Norsk Data

ACCESS
User Guide

ND-60.153.03

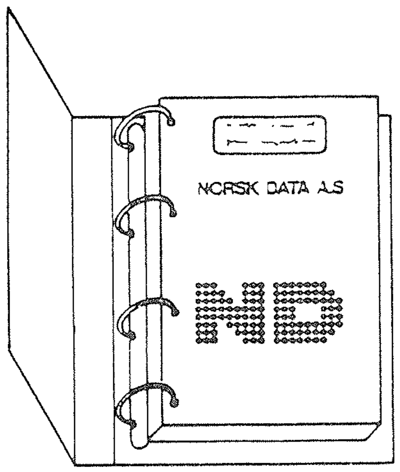
ACCESS
User Guide

ND-60.153.03

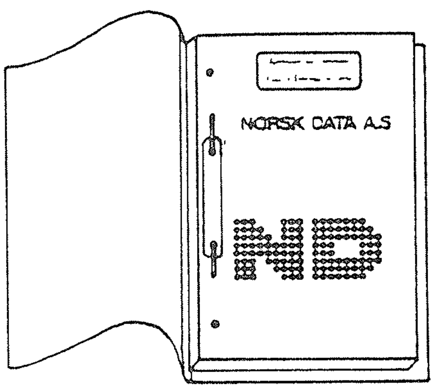
This manual is in loose leaf form for ease of updating. Old pages may be removed and new pages easily inserted if the manual is revised.

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
Documentation Department
P.O. Box 25, Bogerud
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 ....

PRINTING RECORD

Printing	Notes
11/82	Version 01
01/83	Version 02
09/84	Version 03

ACCESS User Guide
Publ.No. ND-60.153.03
September 1984

NORSK DATA A.S
P.O. Box 25, Bogerud
0621 Oslo 6, Norway

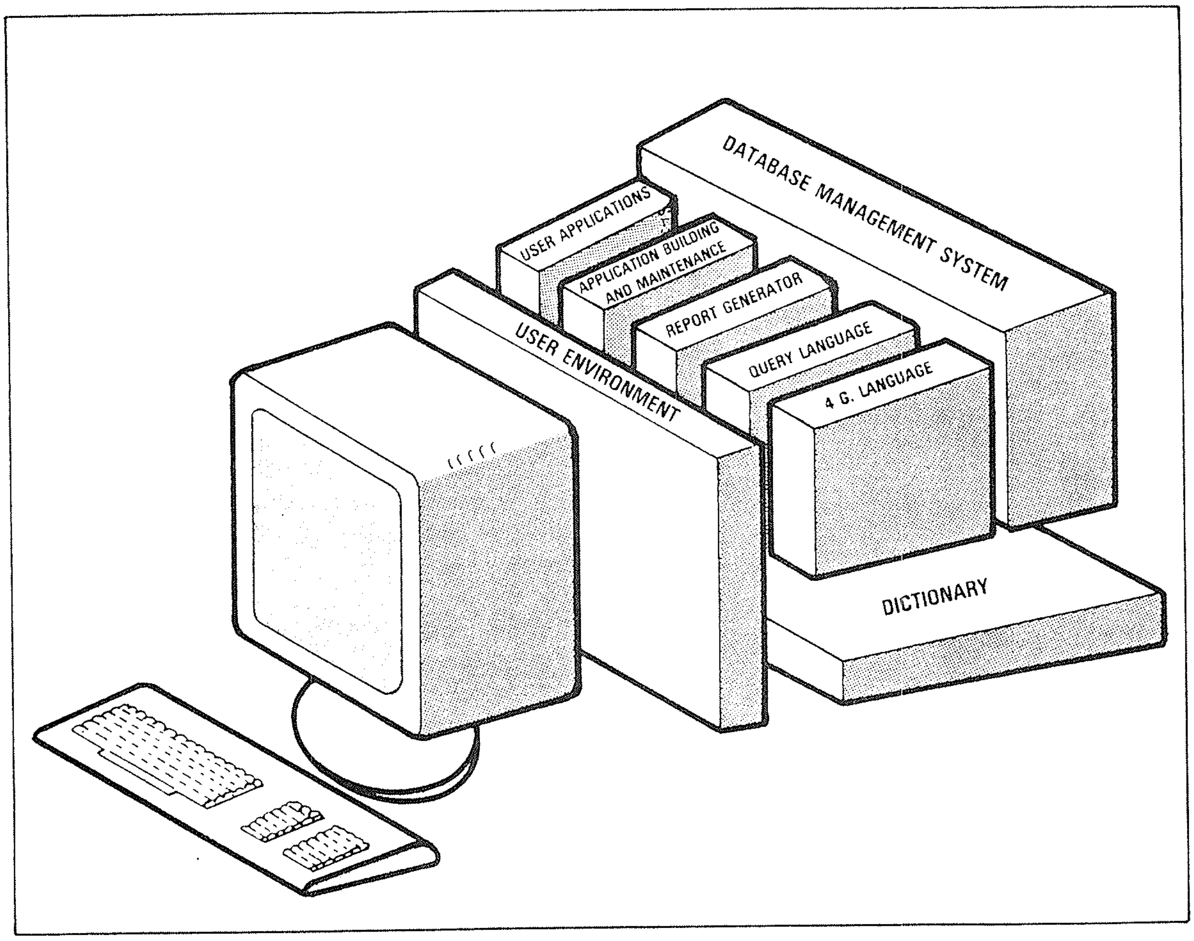
Manuals can be updated in two ways, new versions and revisions. New versions consist of a complete new manual which replaces the old manual. New versions incorporate all revisions since the previous version. Revisions consist of one or more single pages to be merged into the manual by the user, each 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 ND Bulletin 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, together with all types of inquiry and requests for documentation should be sent to the local ND office or (in Norway) to:

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


DATABASE MANAGEMENT SYSTEM
USER APPLICATIONS
APPLICATION BUILDING AND MAINTENANCE
REPORT GENERATOR
QUERY LANGUAGE
4 G. LANGUAGE
USER ENVIRONMENT
DICTIONARY

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 tailormade, 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 user guide describes ACCESS, version D. The product is registered with ND-number ND 10185D.

ACCESS can be used by anyone who is interested in using a computer based "filing system" which can contain large amounts of data.

The system requires a person called the ACCESS supervisor who will have the main responsibility for the system.

THE READER

It is not necessary for the user to have any background in data processing. This manual gives sufficient information for using the system.

THE USER GUIDE

ACCESS consists of three parts:

- The ACCESS query and transaction system
- The DDI process which gives information on data to the query and transaction system
- The DBA program which administrates the whole system

This manual describes those parts of ACCESS which the ordinary user comes into contact with. It explains how to use data bases and files.

Chapter 1 is a general introduction to data bases and the use of ACCESS.

Chapter 2 explains some elementary things such as logging in.

Chapters 3 to 8 give a detailed description of the different functions in ACCESS, illustrated by examples.

The examples in this manual use an example data base delivered with the system. If your installation has not had earlier versions of ACCESS, and no one has changed the data base, the examples should work as shown. Otherwise the results may be different.

The DBA program is described in the ACCESS DBA Manual, ND-30.022.03. It describes how data bases, users, presentation and storage formats are defined. The manual is intended for the ACCESS supervisor, and need not be read by ordinary users.

NEW FEATURES AND CHANGES IN ACCESS

The D version of ACCESS has the following new features:

- More than one table may be used in one query (see section 3.7.10).
- Queries are stored on a single query library, instead of several :TRAN files (see section 8.3). Old queries must be converted to the new format (see section 8.6).
- Tables, data bases and users may be created from ACCESS; and table, user and access descriptions modified (see the DBA manual).
- Menu selection of tables and data bases (see section 3.3).
- User defined menus (see chapters 2 and 7).
- BETWEEN has been replaced by the more general function IN (see page 47).
- The condition box (see section 3.7.12).
- The command **CLEAR-QUERY (see section 8.12).

In addition, the following changes have been made:

- Text used in a query must now <u>always</u> be enclosed in single quotes, even if it contains no blanks.
- When the update operator is used to replace a value with a new one, an equals sign <u>must</u> be used. If the equals sign is omitted, the given value will be added to the old one, instead of replacing it.

The following command names have been simplified:

Old command name	New command name
*RUN-STORED-QUERY	*RUN-QUERY
**NEW-TABLE-SKELETON	**NEW-TABLE
**DISPLAY-DATABASE-NAME	**DATABASE-NAME
***WRITE-WITHOUT-SKELETON	***WRITE-WITHOUT-FRAME
***WRITE-NOTIS-MACRO-CALL	***WRITE-MACRO-CALL
***RUN-AGAINST-RESULT	***REFINE-RESULT

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

TABLE OF CONTENTS

Section Page

C H A P T E R 1

INTRODUCTION TO ACCESS

1 INTRODUCTION TO ACCESS

1.1 WHAT CAN ACCESS BE USED FOR?

The following list gives some possible uses for ACCESS:

- Customer registers
- Subscription lists
- Inventories
- Staff registers
- Purchasing lists
- Order lists

ACCESS is used directly at a terminal. As a user, you tell ACCESS <u>what</u> you want to do, but leave it to ACCESS to find out <u>how</u>. With a few commands you can create, delete and update data contained in data bases without needing to know anything about data processing or programming.

You use so-called operators to tell ACCESS what you want to do, and selection criteria to tell the system which data from the data base you are interested in.

But what is a data base?

1.2 DATA BASES AND STORAGE STRUCTURE

A data base is something which contains information. It can be compared with a filing cabinet, a drawer or an archive.

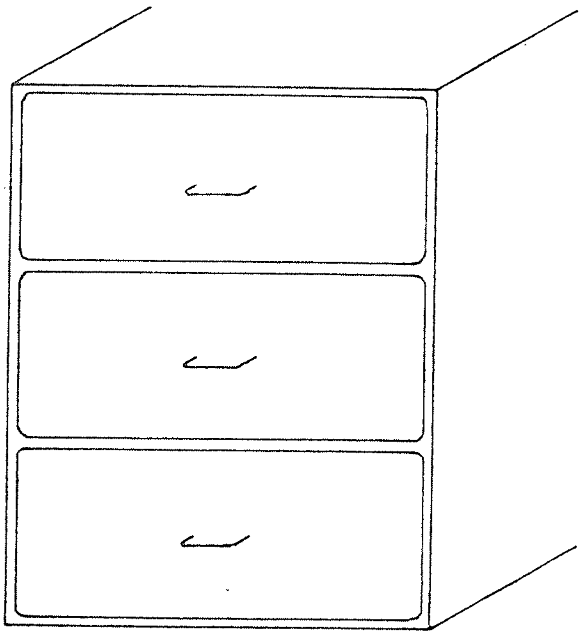


Fig. 1. A filing cabinet is a data base

The filing cabinet in fig.1 has three drawers. In other words, it is a data base divided into three parts, where each part is a drawer. In the same way the permanent storage of the computer (on magnetic disks) is divided into physical areas called files. Just as a drawer has a label on the outside showing the contents, a file has a name for identification. The file name should be descriptive, for example EMPLOYEES.

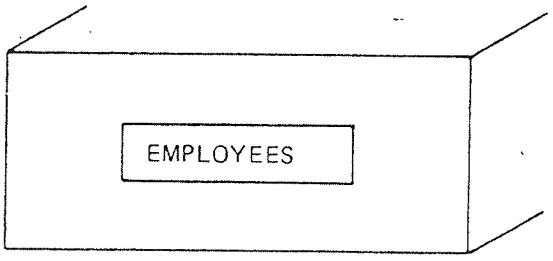
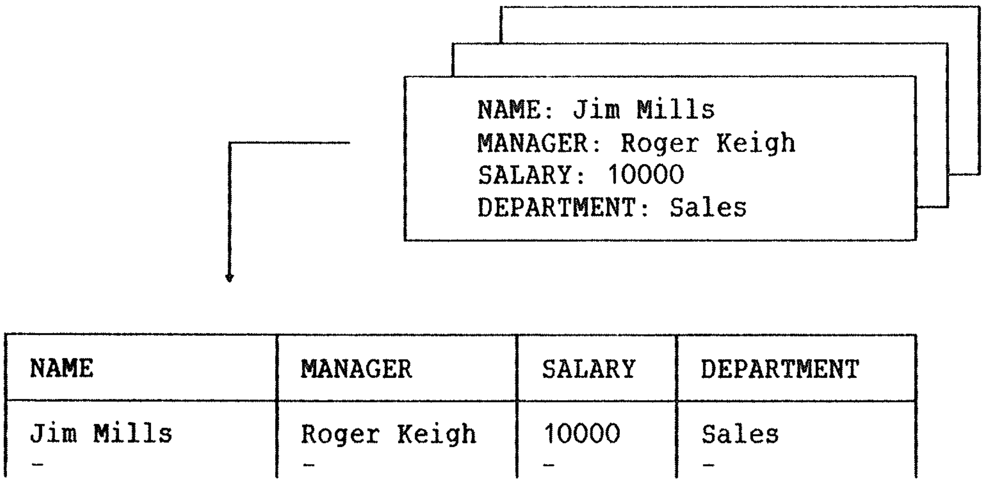


Fig. 2. Each drawer is a file and has a file name



NAME: Jim Mills
MANAGER: Roger Keigh
SALARY: 10000
DEPARTMENT: Sales

NAME	MANAGER	SALARY	DEPARTMENT
Jim Mills	Roger Keigh	10000	Sales
-	-	-	-

Fig. 3. A card in the file corresponds to a row in a table

The drawers in a filing cabinet may contain several cards. One card gives all the information about a particular object, for example an employee. In ACCESS, each card corresponds to one row in a table, where the table is the whole file. One table row is identical to one record. Each table is divided into one or more columns.

The record for one employee, Jim Mills in fig. 3 is:

Jim Mills (name), Roger Keigh (manager)
10000 (salary), Sales (department)

Each record is subdivided into fields which can be empty or contain data. For instance, if the field for boss is empty in the record for Carol Trew, this means that she has no boss or her boss' name is not known.

In ACCESS, the user works with a definite number of tables, where each table is a file <1>, and each table can contain different kinds of information. This information is represented by a set of field names, which is part of the table when it is displayed on the screen. For example, the table EMPLOYEES may contain information about an employee's NAME, MANAGER, SALARY and DEPARTMENT. This is represented on the screen as shown in fig. 3.

It is possible to have several data bases. It can be advantageous to divide information into groups of tables which contain related data. You may, for example, have the two data bases: PERSON-DATABASE and ORDER-DATABASE, each consisting of one or more tables.

<1> If you use ACCESS with a SIBAS data base, each table is a SIBAS realm.

ACCESS makes it possible for a user to handle data that is stored in a data base. The structure of the data base has been defined by the ACCESS supervisor, and you can display data on the screen; put in, delete or change data.

1.3 *AN EXAMPLE FROM THE TRAVEL AGENCY SUNSEEKER LTD.*

Take an everyday situation in the travel agency Sunseeker Ltd. Your job is to answer customers' inquiries and make bookings.

A customer wants to know which hotels in Paris have facilities for playing squash, and the price for a double room at these hotels.

Assume Sunseeker uses ACCESS to keep information about hotels around the world, how much it costs to stay there and what facilities each hotel has for sport. You will then quickly and easily be able to answer your customer's question.

