannot find the file ID specified in the file command. The file ID may be misspelled, or it may have been released. Verify that the file ID is assigned and is spelled correctly.
NH	No hold	An attempt is being made to delete a line that has not been held. Prior to deleting a line, the read with hold option must have been specified. The only line available for the delete (DL) is the last one read; with the hold option, the task can only hold one line at a time.

Table C-1. Error Codes (Continued)

Code	Type of Error	Probable Cause
NK	No key found	DBMS-990 cannot find the primary or secondary key value for a read forward (RF) or a read backward (RB) function. Check the program logic and input data.
NL	No logging	DBMS-990 has back-up logging installed, but a log file has not been opened.
OA	Open assign LUNO error	An operating system error occurred while the user was trying to assign a LUNO to the file. Verify that the file exists.
OE	Open error	An operating system error occurred while trying to open the file. Note that DBMS-990 was successful in assigning a LUNO to the file but could not open it, implying that the file is already in use, or that the file was not created by the DDL translator.
OL	Open log error	The log must be either a magnetic tape, cassette, or sequential file. The type found is none of these, or the logical record length is too small.
ON	Open name	The file ID specified does not match the file ID internally stored in the file of the pathname specified.
PF	Preimage buffer full	The value given for the MAX LINE IMAGES at DBGEN is exceeded. Reduce the size of the transactions.
RL	Record length	The file being assigned has a record length that is not a valid page size. This file was not created by the DDL translator.
R1	ROEX access error	An attempt has been made to open a file with SHRD or EXCL access while the file is already open with ROEX access, or the same task is trying to execute multiple opens with ROEX access.
SV	Security violation	The user has entered an invalid password and is not authorized to use a data item specified in the call.
S1	SHRD access error	An attempt has been made to open a file with EXCL or ROEX access while the file is already open with SHRD access, or the same task has already opened the file with SHRD access.
UF	Undefined field	The field name in the line list is not defined in the DDL.
UL	Undefined line	The DDL does not define the line type specified for this file.
UT	Undefined key type	An attempt has been made to access a data base file with a key type that was not included in DBMS-990 during DBGEN.

Table C-1. Error Codes (Continued)

Code	Type of Error	Probable Cause
WF	Wrong file	All or part of a line that was encoded by a previous call to DBMS-990 has been used in a subsequent call, but the file ID was changed and the new file specified does not contain one of the fields.
XX	Call error	The DML call parameter list is too large for the interface buffer. Possible causes include the following: the call has too many parameters; the wrong parameter list has been sent; or the buffer size allocated during DBGEN is not large enough for the parameter list.
X1	EXCL error access	An attempt has been made to open a file with EXCL, SHRD, or ROEX access when the file is already open with EXCL access.
01	Line = 01 error	An attempt is being made to delete a line type 01 when other line types for the key still exist. A line type 01 cannot be deleted until all other line types for the key have been deleted.

Table C-2. Utility Error Codes

Error Code	Meaning
ABORT	RECOVR forced to abort (entered Q).
BADDAT	A log record has an invalid date and time stamp. If the log file is being used to recover from an operating system crash, BADDAT might signal the end of a log file that has no EOF, instead of signaling an error condition.
BADFIL	The DB FILE ID specified does not match the file ID of the DB FILE PATHNAME specified.
BADKEY	Primary key ID must be the same for the copy file and the DBMS file.
BADLOG	Log file pathname cannot be opened or contains inconsistent data.
BADPSW	Invalid password entry.
BADSFL	The specified copy file cannot be opened or is not of the proper type.
ERROR	A command function name is not valid.
FERROR	A disk I/O operation exceeded the bounds of the file, and all range checks passed. (An operating system error, indicated by hexadecimal 30, occurred.)
FILNTF	The specified pathname does not exist. (An operating system error, either hexadecimal 21 or 27, occurred.)
FNAME	The file ID specified for RLDFIL does not match the copy file.
INVTYP	Invalid type or field change in RLDFIL.
LINYP	The number of line types that CPYFILE found exceeds the maximum allowed.
LOGI/O	I/O error trying to read log file.
NO OUT	The output listing file cannot be opened.
WRPROT	The utility cannot write to the file. (An operating system error, hexadecimal 1A, occurred; the disk drive is write protected.)

When the utility error code received ends in two asterisks, an error is returned to the utility indicated by the middle two characters of the utility error code. Refer to Table C-1 for more information.

Table C-3. DBMS Error Messages

U	DBMS-0001	ERROR ON OPEN OR CLOSE DBMS FILE, STATUS = ?1 Explanation: A file access error status was returned. This is a result of some other user having exclusive (X1 status) or read-only exclusive (R1 status) access to the file. User Action: Retry when file exclusive or read-only exclusive access privileges are released from the file.
U	DBMS-0002	FIELDS ARE ON DIFFERENT LINES, BAD FIELD = ?1 Explanation: All the output fields must be from the same line. The field identified as the bad field is contained in a different data line from the previous field(s). User Action: Retry excluding the bad field ID.
U	DBMS-0003	ILLEGAL FUNCTION "?1", MUST BE "RS","RF", OR "RB" Explanation: The function entered is not a legal function for the PQUERY utility. User Action: Retry using a valid function, must be 'RS','RF', or 'RB'.
U	DBMS-0004	INVALID KEY VALUE Explanation: The value entered for the primary key field does not exist. User Action: Retry with a key value that exists.
U	DBMS-0005	NO FIELD OR GROUP ID SPECIFIED Explanation: No field or group names were entered for the FIELD IDS prompt. User Action: Retry using valid field(s) and/or group(s) ID for prompt.

