

Norsk Data



ACCESS DBA Manual

ND-30.022.03

A decorative graphic consisting of a grid of yellow and black dots. The dots are arranged in a rectangular pattern, with some dots being yellow and others black, creating a pixelated or halftone effect. The pattern is located in the lower right portion of the cover.

ACCESS DBA Manual

ND-30.022.03

NOTICE

The information in this document is subject to change without notice. Norsk Data A.S. assumes no responsibility for any errors that may appear in this document. Norsk Data A.S. assumes no responsibility for the use or reliability of its software on equipment that is not furnished or supported by Norsk Data A.S.

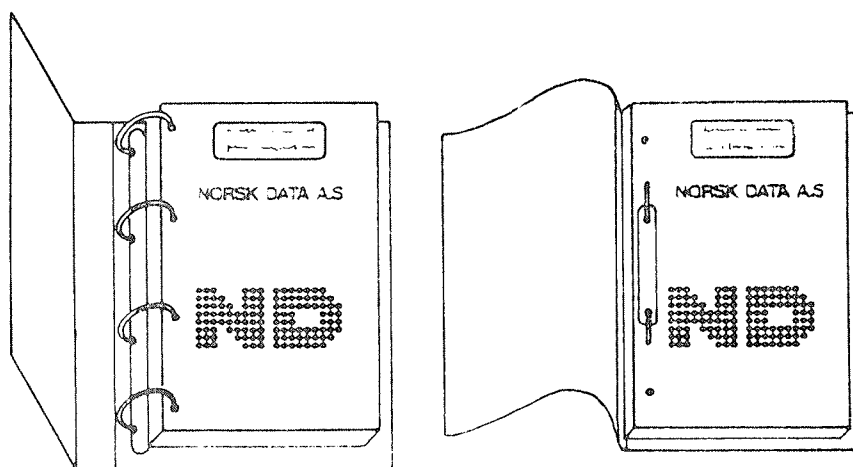
The information described in this document is protected by copyright. It may not be photocopied, reproduced or translated without the prior consent of Norsk Data A.S.

Copyright © 1984 by Norsk Data A.S.

This manual is in loose leaf form for ease of updating. Old pages may be removed and new pages easily inserted if the manual is revised.

The loose leaf form also allows you to place the manual in a ring binder (A) for greater protection and convenience of use. Ring binders with 4 rings corresponding to the holes in the manual may be ordered in two widths, 30 mm and 40 mm. Use the order form below.

The manual may also be placed in a plastic cover (B). This cover is more suitable for manuals of less than 100 pages than for large manuals. Plastic covers may also be ordered below.



A. Ring Binder

B. Plastic Cover

Please send your order to the local ND office or (in Norway) to:

Norsk Data A.S

Graphic Center

P.O. Box 25, Bogerud

0621 Oslo 6, Norway

ORDER FORM

I would like to order

..... Ring Binders, 30 mm, at nkr 20,- per binder

..... Ring Binders, 40 mm, at nkr 25,- per binder

..... Plastic Covers at nkr 10,- per cover

Name

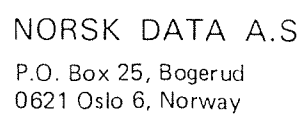
Company

Address

.....

City

ACCESS DBA Manual
Publ. No. ND-30.022.03
August 1984



Manuals can be updated in two ways, new versions and revisions. New versions consist of a complete new manual which replaces the old manual. New versions incorporate all revisions since the previous version. Revisions consist of one or more single pages to be merged into the manual by the user, each revised page being listed on the new printing record sent out with the revision. The old printing record should be replaced by the new one.

New versions and revisions are announced in the Customer Support Information (CSI) and can be ordered as described below.

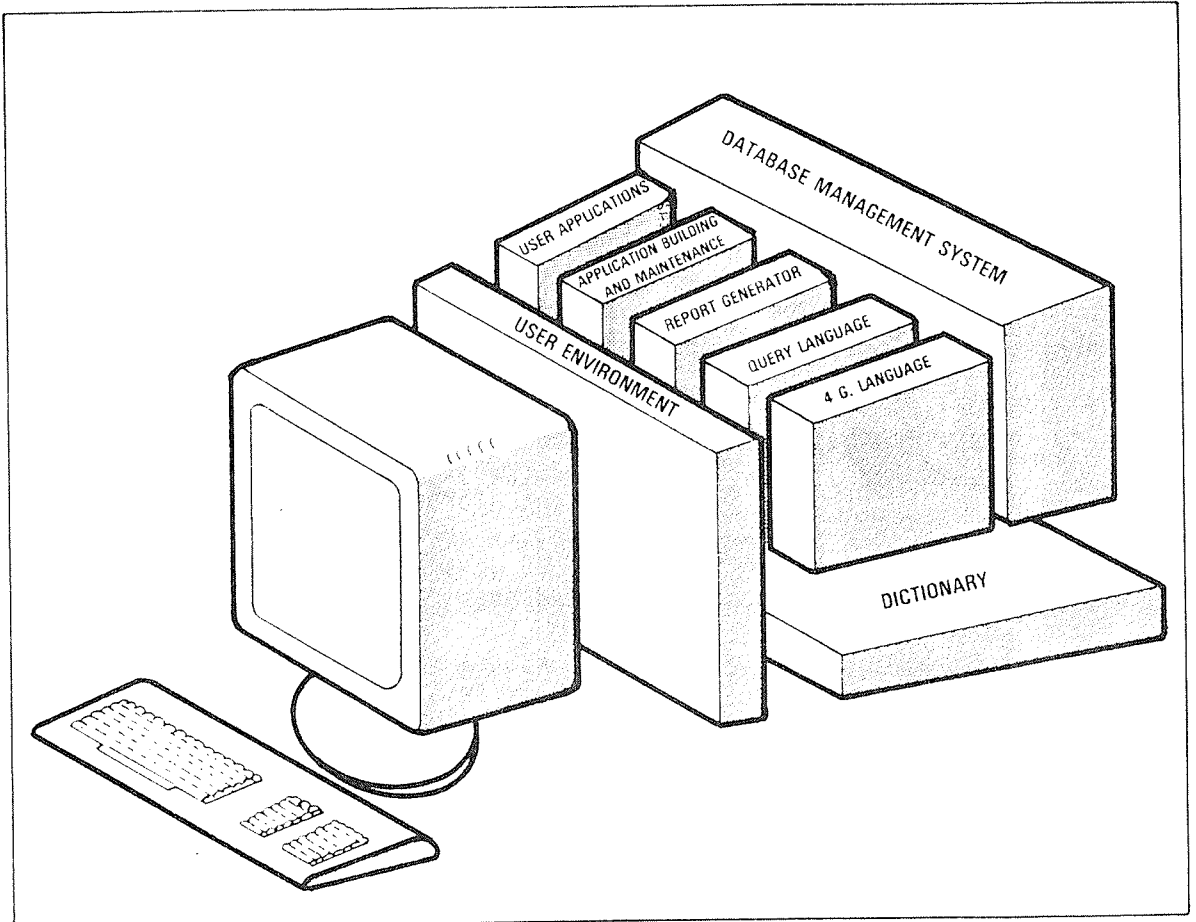
The reader's comments form at the back of this manual can be used both to report errors in the manual and to give an evaluation of the manual. Both detailed and general comments are welcome.

These forms and comments should be sent to:

Documentation Department
Norsk Data A.S
P.O. Box 25, Bogerud
0621 Oslo 6, Norway

Requests for documentation should be sent to the local ND office or (in Norway) to:

Graphic Center
Norsk Data A.S
P.O. Box 25, Bogerud
0621 Oslo 6, Norway



DIALOGUE

DIALOGUE

DIALOGUE is Norsk Data's total concept in data base management. It has the complete set of tools and utilities for:

- high performance, easy expansion, and redefinition of a data base;
- creating a tailored user interface;
- creating and maintaining applications easily and efficiently;
- generating advanced reports;
- common data dictionary information for easy coordination and maintenance of the data base and applications.

The modules of DIALOGUE are described below:

USER ENVIRONMENT

UE is an integrated part of the SINTRAN operating system. It can be used together with DIALOGUE to create a tailor-made, individual interface for the ND system.

4TH GENERATION LANGUAGE

UNIQUE is a tool for application development. It can be used to develop screen pictures and specify transactions directly on the screen. It saves about 90% of development time and maintenance resources.

REPORT GENERATOR

RG allows the definition of advanced reports in an easy manner by drawing the desired layout on the screen.

QUERY LANGUAGE

ACCESS is a tool which can be used to look at data base information in terms of tables. It is suitable for online use.

APPLICATION BUILDING AND MAINTENANCE

ABM can be used to make demanding transaction systems. It is used interactively with simple directives. It saves about 50% of development time and 90% of maintenance resources.

DATABASE MANAGEMENT

SIBAS is a full CODASYL data base management system. Its features include high performance, as well as easy expansion and redefinition of database. It is a flexible and a highly secure system, well suited for distributed processing environments.

Preface:

THE PRODUCT

This manual describes the Data Dictionary-DBA module in the query and transaction system ACCESS, version D. The product is registered with ND-number ND 10185 D.

THE READER

This manual is written for persons who will be responsible for supervising and maintaining an ACCESS query system.

PREREQUISITE KNOWLEDGE

The reader should have a basic knowledge of data processing. She or he should have some knowledge of the ND operating system SINTRAN III, ISAM files, and the ACCESS system itself. To define SIBAS tables, it is also desirable to have a basic knowledge of the structure of a SIBAS data base.

THE MANUAL

This manual describes the DBA program in ACCESS. It contains the information needed to supervise and maintain the ACCESS system. It is more a reference manual than a book intended to be read from beginning to end.

CHANGES IN THE PRODUCT AND THE MANUAL

- The names of the storage formats have been altered, and some new ones (NUMERIC TEXT, BCD, UNPACKED DECIMAL) added (p. 20).
- SIBAS realm descriptions may be copied automatically to ACCESS table descriptions. Group items, sets and CALC-keys may be defined (p. 34).
- Tables, data bases, users and user access to tables may be defined from ACCESS (pages 37, 25, 18 and 29).
- The D version of ACCESS stores several queries on one file (a query library). A section on multiuser query libraries has been added (p. 56).

- The DDI process should now be stopped with the mode file ACC-DDI-STOP:MODE, not with the SINTRAN command ABORT-JOB, as the latter method may cause inconsistencies in the DDI files.

All major changes have been marked with a vertical line in the margin.

See also the preface to the ACCESS User Guide.