With ACCESS you can search through all known hotels and pick out those which would satisfy the customer. So in a few moments you could answer that there is only one hotel in Paris with its own squash hall and that the price for a double room is 210 Fr. per night.

How did you do that?

First you call to the terminal a table frame for the table HOTELS by giving the command NEW-TABLE followed by the table name HOTELS.

The table frame consists of a table name and a set of field names (see fig. 3). This is a picture of the file HOTELS. That the frame is empty does not mean that the file is empty! The frame is only a tool for operating on the data base.

HOTELS	NAME	TOWN	SPORT	PRICE
→				

Fig. 4. The table frame

By filling in the frame with operators and selection criteria, you specify what you want to search for.

In this query you want to display information on the screen, so you use the operator PRINT. But you do not want to display the whole file HOTELS, only a small part. You do that by specifying the selection criteria, in this case 'Paris' in the field TOWN and 'Squash' in the field SPORT.

HOTELS	NAME	TOWN	SPORT	PRICE
→	PRINT.	'PARIS'	'SQUASH'	PRINT.

Fig. 5. The query description

The completed table frame is called a query description. It describes the information you require from the data base (see fig. 5). The query description tells you what type of query you want to execute (whether you want to add records, delete records, update fields, or just look at the information) and which data will be affected by the query.

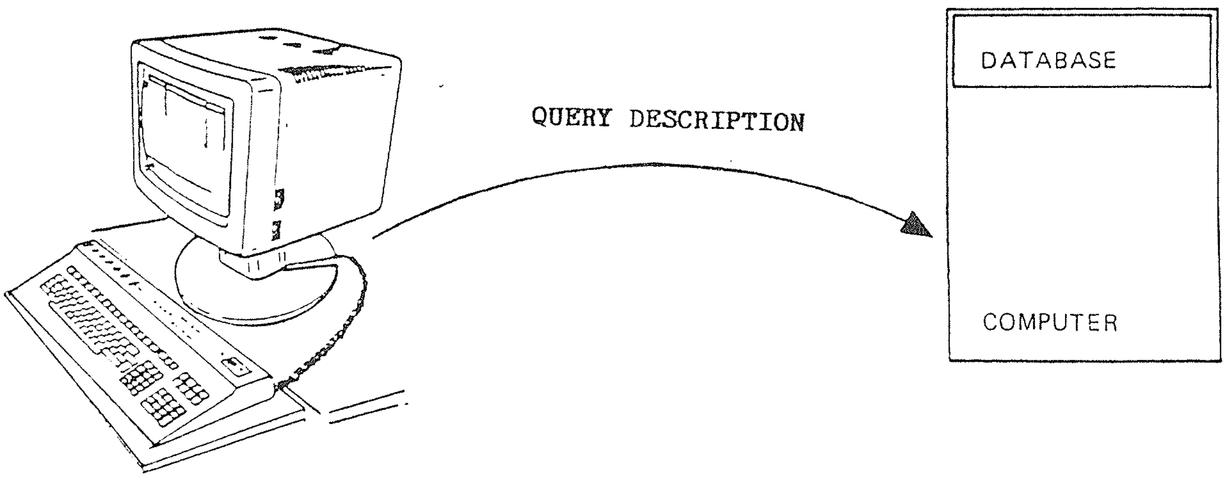


Fig. 6. The query description is interpreted and the query then executed by ACCESS

You then start the query. ACCESS interprets the query description and then executes it. If the system has found some information for you, the result of the query is displayed on the screen as a result table showing the records which correspond to the selection criteria (see fig. 7). This whole process is called a query (exchange of data between the terminal and the data base).

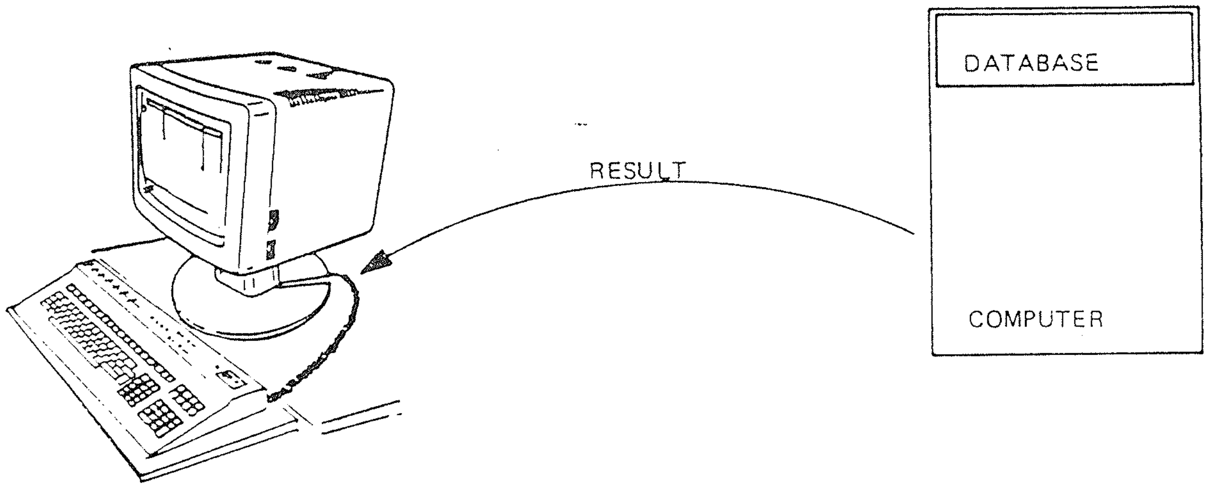


Fig. 7. The result of the query is displayed on the screen

NAME	PRICE
Hotel de France	210 Fr

Fig. 8. The result table

1.4 HOW IS ACCESS USED?

1.4.1 HOW TO USE THE TERMINAL

You communicate with ACCESS from a screen terminal.

An empty screen picture in ACCESS looks like fig. 9. In the lower part of the picture there is a horizontal line.

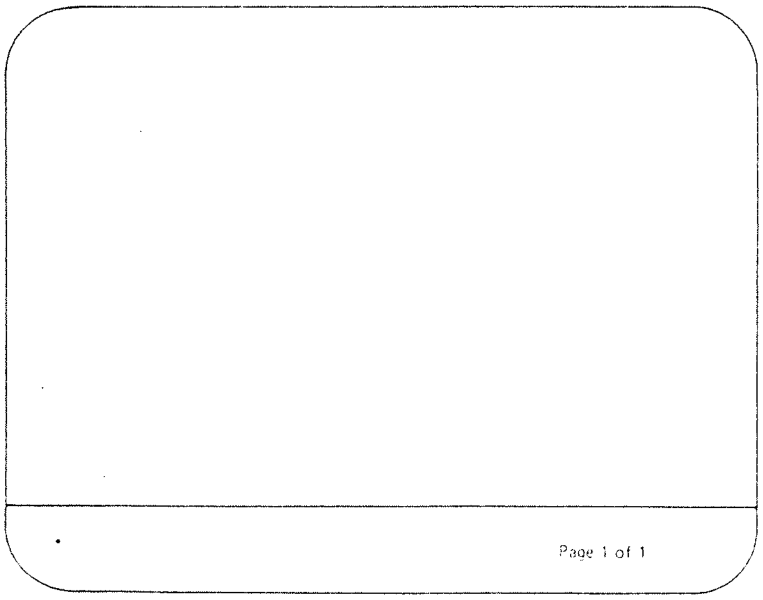


Fig. 9. An empty screen picture

The area under this line is called the command and message position. Two lines are reserved there: one line for messages from ACCESS to the user and one line for commands from the user to ACCESS. To the far left on the command line there are one, two or three asterisks (*), depending on which command level you are on. The asterisk(s) mean that ACCESS is ready to receive commands from the user.

The area above the line is called the window. Since the screen is of limited size, sometimes only part of a background picture is displayed. By using the arrow keys on the terminal you can move the window to see other parts of the background picture. The background picture is divided into horizontal pages, and the message

    Page X of Y

appears at the right of the command line (see fig. 9).

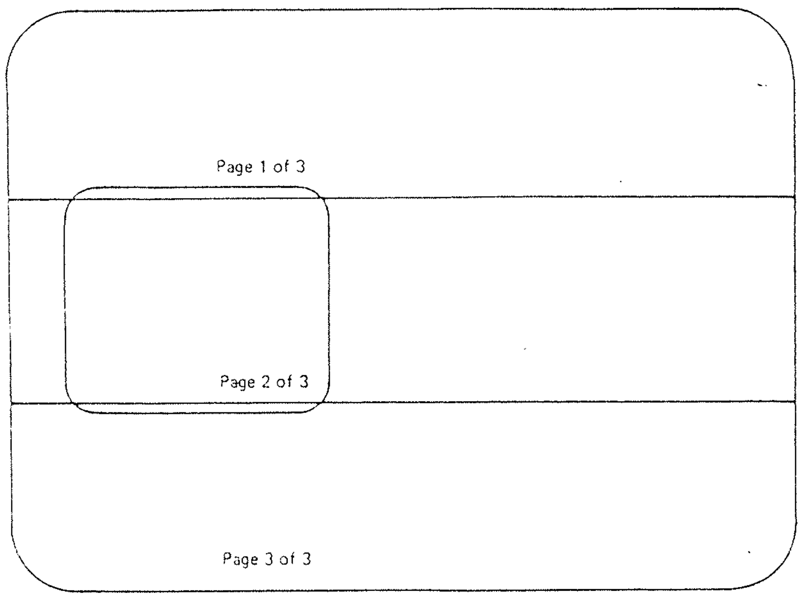


Fig. 10. The background picture is divided into horizontal pages

On the screen there is a blinking line or block called the cursor. This tells you where you are positioned to write on the screen. Whether you type in text or give commands, the cursor follows.

The arrow keys on your keyboard move the cursor around the screen.

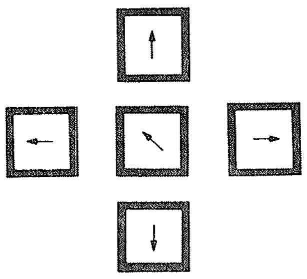


Fig. 11. The arrow keys

Down arrow	↓	moves the cursor to the same position on the line below.
Up arrow	↑	moves the cursor to the same position on the line above.
Right arrow	→	moves the cursor one position to the right.
Left arrow	←	moves the cursor one position to the left.
Home key	↖	moves the cursor to the command position (the position after the asterisk).
Forward tabulator	⇥	moves the cursor to the next column
Backward tabulator	⇤	moves the cursor to the previous column

If you press the home key (slanted arrow) in the command position, the cursor jumps back to the position it had before it came to the command position.

1.4.2 COMMAND TYPES AND PARAMETERS

Instructions from the user to the system are called commands. There are three types of commands in ACCESS:

1) The NOTIS keys and control key commands.

   These are used for editing and navigating in the screen picture and can be used anywhere in the window. Control key commands are written thus: CTRL+D. This means you first press the control key marked CTRL and hold it down while you press D.

2) Direct commands.

   These are used to carry out some types of operations, move from one level of ACCESS to another or call something onto the screen. They must always be given in the command position.

   A few direct commands require additional information, parameters, given together with the command. A command with one or more parameters is written in the form:

   COMMAND <parameter 1> <parameter 2> etc.

   Example: DELETE-QUERY <query name>.

This means you must give a query name in addition to the command, for example DELETE-QUERY QUERY-A. ACCESS will ask for parameters which are not given.

A command must always be terminated by pressing the "carriage return" key:

↵

3) Operators.

These are commands written in the table frame to describe a query. They are used to specify what you want the query to do and are interpreted and carried out by ACCESS. They are always terminated by a period.

For example:	print.
	insert.
	etc.

1.4.3 ABBREVIATIONS

Commands and parameters can be abbreviated as long as this does not make them ambiguous. For example, you can type H instead of HELP or NEW instead of NEW-TABLE. If the abbreviation is ambiguous or unknown you get an error message and must try again.

Many of the commands are split into several hyphenated words, and this makes it easy to make efficient abbreviations. This same system can also be used when you name queries. The hyphens between the words are used to indicate the separation of words. The hyphens indicate the end of the previous word and the beginning of the next word. For example, if you have the queries "ADJUST-SALARY", "ADJUST-TRAVELING-BUDGET", and "ADJUST-TRAVELING-SCHEDULE", these can be abbreviated "A-L", "A--B", and "A--S". Note the use of double hyphens. You can skip identical words.

NB!! ACCESS makes no distinction between uppercase and lowercase letters in commands and parameters, but this is not the case for data. See the next section on data types and how to enter data at the terminal.

1.4.4 DATA TYPES

Data is separated into data types, which are understood and handled in different ways by the computer.

The ACCESS supervisor defines the rules for handling records, who can work with them, if they can be updated, etc. S/he also decides legal data types for each field in the record.

Since the ACCESS supervisor takes care of the formalities, it is sufficient for you to know which data types are used in the different fields.

You only need to remember two types of data. A field can be defined as:

1) Text (or alphanumeric): a collection of characters which may be letters or digits or other symbols.

2) Numeric or numbers: written with digits. Two hundred and seven is written: 207

Text

Data of text type must always be enclosed in single quotes ('). Everything between the quotes is understood by ACCESS as symbols in the text.

Each symbol has a certain value for the computer. Upper case A is different from lower case a. In the same way 'ROLLING STONES' is different from 'Rolling Stones'. It is important to remember this when you put information in the data base and later when you are searching for it by defining the data in the table frame.

Numeric

For numeric type the rules are slightly different. A number, for instance twenty three, is written: 23

Two and a half is written: 2.5

ACCESS does not accept a comma as a decimal point.

In a record, some fields may contain text data, like the field NAME, and other fields may contain numeric data, like the field SALARY.

The ACCESS supervisor decides in advance which types are legal for each field, and if you try to put in illegal data you get an error message. For example, it is not possible to put the data value 'Jim' into the field SALARY.

1.4.5 HOW TO ENTER ACCESS

If your installation has USER ENVIRONMENT, you enter ACCESS in the same way as with other systems, through one of the menus. If you do not have USER ENVIRONMENT, however, you start by typing ACCESS after the @:

```
    @ACCESS↵
```

ACCESS in large letters appears on the screen:

```
   ***         ***        ***     *******     ****       ****
  *   *       *   *      *   *               *    *     *    *
 *******     ***  **    ***  **   ***       **    **   **    **
**     **   ****       ****       **        ***        ***
***   ***   ****       ****       ***       *****      *****
*********   ****       ****       *******     ****       ****
***   ***   ****       ****       ***          ****       ****
**     **    ***        ***       **            ****       ****
**     **     **  **     **  **   ***       **    **   **    **
               *  *       *  *               *    *     *    *
**     **       **         **     ********    ****       ****

                        ND-10185D
```

After a short waiting period, the following words appear:

Enter user identification:

Here you must give your ACCESS user name, which may or may not be the same as the name you used when you logged in.

If you are a beginner, you probably don't have a password, but if you do, ACCESS will ask for it.

You then get the message:

```
    Wait, Data base information is being read in.......
```

When this message disappears, one of two things may happen. The first possibility is that "ACCESS" remains on the screen while an asterisk appears in the lower left corner of the screen. You are now at the start level in ACCESS. Here you can choose whether to run a predefined query or to define your own. This is explained in chapter 3 and onwards.