Table C-3 DBMS Error Messages (Continued)

U	DBMS-0006	STATUS EXCEPTION FROM DBMS, STATUS = ?1 Explanation: The data base manager returned the error status defined in the return message when PQUERY tried to process the request. User Action: Consult the error code table to determine the reason for the error status.
U	DBMS-0007	UNABLE TO OPEN FILE, DBMS STATUS = ?1 Explanation: PQUERY could not open the file with shared access. Another task has the file open with either exclusive or read-only exclusive. User Action: Retry operation when PQUERY can get access to the file.
U	DBMS-0008	UNABLE TO OPEN LISTING FILE Explanation: PQUERY utility could not open the listing file requested. User Action: Probably caused by an invalid pathname, does not exist or is a directory name. Another possibility is there is not enough disk space for the listing file.
U	DBMS-0009	UNDEFINED FIELD NAME "?1" Explanation: The field name given in the message is not defined for the file ID entered. User Action: Enter correct field name and retry.
U	DBMS-0010	UNABLE TO OPEN THE LISTING FILE, SVC ERROR ?1 Explanation: The data base utility could not open the requested listing file. The error could be caused by an invalid pathname, a full directory, or a full disk space. User Action: Refer to the SVC error code for the exact cause of the error and respond accordingly.

Table C-3 DBMS Error Messages (Continued)

U	DBMS-0011	DBMS UTILITY ERROR: ?1 Explanation: The DBMS-990 utility detected an error during processing. User Action: See the table containing the utility error codes for further explanation.
U	DBMS-0012	CANNOT CLOSE SECURITY FILE Explanation: An error was received from closing the security file. User Action: Possible hardware error.
U	DBMS-0013	CANNOT GET TCA FILE Explanation: An error occurred when trying to access the TCA region. User Action: Possible hardware error.
U	DBMS-0014	CANNOT GET OPEN SECURITY FILE Explanation: An error was received when opening the security file. User Action: Possible hardware error. Another possibility is that another task has the file open with access privileges that conflict.
U	DBMS-0015	ERROR IN SYNONYM ASSIGNMENT Explanation: An error was received when trying to assign a synonym. User Action: Delete some synonyms and retry the operation.
U	DBMS-0016	ERROR WHEN ASSIGNING LUNO TO SECURITY FILE Explanation: An error was received when trying to assign a LUNO to the security file. User Action: Verify the security file exists.

Table C-3 DBMS Error Messages (Continued)

U	DBMS-0017	ERROR WHEN READING SECURITY FILE Explanation: An error was received when trying to read the security file. User Action: Possible hardware error.
U	DBMS-0018	ERROR WHEN RELEASING LUNO FOR SECURITY FILE Explanation: An error was received when trying to release a LUNO assigned to the security file. User Action: Possible hardware error.
U	DBMS-0019	INVALID MASTER PASSWORD Explanation: The password entered is not the correct master password. User Action: Retry with the correct master password.
U	DBMS-0020	INVALID DATA FORMAT CONVERSION FOR RLDFIL Explanation: RLDFIL does not support conversion of the data for format specified. User Action: Do not use RLDFIL for changing the format for data types that are not supported by RLDFIL.
U	DBMS-0021	DBMS FILE IS NOT EMPTY Explanation: The file into which the data is to be loaded is not empty. The reload does not occur. User Action: Format the file with the DDL translator before trying the reload.

Table C-3 DBMS Error Messages (Continued)

U	DBMS-0022	INVALID FIELD CONVERSION IN RLDFIL Explanation: RLDFIL encountered a field conversion request that is invalid. User Action: Refer to the documentation for RLDFIL for the valid type changes. Compare the DDL for the file that CPYFIL copied to a sequential file against the DDL for the data base file that is being reloaded. Change the invalid conversion to be valid.
U	DBMS-0023	FIELD DROPPED IN NEW DDL — ?1 Explanation: The field was found in the old DDL but is not in the new DDL. User Action: Verify that the field should not be in the new DDL.
U	DBMS-0024	?1 LINE TYPE MISSING IN NEW DDL Explanation: Line identifier is missing in new DDL. This message occurs if the old DDL contains a line type not in the new DDL. User Action: Verify that the line type should not be in the new DDL.
U	DBMS-0025	POSSIBLE LOSS OF SIGNIFICANCE IN FIELD — ?1 Explanation: This message results if the field in the old DDL has more significant digits than the field declared in the new DDL. User Action: Verify that loss of significance is valid.
U	DBMS-0026	TYPE TRANSFER FOR RPG DATA NOT IMPLEMENTED Explanation: RPG data types are not supported by the RLDFIL utility. User Action: None.