RELATED MANUALS

ACCESS User Guide	ND-60.135
SINTRAN III Reference Manual	ND-60.128
SINTRAN III Timesharing/Batch Guide	ND-60.132
ISAM User Guide	ND-60.108
The Data Base System SIBAS II	
ND User Guide	ND-60.127

T A B L E O F C O N T E N T S

Section	Page
1 ACCESS AND THE DATA BASE ADMINISTRATOR	1
1.1 Introduction	3
1.2 An overview of ACCESS and DBA	3
2 DBA IN SINTRAN.	7
2.1 Starting and stopping the DDI	9
2.2 The user concept in the system	10
2.3 Giving several users access to the same data base	10
2.4 DBA and user areas in the SINTRAN file system	11
2.5 Commands	12
2.6 Starting the DBA	12
2.7 Organizing the daily use of the program	13
3 THE START LEVEL	15
3.1 How to define ACCESS users	17
3.1.1 From DBA	17
3.1.2 From ACCESS	18
3.2 How to define DBA users	18
3.3 How to define the data elements	19
3.3.1 How to specify the storage format	19
3.3.2 The correspondence with COBOL data descriptions	22
3.3.3 Reserved data type names	22
3.4 Finding out where a type is used	23
3.5 Undefined types	23
3.6 How to define a new data base	23
3.6.1 From DBA	23
3.6.2 From ACCESS	25
3.7 How to get a list of the available data bases	25
3.8 How to delete a data base	25
3.9 Modifying an existing data base description	25
3.10 Terminating the DBA start level	26
4 THE TABLE-DEFINITION LEVEL	27
4.1 How to specify user access to a data base	29
4.1.1 Specifying access from ACCESS	29
4.2 How to define the data elements in the tables	30
4.3 How to define and edit a table	30
4.3.1 Search keys - ISAM tables	31
4.3.2 Column width and result length	32
4.3.3 The use of FLAT files	32

<u>Section</u>	<u>Page</u>
4.4 Tables and files in ACCESS	33
4.5 How to define SIBAS tables	34
4.6 How to delete a table description	36
4.7 How to get a list of available tables in the data base . .	36
4.8 Terminating the table-definition level	36
4.9 Creating and editing a table from ACCESS	37
4.9.1 Creating a table from scratch	37
4.9.2 Changing an existing table definition	41
4.10 Creating views	42
 5 THE COMMAND LOAD-DATABASE IN ACCESS	 45
 6 GENERAL INFORMATION ON RUNNING THE ACCESS SYSTEM	 53
6.1 The command files	55
6.1.1 The command DUMP-HELP-FILE in ACCESS	56
6.1.2 The default query library and the default query	56
6.2 Multiuser query libraries	56
6.3 Something is wrong with the work area	57
6.4 DDI not active	57
6.5 Communication problems between the DDI and ACCESS	57
6.6 Improvements in the ACCESS search time	58
6.7 Other error messages	58
 7 COMPLETE EXAMPLES	 61
7.1 Example of how to define users	63
7.2 Example of how to define a data base	64
 <u>APPENDIXES</u>	
 A THE DDI PROCESS AS A BATCH PROCESS	 69
B SYSTEM SUPERVISOR INFORMATION	73
C THE CONTROL COMMANDS IN THE DBA MODULE	79
 Index	 82

CHAPTER 1

ACCESS AND THE DATA BASE ADMINISTRATOR

CHAPTER 1

ACCESS AND THE DATA BASE ADMINISTRATION

1 ACCESS AND THE DATA BASE ADMINISTRATOR

1.1 INTRODUCTION

ACCESS is a system that allows users with little or no experience in data processing to access and manipulate large amounts of stored data. ACCESS itself must retrieve a description of the structure of the data base. This is done through a process called the Data dictionary (DDI) process. The description resides on a set of ISAM files (the DDI files, see p. 13). These files are normally under user DATA-DICTIONARY, but may also be placed under another user (See section 2.4). In the following, it will be assumed that the DDI process is run under user DATA-DICTIONARY, which means that the files must also be located under this user.

The DDI process is implemented as a batch process <1> that has to be active when ACCESS is running. It supplies ACCESS with a complete description of the data structures the system operates on.

We need a tool to create new descriptions and modify old ones. This is the DBA, the Data Base ADMINISTRATOR, described in this manual.

Only a person who is defined as a DBA user is allowed access to the DBA program.

The DBA program described in this manual is designed to provide the information needed during daily use of ACCESS. The DBA program is used to define data bases and users, and to modify them and remove them when no longer needed. The ACCESS system depends on a consistent description of the data structures, and the performance of the system is greatly influenced by how the data is structured.

1.2 AN OVERVIEW OF ACCESS AND DBA

An ACCESS user may have access to one or more data bases. A data base is a collection of tables which may be FLAT files, ISAM files, or SIBAS realms. The SIBAS process must be running if you want to access a SIBAS data base.

The description of the ACCESS data bases is placed on a collection of files which is automatically created by the DBA program. These files are assigned to user DATA-DICTIONARY, the same user under which the DDI process is run.

<1> The DDI system is linked to a batch process in SINTRAN. This means that it does not have to occupy a terminal. If a terminal can be reserved for the DDI-process, you do not have to use the batch process. See Appendix A.

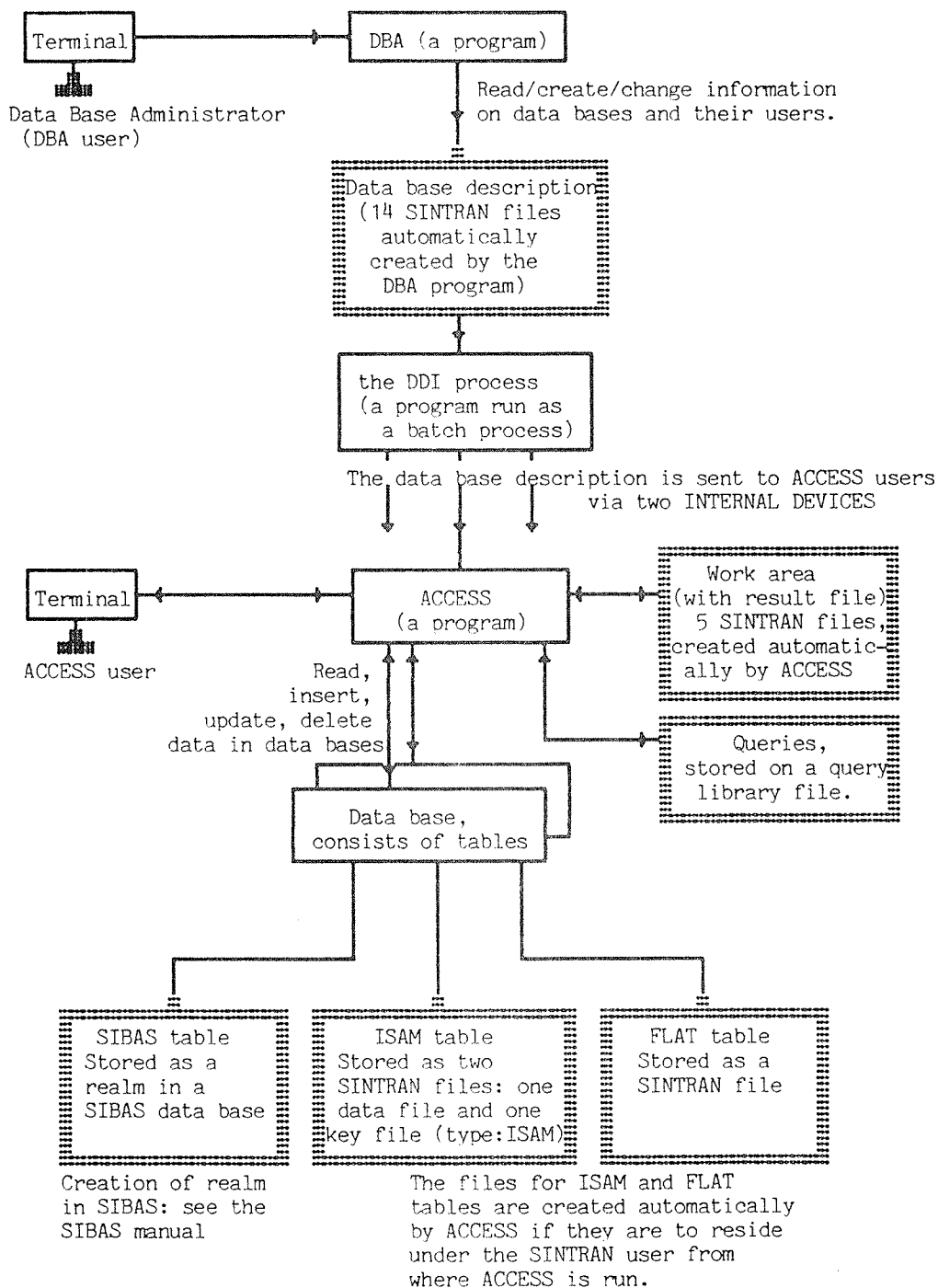


Fig. 1. An overview of ACCESS and DBA

CHAPTER 2

DBA IN SINTRAN.

CHAPTER 2

DBA IN LIBRARY

2 DBA IN SINTRAN.

2.1 STARTING AND STOPPING THE DDI

To start the DBA program, log in as user DATA-DICTIONARY, where the description of the data bases will be found.

If you log in under another user name, you will not have access to the data base descriptions residing under user DATA-DICTIONARY. Moreover, the information you may enter under the other user, will be ignored by ACCESS later! See further information in section 2.4.

Make sure no one is using ACCESS at the moment, by using the SINTRAN command TERM-STAT,,, (look for abbreviations of "ACCESS"). If anyone is running ACCESS, they must log out, or they will have problems when you proceed as described below. Make sure that the DDI process is passive by using the SINTRAN command:

@LIST-BATCH-PROCESS↵

A list containing the status of each batch process is displayed. If user DATA-DICTIONARY is using a batch process, this must be interrupted or "aborted" in the following way:

@MODE ACC-DDI-STOP:MODE↵

You are now ready to enter the DBA program. Give the command:

@DBA↵

After you have finished your work in the DBA program, ACCESS may be run again. Prior to starting up a new ACCESS session, activate the DDI process with the SINTRAN command:

@APPEND-BATCH, <batchno>, (DA-DIC)ACCESS-DDI:BATC, (DA-DIC)ACC-DDI:LOG

The two last parameters are the input and output files for the batch process.

You may also arrange it so that the DDI process will start automatically when you exit the DBA program. You do this by replacing line 3 in the file (DATA-DIC)DBA-HELP:TEXT with the command described above, but without the "@". (You may use NOTIS-WP, but make sure that, except for replacing line 3, you do not add or delete lines in the file.)

NOTE: "APP-" must be placed to the far left, and the command must not go beyond character position 59.

The ACCESS program depends on response from the DDI process in order to function, so the DDI must be active while ACCESS is used.

Further information on the batch system which is used to keep the DDI process active is provided in the SINTRAN III Timesharing/Batch Guide, ND-60.132. See also Appendix A.

All commands in SINTRAN may be abbreviated as long as this does not make them ambiguous. In the examples above, the same commands may be written:

@LI-B-P,,,←

@MODE -DDI:MODE←

@AP-B <batch no>,-DDI:BATC,-DDI:LOG←

The file ACCESS-DDI:BATC is a file which describes the running of the DDI as a batch process. ACCESS-DDI:LOG is the output file for the batch process.

Check that all these files exist in the user area DATA-DICTIONARY with the command

@LIST-FILES←

2.2 THE USER CONCEPT IN THE SYSTEM

ACCESS has its own user identification which is not the same as SINTRAN's user identification.