The other possibility is that you get a menu. This is explained in the following chapter.

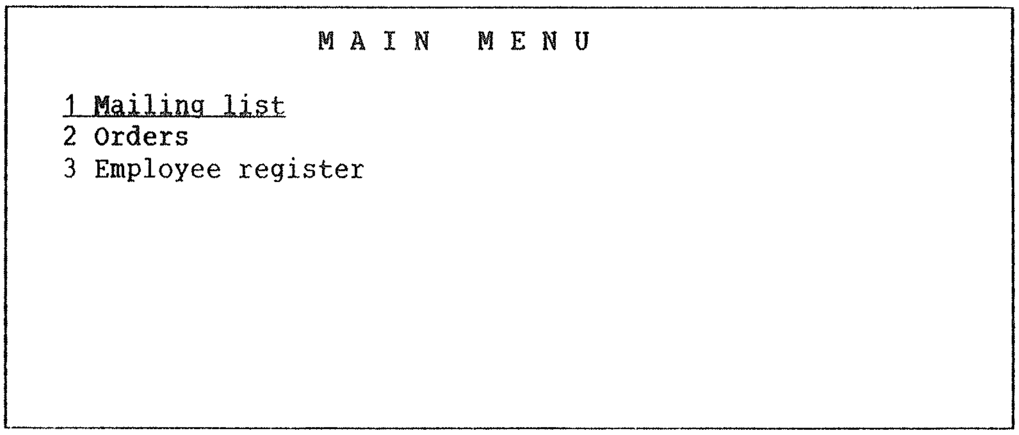
CHAPTER 2

HOW TO USE A MENU SYSTEM

2 HOW TO USE A MENU SYSTEM

2.1 USING A MENU

When you have entered ACCESS you may find a menu on your screen. A menu is a screen picture where you can select courses of action from alternatives displayed. If you get a menu right after entering ACCESS, this will be a menu put in locally by someone at your computer installation. The text on these menus will differ. However, the way they work is the same.



```
                 M A I N   M E N U

    1 Mailing list
    2 Orders
    3 Employee register
```

When the menu is dispayed, the uppermost item will be shown in inverse video (dark letters on a light background). There are two ways of selecting the item you want:

1) By using the up and down arrows

   ↑ ↓

   you can move the inverse video bar to the item you want, and then select the item by pressing the carriage return key:

   ↵

2) You can also select by typing the number in front of the item. For example, to select "Employee register" you can type "3" instead of moving two lines down and then pressing the ↵ key. This can save time if the menu is long.

What happens now may depend on what you have selected. Let us assume that you have selected "Employee register". The next screen picture might be another menu, for choosing what to do with your employee register:

```
          E M P L O Y E E   R E G I S T E R

    1 Display list
    2 Update salary
    3 Delete employee
    4 Add a new employee
    5 Change of address
```

Let us say you select "Add a new employee". Now something new happens:

2.2 USING AN INPUT FORM

```
Name      : ..............................

Address   : .....................................

Manager   : ..............................

Salary    : ........

Department: ...................
```

This type of screen picture is called an <u>input form.</u> Here you simply type on the dotted lines the information on the person you want to add to the list. The ↵ key will automatically take you to the next item.

Note: Like the menus shown earlier, the above input form is only an example. The layout of input forms may differ widely; this example only illustrates the basic procedure for entering data into a form.

When you have finished the last item, press the

⇨ key.

ACCESS will start storing the material you have typed. The message:

WAIT, the query is being executed

appears on the screen. And then:

The query caused 1 record(s) to be accessed. Time used is 0

This means that ACCESS has finished adding the new name to the 1 ist. After this the input form will appear again, and then you can add another person to the list, or go back again to the "employee register" menu. To go back, you simply press the EXIT key. Now you can select a different item on the menu, or go back to the main menu by pressing the exit key once more. To leave ACCESS, press the EXIT key <u>twice</u> when you are in the main menu.

CHAPTER 3

DEFINING AND RUNNING QUERIES

3 DEFINING AND RUNNING QUERIES

3.1 WHAT NEXT?

When you have entered ACCESS as explained in section 1.4.5, you are at the start level (there will be one asterisk in the lower left corner of the screen.)

Now you have the following main alternatives:

Action	Command	Explained in Section
Execute a predefined query	*RUN-QUERY	3.2
Define your own query	*DEFINE-QUERY	3.3 and onwards
Modify a predefined query	*FETCH-QUERY	3.8

3.2 HOW TO EXECUTE A PREDEFINED QUERY

First you may want to see what queries are available for execution. You can do this by giving the command:

```
     *LIST-QUERIES↵
```

ACCESS displays a list of stored queries. An example might look like this:

```
          L I S T   O F   S T O R E D   Q U E R I E S

     Query name                    Query size in bytes

     INSERT-EMPLOYEES                    12345
     INSERT-DEPARTMENTS                   9807
     INSERT-CUSTOMERS                     1032
```

To execute the query "insert-employees", give the command:

```
*RUN-QUERY INSERT-EMPLOYEES↵
```

For an example of what might happen after this, see the previous chapter, section 2.2.

Query execution can be aborted while in progress, by using the EXIT key.

3.3 HOW TO START DEFINING YOUR OWN QUERIES

You want to work with the data base ENGLISH-BASE and get a table frame showing the structure of the file EMPLOYEES on the screen.

- Log in.

- Enter ACCESS.

- Give ACCESS your user name.

- Give ACCESS your password, if you have one.

a) You are now in command position on the start level. On the start level there is always one asterisk in the lower left corner of the screen.

Give the command:

```
*DEFINE-QUERY↵
```

If you have access to only one data base, ACCESS chooses it for you. If you have access to more than one data base, these data bases are shown on the screen.

The screen picture may at this point look like this:

```
          A V A I L A B L E   D A T A   B A S E S

Data base name                       Supervisor

EXAMPLE-DATABASE                     CUSTOMER-1
SIBAS-DATABASE                       SIBAS-USER
TEST-BASE                            SIBAS-USER
NORWEGIAN-BASE                       ACCESS
ENGLISH-BASE                         ACCESS
```

It is possible to move between the data bases with the cursor keys ↓ and ↑. To select ENGLISH-BASE, go down to the appropriate line, and enter ↵.

Alternatively, you can specify the data base when you give the command:

```
*DEFINE-QUERY ENGLISH-BASE↵
```

Ask the ACCESS supervisor to give you access to the ENGLISH-BASE if you don't already have it.

b) You come directly to the query definition level (there will be two asterisks in the lower left corner of the screen).
Now give the command:

```
**NEW-TABLE↵
```

ACCESS will display the available tables on the screen. Select the table EMPLOYEES in exactly the same way that you selected ENGLISH-BASE.

c) Now the left part of the frame appears and you have come into editing position, with the following picture:

EMPLOYEES	NAME	MANAGER	SALARY
→			

You are now ready to complete the table frame with operators and selection criteria.

It is also possible to specify <u>both</u> the data base name and the table name in the command *DEFINE-QUERY. This way you skip the command **NEW-TABLE, and the whole process described above is replaced by one command with two parameters:

```
*DEFINE-QUERY ENGLISH-BASE EMPLOYEES↵
```

3.4 HOW TO LEAVE ACCESS

A) On the NOTIS terminal:

Press the EXIT key twice. You then get to the start level. Press the EXIT key once more, and you leave ACCESS.

B) On other terminals:

Press the HOME key ↖

You enter the table position on the query definition level. When you type in commands here, you automatically go to command position. Give the command:

```
**EXIT↵
```

You enter the command position on the start level, and give the EXIT command once more:

```
*EXIT↵
```

3.5 CALLING A TABLE FRAME

Call a new table frame from the query definition level with the command:

```
    **NEW-TABLE <table name>↵
```

The frame corresponding to the given table name appears on the screen, ready for input.

You go from editing position in the table frame to table position by typing HOME or pressing the EXIT key.

You go back to editing position by pressing the ↵ key.

You go back and forth between table position and command position with the HOME key.

In this example we use the table with the name EMPLOYEES. The table frame looks like this:

EMPLOYEES	NAME	MANAGER	SALARY	DEPARTMENT
→				

3.6 HOW TO EDIT A TABLE FRAME

When the table frame is displayed on the screen, the next task is to make it ready for a particular query. The columns in the table frame may be widened or compressed both before and while you fill in the frame.

3.6.1 WIDENING, COMPRESSING AND EDITING THE COLUMNS

A column is the whole area under one field name in a table frame. A field description consists of all the operators, selection criteria and example elements positioned in the column, within one subquery. We may have several queries in one table frame. These are called subqueries, and are executed in an order determined by ACCESS. You get a new subquery by typing down arrow (↓).

See section 3.7.6 for more information on subqueries.

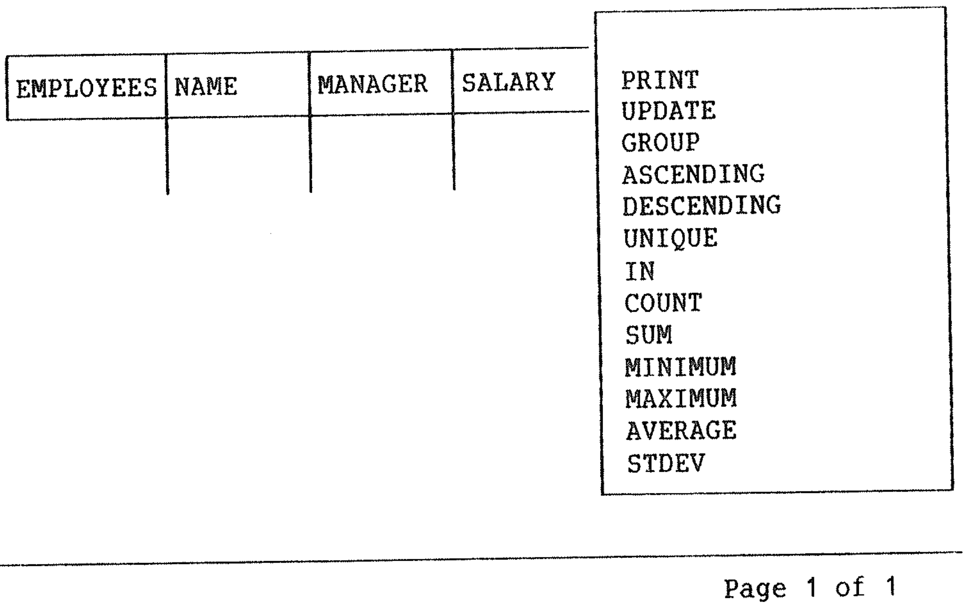
You may write a long field description over more than one line, by giving ↵ at the end of each line, or all on one long line by increasing the column width.

To move the cursor rapidly from one column to another, use the left- and right tabulator arrows (alternatively CTRL+T and CTRL+U).

When you have moved to the desired column, you may use the following keys or screen commands:

>< <>	or CTRL+W	increases the column width (widen).
SHIFT+ >< <>	or CTRL+P	reduces the column width (compress).
MARK		marks an item that is to be copied (the marking is not visible on the screen).
COPY		copies a marked item.
DELETE		deletes an item that is not to be used in a query.
SHIFT+ DELETE		deletes a line that is not to be used in a query.
CTRL+D CTRL+C		deletes a column that is not to be used in a query.

HELP or CTRL+G gives you a menu where you can select operators from the ones you are allowed to use in your current position. For example, when you are inside the table frame:



EMPLOYEES	NAME	MANAGER	SALARY

PRINT
UPDATE
GROUP
ASCENDING
DESCENDING
UNIQUE
IN
COUNT
SUM
MINIMUM
MAXIMUM
AVERAGE
STDEV

Page 1 of 1

Here you can select operators using the same system as before, that is with the up and down arrows and the ↵ key. They will then appear in the table frame just as if you had typed them in manually.

Extension or reductions of column width is carried out in steps of 6 positions at a time.

A column can never be wider than the width of the screen minus the width of the margin and the table name.

Note that the ><<> key (or the commands CTRL+P and CTRL+W) is only used in the query definition to make more room for the field description. This will not change the column width of the <u>result</u> which is displayed on the printing of result level.

CHANGING THE TABLE FRAME DOES <u>NOT</u> CHANGE THE INFORMATION IN THE DATA BASE.

EXAMPLE

In the table EMPLOYEES, give the commands CTRL+D plus CTRL+C in the column MANAGER and then

$\boxed{\begin{matrix}><\\<>\end{matrix}}$

eight times in the column NAME. The table then looks like this:

EMPLOYEES	NAME	SALARY
→		

3.6.2 PAGE SCROLLING

When a column is extended, other columns may fall outside the window, as in the previous example. There the column SALARY and DEPARTMENT are out to the right of the window. You can see this because the last complete column is followed by a new but unfinished column.

If there are more columns to the left of the window, the vertical separation line in the table name column is not drawn. The table name is always shown to the left in the window.

Use the $\boxed{\vdash\!\!-}$ key or CTRL+U

in the lefthand column to move the window to the left,

and the $\boxed{-\!\!\dashv}$ key or CTRL+T

in the right-hand column to move the window to the right. You may also use the arrow keys in table position.

EXAMPLE

To look at the right part of the table in the previous example,

press ⇥ or CTRL+T in the NAME column. This part looks like this:

EMPLOYEES	SALARY	DEPARTMENT
→		

To go back to the left part, press ⇤ or CTRL+U in the SALARY column.

3.7 FILLING IN THE TABLE FRAME

To get an overview of commands available for editing text in the table frame, see section 3.9.

3.7.1 AN EXAMPLE: LOOKING AT DATA IN THE DATA BASE

To find out if employee Mike Platt is in the data base, you must fill in the table frame and execute the query and you will get the result on the screen.

Fill in the table frame as follows (use ↵ to get a new line):

EMPLOYEES	NAME	MANAGER	SALARY	DEPARTMENT
→ print.	'Mike Plat t'			

Here "print." is an operator which tells ACCESS what you want to do (in this case print a result on the screen), and 'Mike Platt' is a selection criterion which tells the system which records you are looking for (ie., those which have 'Mike Platt' as name). Use the HELP key or CTRL+G to see the operators that are available in the field the cursor is positioned in. Note the period after the operator!

To execute the query:

Type ⇨ on the NOTIS terminal

Type ESC + ↵ on other terminals (including the NOTIS terminal)

(or press HOME and give the command **RUN-QUERY).

ACCESS interprets what you have written and executes the query. The result appears on the screen with the message:

"The query caused 1 record(s) to be accessed. Time used: 0/1"

You are now at the printing of result level. When you have looked at the result and want to do something else, use the EXIT key or type:

***<u>EXIT↵</u>

You enter the query definition level in table position. If you want to terminate, press HOME and follow the procedure in the previous example.

The EXIT key can also be used to abort query execution.

3.7.2 HOW TO USE OPERATORS IN THE TABLE FRAME

As previously mentioned, you define a query by filling in the table frame with operators and/or selection criteria. The HELP key or CTRL+G will give you a menu of the available operators in the column in which the cursor is positioned (select with the up and down arrows and ↵).

An operator is a command which is typed either in the first column, the table column, or in any of the other columns. When you use a <u>table operator</u> the whole table is affected; when you use a <u>column operator</u> only the column in which the operator is written is affected. There are also other types of operators:

- Arithmetic operators, used together with the "update." operator.