Table C-3 DBMS Error Messages (Continued)

U	DBMS-0027	UNABLE TO OPEN THE INPUT FILE, SVC ERROR ?1 Explanation: The DDL translator could not open the requested input file. The error could be caused by an invalid pathname or by specifying a file that does not exist. User Action: Refer to the SVC error code for the exact cause of the error and respond accordingly.
U	DBMS-0028	CANNOT OBTAIN TCA FILE, SVC ERROR ?1 Explanation: The data base utility could not obtain the TCA file. User Action: Refer to the SVC error code for the exact cause of the error and respond accordingly.
U	DBMS-0029	ERROR ON READ OF INPUT FILE, SVC ERROR ?1 Explanation: The DDL translator has detected an error on a read from the input file. User Action: Refer to the SVC error code for the exact cause of the error and respond accordingly.
U	DBMS-0030	DATABASE MANAGER ALREADY RUNNING Explanation: The data base manager is already running. User Action: An EDBMS command must be issued prior to reissuing the SDBMS command.
U	DBMS-0031	INVALID OR MISSING PARAMETER Explanation: A parameter required by the data base is either missing or invalid. User Action: Verify that the parameters used in the bid are correct.

Table C-3 DBMS Error Messages (Continued)

U	DBMS-0032	UNABLE TO BID DATABASE MANAGER. SVC ERROR: 2B?1 Explanation: An SVC error occurred during the bid task. User Action: Refer to the SVC error code for the correct action.
U	DBMS-0033	DBMS UTILITY ERROR: ?1 Explanation: An error was detected during the execution of the DBMS-990 utility. The six-character code defines the type of error that occurred. User Action: Look up the six-character code in the Utility Error Codes table. Determine the cause of the error. Retry the operation.
U	DBMS-0034	INVALID FUNCTION Explanation: The SECFUNC task was bid with an invalid function code. User Action: Check to see that the SCI procedure has not been altered.
U	DBMS-0035	INVALID PATHNAME Explanation: An invalid pathname was entered. User Action: Retry using a valid pathname.
U	DBMS-0036	INVALID STATUS CODE: ?1 Explanation: An error status code was returned by the data base manager. User Action: Refer to the DBMS-990 status codes for further information.
U	DBMS-0037	INVALID FILE Explanation: The file ID specified is longer than four characters. User Action: Retry using a maximum of four characters for a file ID.

Table C-3 DBMS Error Messages (Continued)

U	DBMS-0038	ERROR ON WRITE TO LISTING FILE, SVC ERROR ?1 Explanation: The DDL translator has detected an error on a write to the listing file. User Action: Refer to the SVC error code for the exact cause of the error and respond accordingly.
U	DBMS-0039	ERRORS DETECTED DURING DDL TRANSLATION Explanation: The DDL translator has detected errors during translation. User Action: Check the listing file for the specific location of syntax errors and for any semantic errors. Make the appropriate revisions and resubmit the DDL.
U	DBMS-0040	STATUS CODE: ?1 Explanation: An invalid status was returned by the data base manager. User Action: Refer to DBMS-990 status codes for further information.
U	DBMS-0041	CANNOT GET ACCESS Explanation: Another task has exclusive (or read-only exclusive) access to the file specified. User Action: Retry the operation when the file has been released from exclusive access.
U	DBMS-0042	CANNOT GET PARAMETER Explanation: An error occurred when trying to read bid parameter. User Action: The task was bid with incorrect number of parameters. Possible hardware error.

Table C-3 DBMS Error Messages (Continued)

U	DBMS-0043	CANNOT OPEN FILES Explanation: The maximum number of open files has been reached or another task has the security or alias files open with access privileges that conflict. User Action: Restart the data base manager (SDBMS) with a larger maximum number of open files. If another task has the files open then retry later.
U	DBMS-0044	CODE CONFLICT Explanation: The entry to be added is not a subset of the authorization of the higher-level entry, or the item to be added does not have delete authority but the associated line does have delete authority. User Action: Resolve the conflict and retry.
U	DBMS-0045	DATABASE DOWN Explanation: The data base is not up; it has not been started. User Action: Start the data base with SDBMS command.
U	DBMS-0046	DUPLICATE FILE Explanation: Tried to add a file that has already been assigned to the password. User Action: Verify access authorizations for the file assigned to the password.
U	DBMS-0047	DUPLICATE ITEM Explanation: Tried to add an item that has already been assigned to the password. User Action: Verify access authorizations for the item assigned to the password.

Table C-3 DBMS Error Messages (Continued)

U	DBMS-0048	DUPLICATE LINE Explanation: Tried to add a line that has already been assigned to the password. User Action: Verify access authorization for the line assigned to the password.
U	DBMS-0049	DUPLICATE PASSWORD Explanation: Tried to add a password that has already been assigned. User Action: Verify password exists.
U	DBMS-0050	ENTRY AREA FULL Explanation: Attempted to add password/alias entry when the password/alias entry area is full. User Action: Expand the disk data area. Users should consult the DBA. The DBA should copy the security/alias file, depending on which file is full, using the CPYFIL utility. The security file ($SC1)/alias file ($AL1) is located in the SC1/AL1 node of the data base library directory. Redo the DBINS procedure specifying new security/alias file. Increase the number of security/alias entries and reload the copied data.
U	DBMS-0051	FILE NOT FOUND Explanation: The file ID specified does not have authorization. The file ID specified for deletion does not exist for the password. User Action: Must first give authorization to the file. Retry deletion with the valid file ID for the password.
U	DBMS-0052	INVALID CODE Explanation: Error when entering the authorizations for the entry. User Action: Reenter the command.