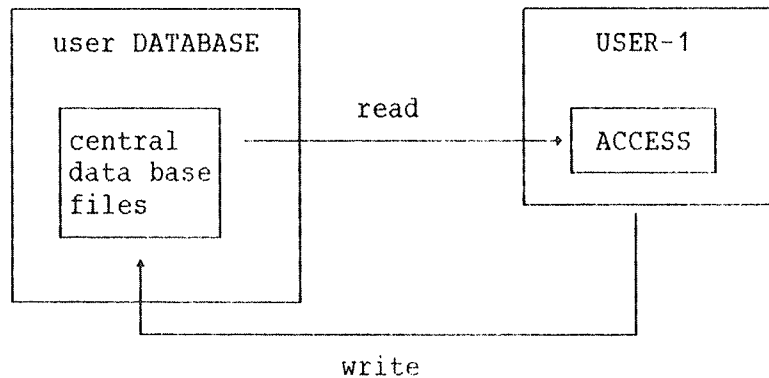
ACCESS users are defined in the DBA program, and are there given access rights to data bases and files. Each ACCESS user has a work area which consists of five SINTRAN files. This work area must be defined for each user.

Only one ACCESS user can use the same work area (files) at a time, whereas several persons (terminals) can be logged into SINTRAN under the same SINTRAN user name. The connection between SINTRAN user names and work area is explained on page 17.

2.3 GIVING SEVERAL USERS ACCESS TO THE SAME DATA BASE

All users who are to add, modify or delete information in a data base (ISAM or FLAT) must have write access to the files in which this information is stored. Since SINTRAN users normally do not have write access to other users' files, this may cause problems when user-1, running ACCESS from his own user area, tries to write on data base files belonging to the user on which the central data base is located. Let us say the user that owns the data base is a user created specifically for the purpose, named DATABASE.

The situation could be illustrated like this:



Problems with this may be avoided in different ways:

- 1) All users of the data base may be created as friends of the user who owns the data base. This, however, is limited to eight users, as this is the maximum number of friends allowed.
- 2) Public access for the data base files may be set to RWA. To make it easier to create new tables, default public access for the user having the data base files may also be set to RWA.

2.4 DBA AND USER AREAS IN THE SINTRAN FILE SYSTEM

If you were to run the DBA from a user other than DATA-DICTIONARY, you would get an independent data base description. Note that ACCESS will not use this description unless the DDI process is run under this particular user. If you want to use an independent description, you must do the following before you start the DDI process: change the contents of the files (DATA-DIC)ACCESS-DDI:BATC and (DATA-DIC)ACC-DDI-STOP:MODE, so that the name DATA-DICTIONARY in the first line is replaced by the name of the SINTRAN user you are logged in as when you run the DBA program.

If you want your installation to have a data base description under another SINTRAN user area in addition to the "normal" description under DATA-DICTIONARY, you must start two DDI processes on two different batch processors. The two processes must use different pairs of "internal devices", and have their individual copies of the input and output files in the APPEND-BATCH command mentioned in section 2.1 (Compare the description of how ACCESS is loaded into SINTRAN from floppy disks.) Each copy of the input file must have its individual SINTRAN user name in the first line.

Since the file ACCESS-HELP:TEXT contains the numbers of the internal devices, it determines which of the two DDI processes ACCESS will communicate with. When you start the ACCESS program from a SINTRAN user area, ACCESS will first search this area for the file ACCESS-HELP:TEXT. If it does not exist there, nor under user SYSTEM, the ACCESS-HELP under user DATA-DICTIONARY will be used. In the file that is found, the two internal devices of the desired DDI process must have been specified.

This means: for each SINTRAN user who is to use a data base description which does not reside under user DATA-DICTIONARY, you must copy the file (DA-DICT)ACCESS-HELP:TEXT to this SINTRAN user, and change the numbers of the internal devices in lines 2 and 3 so that they correspond with the internal device numbers in the input file to the desired DDI process.

An independent data base description allows you to develop the data base without interfering with the daily use of the program. It also gives beginners an opportunity to test and experiment without interfering with the "real" running of the program under user DATA-DICTIONARY. It can also be used to physically separate two different views of the same data.

2.5 COMMANDS

Commands are given in the command position, which is marked with * or ** under the line at the bottom of the screen.

The DBA program has two levels. Level 1 is called the start level. All the tables are defined on level 2, the table-definition level. The number of asterisks shows you which level you are on:

```
*   → Start level
**  → Table-definition level
```

All the commands in the DBA can be abbreviated in the same way as in SINTRAN. This means that *E-TA is sufficient to distinguish the EDIT-TABLE command from the *EDIT-TYPES command. Blanks in front of command names are not allowed.

A command is interrupted with the "HOME" key:



2.6 STARTING THE DBA

The DBA program is started by giving the command:

@DBA

Remember that you must be logged in as user DATA-DICTIONARY if you want to edit the descriptions the DDI process is to give to ACCESS.

The DBA will prompt for user identification. The user identification is not the SINTRAN user identification, which here would be DATA-DICTIONARY, but ACCESS' own user identification. All ACCESS users who are to use the DBA program must be defined as DBA users.

If you want to create an independent data base as described in section 2.4, the DBA will not ask for user identification the first time. You will get directly into the user description, where so far only user "SYSTEM" is defined. Change "SYSTEM" to the desired user name, and give work area and password, as desired. (See section 17.) The new user name will automatically be registered as a DBA user.

2.7 ORGANIZING THE DAILY USE OF THE PROGRAM

If your company or organization wishes to work on the development of the use of ACCESS on a data base while interfering as little as possible with the normal use of the program, the work could be organized in this way:

Copy the data base descriptions (the DDI files) from user DATA-DICTIONARY to user ACCESS. User DATA-DICTIONARY must be listed as "friend" under user ACCESS. The DDI files are the following:

```
DATABASE (:DATA and :ISAM)
TABLE      (:DATA and :ISAM)
ATTRIBUTE  (:DATA and :ISAM)
GROUP-SET  (:DATA and :ISAM)
TYPE       (:DATA and :ISAM)
USER       (:DATA and :ISAM)
ACCESS     (:DATA and :ISAM)
```

Enter the DBA program when you are logged in as SINTRAN-user ACCESS, and define the new data base.

If there are tables of type ISAM or FLAT, copy the files containing the tables to the user area ACCESS. Use the same file names! If you give the command **ISAM-FILES or **FLAT-FILES in the DBA program, the file names should, for the time being, be listed without SINTRAN user names.

Start an individual "test" DDI process for the data base descriptions on user ACCESS, see section 2.4.

Now ACCESS may be run against the copies of the data base to test that everything is working properly, or it may be used for training new ACCESS operators. The user must be logged in as ACCESS. Note that there must be a copy of the file ACCESS-HELP:TEXT under SINTRAN-user ACCESS, with the correct device numbers for the test DDI process.

After the data base description has been tested and found in order, the DDI process must be stopped. Then the DBA program must be run once more under SINTRAN-user ACCESS to enter the SINTRAN user names in the file names of the "real" FLAT and ISAM tables (the commands **ISAM-FILES and **FLAT-FILES). Then the data base description must be copied to DATA-DICTIONARY, and the real work with ACCESS against this data base can start. A "backup" of the data base descriptions on user ACCESS may be taken at the same time.

CHAPTER 3

THE START LEVEL

CHAPTER 3

THE STATE LEVEL

3 THE START LEVEL

3.1 HOW TO DEFINE ACCESS USERS

3.1.1 FROM DBA

Give the command:

*EDIT-USERS+↓

The following picture will be displayed on the screen:

ACCESS users		
ACCESS user name	Password	Work area

User name

The ACCESS user names are displayed in the first column, and new users are entered at the end of the list. If there are no current users in ACCESS, the list will be empty and the first new user can be written on the first line. Users can be deleted with the SHIFT + DELETE key on the NOTIS terminal (or CTRL+D CTRL+L).

Password

To ensure safe use of the ACCESS system, each user may have a secret password, which is displayed in this column. The DBA user is allowed to see the password, change it, delete it and create new passwords. A DBA user is not allowed to see or change the password of another DBA user. It is displayed overwritten with asterisks.

Work area

Every user must have five files that ACCESS can use as a temporary work area.

ACCESS will create these files automatically, but their name must be specified in the "work area" column here. If the work area for USER-1 is specified as (USER-1)WORKAREA", and USER-1 runs access when logged in as SINTRAN-user USER-1, ACCESS will create the following five files:

```
(USER-1)WORKAREA:DATA
(USER-1)WORKAREA:SORT
(USER-1)WORKAREA:SYS1
(USER-1)WORKAREA:SYS2
(USER-1)WORKAREA:SYS3
```

If USER-1 tries to run ACCESS when logged in as another SINTRAN-user, ACCESS will not succeed in creating the work area files, and will display the error message 'Something is wrong with the work area : "(USER-1)WORKAREA:DATA"/75'.

3.1.2 FROM ACCESS

You can create a new user from ACCESS with the command CREATE-USER (on the start level).
The call sequence is:

```
*CREATE-USER <user name> <ACCESS work area>←↵
```

If the user already is defined, the name of the user's work area can be changed.

3.2 HOW TO DEFINE DBA USERS

The DBA user is a special user in the ACCESS system, allowed to create new data bases, delete old data, and change existing descriptions of files belonging to his data bases when needed.

To define an ACCESS user as DBA user, give the command:

```
*DBA-USERS←↵
```

The following picture appears on the screen:

Data base administrators
DBA user

The column is filled in with the names of those ACCESS users who are to become DBAs. Pressing the HELP key will give you a list of all defined users.

Note that you will not be allowed to edit/delete information on DBA users other than yourself.

3.3 HOW TO DEFINE THE DATA ELEMENTS

So that the ordinary users of ACCESS need not bother with the details of storage and retrieval, these are defined once and for all in the DBA program.

All data elements have a TYPE. This defines how the data element is stored, but it also restricts the use of the element. Data elements of the same type can represent each other, "replace one another".

There is a special command used to define types of data elements. Give the command:

***EDIT-TYPES←**

The following picture is displayed:

Data types		
Name of data type	Display code	Storage format

Each data type must be given a name which later will be used when referring to the type in the table description.

It is sufficient to fill in the fields "Name of data type" and "Storage format". The display code, which shows how the data will be displayed on the screen in query results, will be filled in automatically.

3.3.1 HOW TO SPECIFY THE STORAGE FORMAT

The Storage format, or "Internal format", determines how the data type will be stored. It also limits the ways in which the item can be manipulated.

The following table lists all legal storage formats.

Storage Format	Arithmetic legal?	Size of data element
TEXT (n)	NO	n bytes (max 2048)
NUMERIC TEXT (n)	YES precision 18 digits	n bytes
INTEGER2	YES (Min/max value: -32 768 to 32 767)	2 bytes/16 bits
INTEGER4	YES (Min/max value: -2 147 483 648 to +2 147 483 647)	4 bytes/32 bits
REAL4 *	YES - Precision 6 digits	4 bytes/32 bits
REAL6 *	YES - Precision 8 digits	6 bytes/48 bits
REAL8	YES Precision 16 digits	8 bytes/64 bits
BCD (n,m) or PACKED DECIMAL (n,m)	YES Precision 18 digits	(n + m + 2)/2 bytes
$\left\{ \begin{array}{l} \text{SIGNED} \\ \text{UNSIGNED} \end{array} \right\} \dagger$		
UNPACKED DECIMAL (n,m)	YES Precision up to 18 digits	n + m, except SIGNED SEPARATE which gives n + m + 1
$\left\{ \begin{array}{l} \text{SIGNED} \\ \text{UNSIGNED} \end{array} \right\} \dagger$		
$\left\{ \begin{array}{l} \text{EMBEDDED} \\ \text{SEPARATE} \end{array} \right\} \dagger$		
$\left\{ \begin{array}{l} \text{TRAILING} \\ \text{LEADING} \end{array} \right\} \dagger$		

* The choice between REAL4 and REAL6 depends on the type of ND-100 CPU. If you write just REAL, The DBA program will choose the correct one of these.