- Logical operators and relational operators. These are used in connection with selection criteria, which will be explained later.

Some operators have special names. The column operators "ASCENDING." and "DESCENDING." are called sorting operators.

You are now ready to fill in the frame with operators and selection criteria.

There are two main kinds of operators in the table: table operators and column operators. They must be terminated by a period to distinguish them from selection criteria.

There are four main groups of operators:

1) PRINT., sorting operators, UNIQUE., function operators (see p. 40), and GROUP.

2) INSERT.

3) DELETE.

4) UPDATE.

Note that only PRINT. and the function operators give a display on the screen.

Rules:

- You must never mix operators from different groups in the same subquery.

- When subqueries give a printout on the screen, the printouts must fit into the same result table. This means that operators from group one can only be used in one subquery.

3.7.3 TABLE OPERATORS

Table operators are placed in the column under the table name, and they define one operation on the whole table. There are three table operators:

PRINT.
INSERT.
DELETE.

Note the period!

PRINT.

- gives a printout on the screen of data from all columns (those which have not been deleted with CTRL+D and CTRL+C).

EXAMPLE

Task: You want all the information about all the employees.

EMPLOYEES	NAME	MANAGER	SALARY	DEPARTMENT
→ print.				

Result:

NAME	MANAGER	SALARY	DEPARTMENT
John Mills	Sid Stone	24700	Building
Ian Rain	Sid Stone	24000	Building
Christopher Howard	Sid Stone	18200	Building
Harold Ritchie	Sid Stone	12400	Building

etc.

INSERT.

- inserts a new record in the table.
- values must be put into the columns. Remember to enclose data of text type in single quotes (').

If no value is written into a column, or if the column has been deleted by CTRL+D followed by CTRL+C, default values are are put into the data base: a string consisting of blanks in text fields, the value 0 in numeric fields.

Instead of putting values directly into the table frame, (some of) the columns may have input example elements. These values will be read into the column when the query is executed. See chapter 5 and section 3.7.8.

Not everybody is necessarily permitted to put new data into the data base. This is decided in advance by the ACCESS supervisor.

EXAMPLE

Task: Add a new employee named Sid Stone whose boss is James Brick. His salary is 10000 and his department Building.

Solution:

EMPLOYEES	NAME	MANAGER	SALARY	DEPARTMENT
→ insert.	'Sid Stone'	'James Brick'	10000	'Building'

DELETE.

- deletes one or more records from the file.

This is a dangerous operator to use incorrectly. Sometimes a user is not permitted to delete records. If you are not authorized to use this command, the query is interrupted, and an error message is displayed on the screen.

EXAMPLE

Task: Jim Butcher has left the firm. Delete his record.

Solution:

You must use a selection criterion to tell the system that you want to delete the record "Jim Butcher", and only that record. If there are more records with the same identification, fill in more columns so you can be sure the criterion selects the right person. For example, here you could give department and salary, just to be on the safe side!

EMPLOYEES	NAME	MANAGER	SALARY	DEPARTMENT
→ Delete.	'Jim Butcher'			

NB:
If you write "Delete." without giving any selection criteria, all the records in the table will be deleted. However, you get the warning "Do you really want to delete all records?", and the chance to change your mind before this happens.

When the table operator DELETE. is used, only selection criteria are allowed in the columns. The only exception to this is that you can use an input example element instead of a value, so that the value is read in when the query is executed instead of when it is defined. See chapter 5 and section 3.7.8.

3.7.4 COLUMN OPERATORS

Column operators are placed in one of the columns to the right of the table name, depending on which of the fields in the record you want to affect.

If an operator and a selection criterion are placed together in the same field, the operator must come before the criterion.

These are the available column operators:

PRINT.
UPDATE.
ASCENDING.
DESCENDING.
UNIQUE.
GROUP.

SUM.
AVERAGE.
STDEV.
MAXIMUM.
MINIMUM.
COUNT.

The last six operators are called function operators.

Note: If you use the sorting operators, ASCENDING. and DESCENDING., they must be written in the same subquery as PRINT., or together with GROUP

PRINT.

- is the only column operator in addition to the function operators that may give a display of results.
- displays on the screen information from the field (column) where the operator is positioned.

EXAMPLE

Task: Display the names of all the employees.

Solution:

EMPLOYEES	NAME	MANAGER	SALARY	DEPARTMENT
→	print.			

Result:

NAME
John Mills
Ian Rain
Christopher Howard
Harold Ritchie
etc.

UPDATE. <new data value>

- updates/changes the value of data in the column where the operator is positioned.

- Only users authorized by the ACCESS supervisor may update the contents of the data base.

- In a subquery where one or more columns have UPDATE., the other columns must be blank or only contain selection criteria.

- It is possible to update the fields in more than one record at a time. All fields that satisfy the selection criteria will be updated in the fields where UPDATE. is used.

- The operator UPDATE. must be followed by an argument. The values for this argument and/or the selection criteria may be replaced by input example elements. You will then be asked for the values when the query is executed. Note that "%" cannot be used when the value is read in via an input example element.

Text updating

If the field is a text field, you can give it a new value (for example replace the old value 'Pat Peters' with a new value 'Pat K. Peters').

EXAMPLE

Task: The Sales department has got a new boss called Roger Keigh.
This must be updated.

Solution:

EMPLOYEES	NAME	MANAGER	SALARY	DEPARTMENT
→		update. ='Roger Keigh'		'Sales'

Solution with an input example element:

EMPLOYEES	NAME	MANAGER	SALARY	DEPARTMENT
→		update. =$manager		'Sales'

When the query is executed, the word "manager" will appear at the bottom of the screen. You type Roger Keigh, give ↵, and everybody in the Sales department will get Roger Keigh as their manager. If you store this query, it can be used every time the Sales department gets a new manager. By using an input example element also in the DEPARTMENT column, the query can be used for all departments.

Numeric updating

You can update numeric fields absolutely or relatively: absolutely by giving the new value preceded by an equals sign (=), or relatively by using one or more of the following arithmetic operators:

+	plus
-	minus
*	multiplied by
/	divided by

For example, for the field SALARY you have the following possibilities:

Upd.=11000	New value is 11000.
Upd.+1000	New value is old value plus 1000.
Upd.-100	New value is old value minus 100.
Upd.*1.1	New value is old value multiplied by 1.1.
Upd.+20%	New value is old value plus 20%.
Upd.-5.2%	New value is old value minus 5.2%.
Upd./2	New value is half of old value.
Upd.=-100	New value is - 100.

EXAMPLE

Task: All employees are getting a 5% increase in salary.
This must be updated in the data base.

Solution:

EMPLOYEES	NAME	MANAGER	SALARY	DEPARTMENT
→			update.+5%	

EXAMPLE

Task: Jim Lee has been promoted and has therefore received an increase in salary. His new salary is 9500.

Solution:

EMPLOYEES	NAME	MANAGER	SALARY	DEPARTMENT
→	'Jim Lee'		update.=9500	

In this example the operator PRINT. is not used, so the result of the updating is not displayed.

Remember that PRINT. cannot be used in the same subquery as UPDATE.

ASCENDING.

- sorts data from the column in which the operator is placed into ascending order. ASCENDING. must be in the same subquery as PRINT or be used together with GROUP.

DESCENDING.

- sorts data from the column in which the operator is placed in descending order. DESCENDING. must be in the same subquery as PRINT or be used together with GROUP.

The operators ASCENDING. and DESCENDING. are called sorting operators. If there are several of them in the same query, separate them by giving priority in parentheses after the operator and before the period.

Example:

```
Ascending(1).
```

You may have up to 5 sorting operators in each query.

EXAMPLE

Task: Display the names of all the employees sorted according to department and salary.

Solution:

EMPLOYEES	NAME	MANAGER	SALARY	DEPARTMENT
→	print.		pr.asc(2).	pr.asc(1).

Result:

NAME	SALARY	DEPARTMENT
Gary Yarwood	5200	Building
Sam Boyle	5300	Building
Patricia Mac Andrews	9000	Building
Ron Briggs	10600	Building
.	.	.
.	.	.
Margaret Biggs	24200	Building
John Mills	24700	Building
Ian Austin	5000	Cleaning
Julie Mason	6400	Cleaning
.	.	.
.	.	.
Steve Bell	23600	Cleaning
John Mason	24000	Cleaning
.	.	.
.	.	.

UNIQUE.

- Use this operator to avoid duplicates (identical data) in the result.

- All the records found will have different data values in the column where UNIQUE. is positioned. The first record that ACCESS finds with one particular data value is written out. Other records with the same value will be skipped.

- UNIQUE. must be in the same subquery as PRINT.

EXAMPLE

Task: Who is manager in which department, if you assume that each department has only one manager?

Solution:

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

Result:

MANAGER	DEPARTMENT
Sid Stone	Building
Reg Howard	Cleaning
James Watt	Development

etc.

If you do not use UNIQUE. here, you will get as many result records as there are employees.

Function operators

Selection criteria are used to select the records the function operators are to affect. If GROUP. is not used, the function is calculated for all the records that satisfy the selection criteria. If GROUP. is used, the result is calculated for each group.

The result is always written out in a column with a heading ("SALARY maximum" if you have used the operator MAXIMUM in the SALARY column). The values in the column where GROUP is placed are printed out. (PRINT. may be used in the other columns to show examples of values found there.)

SUM.

- displays the sum of all values in a numeric column.

AVERAGE.;

- displays the average of the values in a numeric column.

STDEV.

- displays the standard deviation of the values in a numeric column.

MAXIMUM.

- displays the highest value in a numeric column.

MINIMUM.

- displays the lowest value in a numeric column.

COUNT.

- displays the number of records in a column.

GROUP.

- is used to group identical values in a numeric or text column.
- This command is to be used together with one of the function operators - SUM., STDEV., AVERAGE., MAXIMUM., MINIMUM., or COUNT. - and selection criteria.
- GROUP. alone will arrange data in ascending order. You can also write GROUP. ASCENDING. or GROUP. DESCENDING. Sorting operators in other columns cannot be used in the same subquery. UNIQUE. is not permitted together with GROUP.

EXAMPLE

Task: Display the maximum salary for each department.

Solution:

EMPLOYEES	NAME	MANAGER	SALARY	DEPARTMENT
→			MAX.	PR.GROUP.

Result:

SALARY Maximum	DEPARTMENT
24700	Building
24000	Cleaning
24300	Development
24500	Finance
24200	Maintenance
23000	Marketing
23600	Personnel
24200	Sales
24200	Support

EXAMPLE

Task: Display the minimum salary, the maximum salary and the sum of the salaries for each department.

Solution:

	EMPLOYEES	SALARY	DEPARTMENT
→		MIN.MAX.SUM.	PR.GROUP.

Result:

SALARY sum	SALARY minimum	SALARY maximum	DEPARTMENT
227100	5200	24700	Building
338800	5000	24000	Cleaning
308200	5300	24300	Development
398300	5800	24500	Finance
301900	5300	24200	Maintenance
356700	5300	23000	Marketing
268200	7300	23600	Personnel
329400	5600	24200	Sales
263900	5900	24200	Support

3.7.5 HOW TO USE SELECTION CRITERIA

To describe the records you want to work with in the different queries, you must specify selection criteria. These are placed in the relevant columns. A selection criterion consists of a data value, or data values combined with one of the relational or logical operators listed below. These operators must always be placed <u>after</u> column operators.

There are three kinds of selection criteria:

1) A data value alone;

2) Part of a data value together with a question mark and/or asterisk;

3) Data values combined with relational operators and/or logical operators.

We have the following relational operators and logical operators:

=	equal to, is assumed if an operator is not given.
<	less than
<=	less than or equal to
>=	greater than or equal to
><	not equal to
>	greater than
IN	between two values, or belonging to a list
AND	and
OR	and/or or both
NOT	not

Note that example elements in the table column may work similarly to the operator OR. See section 3.7.8.

<u>Value alone</u>

FOR NUMBERS

In a numeric field, write the number you want to search for, either alone or with logical operators. If you get the error message "Illegal type of constant in this column", it may be that the column is actually defined as a text column, even though the "text" it contains consists only of digits. If this is the case, you will have to use single quotes around the numbers (see below).

FOR TEXT

You can also search in text fields. You may use logical operators, but the search criteria will be more complicated.

ACCESS always assumes the text field to be the search area, and if the text field is longer, ACCESS assumes that the rest of the field is blank. See section on "Searching in text" below.

<u>All text strings must be enclosed in single quotes.</u>

ACCESS distinguishes between upper and lower case letters in selection criteria, so it is relevant whether you specify the search in uppercase or lowercase letters; the text must be typed in exactly as it was written into the data base, or the record will not be found.

EXAMPLE

Task: Which persons in Marketing earn 17800?

Solution:

EMPLOYEES	NAME	MANAGER	SALARY	DEPARTMENT
→	print.		17800	'Marketing'

Result:

NAME
Alex Sykes Ann Mills

Searching in text using * and ?

ONLY FOR SEARCHING IN TEXT.

If you want to pick a special text from fields containing long texts, you can do this by using the characters asterisk (*) or question mark (?).

The asterisk replaces an arbitrary character in the text. A question mark replaces several characters; there may be any number of characters behind the question mark. The question mark can only be behind, or in front of and behind the text you are searching for.

Examples:

You want to find everybody with the first name Peter.
Write: 'Peter'?

You want to find everybody with Peter as part of their name, for example James Peter Ball, Tim Peters.
Write: ?'Peter'?

You want to find everybody with 'at' as part of their name, for example Patricia, Platt.
Write: ?'at'?

You can combine ? and * with logical operators to express complicated search criteria. But remember that the operators must be put outside the quotes; if they are placed inside the quotes, they are interpreted as part of the text.

You want to find everybody with a surname of six
letters, ending in "son",
for example Hanson, Benson, Larson.
Write: ?' '***'son'?

The first question mark shows that an unknown number of characters should be skipped, then there should be a blank, then any three characters before the three letters "son". If the first question mark is omitted, the first name will not be skipped. If you only had surnames in the field, it would be sufficient to write ***'son'

Remember that blanks are characters like any other symbol! If * or ? are enclosed in single quotes, they will be interpreted as ordinary characters.

Data value with relational operators and logical operators

Values can be combined with a relational operator. This applies both to numeric and text string fields. Use relational operators if you want to search for values:

greater than	(>)
less than	(<)
greater than or equal to	(>=)
less than or equal to	(<=)
not equal to	(><)

If you want to search for a combination of these, you must also use logical operators, or IN (see p. 47).

EXAMPLE

Task: Who works in Finance and who works in Sales? (In other words, write out the names of those who work in Finance or Sales.)

Solution:

EMPLOYEES	NAME	MANAGER	SALARY	DEPARTMENT
→	print.			print. 'Finance' OR 'Sales'

Result:

NAME	DEPARTMENT
Anne Lee	Finance
Sid Grant	Finance
.	.
.	.
Tony Ritchie	Sales
Ruth Mason	Sales
.	.
.	.

etc.

IN

This operator can be used to find:

All values between two limits:	IN(10:60)
All values belonging to a list:	IN(1,5,6,18)
A combination of these two functions:	IN(1,2,5:9,18)

The first of these formats can be used to search for data values which lie between certain limits. The search is carried out in a special way.