Table C-3 DBMS Error Messages (Continued)

U	DBMS-0053	INVALID LISTING ACCESS NAME Explanation: The listing access name is not valid. User Action: Retry using a valid listing access name. Check available space in directory and/or disk volume.
U	DBMS-0054	INVALID FUNCTION Explanation: The task was bid with a function that is not defined. User Action: Check to see that the SCI procedure has not been altered.
U	DBMS-0055	INVALID ITEM Explanation: Item ID is longer than four characters. User Action: Retry using four characters or less for the item name.
U	DBMS-0056	INVALID LINE Explanation: The line ID specified is longer than two characters. User Action: Retry using two characters for the line ID.
U	DBMS-0057	INVALID MASTER Explanation: Master password is longer than four characters. User Action: Retry using four characters or less for master password.
U	DBMS-0058	INVALID MAX VALUE Explanation: When creating data base security, the MAX PASSWORDS or MAX ENTRIES prompt value was greater than 999999. User Action: Redo the DBINS process specifying values less than 999999 for the MAX PASSWORDS and/or MAX ENTRIES prompt.

Table C-3 DBMS Error Messages (Continued)

U	DBMS-0059	INVALID PASSWORD Explanation: The password specified does not have access authorization or has not been assigned. Also, the password specified is longer than four characters. User Action: Verify the validity and/or authorization of the password.
U	DBMS-0060	INVALID TYPE Explanation: Type is not FILE, LINE or ITEM. User Action: Retry using a valid type.
U	DBMS-0061	ITEM NOT FOUND Explanation: The item ID specified does not exist. User Action: Retry with different item ID.
U	DBMS-0062	LINE NOT FOUND Explanation: The line ID specified does not have authorization. The line ID specified for deletion does not exist for password. User Action: You must first give the authorization to the line. Retry with a valid line ID for the password.
U	DBMS-0063	NO BUFFERS Explanation: No open file buffers. Other tasks are already using the buffers. User Action: Wait until buffers are available.

Table C-3 DBMS Error Messages (Continued)

U	DBMS-0064	PASSWORD AREA FULL Explanation: The maximum number of passwords specified in DBINS has been exceeded. User Action: Expand the disk data area. Users should consult the DBA. The DBA should copy the security file using the CPYFIL utility. The security file ($SC1) is located in the SC1 node of the data base library directory. Redo the DBINS procedure specifying new security file. Increase the number of passwords and reload the copied data.
U	DBMS-0065	PASSWORD NOT FOUND Explanation: The password used is not a valid password. User Action: Verify the correct password was used.
U	DBMS-0066	DB FILE NOT ASSIGNED Explanation: An attempt was made to access the security or alias files without the files being assigned. User Action: Stop the data base using the EDBMS command. Restart the data base using the SDBMS command.
U	DBMS-0067	ALTERNATE COLLATING SEQUENCE SPECIFIED AND S1 ROUTINES WERE NOT INCLUDED Explanation: A pathname for an alternate collating sequence was specified as a bid parameter to the data base manager that was generated without sequential keys. Alternate collating sequences are only valid for the ordering of sequential keys; thus a data base manager generated without sequential keys cannot use the alternate collating sequence file. User Action: Remove the alternate collating sequence pathname from the bid parameter.

Table C-3 DBMS Error Messages (Continued)

U	DBMS-0068	UNABLE TO ASSIGN LUNO TO ALTERNATE COLLATING SEQUENCE FILE Explanation: The pathname defined for the alternate collating sequence file is invalid. User Action: Change the pathname used for the alternate collating sequence to be valid or null.
U	DBMS-0069	UNABLE TO OPEN ALTERNATE COLLATING SEQUENCE FILE Explanation: An error occurred when the data base manager attempted to open the alternate collating sequence file. User Action: Verify the validity of the pathname in the bid parameter for the alternate collating sequence file.
U	DBMS-0070	UNABLE TO READ ALTERNATE COLLATING SEQUENCE FILE Explanation: An error occurred when the data base manager attempted to read the alternate collating sequence file. User Action: Verify the validity of the pathname in the bid parameter for the alternate collating sequence file.
U	DBMS-0071	UNABLE TO OPEN THE DATABASE FILE, SVC ERROR ?1 Explanation: The DDL translator could not open the requested data base file. The error could be caused by an invalid pathname, a full directory, or a full disk space. User Action: Refer to the SVC code for the exact cause of the error and respond accordingly.