† Defaults are SIGNED EMBEDDED TRAILING

Relatively inexperienced DBA users who want to define their own tables will need mainly the storage types TEXT and NUMERIC TEXT. See the description of the ACCESS command CREATE-TABLE (p. 37) for a more detailed explanation of these types (in ACCESS called CHARACTER and NUMERIC).

If the data type is to be used in the definition of data items in SIBAS, the size of the data element must be identical to the LENGTH of the data element as this is defined in the SIBAS-DRL. Note that LENGTH in SIBAS-DRL is specified as the number of 2-byte words. The SIBAS type chosen in the DRL does not have to correspond.

Example: we use a 48-bit floating point ND-100 CPU, and "no1" and "no2" are defined here with storage format REAL. These may be defined in SIB-DRL as:

```
NEW ITEM <realm name> no1 TYPE integer  START 1 LENGTH 3
```

```
NEW ITEM <realm name> no2 TYPE character START 4 LENGTH 3
```

Note that the storage format in ACCESS must be identical with that defined for the data base (and which other users use when data is put in). ACCESS does not set parity bits in the types TEXT and CHARACTER.

If you write just REAL, DBA will choose between REAL4 and REAL6, according to the CPU.

If you want to use the same data base on both an ND-100 and an ND-500, REAL8 should be used.

3.3.2 THE CORRESPONDENCE WITH COBOL DATA DESCRIPTIONS

The following table lists the storage formats used in the DBA program and the corresponding COBOL data descriptions. It is intended to make it simpler to define tables against data generated by COBOL programs.

Storage Format	COBOL 100	COBOL 500
TEXT (n)	PIC X(n)	PIC X(n)
NUMERIC TEXT (n)		
INTEGER2	COMP-1	
INTEGER4	COMP-1 PIC 9(n) where n >= 5	COMP-1
REAL4	COMP-2 on a 32 bit CPU	COMP-2 PIC S9(n)V9(m) where n + m <= 6
REAL6	COMP-2 on a 48 bit CPU	COMP-2 PIC S9(n)V9(m) where n + m >= 7
REAL8		
BCD (n,m) or PACKED DECIMAL (n,m)	COMP-3 PIC 9(n)V9(m)	COMP-3 PIC 9(n)V9(m)
{ SIGNED } { UNSIGNED }	PIC S9(n)V9(m) PIC 9(n)V9(m)	PIC S9(n)V9(m) PIC 9(n)V9(m)
UNPACKED DECIMAL (n,m)		
{ SIGNED } { UNSIGNED }	PIC S9(n)V9(m) PIC 9(n)V9(m)	PIC S9(n)V9(m) PIC 9(n)V9(m)

3.3.3 RESERVED DATA TYPE NAMES

In the Data Dictionary some data type names are reserved for ACCESS usage. They are created if a table is defined from ACCESS, or if the command COPY-REALMS is used in the DBA program.

The reserved data-type names are:

INTEGER-<n>
REAL-<n>
BCD-<n>-<m>
NUMERIC-<n>
CHARACTER-<n>
FILLER-<n>

3.4 FINDING OUT WHERE A TYPE IS USED

To obtain information on the use of the types that are defined at a particular time, use the command:

*WHERE-IS-TYPE-USED <type name>←↓

It is often useful to know if a type is used at all; if not, the type definition can be deleted. It can also be useful to know where a type is used if you want to define a new type, and to use this in already defined tables where the new type has the same storage format as the old one. It may also be useful to compare types that use an almost identical storage format, to see if one of them is superfluous.

You will not be allowed to delete a type that is in use.

3.5 UNDEFINED TYPES

To obtain a list of all the types that have been used in the "*EDIT-TABLE" command, but not yet defined by using the "*EDIT-TYPES" command, use the command:

*UNDEFINED-TYPES←↓

If you, for instance, have misspelled a data type in a table ("APPPLES"), this will be displayed. You must not use a table containing undefined types, as this will give an erroneous result.

3.6 HOW TO DEFINE A NEW DATA BASE

3.6.1 FROM DBA

In ACCESS, a data base is the name of a group of tables that may be realms in a SIBAS data base, indexed ISAM files, or so-called FLAT files without search keys. The records in the files must have a fixed length. Each data element must have the same length and location in each record in the table.

Using the DBA program, any user of the system may be given access to such a data base. This description must contain information on how the data is stored, who is allowed access to the data base, and how the information in the data base is to be presented to the users.

A new ACCESS data base may be created by giving the following command (if the data base is to contain SIBAS tables, it is simpler to use command LIST-DATABASES; this is explained below):

*CREATE-DATABASE↵

DBA will ask for the name of the new data base:

Data base name: SAMPLE-BASE

Alternatively, you can give the data base name along with the command:

*CREATE-DATABASE SAMPLE-BASE

The ACCESS user name you gave when you entered the DBA will be recorded as the owner of the data base. This name can be altered later with the LIST-DATABASES command (described below).

The same data base may have tables of different types: SIBAS, ISAM and "FLAT".

The information/data you specify in the DBA program under user DATA-Dictionary is stored in files in this user's area, and is not transferred to the SIBAS data base or the file system.

The command *CREATE-DATABASE makes the system go from the start level to the table-definition level.

If the data base is to contain SIBAS tables, it is more practical to create it with the command *LIST-DATABASES. A table like the following will be displayed:

Defined data bases				
Data base name	SIBAS name	SIBAS no	SIBAS password	Responsible DBA
ENGLISH-BASE		0		ACCESS

New ACCESS data bases may be created by adding to this screen picture.

Data base name is the name of the ACCESS data base.

SIBAS name is the name of the SIBAS data base that may be used in this ACCESS data base. Realms from this SIBAS data base may be defined as tables in the ACCESS data base. Only one SIBAS data base may be used with one ACCESS data base.

SIBAS no is the number of the SIBAS process that controls the SIBAS data base.

SIBAS password is the password on the data base level in the SIBAS data base.

Responsible DBA is the DBA user that created the ACCESS data base. In other words, enter your DBA user name.

3.6.2 FROM ACCESS

The command CREATE-DATABASE (described in the previous section) may also be given on the start level in ACCESS. It is used in exactly the same way as in DBA.

3.7 HOW TO GET A LIST OF THE AVAILABLE DATA BASES

A list of the available data bases in the system is produced by the command:

*LIST-DATABASES<-J

The names of all data bases are displayed on the screen, together with some information that is only relevant when SIBAS tables are used. This information is explained above (section 3.6.1).

3.8 HOW TO DELETE A DATA BASE

A data base description is deleted from the ACCESS system by giving the command:

*DELETE-DATABASE<-J

followed by the name of the data base. If the name is not given, the system prompts for the name as in the *CREATE-DATABASE command.

For this command to function, the system requires that the DBA giving the command must be the DBA responsible for that data base, the "owner" of the data base.

The data files that are defined as belonging to the data base are not affected by the command, but if they are to be used again, they must be redefined in the DBA program.

3.9 MODIFYING AN EXISTING DATA BASE DESCRIPTION

Usually, it is more convenient to modify an existing data base description than to create a new description. This is done with the command:

*EDIT-DATABASE<-J

The name of the data base is given directly after the command, or as an answer to the question asked by the system.

The description of the data base is loaded into the system and the system proceeds to the table-definition level (level 2).

Only the owner of the data base may edit the data base description.

3.10 TERMINATING THE DBA START LEVEL

To leave the DBA program from the start level, press the



key, or give the command:

***EXIT**↵

C H A P T E R 4

THE TABLE-DEFINITION LEVEL

CHAPTER 4

THE TABLE-DEFINITION LEVEL

4 THE TABLE-DEFINITION LEVEL

The table-definition level (level 2) is for creating, editing and deleting tables or description of files. You get to this level by giving the commands

*CREATE-DATABASE←↓

or

*EDIT-DATABASE←↓

Work on this level requires that both the user and a data base are defined.

4.1 HOW TO SPECIFY USER ACCESS TO A DATA BASE

Before a user can execute queries against a data base, s/he must be given access rights to the data base. Only the DBA user who created the data base (with the command CREATE-DATABASE or LIST-DATABASE) can give other users access. This is done with the command:

**EDIT-ACCESS←↓

The data base has already been defined on the start level, so the system does not require the data base name here. The following picture will be displayed on the screen:

User access to data base				
ACCESS user name	Print.	Insert.	Update.	Delete.

In the user name column you type in the names of the ACCESS users you want to give access to the data base. (Pressing the HELP key will give you a list of defined users.) In the columns under each ACCESS operator you type YES or NO depending on whether you allow this user to use this particular operator. Note that blank is identical to NO, but the owner of the data base has unlimited access.

4.1.1 SPECIFYING ACCESS FROM ACCESS

The command CREATE-ACCESS on the start level of ACCESS has the same function as the DBA command described above, but it does not allow you to look at previously defined access authorizations.

The command is given as:

***CREATE-ACCESS** <data base name><user name><P.><I.><U.><D.>+↵

If the user already has access to the data base, the user's access rights will be changed.

The access authorizations are given as YES or NO and are:

<P.> - Print is permitted.
<I.> - Insertion of new records is permitted.
<U.> - Update of already existing records is permitted.
<D.> - Deletion of records is permitted.

4.2 HOW TO DEFINE THE DATA ELEMENTS IN THE TABLES

The types for all data elements must be defined with the command

****EDIT-TYPES**+↵

This command is described in detail in section 3.3. It can be used either before or after the table itself is defined.

4.3 HOW TO DEFINE AND EDIT A TABLE

A new table is created by giving the command:

****CREATE-TABLE**+↵

The DBA system will ask for the table name, and then for the table type. The possible table types are:

ISAM: search keys may be specified
FLAT: no search keys allowed
SIBAS: A SIBAS DBMS realm

To edit a table that already exists, use the command:

****EDIT-TABLE**+↵

In this case DBA will only ask for the table name.

With both these commands, the following picture will appear on the screen with empty columns:

<Name of the table>			
Name of data element	Data type used	Number	Sear

You can then fill in the information describing the data elements used in the table. The table will later be identified by the table name.

The names given to each data element in your table appear as column names when you later use ACCESS to define and execute queries against the table.

The "Data type used" field is filled with the names of the data types you defined with the EDIT-TYPES command.

The "Number" field is filled with a digit to indicate the order in which the data elements will be displayed later in the table frame and result table. This field may be left blank; it will be filled in automatically.

The previous table is the left part of the screen picture used for defining and editing tables. As the fields are filled in, the window moves to the right to display the rest of the table:

<Name of the Table>			
Number	Search key	Column width	Result length

4.3.1 SEARCH KEYS - ISAM TABLES

Search keys in ISAM tables can be either PRIMARY or SECONDARY. For FLAT files this field is not used.

When to use search keys

Using search keys will mean faster searches. Queries using selection criteria and sorts in a column will run faster if that column is defined as a search key.