The search looks for values between and including the limits. For numeric data values, all numbers in the range are picked out. For text strings, the values are picked according to alphabetical order. However, the "alphabet" is slightly extended, to include special signs such as numbers and blanks.

The value of each symbol is calculated according to the ASCII code. The order of the symbols is given in the SINTRAN III Reference Manual, ND-60.128.02, Appendix G. In outline it is as follows (from lowest to highest values):

blank, special characters,(0-9),(A-Z),special characters,(a-z)

The letters are in alphabetical order, with all uppercase before all lowercase.

<u>EXAMPLE</u>

IN('Jim Johns':'Olive Osmond')

- Here names between "Jim Johns" and "Olive Osmond" are found in alphabetical order.

- These names are found

Jim Johnson	(o has a greater value than blank)
Jim Wright	(W has a greater value than J)
Jimmy Johns	(m has a greater value than blank)
Mike Brook	(M comes between J and O)

- These names are not found

Jim Johns	(The extra blank makes the value lower)
Eric Abbs	(E has a lower value than J in Jim)
Jim Biddulph	(B has a lower value than J in Johns)
Pat Oliver	(P has a greater value than O in Olive)

<u>EXAMPLE</u>

IN('J':'O')

- Here names beginning with the letters from J to O are found.

- These names are found

Jan Adamson	(a is greater than blank)
John Addy	(o is greater than blank)

- These names are not found

Otto Askey	(t is greater than blank)
O A Ask	(A is greater than blank)
Pat Oliver	(P is greater than O)

If you want to search for all names that begin with J, K, L, M, N, O, you must write J:Ozzz or J:P.

EXAMPLE

IN(10000:12000)

- Here you are looking for numbers between 10000 and 12000 inclusive.

IN can also be used to find all values in a list. For example, the following query will display all employees who earn exactly 13000, 18000, or 22000:

EMPLOYEES	NAME	MANAGER	SALARY	DEPARTMENT
→ print.			in(13000,18000, 22000)	

The two ways of using IN (with a colon and with commas) can be combined. If in the above query you had used IN(13000,14000:15000,18000,22000), you would also get all employees earning between 14000 and 15000.

3.7.6 SUBQUERIES

Sometimes it is necessary or desirable to build up a query using several subqueries.

To start a new subquery, give down arrow in the last subquery, or alternatively right tabulator arrow or CTRL+T in the last column in the table frame.

ACCESS responds with an arrow (→) on the first line in each subquery as shown in the next example.

EXAMPLE

Task: Who in Cleaning earns between 5000 and 10000?
Also update the salary of those with Jane Wright as their boss by 4%.

Solution:

	EMPLOYEES	NAME	MANAGER	SALARY	DEPARTMENT
→		print.		in(5000: 10000)	'Cleaning'
→			'Jane Wright'	update.+4%	

Result:

NAME
Ian Austin
Harold Moss
Gary Howard
Julie Mason
Peter Beever
etc.

The records which are updated are not displayed. The "greater than" or arrow symbols to the left in the table frame each start a new subquery.

3.7.7 POSSIBLE SEQUENCES OF COLUMN OPERATORS

Not all sequences of column operators are allowed. The two diagrams on the next page illustrate the legal sequences. The first one simply illustrates the rule that the function operators must precede the selection criteria.

The second illustrates the more complex relations between PRINT., GROUP., UNIQUE., ASCENDING. and DESCENDING. Any sequence of operators you get by following the arrows is legal. It is not necessary to start at the top, or end up at the bottom. For example, GROUP.COUNT. is allowed. ">18000" is only an example of a selection criterion, and may be replaced by any expression that uses the operators explained in the previous section: asterisk, question mark, logical operators and relational operators (including IN).

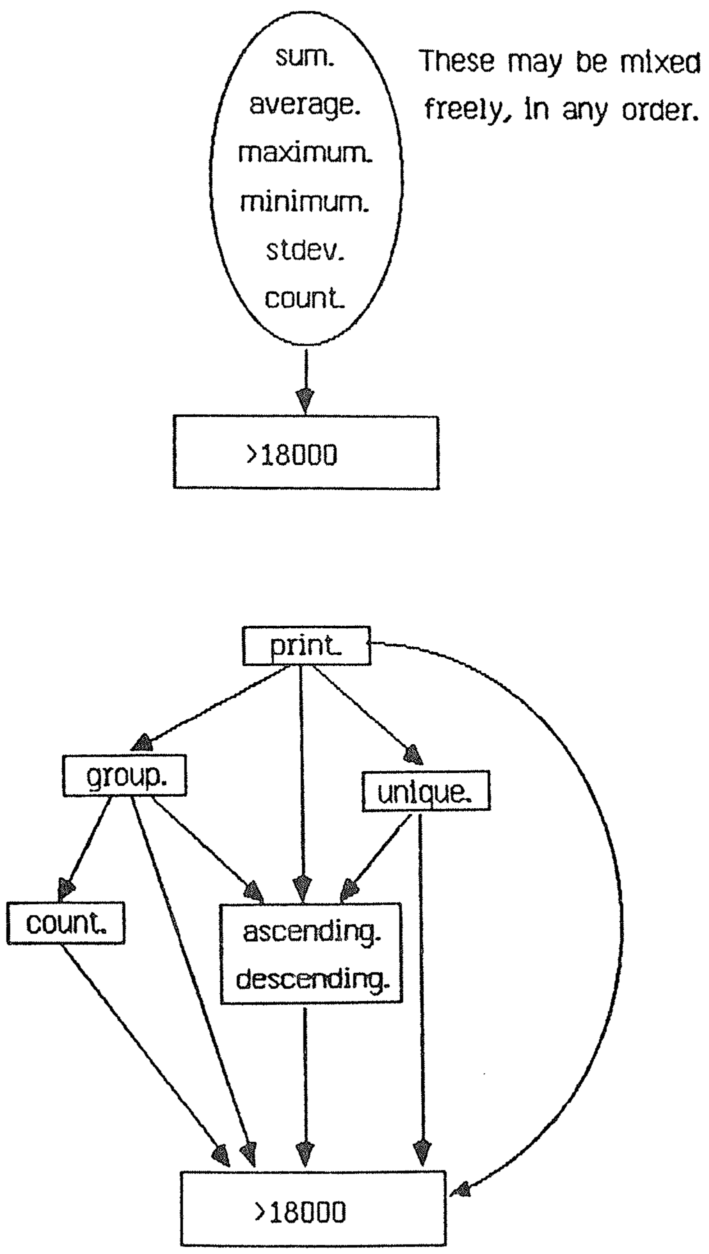


Fig. 12. Column operator sequences

UPDATE. can only be used alone.

3.7.8 EXAMPLE ELEMENTS

There are four types of example elements. All are written as _<name>, except no. 3 below:

1) Example elements used in the table column, to connect several subqueries.

2) Output example elements in one or more of the field columns. These are used to identify the data fields which are to be written out when an output form is used. See chapter 4

3) Input example elements in field columns, written as $<name>. These are used instead of a data value. When the query is executed, ACCESS will ask for a data value which then will replace the input example element. Chapter 5 contains examples of the use of input example elements.

4) Example elements used in the field columns to create logical links (or connections) between different table frames on the screen, or between subqueries. See sections 3.7.10 and 3.7.11.

This section will deal with example elements of the first type.

When two selection criteria in different data fields are used in the same subquery, those records which satisfy <u>both</u> criteria are picked out.

If you want to pick out records which satisfy either one <u>or</u> the other, you have to write the selection criteria in two different subqueries, and connect them with an example element. All records which satisfy at least one of the criteria will then be picked out.

If two subqueries are connected with an example element, and they both have selection criteria in the same column, the effect will be the same as if the selection criteria were written in <u>one</u> subquery with the logical operator OR between them.

Subqueries that are connected with example elements must follow this rule:

> <u>One</u> of the subqueries must have all the operators, and the output example elements if they are used. The other subqueries can only have selection criteria where data values may be replaced by input example elements.

<u>EXAMPLE</u>

"Mark" everyone with Vera Kendal as their boss. "Mark" everyone earning more than 10000 per annum. Display the names in the "marked" records.

What happens here is that the names of all employees who either have Vera Kendal as their boss <u>and/or</u> earn over 10000 are displayed. Observe that only the names are displayed. If you want the salaries displayed as well, you have to put "print." in the salary column.

Solution:

EMPLOYEES	NAME	MANAGER	SALARY	DEPARTMENT
→ _x		'Vera Kendal'		
→ _x			>10000	
→ _x	print.unique.			

If you put all the operators on one line, as shown below, do you get the same result?

EMPLOYEES	NAME	MANAGER	SALARY	DEPARTMENT
→	pri.	'Vera Kendal'	>10000	

No, in that case you get the names of all the employees who have <u>both</u> Vera Kendal as their boss <u>and</u> earn more than 10000.

In the first table above we have written "unique." in the name column, because we only want to have each name displayed once. If we had not done that, all those who satisfied both search criteria (those that the last example query will find) would have been displayed twice.

<u>EXAMPLE</u>

Task: Display the names of those who work in Sales or Finance and earn over 11000, and the names of those who work in other departments and earn between 10000 and 12000.

Solution:

EMPLOYEES	NAME	SALARY	DEPARTMENT
→ _x		>11000	'Sales' OR 'Finance'
→ _x		in(10000:12000)	><'Sales' AND ><'Finance'
→ _x	print.		

Note: The sorting operators ASCENDING and DESCENDING, must, if they are used, appear in the same subquery as PRINT. Other subqueries must include at least one selection criterion.

3.7.9 COMMANDS IN THE COMMAND BOX

A command box is used to specify commands to be executed with the query. Several categories of commands are available.

- Commands for the further processing of results. The same commands can also be executed manually after query execution is finished (on the printing of result level, see chapter 6). With a command box, these commands can be stored as part of the query and executed automatically when the query is run.

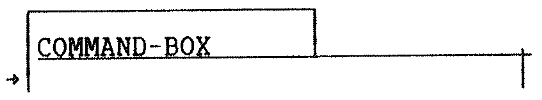
- Commands controlling input and output. These are used with advanced input and output functions.

- Commands controlling query execution. These are used to define menus, to restrict the number of records searched, and to execute several queries in succession automatically.

The command box and its operators are stored in the query description and can be used repeatedly. Operators must be written in a special frame or box which you get on the screen by typing the command:

**<u>COMMAND-BOX↵</u>

A command box then appears on the screen:



As in a table frame, you can select commands by pressing the HELP key, selecting the desired command with the up and down arrows, and finally pressing ↵.

The command box can be edited in the same way as a table frame; you can use:

><<> or CTRL+W to increase the width of the box, and

SHIFT+><<> or CTRL+P to reduce the width.

A command with its parameters can extend over several lines in the command box. You get a new line for the same command by pressing ↵, while the down arrow gives you a new line for a new command.

A new command line is indicated by ">" or "→".

Press HOME or the EXIT key when you want to finish filling in the command box. Then press the up or down arrow to jump between the command box and the table frame(s).

Here are the command box commands:

Result processing commands

REORDER-COLUMNS	<Column-1> <Column-2> .... <Column-n>
HEADING	<Heading string in quotes>
RENAME-COLUMN	<Old column name> <New column text>
PRINT	<File name with default file type :SYMB>
STORE-DATA	<File name with default file type :DATA>
WRITE-WITHOUT-FRAME	<File name with default file type :SYMB>
DESCRIBE-MACRO-CALL	<NOTIS macro description>
WRITE-MACRO-CALL	<File name with default file type :TEXT> <Macro name>

Commands controlling input and output

INPUT	<Input element> <Prompt text>
REPEAT	
KEEP-VALUES	<Input element-1> <Input element-2> .... <Input element-n>
DEFAULT	<Input element> <Default value>
NO-FORMFEED	

Commands controlling query execution

MENU	<Command to execute> <Query name> <Menu item text>
MENU-HEADING	<Heading text>
MENU-FOOTING	<Line no> <Footing text>
EXECUTE	<Query name>
NO-SCREEN	
RESTRICT-SEARCH	<Number of records>
SKIP-FOUND-RECORDS	<Number of records>

The result processing commands are explained in chapter 6. The commands controlling input and output are described in chapters 5 and 4. The MENU commands are explained in chapter 7. The remaining commands are described in the following sections.

3.7.9.1 EXECUTE AND NO-SCREEN

The EXECUTE command allows you to define a query that causes the execution of several other queries in succession. Such a query must consist only of a command box, as in the following example:

```
 | COMMAND-BOX
→| EXECUTE query-3
→| EXECUTE query-2
→| EXECUTE query-1
```

This query must be stored, and then run from the start level. The reason the query names have been entered in this order is that the EXECUTE commands are executed from the bottom up: query-1 will be executed first, then query-2, and finally query-3.

In the above example the EXIT key must be pressed between each query execution, <u>unless</u> query-1 and query-2 each contains a command box with the command NO-SCREEN. In the latter case, all three queries will be executed automatically in succession, with no chance to inspect the results of query-1 and query-2 on the screen.

3.7.9.2 RESTRICT-SEARCH AND SKIP-FOUND-RECORDS

The command RESTRICT-SEARCH can be used in the command box to restrict the number of records found in a search. For example, the following query will display only the first 10 employees that earn more than 15000:

EMPLOYEES	NAME	MANAGER	SALARY	DEPARTMENT
→ PRI.			>15000	

```
 | COMMAND-BOX
→| RESTRICT-SEARCH 10
```

SKIP-FOUND-RECORDS causes ACCESS to skip the first records satisfying your selection criteria. In the above query, if you replace RESTRICT-SEARCH 10 with

SKIP-FOUND-RECORDS 10

the 11th to last employees earning more than 15000 will be displayed.

These commands may be used together. To display "the second 10" (no. 11 through no. 20) employees earning more than 15000, use the following query:

EMPLOYEES	NAME	MANAGER	SALARY	DEPARTMENT
→ PRI.			>15000	

	COMMAND-BOX
→	RESTRICT-SEARCH 10
→	SKIP-FOUND-RECORDS 10

3.7.10 QUERIES WITH MULTIPLE TABLES

You now want to find all employees who work in a department located on the second floor.

This query uses two tables rather than one. You are already familiar with the EMPLOYEES table which we have used in the queries so far. The second table, DEPARTMENTS, contains information about the location of each department.

DEPARTMENTS	NAME	FLOOR
→		

You can get table frames for both tables on the screen by using the command NEW-TABLE twice. What information is common to the two tables? The department information appears in both and is the common information or link between the two tables. You can think of the link as a logical connection. Now we will phrase the query to make use of the link information between the two tables:

Print all employees who work in a department, for example DEPT, such that DEPT is located on the second floor.

We do not know what the actual department name (or names) is. What we do know is that the name must be the same in both tables. Therefore we use an example element instead of the name.

The example element says that we do not care what the department name is, but we do want the relationship, the link, to exist. Instead of the example element name _DEPT used in this example, you could use any name (for example _X1, __23) as long as you remember to start it with an underline character.

Now let us use ACCESS to get the result. The following procedure is used:

1) Get the EMPLOYEES table frame onto the screen by using the NEW-TABLE command.

2) Write print. in the table column.

3) Write _DEPT in the DEPARTMENT column.

4) Press the home key (\) twice.