Table C-3 DBMS Error Messages (Continued)

U	DBMS-0072	** AN EQUAL SIGN (' = ') WAS EXPECTED ** Explanation: The DDL translator expected an equal sign in the location indicated by the up arrow. User Action: Correct the syntax and resubmit the DDL.
U	DBMS-0073	** INVALID LINE/VOLUME VALUE ** Explanation: A zero (0) has been entered for the value of a line or volume. User Action: Change the line or volume to a nonzero value and resubmit the DDL.
U	DBMS-0074	**INVALID CHARACTERS IN ID ** Explanation: A four character ID has been entered that contains an invalid character. The invalid character has been flagged with an up arrow. An ID may contain only alphanumeric characters, numeric characters, dollar signs ($), and blanks. An ID must start with a dollar sign or an alphanumeric character and may not contain embedded blanks. User Action: Correct the invalid character and resubmit the DDL.
U	DBMS-0075	** A COMMA (',') WAS EXPECTED ** Explanation: The DDl translator expected a comma in the location indicated by the up arrow. User Action: Correct the syntax and resubmit the DDL.

Table C-3 DBMS Error Messages (Continued)

U	DBMS-0076	** KEYWORD EXPECTED ** Explanation: The DDL translator expected a keyword before the location indicated by the up arrow. User Action: Verify the entry is a valid keyword and resubmit the DDL.
U	DBMS-0077	** A NUMERIC VALUE WAS EXPECTED ** Explanation: The DDL translator expected a numeric value before the location indicated by the up arrow. User Action: Verify the entry is a valid numeric value and resubmit the DDL.
U	DBMS-0078	** A FORMAT WAS EXPECTED AFTER THIS DATA TYPE ** Explanation: The DDL translator expected a format after the data type specified in a field definition line. User Action: Add a valid format to the data type and resubmit the DDL.
U	DBMS-0079	** INVALID DATA TYPE LENGTH ** Explanation: An invalid data type length has been entered. A CX or RD field must be 8 bytes long, an RS or ID field must be 4 bytes long, and an LG or IS field must be 2 bytes long. An FX field must have a length of 2 bytes with between 0 and 16 bits. A PK, AN, AS, CN, or CS data type must have a length of between 1 and 18 bytes. User Action: Verify the data length is valid and resubmit the DDL.

Table C-3 DBMS Error Messages (Continued)

U	DBMS-0080	** A PERIOD ('.') WAS EXPECTED ** Explanation: The DDL translator expected a period in the location indicated by the up arrow. User Action: Correct the syntax and resubmit the DDL.
U	DBMS-0081	** MAXIMUM LINE/VOLUME COUNT EXCEEDED — ?1 ** Explanation: The number specified for the line/volume count is greater than that allowed by the DDL translator. User Action: Revise the number and resubmit the DDL.
U	DBMS-0082	CANNOT FORMAT DATABASE FILE WITH PATHNAME SPECIFIED Explanation: The file specified for the new data base file already exists and does not match the characteristics of the file to be created. The preexisting file must be a relative record file with the same physical and logical page size as the file to be created, and both files must have the same number of records. User Action: Choose another pathname for the data base file or delete the file that currently resides at the pathname specified.
U	DBMS-0083	ALIAS AREA FULL Explanation: Attempted to add an alias when the alias area was full. User Action: Expand the disk data area. Users should consult the DBA. The DBA should copy the alias file using the CPYFIL utility. The alias file ($AL1) is located in the data base directory in the AL1 node. Redo the DBINS procedure specifying new alias file. Increase the number of alias entries and reload the copied data.

Table C-3 DBMS Error Messages (Continued)

U	DBMS-0084	ALIAS NOT FOUND Explanation: The alias specified does not exist. User Action Check for spelling error and try again.
U	DBMS-0085	DUPLICATE ENTRY FOR ALIAS Explanation: The specified alias name is already assigned. User Action: Select another alias name.
U	DBMS-0086	ERROR WHEN OPENING LISTING FILE Explanation: An error was received when trying to open the listing file. User Action: Verify the validity of the listing file access name. Check to ensure that the file can be created with access name specified.
U	DBMS-0087	ERROR WHEN WRITING TO LISTING Explanation: An error was received when writing the listing file. User Action: Check to ensure listing file device is ready. If listing access name is a disk then check that space is available for the file.
U	DBMS-0088	FILE NOT DEFINED FOR ALIAS Explanation: The specified alias is not assigned to the specified file. User Action: Verify the correct alias and file and retry the operation.
U	DBMS-0089	ILLEGAL ALIAS NAME Explanation: An alias name must begin with an alphabetic character; be 20 characters or fewer in length; and consist of alphanumeric, dollar sign ($), dash (-), or underscore characters only. User Action: Modify the alias name to fit the description above.