On the other hand, keys take up space. When a column is defined as a key, the whole column will be stored twice: once on the :DATA file and once on the :ISAM file. This also means that queries using UPDATE., DELETE., and INSERT. will run more slowly, since the key fields have to be stored twice.

As a rule of thumb, use as keys only columns that you expect to do searches in often. Do not use long text fields as keys.

Secondary or primary?

Primary keys give faster searches than secondary keys. On the other hand, a primary key column cannot contain duplicate entries, nor the value zero in numeric fields. For example, if a column is to contain

surnames, it would not be wise to define it as a primary key, since there might well be two or more people called Jones. Use "PRIMARY" only when you are sure that the column will not contain duplicates.

It is sufficient to type "S" or "P" in the search key field. All other values, including misspellings, will mean that the data element has not been given a search key. It will therefore be more time-consuming to search for this element, as the search must be carried out sequentially.

An ISAM table must have at least one data element, maximum 64, defined as a search key.

4.3.2 COLUMN WIDTH AND RESULT LENGTH

The column width is used to define the initial width of the column in the table frame in ACCESS. The column will get the defined width or the minimum width needed for the data element names.

The result length is the width of the column when the result of a query is displayed on the standard table format. For example:

A text field with a length of 80 characters and a result length of 20 will fill 4 lines with 20 characters. The result will then be displayed with at least 4 lines per record. The number of lines per record is defined by the field which occupies the most lines.

If nothing is defined, ACCESS will see to it that the result is displayed on a sensible format.

4.3.3 THE USE OF FLAT FILES

A FLAT file is a sequential file with a fixed record length. All the records in the file are assumed to have an identical structure. No search keys may be assigned to such a file. This means that each time ACCESS is to find anything in the file, it has to search through the entire file from the beginning. Such files are nevertheless much used, because they are simple and reliable. A FLAT table is defined as previously described, with the difference that no search keys are defined in the search key column. The table for defining the contents is identical with the ones used for the other table types.

The file name (in SINTRAN) can be redefined with the command ****FLAT-FILES**. The following table appears:

FLAT files	
Table name	SINTRAN file name

The file name is redefined by editing the right column.

4.4 TABLES AND FILES IN ACCESS

If the table you have defined is to be used by different users running ACCESS from different SINTRAN user areas, accessing the same data base, you should perform the following two steps:

- 1) For ISAM tables, the following command is used: ***ISAM-FILES**.

The following picture appears on your screen (As an example, we have assumed that you have defined one table called "EMPLOYEES"):

ISAM files	
Table name	SINTRAN File Name
EMPLOYEES	EMPLOYEES

In the "SINTRAN file name" column the name of the SINTRAN-user where the file is to be located must be added. In other words, EMPLOYEES must be changed to, say, (DATABASE)EMPLOYEES.

- 2) Create two files with the name from the column "SINTRAN file name", with file types :ISAM and :DATA. In the above example, you would need to create the files EMPLOYEES:DATA and EMPLOYEES:ISAM under SINTRAN-user DATABASE.

For FLAT files, use the same procedure, but replace the command ISAM-FILES with FLAT-FILES, and create only the :DATA file.

If you do not include the user name in the "SINTRAN file name" column, ACCESS will later create one file for this table for every SINTRAN-user running ACCESS against this table. Each SINTRAN user will thus get a different version of the table. The data will be spread on several files, and an ACCESS query will only be able to find part of the data: that part belonging to the SINTRAN user currently running ACCESS. In effect, the result will be different data bases with identical names.

This could be a practical solution when several users have their private data with a common description, for example private accounts. They must have their own SINTRAN user area. Then you need only make one description in the DBA program, covering several data bases.

On the other hand, if you want just one data base, you must include the SINTRAN user name in the file name!

4.5 HOW TO DEFINE SIBAS TABLES

A SIBAS realm can be defined as a table in ACCESS. The SIBAS single items will be the data elements in the table.

The SIBAS data base (and system number) to be used is specified when the ACCESS data base is created (see p. 24).

SIBAS tables may be defined manually, as ISAM and FLAT tables; or they may be defined automatically, by copying information directly from the description in SIBAS-DRL.

Automatic table definition

The following commands may be used:

- **LIST-REALMS Will list all realms in the SIBAS data base.
- **COPY-REALM Will copy a SIBAS realm description to an ACCESS table
 <realm name> description including all SIBAS items, all
 group items <1>, and all sets using the given realm.
- **COPY-DATABASE Will copy all SIBAS realm descriptions to ACCESS table
 descriptions, including all items, group items and
 sets.

If you have SIBAS version D, there is one limitation to the commands COPY-REALM and COPY-DATABASE: the set definitions produced may be incomplete. In this case you will get the following message:

Set definitions incomplete. Use the command EDIT-SETS.

When you now use the command EDIT-SETS, there will be some blank fields which must be filled with information from the SIBAS data base definition.

The table and data element names created by COPY-REALM and COPY-DATABASE will be identical to the SIBAS realm and item names. If you want the table and data element names that ACCESS displays on the screen to be more descriptive, you can edit them with the commands SIBAS-REALMS and SIBAS-ITEMS.

-
- <1> The group items will not be defined as ACCESS data elements.
 If you want this, you must do it manually.

The command SIBAS-ITEMS will give a screen picture like the following:

EMPLOYEES	
Data element name	Item name
NAME	NAME
MANAGER	MANAGER
SALARY	SALARY
DEPT	DEPT

The left column contains the data element names that the ACCESS users see on the screen. These may be edited; for example, "DEPT" may be changed to "Department".

Manual table definition

The CREATE-TABLE, EDIT-TABLE and LIST-TABLES commands are used in the same way as when you define ISAM or FLAT tables (see section 4.3).

Note that you cannot use these commands to alter the structure of the SIBAS data base itself. Your definitions have to correspond to the definitions that already exist in SIBAS-DRL.

The following search-key types can be defined:

- PRIMARY-INDEX - Index key, duplicates not allowed
- SECONDARY-INDEX - Index key, duplicates allowed
- PRIMARY-CALC - Calc key, duplicates not allowed
- SECONDARY-CALC - Calc key, duplicates allowed

You can give the tables and data elements any names you want, but since these correspond to realms and items in SIBAS, you must use the commands SIBAS-ITEMS and SIBAS-REALMS afterwards. Let us again use SIBAS-ITEMS as an example:

EMPLOYEES	
Data element name	Item name
Full name	Full nam
Manager	Manager
Salary	Salary
Department	Departme

In the previous example both columns had the correct SIBAS item name, copied directly from SIBAS, and so we needed to change the left column to get more reasonable data element names. Here both columns have the data element name defined with the command EDIT-TABLE, and therefore we need to change the right column to correspond to the SIBAS item names. As the names in this column are used by ACCESS to retrieve data from the SIBAS data base, the names must be identical to those defined in SIBAS-DRL.

In addition, the group items and sets in the SIBAS data base can be defined with the commands EDIT-GROUPS and EDIT-SETS. This should be done to ensure maximum efficiency. (EDIT-SETS is only relevant if you have more than one SIBAS table in the data base and the corresponding realms are connected by a set.)

4.6 HOW TO DELETE A TABLE DESCRIPTION

By giving the command:

****DELETE-TABLE↵**

with the table name as parameter, the table description will be deleted. The file itself will not be affected. The table may occur in other descriptions, and all descriptions where the table is included will be deleted. For instance, a deleted table will not be displayed on the screen, even if the ISAM files still exist.

4.7 HOW TO GET A LIST OF AVAILABLE TABLES IN THE DATA BASE

The command

****LIST-TABLES↵**

displays a list of all the tables in the data base.

4.8 TERMINATING THE TABLE-DEFINITION LEVEL

To terminate level 2 - the table-definition level - and go to the start level, press the



key, or give the command:

****EXIT↵**

By giving EXIT twice from this level, you leave the DBA program.

The command CREATE-TABLE can also be used from the start level in ACCESS. Here it is used to edit a table or to create a new table.

4.9.1 CREATING A TABLE FROM SCRATCH

*CREATE-TABLE←┐

Enter table name: This must be a table name that
 does not already exist.

Enter file name: For ISAM or FLAT tables, enter the name of the file on which the data in the table is to be stored. For SIBAS tables, enter the realm name.

The following picture will be displayed on the screen:

[illegible]

You can now start to enter the information you want into the screen picture.

Each line represents a data element in the table, and the ordering of the data elements within the table is the same as on the screen.

Each data element is described by the following three fields:

Data element name A name of up to 32 characters, which is unique within the table. The uniqueness will be checked by ACCESS.

Key type Possible key types are PRIMARY, SECONDARY or none. See p. 31 for a more complete explanation.

Keys are not used with flat files.

Data type

It is possible to use the data types defined with the EDIT-TYPES command in the DBA-program. In addition, some predefined data types are available. If you are using a data base that already exists (for example, a SIBAS data base) you will need the data types used in that data base. However, if you are creating a new data base for ACCESS, the types CHARACTER, NUMERIC and FILLER will usually be sufficient:

CHARACTER-n

Use this for all text items. It can contain up to n characters. For example, a data element using the data type CHARACTER-3 can contain up to 3 characters. It may also contain numbers, but you will not be able to do arithmetic on them. This type corresponds to the storage format TEXT in DBA.

NUMERIC-n

Use this for all data elements that are to contain numbers on which you want to perform arithmetic. "n" is the maximum number of characters the number occupies: the number of digits plus one extra for the decimal point, if you plan to use decimals.

FILLER-n

If you define an item as filler, ACCESS will simply disregard it, and so it will not be used in the table. This is useful if you want to define different tables that use the same data stored on the same file. Example: You want to define a table using the same data as the "employees" table in the ENGLISH-BASE. This table has been defined as follows:

NAME	CHARACTER-32
MANAGER	CHARACTER-32
SALARY	INTEGER-4
DEPT	CHARACTER-20

If you want to define a table containing only the names and salaries, you can do it like this:

NAME	CHARACTER-32
MANAGER	FILLER-32
SALARY	INTEGER-4
DEPT	FILLER-20

Of course, the SINTRAN file name for the new table must be defined to be the same as that for the old table.

The remaining predefined data types may be necessary if ACCESS is to use data created by other programs, or data in a SIBAS data base. These data types are:

INTEGER-2	- A 16 bit integer
INTEGER-4	- A 32 bit integer
REAL-4	- A 32 bit floating real
REAL-6	- A 48 bit floating real
REAL-8	- A 64 bit floating real
BCD-n-m	- A BCD variable with n digits before the decimal point and m digits after

Some more details on the storage formats corresponding to these data types can be found on page 20.

You can move between the fields with the following cursor keys:



Move to the field above.



Move to the field below.



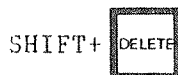
or



Move to the next field. The next field can be the first field on the line below.



Move to the previous field. The previous field can be the last field on the line above.



Deletes the current item.

CTRL+L

Inserts a new item before the current item.



or



Terminates the command without storing the table description.



Inserts the table description into the Data Dictionary

4.9.2 CHANGING AN EXISTING TABLE DEFINITION

The example above showed how we can define a table from scratch.

What will happen if you use the command *EDIT-TABLE to modify a table that already exists?

To change the description of the "employees" table, you would start with the command (you would have to enter ACCESS under the user name ACCESS to actually do this):

*CREATE-TABLE ENGLISH-BASE EMPLOYEES←J

ACCESS will respond:

Enter the new table name: The old table name will be displayed.