5) Get the DEPARTMENTS table frame onto the screen by using the NEW-TABLE command.

6) Write _DEPT in the NAME column.

7) Write 2 in the LOCATION column.

8) Run the query.

The screen picture will look like this:

EMPLOYEES	NAME	MANAGER	SALARY	DEPARTMENT
→print.				_DEPT

DEPARTMENTS	NAME	FLOOR
→	_DEPT	2

This query illustrates the use of example elements as a link between tables. The same example element must be used in both tables.

The sequence of the table frames on the screen is not important. If the DEPARTMENTS table had been displayed before the EMPLOYEES table, it would have no effect on the result of the query.

Perhaps you are still confused about when to use an example element. Remember that an example element is only required when you want to link two or more items (columns). With this in mind, if you enter the query in the following sequence, you can see where links (example elements) are required.

1) Get all the table frames you need onto the screen.

2) Enter "print." in all columns from which you want output (or use "print." in the leftmost column).

3) Enter constants in all columns whose output must meet certain specifications.

4) The "Employees located in Oslo" query will now look like this:

EMPLOYEES	NAME	MANAGER	SALARY	DEPARTMENT
→print.				

DEPARTMENTS	NAME	ADDRESS	LOCATION
→			'Oslo'

As you can see, there is nothing to link or associate the EMPLOYEES table print request with the DEPARTMENTS table constant 'Oslo'. Therefore, the last step in the process is:

5) Enter example elements where necessary to link the parts of the query.

It is also easy to see the link or common information between the two tables: departments. By entering the same example element in the DEPARTMENT column of the EMPLOYEES table and the NAME column of the DEPARTMENTS table, the link is established between the two tables.

However, remember also that this is merely a suggested sequence of defining queries with links.

You may have your own way of thinking about links and your own sequence of building a query. Since ACCESS allows you to follow your own thought processes, you can formulate your queries in any sequence you want.

3.7.11 MORE ABOUT USING EXAMPLE ELEMENTS

Once you understand the concept of linking example elements, you can link any number of tables and any number of rows with single or multiple tables, as in the following example.

EXAMPLE

Task: Display the names and salaries of all employees who earn more than their manager.

Solution:

EMPLOYEES	NAME	MANAGER	SALARY	DEPARTMENT
→	print.	_BOSSNAME	print.>_SAL	
→	_BOSSNAME		_SAL	

This query is easier to understand if we state it in the following way: Print the name and salary of the employees whose manager may be BOSSNAME (as an example), and who earn more than SAL when BOSSNAME earns SAL. Here the example element is used to link the manager in the first row with the name in the second row. In addition, another example element is used to link and compare the two salaries.

3.7.12 THE CONDITION BOX

Selection criteria that are ordinarily written in the table columns may also be put in a condition box. This makes it possible to use some selection functions in addition to those allowed in the table frame. The condition box can also be used to define relationships between different columns.

To get a condition box on the screen, use the command:

```
**CONDITION-BOX↵
```

The following operators are allowed in the condition box. Note in particular that two relational operators may be used in one expression.

OPERATOR TYPE	EXAMPLES
Example elements	_a _salary
Relational operators: = < > <= >= ><	_a > 2 0 < _a <= 10
IN	_a in(1:5,7,14)
Logical operators: AND, OR, NOT	_a > 10 or _a < 2
Arithmetic operators: + - * /	_a/7+3 = 13*_b

As mentioned, selection criteria may be written in the condition box instead of in the table frame. This is done with example elements:

EMPLOYEES	NAME	MANAGER	SALARY	DEPARTMENT
→ p.			_sal	

CONDITION-BOX
→ _sal in(10000:15000)

The above query will give the same result as the following:

EMPLOYEES	NAME	MANAGER	SALARY	DEPARTMENT
→ p.			in(10000: 15000)	

3.7.13 RELATIONSHIPS BETWEEN COLUMNS

You can compare the values in two different columns, as in the following example, which will display all the records where the result is greater than the budgeted amount:

DEPT_174	BUDGET	RESULT	
→	_budget	_result	

	CONDITION-BOX
→	_result > _budget

This query can also be written without the condition box:

DEPT_174	BUDGET	RESULT	
→	_budget	>_budget	

You can update a column to contain the result of arithmetic operations on other columns, as in the following example, where the sum paid is updated to be equal to price multiplied by quantity:

ORDER	CUST_NO	ORDER_NO	ARTICLE	QUANTITY	SUM
→ p.			_art	_q	upd._sum

ARTICLE	ARTICLE	PRICE_PER_ARTICLE
→ p.	_art	_price

	CONDITION-BOX
→	_sum = _q*_price

3.8 HANDLING A STORED QUERY

You want to make a simple system which your company can use to adjust salaries. You want to adjust all salaries between 0 and 20000 by 5 %, and write the following query:

EMPLOYEES	NAME	MANAGER	SALARY	DEPARTMENT
→ _X			IN(0:20000)	
→ _X			updat.+5%	

Here we have used the down arrow after 20000), in order to get a new subquery (with → to the far left). The two parts of the query are connected with the example element _X in the first column (table column).

Press HOME twice, and you enter command position. Give the command:

```
*STORE-QUERY "SALARY-ADJUSTMENT"↵
```

Use the EXIT key (or give the command EXIT) twice to leave ACCESS. The query has now been stored in the query library file for later use.

In the next example, you want to edit this query so that the salary of all those who earn between 95230 and 111190 is adjusted by 7%.

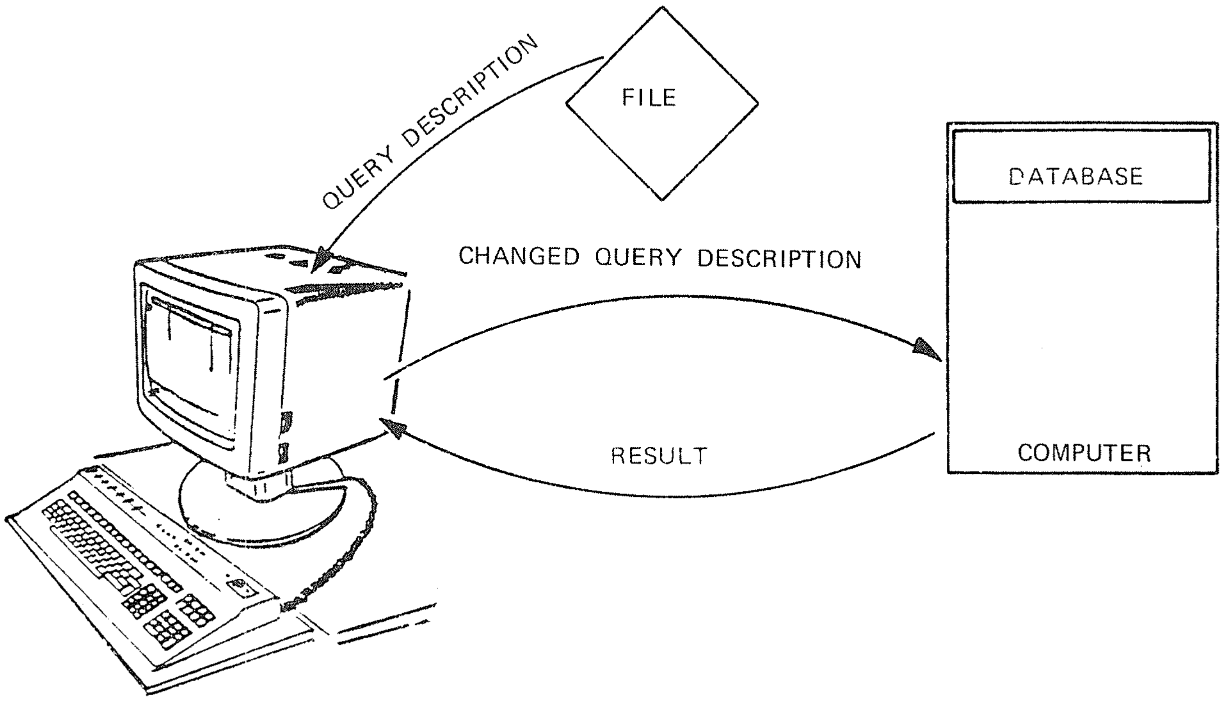


Fig. 13. Change and execute an old query

The query description on file is not to be changed. The query SALARY-ADJUSTMENT may be used later for other adjustments.

Fetch the query description SALARY-ADJUSTMENT. Enter ACCESS, and give the command:

```
*FETCH-QUERY SALARY-ADJUSTMENT↵
```

The table frame appears immediately, and you can edit it to suit your task.

EMPLOYEES	NAME	MANAGER	SALARY	DEPARTMENT
→ _X			in(95230 :111190)	
→ _X			update.+7%	

The query description remains unchanged, so you can use it later for other purposes.

3.9 FUNCTIONS IN EDITING POSITION

Using the functions available in editing position you can move inside and between subqueries in the table frame.

To get a new line in the same subquery, press ↵. To move to another column, use the tabulator arrows.

To show where a subquery starts and ends (vertically), each new subquery is marked with → in the left margin of the table frame.

1) The following screen commands are available for editing text in a table frame. For the most part, these are the same commands as those used in NOTIS-WP.

$\cancel{a}$ or CTRL+A or DEL	deletes one character.
CTRL+C	copies a character from the previous line.
CTRL+D CTRL+C	deletes the current column.
CTRL+D ↵	deletes the rest of the line in the column from the current cursor position.
CTRL+D "X"	deletes the rest of the line in the column up to the character "X".
CTRL+D CTRL+D	deletes the entire current line in the column.
DELETE or CTRL+D CTRL+F	deletes part of the current subquery in this column.
SHIFT + DELETE or CTRL+D CTRL+L	deletes the entire current subquery
INS EXP or CTRL+E	turns expand mode on/off (allows extra characters to be written in the text).
SHIFT+ INS EXP or CTRL+B	turns insert mode on/off (allows you to insert lines between existing lines in a subquery).

HELP or CTRL+G	gives a menu of available operators in the current column (see section 3.6.1). If you press SHIFT+HELP you will get information on how the column has been defined.
PRINT or CTRL+O	prints the screen picture (table frame or result table), on a local printer connected to your terminal.

2) The following screen commands are available for navigating within a table frame:

CTRL+F	moves the cursor to the first position after the last input character in the column.
CTRL+R	moves the cursor to the first character in the column.
⇥ or CTRL+T	moves the cursor to the start of the next column. If the cursor is in the last position of the column, the right arrow has the same effect
⇤ or CTRL+U	moves the cursor to the start of the previous column. If the cursor is in the first position of the column, the left arrow has the same effect
MOVE <column>	moves the cursor to the given column.

Using the arrow keys for navigating within a table frame.

Inside a column, the arrow keys have the following effects:

- Left arrow
  moves the cursor to the previous character position.

- Right arrow
  moves the cursor to the next character position.

- Up arrow
  moves the cursor to the same position on the line above, or to the subquery above.

- Down arrow
  moves the cursor to the same position on the line below, or to the subquery below.

Special functions:

EXIT or ↖	brings you to table position.
↵	gives a new line in the field you are positioned in the current subquery.
⇨	starts the execution of the query description.

3.10 COMMANDS IN TABLE POSITION

If you want to change a query description after having been in command position, you can:

1) press HOME to change to table position in order to move to the right table frame/command box/condition box.

2) use left or right arrow for page scrolling.

3) find the right page and frame/box by means of the arrow keys. Use the up and down arrows to jump between table frame(s) and box(es).

4) go into editing position to edit the frame/box. Press ↵ to go from table position to editing position. Make desired alterations.

5) if you want to make corrections in another frame or box, press HOME to go back to table position and continue from point 2.

6) if you have completed the editing, press HOME to go into table position.

If you want to expand or change the command box, you may type COMMAND-BOX again, and the cursor will jump to the box which has already been defined.

The following commands are available in table position:

```
HELP or CTRL+G..........Display HELP information
1.......................Move to the first page
$ ......................Move to the last page
- ......................Move to the previous page
+ ......................Move to next page
left arrow .............Scroll to the left
right arrow ............Scroll to the right
up arrow ...............Move to the previous table/box
down arrow .............Move to the next table/box
HOME or any character...Brings you into command position
↵ ......................Brings you into editing position
DELETE or CTRL+D........Deletes the whole current table frame/box
```

CHAPTER 4

ALTERNATIVE DISPLAY OF RESULTS

4 ALTERNATIVE DISPLAY OF RESULTS

4.1 THE OUTPUT FORM

If you do not want to present data in the standard table format, you can design another format for your particular need.

This part of ACCESS may, to a limited extent, be compared with a report generator. It can be used to create general reports consisting of data from the data base and additional text.

One page of a report generated with the help of ACCESS and written out on a line printer may, for example, look like this:

```
EMPLOYEES

NAME:.....................

DEPARTMENT:.........

MANAGER:...................

SALARY:.........
```

Fig. 14. Example of an output form

4.2 FILLING IN THE TABLE FRAME

When you want to present data in a format other than the standard table format, the first thing you have to do is fill in a table frame in the usual way. Output example elements must be used in the fields that are to be displayed in the output form. to connect them to the fields in the output form. This can be illustrated by the following example:

Task: You want to write out on a screen form all information about those working in the Sales department. The table frame is filled in as follows:

EMPLOYEES	NAME	MANAGER	SALARY	DEPARTMENT
→ _X				'Sales'
→ print._X	_IDENT	_BOSS	_SALARY	_GROUP

_IDENT, _BOSS, etc., are output example elements. Note that the output example element starts with the underline sign.
The example element _X used in the table column is a special example element and is only used for linking the two subqueries together.

The example elements connect the information known in the table frame with that necessary for ACCESS to use the OUTPUT-FORM definition (for example, where the relevant data can be found and how much space you need for displaying the data in your output form).

4.3 CALLING THE OUTPUT-FORM

When the fields to be used have been defined by entering example elements in the necessary columns, give the command:

```
**OUTPUT-FORM <number of lines in form>↵
```

Give the number of lines only when you want an output form larger than the screen window. The maximum number of lines is 72.

The screen is cleared except for the two bottom lines, where "***" tells you that you can start defining a form. If the form is bigger than the screen window, the first and last line numbers displayed are also given.

The cursor is positioned in the upper left corner of the screen. You are in editing position.

Press HOME to go to command position. The HELP function now gives you a list of available commands. When you press HELP or CTRL+G in editing position, a list of the fields you have available and information on how much space they occupy on the screen is displayed. Pressing HOME once more returns you to editing position.

4.4 DEFINING AN OUTPUT-FORM

An output form consists of two parts:

1) Prompt text;

2) information from the data base.

Text can be written anywhere on the screen and edited in the usual way. CTRL+L inserts a new line where the cursor is positioned. Those lines below are moved down on the screen. CTRL+D CTRL+D deletes a line.

4.5 DEFINING, MOVING, COPYING AND DELETING FIELDS

Let us have a look at how you insert a field from the table frame in the output form.

DEFINE-FIELD

A field may be placed in the form in the following way:

1) Move the cursor to the position in the form where you want to place the field.

2) On the NOTIS terminal, press the FIELD key (or press HOME and give the command ***DEFINE-FIELD). ACCESS asks for the field name. Here you type in the name of one of the example elements in the table frame, in this case "IDENT". ACCESS places the field on the screen where the cursor was positioned before you pressed FIELD or HOME.