Table C-3 DBMS Error Messages (Continued)

U	DBMS-0090	ILLEGAL FIELD FOR ALIAS Explanation: The specified field does not have access authorization or has not been assigned. User Action: Verify the field ID and retry the operation.
U	DBMS-0091	LINE TYPE NOT DEFINED FOR ALIAS Explanation: The specified alias is not assigned to the specified line type. User Action: Verify the alias and line type and retry the operation.
U	DBMS-0092	** AN '01' LINE TYPE MAY ONLY FOLLOW THE PRIMARY KEY DEFINITION ** Explanation: The DDL translator has encountered an 01 line type after other line types have been defined. If a file contains an 01 line, it must precede all other line type definitions. User Action: Position the 01 definition to precede all others or change the ID of the line; then resubmit the DDL.
U	DBMS-0093	** DUPLICATE FIELD/GROUP NAME ENCOUNTERED — ?1 ** Explanation: The DDL translator has encountered a field or group ID that has been defined twice. User Action: Change one of the occurrences of the duplicate ID and resubmit the DDL.

Table C-3 DBMS Error Messages (Continued)

U	DBMS-0094	** INVALID ACCESS DECLARATION ** Explanation: The DDL translator has encountered an invalid access declaration in the optional ACCESS clause. One of the access keywords RANDOM or SEQUENTIAL is required. The /1 designator is optional. User Action: Correct the syntax error indicated by the location of the up arrow and resubmit the DDL.
U	DBMS-0095	** DUPLICATE LINE TYPE ENCOUNTERED ** Explanation: The DDL translator has encountered a line type ID that has been previously defined. User Action: Change the previous occurrence of the ID or the occurrence flagged and resubmit the DDL.
U	DBMS-0096	** INVALID DATA TYPE ** Explanation: The DDL translator has encountered an invalid data type. A list of valid data types is listed in the DBMS Programmer's Guide. User Action: Verify that a valid data type has been entered and resubmit the DDL.
U	DBMS-0097	** SYNTAX ERROR ** Explanation: The DDL translator has encountered a syntax error. Usually, this error is caused by a nonblank character at the end of a line. User Action: Correct the syntax error marked by the location of the up arrow and resubmit the DDL.

Table C-3 DBMS Error Messages (Continued)

U	DBMS-0098	** ERROR ON CONVERSION OF ASCII VALUE ** Explanation: The DDL translator could not convert a numeric ASCII string to internal format. The ASCII string probably contains a non-numeric character. User Action: Verify that the value flagged by the up arrow contains only numeric characters and resubmit the DDL.
U	DBMS-0099	** STATEMENT OUT OF ORDER ** Explanation: The DDL translator has encountered a statement that it did not expect. The statement is not valid in its present location. User Action: Relocate the statement in a valid location and resubmit the DDL.
U	DBMS-0100	** MAXIMUM LINE LENGTH EXCEEDED ** Explanation: The data line identified exceeds the maximum length allowable, 512 bytes. The maximum size of a data line is 512 bytes minus the primary key length, minus 10 bytes overhead, minus eight times the number of secondary keys in the line. User Action: Shorten the line and resubmit the DDL.
U	DBMS-0101	ERROR ON CLOSE OF THE DATA BASE FILE, SVC ERROR ?1 Explanation: The DDL translator has detected an error while trying to close the data base file. User Action: Refer to the SVC error code for the exact cause of the error and respond accordingly.

Table C-3 DBMS Error Messages (Continued)

U	DBMS-0102	ERROR ON WRITING TO THE DATA BASE FILE, SVC ERROR ?1 Explanation: The DDL translator has detected an error while trying to write to the data base file. User Action: Refer to the SVC error code for the exact cause of the error and respond accordingly.
U	DBMS-0103	END OF TAPE ENCOUNTERED BY CPYFIL Explanation: CPYFIL utility encountered an end-of-tape mark. User Action: Mount a new reel of tape. Press the "return" key when the tape unit becomes ready.
U	DBMS-0104	** FILE NOT PROCESSED DUE TO SYNTAX ERRORS OR 'DUMY' PATHNAME ** Explanation: No data base file has been created. Either the DDL translator detected syntax errors during parsing or the user specified DUMY in response to the DB FILE PATHNAME prompt, indicating no file should be created. User Action: Correct any syntax errors flagged in the listing and enter a valid file pathname in response to the DB FILE PATHNAME prompt. User Action: None.
U	DBMS-0105	END OF TAPE ENCOUNTERED BY RLDFIL Explanation: RLDFIL utility encountered an end of tape mark. User Action: Mount the next reel of tape from the multiple tapes produced by CPYFIL. Press "return" when the tape unit becomes ready.

Table C-3 DBMS Error Messages (Continued)

U	DBMS-0106	END OF TAPE ENCOUNTERED BY RECOVR Explanation: RECOVR utility encountered an end-of-tape mark. User Action: Mount the next reel of tape from the multiple tapes containing the log file. Press "return" when the tape unit becomes ready.
U	DBMS-0107	?1: ?2 IS NOT A VALID KEYTYPE Explanation: The key type found in the file is not a valid DBMS-990 key type. DBMS-990 supports two key types, sequential (S1) and random (R1). The key type reported is not supported. User Action: None.
U	DBMS-0109	NO VALID DATABASE FUNCTIONS ENTERED Explanation: No valid data base functions were entered in response to the FUNCTIONS prompt. User Action: Refer to the DBSTAT section of the DBA User's Manual for a list of legal functions or enter ALL for a report listing the statistics on all the functions.
U	DBMS-0110	** MAXIMUM GROUP LENGTH EXCEEDED ** Explanation: The group flagged by the DDL translator exceeds the maximum length allowable. User Action: Reduce the size of the group and resubmit the DDL.
U	DBMS-0111	** MAXIMUM NUMBER OF SECONDARY KEYS EXCEEDED ** Explanation: More than 13 secondary keys have been defined. User Action: Reduce the number of secondary keys and resubmit the DDL.