If you change it, there will be two tables: the old one, with the old name; and a new one, with the changed name. This can be useful if you want to create a new table much like the old one.

Enter table type: The table type can be modified. You can, for instance, change a table from FLAT to ISAM. The data has to be reloaded with the *LOAD-DATABASE command.

Enter file name: The SINTRAN file name/SIBAS realm name can be modified.

The items of the table are displayed on the screen, and you can edit them. An example:

TABLE DEFINITION		
Item name	Data type	Key type
Customer name	character-30	primary
Address	character-40	
City	cityname	secondary
Telephone number	telephone	secondary
.....
.....
.....
.....
.....
.....
.....
.....

4.10 CREATING VIEWS

Different users may have the right to access different parts of the same mass of data. Since access rights can only be defined for a whole data base in ACCESS, this must be done by defining different ACCESS data bases against the same files or the same SIBAS data base. As an example, let us say two users, USER-A and USER-B, are to be allowed to work with different parts of the same SIBAS data base:

USER-A

USER-B

Departments

Departments

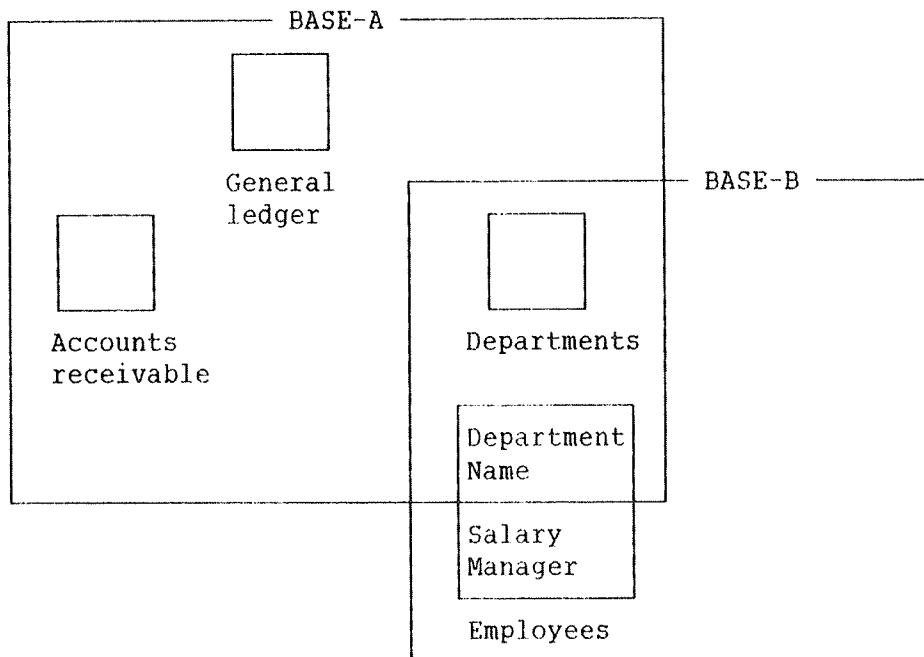
General ledger

Employees

Accounts receivable

Employees - only department and name

To make this possible, we must define two overlapping ACCESS data bases: BASE-A for USER-A and BASE-B for USER-B:



The tables Employees and Departments have to be defined in both BASE-A and BASE-B.

The Employees table must be defined in two different ways. If it is a SIBAS realm, this simply means that you define tables that do not contain all items in the realm. With ISAM or FLAT files, the fields that are not to be seen must be defined as FILLER (see p. 39).

CHAPTER 5

THE COMMAND LOAD-DATABASE IN ACCESS

CHAPTER 3

THE COMMAND-LINE DATABASE IN ACCESS

5 THE COMMAND LOAD-DATABASE IN ACCESS

Only the user who has defined the data base is allowed to use this command, which makes it possible to load new data into the data base or change the structure of files. This may for example be necessary after a new item has been added.

The sequence is started on the start level of ACCESS, with the command:

*LOAD-DATABASE <data base name>←J

Data can be loaded from two different types of files.

TEXT files Data is stored as text and must be converted before it is put into the table. The records are separated with characters for CR/LF. Such a file is made in NOTIS-WP or PED with one record on each line. The command ***WRITE-WITHOUT-FRAME also produces files of this type.

DATA files Data is stored on a binary format. The records have a constant length and are not separated by separating characters. The command ***STORE-DATA also produces such files.

We have the following commands:

****LOAD-TEXT-FILE <table name>,<file name>**
Loads data from a text file to a table.
(SIBAS, ISAM or "FLAT".)

****LOAD-DATA-FILE <table name>,<file name>,<record length in bytes>**
Loads data from a data file with a given
record length to a table. (SIBAS, ISAM or "FLAT".)

****OPTIMIZE-TABLE <table name>**
Sorts the table according to desired index key(s)
and reloads the table.
Only for ISAM tables.

The commands **LOAD-TEXT-FILE and **LOAD-DATA-FILE will give the following picture for the table EMPLOYEES (after you have been asked whether you want default values inserted):

EMPLOYEES	NAME	MANAGER	SALARY	DEPARTMENT
→ Type:				
→ Start:				
→ Length:				

If you have answered "yes" to the default value question, ACCESS will complete the table with values in the fields. You may change these values if necessary.

The parameters Type, Start, and Length have the following meaning:

Type indicates the field's data type. If you use text files, the type should always be CHARACTER. Numbers are converted automatically from text to binary format.

If you use data files, you can have the following types:

CHARACTER	Text field
NUMERIC	Numeric text
INTEGER-2	16 bit integer
INTEGER-4	32 bit integer
REAL-4	32 bit real
REAL-6	48 bit real
REAL-8	64 bit real
BCD	BCD field

If you leave the type field blank, ACCESS will give this column a blank or zero value in all the records.

Start indicates where the field begins in the record from the input file. The first position is 1.

Length tells you how long the field is in the input file (in number of bytes). If the length is less than described in the DDI (the record description), the remainder is filled with blanks. If the length is greater, the exceeding part is deleted. If you have a field in the record that you do not want to include, you can omit this by setting start in the field following the unwanted field. The system will check that the fields do not overlap, for instance if you give the wrong start and length values. If start and length values are not given, the values defined by the DBA will be taken as default values.

The commands can also be used to redefine the data base tables. See example 3, p. 51.

Example 1: Load-text-file

Suppose that the records in the input file have the following structure:

start:	1	33	44	76	85
	NAME	BIRTH NO	MANAGER	SALARY	DEPARTMENT
no.of char:	32	11	32	9	12
rec.length: 96					

You want to load this file into the "employees" table. This means that the BIRTH NO field must be skipped during loading. Fill in the skeleton in the following way after giving the command ****LOAD-TEXT-FILE:**

EMPLOYEES	NAME	MANAGER	SALARY	DEPARTMENT
→ Type:	character	character	character	character
→ Start:	1	44	76	85
→ Length:	32	32	9	12

You terminate this sequence by pressing:



ACCESS then starts to read data from the input file. Data is presented on the screen in the usual table result format, and you can check that the formatting was successful.

You leave this sequence with the EXIT command, and you will then get the question:

Do you want to put data into the table? Y/N.

- a) If Yes, the data base is loaded with the registered data after the question of whether or not to delete old data is answered.
- b) If No, you will get the question:
Do you want to edit the table description? Y/N
 - i) If Yes, you can change the record description: type, start, length.
 - ii) If No, the command is terminated.

Example 2: Optimizing the table

When you use a table to delete, update, and insert data, the contents of the table become unorganized after a time. This means that the search time becomes long.

To arrange data in such a way that the search time is reduced, you can use the command:

****OPTIMIZE-TABLE***

This command restructures the contents of the table to the effect that the number of disk accesses will be reduced. This is achieved by removing all deleted records and sorting the data according to the most frequently used index keys.

WARNING: This command should never be given without first taking a backup of the data files. If there is an error, all the data in the table may be deleted.

As an example, let us optimize the table EMPLOYEES: Type **OPTIMIZE-TABLE EMPLOYEES

The following picture appears on the screen. Answers typed by the user are underlined:

Table name: EMPLOYEES

Search key	Duplicates
NAME	NO
MANAGER	YES
DEPARTMENT	YES

How many keys do you want to sort on: 1

1. The sort key is: DEPARTMENT

The data in the table is now sorted, and the ISAM file regenerated. The table is now optimized for searching in the column DEPARTMENTS.

Note that the command only applies to ISAM files, and not for SIBAS realms and FLAT files.

Example 3: Changing the table definition

If you want to change some of the items or add new items in an existing table, this must be done in three steps.

- 1) Use ACCESS to print out data in the original table with ***STORE-DATA (or ***WRITE-WITHOUT-SKELETON)
- 2) Enter the DBA and redefine the table description.
- 3) Use ACCESS to load data to the new table using **LOAD-DATA-FILE (or **LOAD-TEXT-FILE) Answer "yes" to the question on whether old data is to be deleted.

Example: You want to change the table EMPLOYEES so that NAME and MANAGER are to have 40 characters. SALARY is to be replaced by SAL. GROUP, and you want to extend the record to include the item DA.OF EMP. (date of employment).

- 1) The contents of the old record is stored. We execute the following query:

EMPLOYEES	NAME	MANAGER	SALARY	DEPARTMENT
→	print.	print.		print.

With the command *****STORE-DATA**, write the result to the file TEMP:DATA. The record length is 32+32+20=84 bytes.

- 2) Then enter DBA and change the table description of EMPLOYEES.
- 3) Enter ACCESS and write the following:

***LOAD-DATABASE ENGLISH-BASE↵**

****LOAD-DATA-FILE EMPLOYEES TEMP:DATA↵**

Answer "no" to the question on default values, and "84" to the question on record length. The following table appears on the screen, and you fill it in thus:

EMPLOYEES	NAME	MANAGER	SAL.GROUP	DEPARTMENT	DA.OF EMP
→ Type:	CHAR	CHARACTER		CHARACTER	
→ Start:	1	33		65	
→ Length:	32	32		20	

The value of SAL.GROUP and DA.OF EMP is set to 0 or blank, depending on whether the fields are defined as numeric or alphanumeric. The extra, new positions in NAME and MANAGER are blank.

C H A P T E R 6

GENERAL INFORMATION ON RUNNING THE ACCESS SYSTEM

CHAPTER 8

GENERAL INFORMATION ON MONITORING THE ACCESS SYSTEM

6 GENERAL INFORMATION ON RUNNING THE ACCESS SYSTEM

6.1 THE COMMAND FILES

The system uses the command files ACCESS-HELP:TEXT and DBA-HELP:TEXT. Before the system is started, the desired language version must be copied to these files from the files ACCESS-NO-D:TEXT or ACCESS-END:TEXT - which come with the system.

Different SINTRAN-users may have personal versions of ACCESS-HELP:TEXT under their own user areas, so that they may use the language version they prefer. It is also possible to use NOTIS-WP on the HELP files and change the commands, both in ACCESS and DBA. You may translate the HELP files to another language, so that all commands, messages and help information is given in this language. It is important that no lines are deleted or added in the HELP files. You may only overwrite existing lines. If you want to delete a command, so that the user(s) will not be able to use it, there are two different ways of doing it, depending on the type of command:

- 1) If the command is an operator (one of the table frame commands that must be followed by a period), replace it with the word VOID.