   The field on the screen will look like this:

   ```
           IDENT..........................
   ```

   If you want the field to extend over several lines, the FIELD key cannot be used, and you have to give the number of lines after the field name:

   ```
       ***DEFINE-FIELD  IDENT  2
   ```

   The field on the screen will then look like this:

   ```
           IDENT...........
           ................
   ```

   Note that the total length of the field must be divided equally over the number of lines. Use the HELP key or CTRL+G to see the length of the field.

3) Move the cursor to the place where you want the next field to start. Continue as above until all the fields are placed.

An alternative way of defining a field is as follows:

1) Position the cursor where you want the field to begin.

2) Press the HELP key. ACCESS will displays a menu showing the names and lengths of all fields that may be defined (in other words, the names of the input example elements in the query and the width of the columns they have been placed in).

3) Select the desired field with the up and down arrows and ↵.

MOVE-FIELD

A field is moved from one place to another in the form in the following way:

1) Place the cursor in any position within the field you want to move and "mark" it with the MARK key (or CTRL+V) Note: The marking is invisible. This does not mean that the field is not marked.

2) Move the cursor to where you want the field to begin.

3) Press the MOVE key (or press HOME and give the command ***MOVE-FIELD).

COPY-FIELD

A field is copied from one place to another in the form in the following way:

1) Place the cursor in the field to be copied and use the MARK key (or CTRL+V).

2) Move the cursor to where you want the copied field to begin.

3) Press the COPY key (or press HOME and give the command ***COPY-FIELD).

DELETE-FIELD

A field is deleted from the form in the one of the following two ways:

a) Place the cursor at any point in the field to be deleted and press the DELETE key (or press HOME and give the command ***DELETE-FIELD).

or

b) Place the cursor in the field and press CTRL+D and CTRL+F. With CTRL+D and CTRL+L a whole line is deleted: both the prompt text and the field.

For example, in front of the name field above it would be natural to use the prompt text "NAME". The field on the screen would then look like this:

```
NAME: IDENT...........................
```

REPEAT-LINES

One or more lines can be copied several times by using the command ***REPEAT-LINES in the following way:

1) Mark the line(s) to be repeated by using the MARK key (or CTRL+V) on the first and last line, or twice on one line if there is only one. Place the cursor where you want the copied lines to begin.

2) Press HOME and give ***REPEAT-LINES and the number of repetitions, for example, ***REPEAT-LINES 12. The default number of repetitions is 1.

When you have several fields with the same example element in the output form, they will be filled with values from <u>several</u> records when the query is executed. Note that the output form <u>must</u> contain the same number of fields for each example element. Some of the example elements may not, however, have any fields in the output form. This means that you do not want to display these fields.

EXAMPLE

Suppose you want to have a printout of all the names in the data base with 5 names per page.

You can fill in the table frame like this:

EMPLOYEES	NAME
→ PRINT.	_N

You define the output form like this:

1) Give the command **OUTPUT-FORM

2) Place the cursor on the screen where you want to have the first name. Use the FIELD key and then type "N" (or use CTRL+V, HOME and DEF N).

3) Place the cursor on the dotted line which is now displayed on the screen. Press the MARK key (or CTRL+V) twice. Place the cursor on the line where you want the next name. Note: not on the same line! Give HOME, and ***REP 4.

4) Type "NAME" to the left of the first dotted line.

   The output form will then look like this:

```
        NAME  N................
              N................
              N................
              N................
              N................
```

5) Use the EXIT key (or give HOME and the command EXIT). The query can now be executed or stored for later use. When it is executed, the first five names will be written out. Press the down arrow to get the next five names.

Note: Remember to position the cursor where the lines are to appear, <u>before</u> you give the REPEAT-LINES command. Otherwise the first of the repeated lines will coincide with the last line to be repeated, and this is not allowed.

4.6 NO-FORMFEED

In the above example, if you print the result on a printer, you get five names on each paper page. If, however, you use the command box command NO-FORMFEED, ACCESS prints the result without page breaks. In this case, you get five names, and then only as many blank lines as the output form contained, and then the next five names. If you want a continuous list of names, with no blank lines, you can REPEAT the name field through all 23 lines of the output form.

4.7 TERMINATION

When you have finished editing a form, leave the form definition level by pressing the EXIT key or giving the command ***EXIT. You are then back on the query definition level with a description of the table frame (and command box) on the screen. You can now proceed from where you left off when you gave the command OUTPUT-FORM.

When the query is executed, the result is presented on the defined format. The form definition is then part of the query description and can be stored on a file with the table frame and command box for later use (**STORE-QUERY), or can be written out on an output device (***PRINT-QUERY).

4.8 DELETING THE OUTPUT FORM

When you are back on the query definition level, you can delete the output form by giving the command:

```
**DELETE-OUTPUT-FORM↵
```

C H A P T E R   5

INPUT OF NEW DATA USING EXAMPLE ELEMENTS

5 INPUT OF NEW DATA USING EXAMPLE ELEMENTS

5.1 INPUT OF DATA USING INPUT FORM

An input form may be used to input data which is used by the query when it is executed. The same commands as for output form are available here, <u>except</u> REPEAT-LINES, COPY-FIELD and the COPY key. See section 4.5.

Before you start making the input form itself you must get the table frame onto the screen and give the various data values names. You do this by defining input example elements in the necessary columns, and use them in the input form. The procedure is similar to the one used for the output form. An example is shown below.

Start by calling the table frame, and use the table operator INSERT. The symbolic names A,B,C have the prefix $ to indicate the use of the names. Remember that "_" is used for output example elements and "$" for input example elements. The following example shows a query definition with input form.

The table frame:

EMPLOYEES	NAME	MANAGER	SALARY	DEPARTMENT
→ insert.	$A	$B		$C

The input form:

```
Registration of new employees

Name : A.............................

Boss : B.............................

Department: C.................
```

The command to start defining the input form is analogous to that used for the output form:

```
    **INPUT-FORM <number of lines in form>↵
```

Here, too, you can omit the number of lines, unless you want a form larger than the screen. The maximum number of lines is 72.

All input example elements in the table frame must be included in the input form, or you will get an error message. But all the columns in the table do not have to have an input example element. In the example, SALARY will be set to 0. It can be updated with another query later. Each example element can only be used once in the input form.

If you want to input a lot of data with the same input form, you do not have to go back to the table frame and give the RUN-QUERY command between each input form you fill in. You may create a command box with the command REPEAT as part of the query. ACCESS will then present a new empty input form after each execution. To end a series of registrations, press the EXIT key (or HOME). Note that when you interrupt the procedure, the data currently on the screen will not be put into the data base.

You may jump between the fields in the input picture with the arrow keys. If you make a typing mistake in one of the fields, this can be corrected if you do it before the query is executed.

To start query execution, use the ⇨ key.

All types of data values can be given as input data via input example elements, instead of being entered directly in the query. As an example, let us see how you can define a query which gives a salary increase for one person:

EMPLOYEES	NAME	SALARY
→	$N	upd.+$L

You can then define an input form which asks for name and increase in salary. The query will then be executed as if the data were put directly into the table frame.

Note that "%" cannot be used here. You can increase the salary by 1000, but not by 10%.

5.2 INPUT OF DATA WITHOUT INPUT FORM

If you have a query description with input example elements, but have not defined an input form together with the query, you are allowed to input data at the bottom of the screen when the query is executed.

Note that only one line is available for the data value. For data values meant to occupy more than one line, an input form must be used.

<u>EXAMPLE</u>

You define the query:

EMPLOYEES	NAME	SALARY
→ PRINT.		in($A:$B)

When the query is executed, first

A: and then B:

are displayed at the bottom of the screen, and you can enter the data after ":".

If you want to repeat the query several times, you can also write REPEAT in a command box. When the query is executed, you will get the following question (answer Yes or No):

"Do you want to run the query once more?"

<u>PROMPT-TEXT</u>

The name of the input example element will be the prompt text for the input of data. If you want to have a prompt text with more than one word, you can use the command

```
PROMPT-TEXT <input example element> <prompt text>
```

in the command box. The prompt text will then be displayed when the query is executed.

Suppose you have defined the table frame as shown in the previous example, where the names and salaries of all those between a minimum and a maximum salary are displayed. If you do not want the prompt texts A: and B:, you can define the following command box:

```
  COMMAND-BOX
→ PROMPT-TEXT $A 'Salary from: '
→ PROMPT-TEXT $B 'Salary to: '
```

"Salary from:" and "Salary to:" will then replace "A:" and "B:".

ACCESS asks for the value of the input example elements in the following order.

- First all input example elements that appear together with a PROMPT-TEXT command, and in the order they are written in the command box.

- Then the rest of the input example elements (not necessarily in the order they are written in the table frame).

If you use the command HEADING (see section 6.5.3) in a query with input example elements, you can use the input value in the heading by including the input example element in the heading text, in the following way: ^$a;

EXAMPLE

Task: Produce a list of all employees earning more than the value input and print it on a philips printer, with an appropriate heading.

Solution:

EMPLOYEES	NAME	MANAGER	SALARY	DEPARTMENT
→	p.		p.>$sal	

```
  COMMAND-BOX
→ prompt-text $sal 'Salary from: '
→ heading 'Employees earning more than ^$sal;'
→ print philips
```

5.3 FACILITATING DATA ENTRY

When you want to enter large amounts of data into a register, you need some data entry functions. ACCESS has two command box commands that enable you to enter records faster in an input form.

The commands are:

KEEP-VALUES	- The command requires a list of input example elements, for example KEEP $A $B $C. The values of the fields that use these example elements will stay in place in the input form until you change them. This means that if you use the KEEP command for the manager column, and you want to enter a number of employees having the same manager, the manager name need not be entered more than once. If the next employee has a different manager, the manager name must be changed on the screen.
DEFAULT	- The command requires the name of an input example element and the desired default value. If you do not enter a value in this field, the default value is automatically inserted.

We now want to define a query for entering data in the employee register. To make this process more efficient, we do the following:

1) Use <u>repeat</u>, because we want to input many records.

2) Use <u>keep $boss $dept</u>, because we want the old values to remain on the screen.

3) Use <u>default $salary 15000</u>, because we want the value 15000 to be inserted if no salary is entered.

4) Use <u>default $dept 'Sales department'</u>, because we want the new employees to belong to the Sales department if no department name is entered.

ACCESS will not allow you to use these commands in the command box before you have defined the fields in the input form. You must therefore define the table frame first, then the input form, and finally the command box.

The table frame:

EMPLOYEES	NAME	MANAGER	SALARY	DEPARTMENT
→insert.	$name	$boss	$salary	$dept

The input form:

```
Registration of new employees

Name : name..........................  Salary per year: salary...

Boss : boss..........................

Department: dept...............
```

The command box:

```
 COMMAND-BOX
→ repeat
→ keep    $boss $dept
→ default $salary 15000
→ default $dept   'Sales department'
```

After entering some records, the screen may look like this:

```
Registration of new employees

Name : _.............................  Salary per year: .........

Boss : Audrey Sands..................

Department: Marketing
```

5.4 DELETING THE INPUT FORM

The input form can be cleared from the query with the command

**DELETE-INPUT-FORM↵

on the query definition level.

C H A P T E R 6

PROCESSING AND PRINTING RESULTS

6 PROCESSING AND PRINTING RESULTS

6.1 THE PRINTING OF RESULT LEVEL

When you execute a query which PRINT. or a function operator, the result is displayed on the screen and you automatically go down to the printing of result level, which has its own set of commands.

You can interrupt a query by pressing the EXIT key or the CANCEL key (or pressing ESC twice). However, queries which include DELETE., INSERT., or UPDATE. will not be interrupted.

6.2 NAVIGATING COMMANDS

Navigate in the screen display by using the navigate commands and the arrow keys:

↑ or PREVIOUS-PAGE	gives the previous page
↓ or NEXT-PAGE	gives the next page
1 or ***FIRST-PAGE	gives the first page
$ or ***LAST-PAGE	gives the last page
←	moves you to the left in the result table
→	moves you to the right in the result table
***MOVE-TO-PAGE <pageno.> or 1-9	moves you to the given page. If the number is less than 10, the number alone can be used.

6.3 FURTHER PROCESSING OF THE RESULT

After you obtain a result from a data base with the operator PRINT or a function operator, you can carry on working with it without having to go back to the data base. This saves time and increases the efficiency of the system. It is also possible to put in new data or update records in the result file before you print it.

Use the command ***REFINE-RESULT, or the

⇨ key.

See figure 15.

The table below is the first result you get from ACCESS.

NAME	MANAGER	SALARY	DEPARTMENT

After the REFINE-RESULT command, a new table is displayed which has the same columns as the result, and the table name is "LAST RESULT", as shown below:

LAST RESULT	NAME	MANAGER	SALARY	DEPARTMENT
→				

This can now be completed as an ordinary table frame. All the usual operators and selection criteria are allowed. The query is executed as before with

⇨

or 'HOME' and REFINE-RESULT. Note that you can also 'update' the result, but the contents of the data base are not affected.

The operators INSERT., DELETE. and UPDATE. do not give a display of the result. After having used one of these, you therefore get back to the 'Printing of result level', where you can define more queries against the result.

When you use PRINT. or a function operator, a new result is diplayed. You can define further queries against this result with ***REFINE-RESULT.

You cannot use the commands OUTPUT-FORM, INPUT-FORM or COMMAND-BOX in connection with REFINE-RESULT. Note that the original result is forgotten when you use REFINE-RESULT again. Only the last result is remembered.