Table C-3 DBMS Error Messages (Continued)

U	DBMS-0112	** NEW DATA BASE FILE CREATED ** Explanation: The data base file has been formatted and is ready to be accessed by DBMS-990. User Action: No user action required.
U	DBMS-0113	** NO FIELDS DEFINED FOR THIS LINE/GROUP ** Explanation: The line or group flagged by the DDL translator has no fields defined. User Action: Add a field definition to the line or group and resubmit the DDL.
U	DBMS-0114	** NO LINES DEFINED FOR THIS FILE ** Explanation: No lines have been defined to the DDL translator for this file. User Action: Add a line definition to the file and resubmit the DDL.
U	DBMS-0115	ERROR ON WRITE TO LISTING FILE, SVC ERROR ?1 Explanation: The DBSTAT utility has detected an error during a write to the listing file. User Action: Refer to the SVC error code for the exact cause of the error and respond accordingly.

Table C-3 DBMS Error Messages (Continued)

U	DBMS-0116	** KEY LENGTH EXCEEDS MAXIMUM ALLOWABLE ** Explanation: The length of the primary or secondary key flagged by the DDL translator is longer than the maximum allowable, 40 bytes. User Action: Reduce the size of the key field or group and resubmit the DDL.
U	DBMS-0117	** INVALID SECONDARY KEY NAME ** Explanation: The field/group name flagged by the DDL translator is invalid as a secondary key. Either the field/group has not been defined in a line, the field/group is defined as the primary key, or the field/group is already defined as a secondary key. User Action: Verify that the field/group is valid as a secondary key and resubmit the DDL.
U	DBMS-0118	ERROR RETURNED FROM DBMS, CODE ?1 Explanation: DBMS-990 has returned an error code to the DBSTAT utility. User Action: Refer to the DBMS Programmer's Guide for an explanation of the two character error code and respond accordingly.
U	DBMS-0122	THE PAGE SIZE HAS TO BE EITHER 256 OR 288 Explanation: The page size for the DB file must be either 256 or 288. Usually, choice is made dependent on sector size of target disk for the DB file. User Action: Retry with valid page size, 256 or 288.

Table C-3 DBMS Error Messages (Continued)

U	DBMS-0123	** DDL ENDS PREMATURELY ** Explanation: The DDL translator has reached an EOF before parsing the END.statement. User Action: Verify that all statements are in their correct order, add the END. statement and resubmit the DDL.
U	DBMS-0124	DATABASE MANAGER UNABLE TO OPEN PRE-IMAGE FILE Explanation: The system files have been damaged. User Action: Perform system generation at this time.
U	DBMS-0126	DATABASE MANAGER CANNOT GET REQUIRED MEMORY Explanation: An attempt was made to start the data base manager (SDBMS) with too many buffers. User Action: Retry the SDBMS command with smaller numbers for the parameters.
U	DBMS-0130	DBMS SUCCESSFULLY STARTED Explanation: The data base manager is running. User Action: No user action required.
U	DBMS-0131	SYSTEM FAILURE OCCURRED WHILE UPDATING FILE ?1 Explanation: The system crashed in the process of updating the file in question. The physical integrity of this file is now in question. User Action: Perform the Copy/Concatenate (CC) command on the backup for the file pathname and the RECOVR utility.

Table C-3 DBMS Error Messages (Continued)

U	DBMS-0132	UNABLE TO ACCESS FILE ?1 Explanation: For some reason the file in question is no longer available. User Action: Check to see if the necessary volume is installed. If the file is no longer available, perform CDBL.
U	DBMS-0133	PATHNAME OF FILE IS ?1 Explanation: Gives the pathname of the affected file. User Action: None
U	DBMS-0134	INTEGRITY ERROR ENCOUNTERED, PERFORM RECOVR Explanation: The system was interrupted in the process of updating a file. The physical integrity of the file is now in question. User Action: Perform the Copy/Concatenate (CC) command on the backup for the file pathname and perform the RECOVR utility.
U	DBMS-0135	UNABLE TO RESTART DBMS, CANNOT REOPEN FILES Explanation: The system is not able to open all the files that were assigned when the data base was last running. User Action: Check to see that the file(s) indicated is accessible, e.g., that the appropriate volume is installed, the file(s) has not been deleted. Perform the CDBL utility if the file is not accessible.
U	DBMS-0136	RESTART FAILED Explanation: The system was unable to restart the data base manager. User Action: None

Table C-3 DBMS Error Messages (Continued)

U	DBMS-0137	FILES REOPENED, ROLLBACK FAILED Explanation: Either the system log has been damaged or one of the files affected by rollback does not match the form it was in when the data base was last running and an error has occurred during rollback. User Action: Perform CDBL.
U	DBMS-0138	DBMS OPEN ERROR IS ?1 Explanation: Returns error status from DBMS. User Action: Check Appendix C of the *Data Base Administrator User's Guide* for an explanation of the two-character code.