For example, if you want to prevent a user from deleting records in the data base, replace the word DELETE with VOID in the following line in the user's ACCESS-HELP file:

```
DELETE                                % 2 Delete a record
```

The user will then be unable to use the operator "delete." in the table frame. The words to the right of the "%" sign are treated by ACCESS as comments. They have no effect on how the system works.

- 2) If the command is not an operator, replace it with blanks.

To prevent a user from using the command RESERVE-DATABASE, blank out "RESERVE-DATABASE" in the following line in the user's ACCESS-HELP file:

```
RESERVE-DATABASE                     % Lock the tables
```

NOTE: If the user has a file called ACCESS-HELP.INIT, ACCESS will read from this file instead of ACCESS-HELP:TEXT. This means that, to make the above process work, you must delete the file ACCESS-HELP:INIT or dump the command file (this is described in the next section).

We suggest that you protect the HELP files by giving them "read only" access. It will be confusing for the ACCESS users if the command names are changed too often.

If ACCESS-HELP does not exist under the user area from which ACCESS is run, nor under user SYSTEM, the version under user DATA-DICTIONARY is always used.

6.1.1 THE COMMAND DUMP-HELP-FILE IN ACCESS

The command DUMP-HELP-FILE, on the start level in ACCESS, converts the command file into a form that is read faster by ACCESS, and places the result in a file called ACCESS-HELP:INIT. It is used in the following way:

- 1) Check that there is a file called ACCESS-HELP-SYMB under user DATA-DICTIONARY, or, if you are doing this with a special version of the command file for a single user, under that user.
- 2) Create a continuous file (size 60 pages) called ACCESS-HELP:INIT under the same user. If the user already has this file, it must be deleted and created anew. (In other words, this file must be empty.)
- 3) Enter ACCESS and give the command DUMP-HELP-FILE. ACCESS will respond with the message: "the command file has been dumped".

6.1.2 THE DEFAULT QUERY LIBRARY AND THE DEFAULT QUERY

The default query library is the library file that is opened automatically when the user enters ACCESS. The name of this file is given in line 5 of the file ACCESS-HELP:TEXT, and can be changed by editing in NOTIS-WP.

A default query is a query that is executed automatically when a user enters ACCESS. Together with the MENU command in the command box in ACCESS, this feature makes it possible to give the user a menu on entering ACCESS (see the ACCESS User Guide).

The name of the default query is entered from the first character position in the empty space at the beginning of line 6 of the file ACCESS-HELP:TEXT.

6.2 MULTIUSER QUERY LIBRARIES

It is possible to have a query library that can be used simultaneously by more than one user. This file's SINTRAN access codes (OWN, FRIEND and PUBLIC) must all be restricted to READ (R).

Example: You have a query library called COMMON-LIB:TRAN under user ACCESS. On this file you have placed "standard" queries that are to be used by several different users. To make this possible, log in as user ACCESS and give the command:

```
@SET-FILE-ACCESS COMMON-LIB:TRAN R R R
```

6.3 SOMETHING IS WRONG WITH THE WORK AREA

Possible reasons for this error message are:

- 1) The work area files do not exist or are located on a different user area, to which you do not have read and write access.
- 2) The work area files are being used by another user. Check the user definitions to see if two users share the same work area. If so, change one of the two work area names.

(See page 17 on how to define users).

6.4 DDI NOT ACTIVE

This message appears if someone tries to run the ACCESS main system without the DDI process being active in a batch process. It may be that you have forgotten to start the batch process "ACCESS-DDI-D:BATC" (see chapter 2). It may also be caused by the ACCESS process being unable to communicate with the DDI process because the numbers in the ACCESS-HELP and ACCESS-DDI-D:BATC files for communication devices do not correspond, or do not exist in the operating system (see next section).

6.5 COMMUNICATION PROBLEMS BETWEEN THE DDI AND ACCESS

The communication between the ACCESS main system and the active DDI process uses a mechanism in SINTRAN called "internal devices". An internal device is a buffer area in SINTRAN used to transfer data between processes. One process reserves the buffer for reading, the other for writing. Two INTERNAL DEVICES are needed for ACCESS to be able to communicate with DDI.

All internal devices are given numbers at SINTRAN system generation (see SINTRAN manuals). ACCESS usually uses device 128 and 129 for communication with the DDI, but other numbers may be assigned if these devices are already in use by other systems.

The numbers of the devices that are used are contained in the beginning of the files ACCESS-HELP:TEXT and ACCESS-DDI-D:BATC.

The numbers in these two files must correspond!

If the devices are occupied by other systems, vacant SIBAS devices may be used (from 160 to 166 or 172). Remember to alter the numbers of the internal devices in both the ACCESS-DDI-D:BATC file and the ACCESS-HELP file.

6.6 IMPROVEMENTS IN THE ACCESS SEARCH TIME

A SINTRAN file is a data area on a disk which can be continuous or indexed. Searching is usually faster on a continuous file, but the drawback comes when the file is full, because it may happen that the file cannot be expanded on a continuous disk area. An indexed file is expanded dynamically, but the search is based on an index table of the pages. The index table is maintained by the SINTRAN file system.

Both file types (which has nothing to do with :TYPE) are created by the command:

```
@CREATE-FILE < SINTRAN file name >,< number of pages >+1
```

If <number of pages> is equal to 0, the file becomes indexed. Any other number gives a continuous file.

ISAM and SIBAS handle continuous files fast, because the addresses of the records lie in the index table of the files. It is therefore not necessary to lookup in the file system's index table first and then in SINTRAN's index table.

When a continuous file is full, it must be expanded if possible. If this is not possible, you must redefine the file with a so-called "unload" and "load" operation (see chapter 5). Queries on continuous files go faster, but you lose flexibility.

Note that an indexed file may be copied onto a continuous file and vice versa.

Special tools are developed to maintain SIBAS and ISAM files. They are not integrated for this release of the DBA. If errors occur in the files, if records disappear or strange error messages appear, it is possible to use the program ISAM-SERVICE for ISAM files to check that the files are consistent. You may also execute so-called "unload/load" operations from this subsystem.

The general documentation of the SINTRAN operating system is very extensive. You are advised to read the SINTRAN III Time-sharing/Batch Guide (ND-60.132) which describes the most important features of the file system.

6.7 OTHER ERROR MESSAGES

Usually an error message from ACCESS is in clear text which may be followed by an error code number.

The error codes correspond to SINTRAN error codes, which are also described in the ACCESS user guide.

SIBAS and ISAM may also return error messages (described in the SIBAS and ISAM manuals). SIBAS will print out the error code on the console ("error device") if anything serious occurs. ISAM will always try to return to ACCESS with an error status, but situations may occur when ISAM "collapses". A message corresponding to this one will be printed out:

ISAM FATAL ERROR IN ROUTINE 'ZABUF'/1

and control is returned to SINTRAN. In this case you should check whether the file has been restructured, whether ACCESS is running against the correct file, etc.

CHAPTER 7

COMPLETE EXAMPLES

CHAPTER 7

COMPLETE EXAMPLES

7 COMPLETE EXAMPLES

7.1 EXAMPLE OF HOW TO DEFINE USERS

Log in as SINTRAN-user DATA-Dictionary. Stop the DDI process as described in section 2.1, and enter the DBA program by typing:

```
@DBA←
```

We assume that the system has been delivered with the test data base that is used in the manual. User ACCESS is then already defined as a DBA user, with no password.

The system will prompt for user identification, and you answer A, which is an abbreviation of "ACCESS".

After a little while, a HELP picture is displayed on the screen, showing the available commands on level 1.

First you have to define who is to be allowed to run the ACCESS part of the system. Then you must state which of these users are allowed to use the DBA program. Start by giving the command:

```
*ED-US←
```

which is an abbreviation of *EDIT-USERS. A table of users where ACCESS is already defined is displayed on the screen. Define MY-SELF and ALBERT-NEUMANN as users:

ACCESS users		
ACCESS user name	Password	Work area
ACCESS		(ACCESS)WORKAREA
MY-SELF	ME	(ACCESS)WORKAREA-1
ALBERT-NEUMANN		(ALBERT-NEUMANN)WORKAREA-2

The user MY-SELF must from now on give the password ME when entering ACCESS, whereas ALBERT-NEUMANN will not be asked for a password.

MY-SELF must always log in as SINTRAN-user ACCESS (or a user defined as "friend" to this user). Otherwise MY-SELF will not be able to use the work area (ACCESS)WORKAREA-1.

Go out of the table by pressing the Home key.

You want to define user MY-SELF as a DBA user. MY-SELF is to be allowed to enter the DBA program and create data bases; ALBERT-NEUMANN is only allowed to operate ACCESS against data bases defined by others.

The command

*DBA←

which is an abbreviation of *DBA-USERS, gives a table of DBA users on the screen. User ACCESS is already defined, and you type in MY-SELF:

Data base Administrators
DBA user
ACCESS
MY-SELF

Now you have to exit the DBA program, so press the EXIT key (or give the command *EXIT). When you define a data base, the user name you gave when you entered the DBA will be registered as the owner of the data base description. In this case you had entered as DBA user ACCESS, and since you want MY-SELF to be the owner of the data base you are to define, you exit the DBA program and re-enter it, this time with MY-SELF as DBA user.

7.2 EXAMPLE OF HOW TO DEFINE A DATA BASE

You want to define the data base EXAMPLE-BASE with MY-SELF as the owner. MY-SELF will then be the only user who is allowed to modify the definition of this data base.

Enter the DBA program by typing

@DBA←

and give M-S (short for MY-SELF) and the password ME. You enter the start level (marked by * in command position), and give the command:

*CREATE-DATABASE EXAMPLE-BASE←

You now enter the table-definition level, level 2.

Here you have to define some data types for the data elements. Give the command:

**EDIT-TYPES←

The following table appears on your screen and you fill it in as indicated here:

Data types		
Name of data type	Display code	Storage format
PERSON-NAME		TEXT(32)
MONEY		INTEGER4
DEP-NAME		TEXT(20)

It is sufficient to fill in the name field and the storage format, and leave it to the system to choose values for the display code.

You can now define a table which you later may define and run queries against.

Give the following command:

```
**CREATE-TABLE EMPLOYEES←
```

DBA will ask for the table type:

Enter table type (ISAM,SIBAS or FLAT): ISAM

and then the following picture appears on the screen and is filled in as shown below:

EMPLOYEES			
Name of data element	Data type used	Number	Sear
NAME	PERSON-NAME	1	
MANAGER	PERSON-NAME	2	
SALARY	MONEY	3	
DEPARTMENT	DEP-NAME	4	

EMPLOYEES			
Number	Search key	Column width	Result length
1	P		
2	S		
3			
4	S		

For column width and result length, we let the system choose values.

Then the ISAM files must be defined. Give the command:

****ISAM-FILES+J**

and the table is filled in as shown:

ISAM Files	
Table name	SINTRAN file name
EMPLOYEES	(ACCESS)EMPLOYEES

The table "EMPLOYEES" is now connected to the ISAM files EMPLOYEES:DATA and EMPLOYEES:ISAM under SINTRAN-user ACCESS. The first file is to contain all the data, the second is to contain the search-key tables.