Table C-4. Internal Message Codes

Internal Message Code	DBMS Message Number	Internal Message Code	DBMS Message Number	Internal Message Code	DBMS Message Number
>0001	0001	>0025	0037	>0056	0086
>0002	0002	>0026	0038	>0057	0087
>0003	0003	>0028	0040	>0058	0088
>0004	0004	>0029	0041	>0059	0089
>0005	0005	>002A	0042	>005A	0090
>0006	0006	>002B	0043	>005B	0091
>0007	0007	>002C	0044	>005C	0092
>0008	0008	>002D	0045	>005D	0093
>0009	0009	>002E	0046	>005E	0094
>000A	0010	>002F	0047	>005F	0095
>000B	0011	>0030	0048	>0061	0097
>000C	0012	>0031	0049	>0062	0098
>000D	0013	>0032	0050	>0063	0099
>000E	0014	>0033	0051	>0065	0101
>000F	0015	>0034	0052	>0066	0102
>0010	0016	>0035	0053	>0067	0103
>0011	0017	>0036	0054	>0068	0104
>0012	0018	>0037	0055	>0069	0105
>0013	0019	>0038	0056	>006A	0106
>0014	0020	>0039	0057	>006B	0107
>0015	0021	>003A	0058	>006C	0108
>0016	0022	>003B	0059	>006D	0109
>0017	0023	>003C	0060	>006E	0110
>0018	0024	>003D	0061	>006F	0111
>0019	0025	>003E	0062	>0070	0112
>001A	0026	>003F	0063	>0071	0113
>001B	0027	>0040	0064	>0073	0115
>001C	0028	>0041	0065	>0074	0116
>001D	0029	>0042	0066	>0077	0119
>001E	0030	>0043	0067	>0078	0120
>001F	0031	>0044	0068	>0079	0121
>0020	0032	>0045	0069	>007A	0122
>0021	0033	>0046	0070	>007B	0123
>0022	0034	>0053	0083	>007E	0126
>0023	0035	>0054	0084		
>0024	0036	>0055	0085		

Alphabetical Index

Introduction

HOW TO USE INDEX

The index, table of contents, list of illustrations, and list of tables are used in conjunction to obtain the location of the desired subject. Once the subject or topic has been located in the index, use the appropriate paragraph number, figure number, or table number to obtain the corresponding page number from the table of contents, list of illustrations, or list of tables.

INDEX ENTRIES

The following index lists key words and concepts from the subject material of the manual together with the area(s) in the manual that supply major coverage of the listed concept. The numbers along the right side of the listing reference the following manual areas:

- Sections — Reference to Sections of the manual appear as "Sections x" with the symbol x representing any numeric quantity.

- Appendixes — Reference to Appendixes of the manual appear as "Appendix y" with the symbol y representing any capital letter.

- Paragraphs — Reference to paragraphs of the manual appear as a series of alphanumeric or numeric characters punctuated with decimal points. Only the first character of the string may be a letter; all subsequent characters are numbers. The first character refers to the section or appendix of the manual in which the paragraph may be found.

- Tables — References to tables in the manual are represented by the capital letter T followed immediately by another alphanumeric character (representing the section or appendix of the manual containing the table). The second character is followed by a dash (-) and a number.

  Tx-yy

- Figures — References to figures in the manual are represented by the capital letter F followed immediately by another alphanumeric character (representing the section or appendix of the manual containing the figure). The second character is followed by a dash (-) and a number.

  Fx-yy

- Other entries in the Index — References to other entries in the index preceded by the word "See" followed by the referenced entry.

CUT ALONG LINE

USER'S RESPONSE SHEET

Manual Title: Model 990 Computer DNOS Data Base Administrator User's Guide (2272059-9701)

Manual Date: 15 July 1982 **Date of This Letter:** _____

User's Name: _____ **Telephone:** _____

Company: _____ **Office/Department:** _____

Street Address: _____

City/State/Zip Code: _____

Please list any discrepancy found in this manual by page, paragraph, figure, or table number in the following space. If there are any other suggestions that you wish to make, feel free to include them. Thank you.

Location in Manual	Comment/Suggestion
_____	_____
	_____
	_____
	_____
_____	_____
	_____
	_____
	_____
_____	_____
	_____
	_____
	_____

NO POSTAGE NECESSARY IF MAILED IN U.S.A.
FOLD ON TWO LINES (LOCATED ON REVERSE SIDE), TAPE AND MAIL

FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 7284 DALLAS, TX

POSTAGE WILL BE PAID BY ADDRESSEE

TEXAS INSTRUMENTS INCORPORATED

DIGITAL SYSTEMS GROUP

ATTN: TECHNICAL PUBLICATIONS
P.O. Box 2909 M/S 2146
Austin, Texas 78769

FOLD

TEXAS INSTRUMENTS
INCORPORATED

DIGITAL SYSTEMS GROUP
P.O. BOX 2909 • AUSTIN, TEXAS 78769