When you have defined the table, you must define who is allowed to operate on the table with ACCESS. Use the command EDIT-ACCESS, and the table is filled in as shown:

User access to data base				
ACCESS user name	Print.	Insert.	Update.	Delete.
MY-SELF	YES	YES	YES	YES
ALBERT-NEUMANN	YES	YES	YES	NO

The user "MY-SELF" is allowed to access and manipulate the data in the table, while the user "ALBERT-NEUMANN" is not allowed to delete records in the data base.

Note that the commands *LIST-DATABASES and *EDIT-DATABASE may now be used, since a data base and a table exist.

This completes the definition of a table, and you go back to SINTRAN by pressing the EXIT key twice.

To give ACCESS the information defined in the DBA program, the DDI process must be active. This is done by attaching it to a batch process with the SINTRAN command:

@APPEND-BATCH 1,ACCESS-DDI:BATC,ACCESS-DDI:LOG+J

If this has been included as a terminating command in the file DBA-HELP, it will be executed automatically each time you exit the DBA program, see section 2.1.

This completes the definition of the data base, the DDI is active, and ACCESS may be used as described in the ACCESS User Guide.

A P P E N D I X A

THE DDI PROCESS AS A BATCH PROCESS

A P P E N D I X

THE DBI PROCESS AS A BATCH PROCESS

ACCESS requires information regarding data bases, access rights, etc. This data base description, which is defined in the DBA program, is normally stored on files under user DATA-DICTIONARY, and it is the same for all users running ACCESS.

An active DDI process keeps all descriptions entered by the DBA program, and conveys this information to ACCESS, when ACCESS asks for various information. Two "internal devices" are currently used to perform the exchange of messages between the various processes.

The DDI process is an ordinary program, which never stops by itself when it has been started. The program is on the file (DATA-DIC)ACCESS-DDI-D:PROG. The program may be started from a terminal, but occupies this terminal as long as it is running. To avoid tying up scarce and costly resources, it should be started from a batch processor. But note that this batch processor will be occupied for as long as the DDI process is running.

Inform the other users of the computer that they must not use the batch processor which is running the DDI process. If they do they will have to wait for quite a long time!

A batch processor must get information on what is to be done from a file called the input file of the process. It also needs an output file for, among other things, error messages. To start the batch process, use the the file

(DATA-DIC)ACCESS-DDI-D:BATC

as the input file, and

(DATA-DIC)ACCESS-DDI-D:SYMB

as the output file. The latter should be empty when you start the DDI process. Give the command @LI-FI to check that it has been created!

The system is delivered with the following input file, ACCESS-DDI-D:BATC:

```
@ENTER DATA-DICTIONARY,,,32000
@(DATA-DICTIONARY)ACCESS-DDI-D:PROG
128
129
CC <DDI PROCESS> CC
@
```

The first line in this file tells the batch process that the DDI process is to use the data base description on the user area DATA-DICTIONARY, and that it is to run as long as possible (32000 minutes).

You may have to modify this line. Its general format is:

```
@ENTER,<SINTRAN user>,<password>,<project password>,<maximum time for
the process>
```

The numbers of the internal devices are to be entered in lines 2 and 3. These numbers are also found in the file ACCESS-HELP:TEXT, and in a mode file, called ACCESS-DDI-STOP:MODE which is used to stop the DDI process. This file contains the following:

```
@(DATA-DICTIONARY)ACC-DDI-STOP  
128  
129
```


A P P E N D I X B

SYSTEM SUPERVISOR INFORMATION

APPENDIX B

SYSTEM SUPERVISOR INFORMATION

In the examples in this appendix, comments are preceded by a "@" character.

HOW TO SET THE XON/XOFF OPTION

If ACCESS is to work properly on the NOTIS terminal TDV-2200/9-ND-NOTIS the "communication handshake" must always be set to xon/xoff. This should be done during installation, but there might still be terminals where it has not been done. This will result in "messy" screen pictures. The procedure for setting xon/xoff is therefore given here:

Enter the menu for communication switches on your terminal (for details look at the Tandberg manual).

Type CTRL + HELP twice and you enter the CONFIGURATION MENU. Use the ↓ key until you have positioned the cursor on the line "Communication switches". Press the ENTER key. You get the following picture on the screen:

C o m m u n i c a t i o n S w i t c h e s	
Send Receive Mode	Simultaneous
Echo	External
Online	Toggle
Communication Clock	ASY
Communication Handshake @Must be set to <u>XON/XOFF</u>
Modem	Inhibit
.	
.	
.	
.	
.	

Move the cursor with arrow keys, and select with the ENTER key.

Log in as user SYSTEM and find the terminal number with @WHO. Here we will use terminal number 53 as an example. Perform the following procedure (the underlined texts must be typed by you):

```
@SINTRAN-SERVICE+  
*CH-DATA+  
LOG.UNIT NO.: 53D+      © Terminal 53 is used as an example  
INPUT/OUTPUT: I+  
MEMORY? Y+  
IMAGE? Y+  
SAVE-AREA? Y+  
  
MEMORY      IMAGE      SAVE-AREA  
  
DFLAG/ <value> <value> <value> 101000+  
+  
*EXIT+  
@
```

SETTING BATCH PRIORITY FOR USE WITH ACCESS

The batch process, or processes, which is to be used by ACCESS should run at a higher priority than ordinary batch processes. Otherwise, response times for the ACCESS users will be unnecessarily long. Like xon/xoff, this should be taken care of during installation. However, if it becomes necessary to start using a new batch process for ACCESS, the following procedure should be performed (here batch no. 3 is used as an example):

Log in as user SYSTEM.

Find the logical unit number of the batch process. These numbers are assigned according to the following system:

Batch no	Logical unit no
1	670
2	672
3	674
4	676
.	.
.	.
.	.

For example, the logical unit number for batch no. 3 is 674. Continue as follows:

```
@SINTRAN-SERVICE+  
*REMOVE-FROM-TIME-SLICE 674D+  © Logical unit number  
MEMORY? Y+  
IMAGE? Y+  
SAVE-AREA? Y+  
*EXIT+  
@PRIOR BCHO3 42D+      © Set the priority to 42
```

The priority can be checked with the command:

@LIST-RT-DESCRIPTION BCH03

For this batch process, the priority should now be 42. For an ordinary batch process, it will usually be less than 20.

RESETTING BATCH PRIORITY

If a batch process is no longer to be used with ACCESS, you should reset it to "normal" with the following procedure:

@SINTRAN-SERVICE
*INSERT-IN-TIME-SLICE 674D @ Logical unit number
MEMORY? Y
IMAGE? Y
SAVE-AREA? Y
*EXIT

A P P E N D I X C

THE CONTROL COMMANDS IN THE DBA MODULE

APPENDIX C

THE CONTROL COMMANDS IN THE DBA MODULE



or CTRL+D CTRL+D

deletes the field in which
the cursor is positioned.

SHIFT +



or CTRL+D CTRL+L

deletes the whole line where
the cursor is positioned



or CTRL+G

displays the available commands
in the current situation.



or CTRL+O

prints the screen picture on a
file. (You will be asked for
the file name.)



or CTRL+A

deletes one character - where
the cursor is positioned.



or CTRL+E

makes it possible to insert
characters in front of the
cursor position.

Index

Abbreviations	12.
Available data bases	25.
Batch	
priority	76.
process	3, 71.
Calc keys	35.
Column width	32.
Command files	55.
Commands	12.
Commands, control	81.
Communication problems	57.
Control commands	81.
COPY-DATABASE	34.
COPY-REALM	34.
CREATE-ACCESS	29.
CREATE-DATABASE	24, 25.
CREATE-TABLE	37.
CREATE-USER	18.
Daily use	13.
Data	
dictionary	3.
element definition	30.
Data base	3.
definition	23, 64.
deletion	25.
description	9.
Data bases, available	25.
Data element definition	19.
DBA	3, 9, 12.
DBA-USERS	18.
DBA user definition	18.
DDI	
files	3, 13.
process	3, 71.
start	9.
stop	9.
Defining	
data elements	19.
DBA users	18.
users	17.
DELETE-DATABASE	25.
DELETE-TABLE	36.
Devices, internal	11.
DUMP-HELP-FILE	56.
EDIT-ACCESS	29.
EDIT-DATABASE	25.
EDIT-TYPES	19, 30.
EDIT-USERS	17.
Error messages	58.
EXIT	26, 36.
Files	33.
Files,	
command	55.

FLAT	3, 30, 32.
ISAM	3, 30, 33.
FLAT files	3, 30, 32.
Handshake	75.
Help information	55.
Independent data base description	11.
Internal devices	11.
ISAM files	3, 30, 33.
ISAM-FILES	33.
Level,	
start	12, 17.
table definition	12, 29.
Library, multiuser	56.
LIST-DATABASES	25.
LIST-REALMS	34.
LIST-TABLES	36.
LOAD-DATA-FILE	47.
LOAD-DATABASE	47.
LOAD-TEXT-FILE	47.
Multiuser library	56.
OPTIMIZE-TABLE	47.
Owner	24.
Password	17.
Priority	76.
Process,	
batch	3, 71.
DDI	3, 71.
Query library, multiuser	56.
Realms	3.
Result length	32.
Search	
keys	31.
time	58.
SIBAS	
realms	3.
tables	34.
SIBAS-ITEMS	35.
SIBAS-REALMS	35.
SIBAS realms	30.
SINTRAN	9.
Start level	12, 17.
Storage format	19.
Supervisor	75.
System supervisor	75.
Table	
access	29.
definition level	12, 29.
Tables	33.
Tables, SIBAS	34.
Time, search	58.
Types, undefined	23.
Undefined types	23.

UNDEFINED-TYPES	23.
User	
area	11.
concept	10.
definition	17, 63.
name	17.
WHERE-IS-TYPE-USED	23.
Work area	10.
Work area	17.
xon/xoff	75.

SEND US YOUR COMMENTS!!!

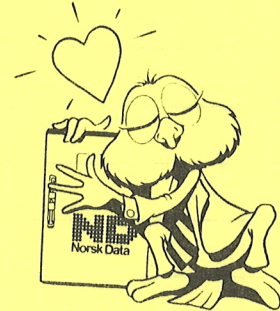


Are you frustrated because of unclear information in this manual? Do you have trouble finding things? Why don't you join the Reader's Club and send us a note? You will receive a membership card — and an answer to your comments.

Please let us know if you

- * find errors
- * cannot understand information
- * cannot find information
- * find needless information

Do you think we could improve the manual by rearranging the contents? You could also tell us if you like the manual!



HELP YOURSELF BY HELPING US!!

Manual name: ACCESS DBA Manual

Manual number: ND-30.022.03

What problems do you have? (use extra pages if needed) _____

Do you have suggestions for improving this manual ? _____

Your name: _____ Date: _____

Company: _____ Position: _____

Address: _____

What are you using this manual for ? _____

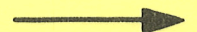
NOTE!

This form is primarily for documentation errors. Software and system errors should be reported on Customer System Reports.

Send to:

Norsk Data A.S
Documentation Department
P.O. Box 25, Bogerud
Oslo 6, Norway

Norsk Data's answer will be found on reverse side



Date _____

9515-24

Systems that put people first

NORSK DATA A.S OLAF HELSETS VEI 5 P.O. BOX 25 BOGERUD 0621 OSLO 6 NORWAY
TEL.: 02 - 29 54 00 - TELEX: 18284 NDN