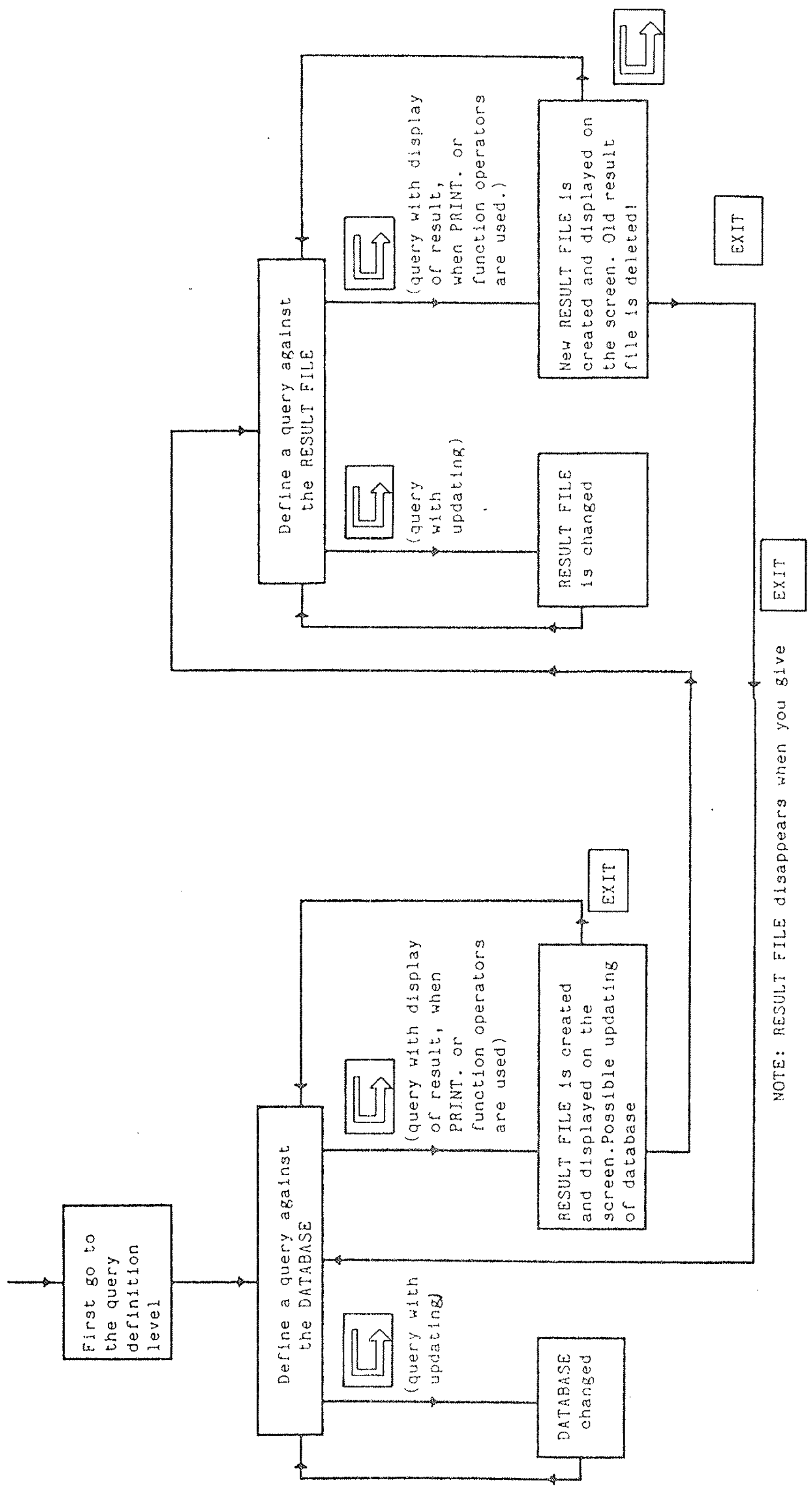

First go to the query definition level
Define a query against the DATABASE
(query with updating)
DATABASE changed
(query with display of result, when PRINT. or function operators are used)
RESULT FILE is created and displayed on the screen.Possible updating of database
EXIT
Define a query against the RESULT FILE
(query with updating)
RESULT FILE is changed
(query with display of result, when PRINT. or function operators are used.)
New RESULT FILE is created and displayed on the screen. Old result file is deleted!
EXIT
EXIT
NOTE: RESULT FILE disappears when you give

Fig. 15. Queries against DATA BASE and RESULT FILE

The legal commands are:

HELP	Display help information.
REFINE-RESULT	Run query.
EXIT	Leave this level.

Example

You find all employees in the example data base by filling in the table frame in the usual way:

EMPLOYEES	NAME	MANAGER	SALARY	DEPARTMENT
→ print.				

With this result as a basis, you want to find everyone who earns more than 20000, and you are only interested in the columns NAME, SALARY and DEPARTMENT. MANAGER has been deleted with CTRL+D and CTRL+C.

You give the command ***REFINE-RESULT, and the new table is displayed on the screen.

Last result	NAME	SALARY	DEPARTMENT
→ print.		>20000	

You have now got a result, but you want to exclude all those who work in Sales from the list. You repeat the command ***REFINE-RESULT and fill in the frame thus:

Last result	NAME	SALARY	DEPARTMENT
→ print.			NOT 'Sales'

The records you have now found are to be printed. See the next section. Press the EXIT key (or give the command "***EXIT") to get back to the query definition level.

6.4 PRINTING THE RESULT

6.4.1 PRINT

```
***PRINT <file name>,<number of lines per page>↵
```

- The command is used to print the result of a query somewhere else than the screen, usually a printer or a text file.

- You can have from 1 to 70 lines per page. On an A-4 sheet, it is advisable not to have more than 64 lines. The default value is the number of lines on the screen (21 on NOTIS terminals).

- If you want to write the result of the query to a file, consider the following points:

  - The file does not have to already exist. You create a new file by enclosing the file name in double quotes:

    ```
    ***PRINT "OSCAR"↵
    ```

  - The file type will be :SYMB, which is the default for printout in ACCESS. The complete name of the file will then be OSCAR:SYMB.

  - If you want to have another file type, this must be specified:

    ```
    ***PRINT "RESULT:TEXT"↵
    ```

EXAMPLE

Task: You want to print on the line printer a list of those working in the Development department, in order of decreasing salary. You also want the column SALARY to be named ANNUAL SALARY and to be positioned before the column NAME. The heading should be 'Annual Salary in Development'. The result will be printed using the command box.

Solution:

EMPLOYEES	NAME	MANAGER	SALARY	DEPARTMENT
→	print.		print.desc.	'Development'

```
  COMMAND-BOX

→ print line-printer
→ heading 'Annual Salary in Deve
  lopment'
→ rename SALARY,'ANNUAL SALARY'
→ reorder 'ANNUAL'
```

Note: the width of the command box can be increased with the

>< <> key or CTRL+V.

Result:

Annual Salary in Development

ANNUAL SALARY	NAME
9200	Sam Boyle
9000	Paul Rosier
8500	John Bell
8000	Diane Hyde
6400	John Mills
6000	Ann Platt
5000	Ann Bow
5000	Richard Hill
3300	Geoff Carver

etc.

The commands RENAME-COLUMNS, REORDER-COLUMNS and HEADING are explained in sections 6.5.1 to 6.5.3.

6.4.2 WRITE-WITHOUT-FRAME

***WRITE-WITHOUT-FRAME <file name> <number of lines per page>↵

- Like ***PRINT, this command prints the result of a query on a printer or a file, but without the column names and frame. This may be useful for instance when the result is to be included in a report written in NOTIS-WP.
- if the number of lines is omitted, no page shifts will be generated, and the file may for example be used as input for the command **LOAD-TEXT-FILE.

6.4.3 STORE-DATA

***STORE-DATA <file name>↵

- This command writes the result on a file as you get it from the data base. It is used when you need a special data file (SINTRAN file) outside the data base, for further processing in another system or for the command **LOAD-DATA-FILE.

6.4.4 DESCRIBE-MACRO-CALL AND WRITE-MACRO-CALL

It is here assumed that the reader is familiar with the word processing system NOTIS-WP and the text formatter NOTIS-TF.

These two commands are particularly useful for standard letters. The command ***WRITE-MACRO-CALL <file name> <macro> <separator> takes data from a result table and writes it as macro calls on the specified file. Let us say you have the following result table on the screen:

NAME	DEPARTMENT
John Mills	Building
Ian Rain	Building
Christopher Howard	Building
Harold Ritchie	Building

If you now give the command

```
    ***WRITE-MACRO-CALL address-list adr =↵
```

ACCESS will write the macro calls on the file "address-list:text", as follows.

```
^adr=Ian Rain           =Building       ;
^adr=Christopher Howard=Building        ;
^adr=Harold Ritchie     =Building       ;
```

Here ACCESS has filled in the first two macro parameters. If the result table had contained a different number of columns, say four, then the first four parameters would be filled in.

It may be that you need to leave one or more macro parameters blank. This can be achieved by using the command DESCRIBE-MACRO-CALL before WRITE-MACRO-CALL. DESCRIBE-MACRO-CALL is used to tell ACCESS where in the macro call you want ACCESS to put the data from the result table. For example, if you want only parameters 1 and 3 filled in, give the command in this way:

```
    ***DESCRIBE-MACRO-CALL ^adr/^//^;↵
```

Let us look at a more complete example.

You have updated the salaries of a group of employees, and you want to send them a standard letter informing them of their new salary. The procedure is as follows.

1) Use NOTIS-WP to write the letter text, as in the following example.

```
This is to inform you that your 1984 yearly salary has been
increased to $^item-1;. The total raise is 10%, which
includes a 6% cost of living raise and a 4% merit raise.




                          Roger Lawrence
                          Manager
```

2) Store this text on a file called "letter-text:text". In the final letter texts, ^item-1; will be replaced by each employee's new salary.

3) Use ACCESS to get a result table, like the following, on the screen:

NAME	DEPARTMENT	SALARY
John Mills	Building	24700
Ian Rain	Building	24000
Christopher Howard	Building	18200
Harold Ritchie	Building	12400

The columns DEPARTMENT and SALARY have been interchanged by using the command REORDER-COLUMNS (see section 6.5.1).

4) Give the command:

```
***DESCRIBE-MACRO-CALL ^adr/^/^/////^;↵
```

Now you have told ACCESS that you want the data from the result table put into parameters 1, 2 and 7 in the "^adr" macro. These parameters are "name", "address" and "item-1".

5) Give the command:

```
***WRITE-MACRO-CALL address-list↵
```

The macro name need not be specified here, since it was given in the previous command.

ACCESS will generate the macro calls on the file ADDRESS-LIST:TEXT

```
^adr/John Mills        /Building        /////        24700;
^adr/Ian Rain          /Building        /////        24000;
^adr/Christopher Howard/Building        /////        18200;
^adr/Harold Ritchie    /Building        /////        12400;
```

Note: As in this example, there may be blanks between a separator and a number. These blanks will appear in the finished letter. If you want to avoid them, remove them with SHIFT + JUST in NOTIS-WP.

6) Use NOTIS-WP to create a file containing the following single line and format it with NOTIS-TF.

```
^stdletter/address-list/letter-text;
```

This one line can be thought of as the recipe you hand over to NOTIS-TF to enable it to find the ingredients for the letters. The text formatter takes the letter text (the vegetables) from the file LETTER-TEXT:TEXT, spices it with the addresses from ADDRESS-LIST:TEXT, and cooks the finished letters.

6.4.5 BAR-CHART

This command is used for the graphic representation (a bar chart) of the result of a query. ACCESS will ask for the name of the output file. This can be a disk file, a printer or a terminal. After having given the command, you will be asked for parameters: variables along the x-axis and y-axis, maximum and minimum values, etc.

The dialog on the screen:

Column along the horizontal axis:	Here you answer with the field name of the parameter along the x-axis.
Column along the vertical axis:	The name of a numeric field.

The parameters along the axes must be given, but ACCESS uses default values for the rest of the parameters unless you specify them:

Enter heading:	Give a text string as the heading.
Distance between bars:	This is the number of character positions from the left edge of one bar to the left edge of an adjacent one. The minimum value is 8. The bar itself always occupies 3 positions horizontally.
Lower value on the vertical axis (Default=Minvalue):	Default is the minimum encountered value
Upper value on the vertical axis (Default=Maxvalue):	Default is the maximum encountered value. If the upper value is set at less than the maximum encountered value, ACCESS will change this to the maximum value. The same is the case for the minimum value.

Number of lines per page:	This is the total height of the bar chart, including the heading and the bottom text line.
Number of values to be displayed on the vertical axis:	The number of values displayed along the vertical axis. Default is 3: minimum, medium and maximum value.

An example of the use of this command is shown in fig. 16.

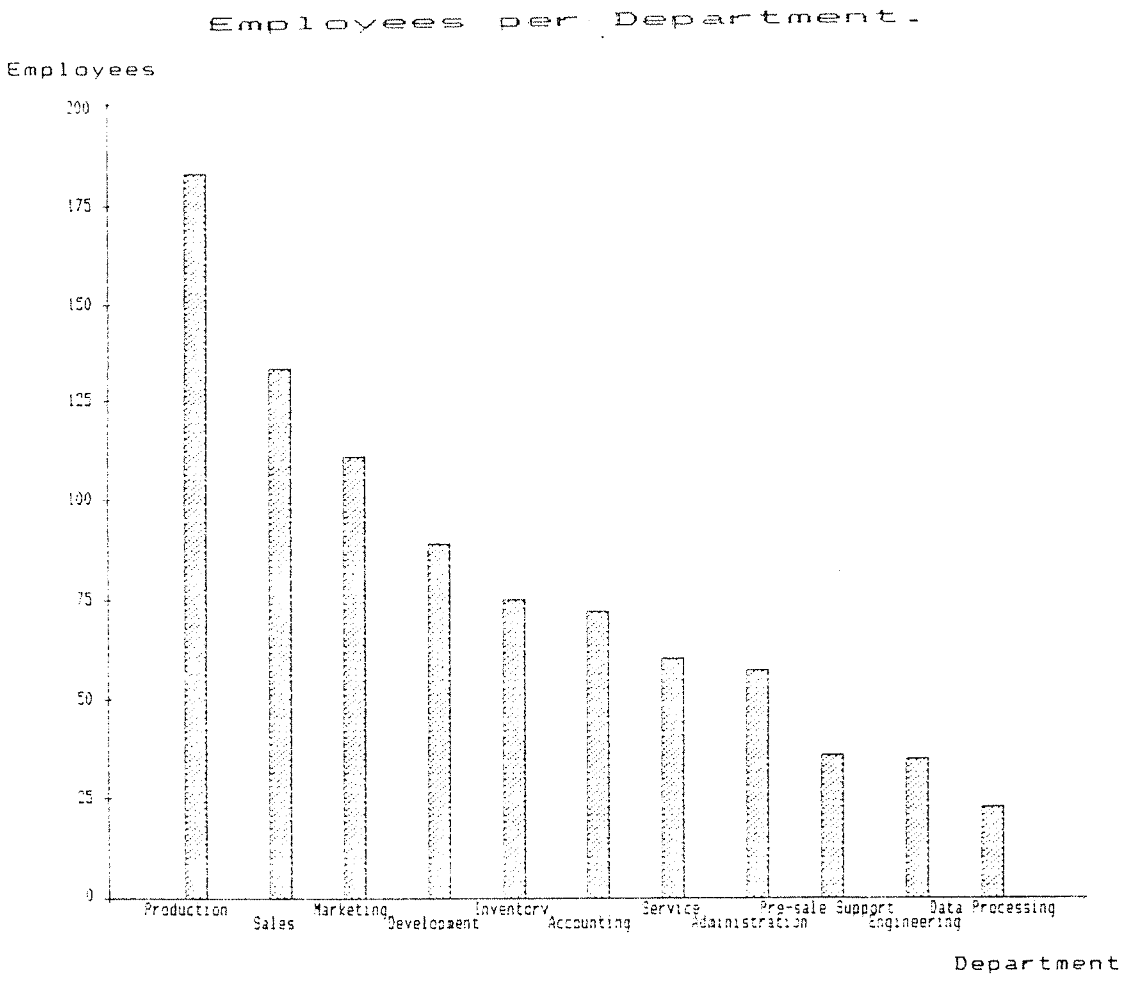


Fig. 16. Example of the use of BAR-CHART

6.5 EDITING THE RESULT

6.5.1 REORDER-COLUMNS

Format:

```
***REORDER-COLUMNS <item-1> <item-2>,....,<item-6>↵
```

- If you want the columns in the printout arranged in a different order, you can use this command.

- If you only write some of the field names in the list, those columns which have not been mentioned will come in the original order, but after the columns given in the command.

6.5.2 RENAME-COLUMNS

Format:

```
***RENAME-COLUMNS <old name> <new name>↵
```

- This command is used to change the name of a column in the table.

- If you want to change more than one name, you must use the command several times.

- If the new name consists of several words, it must be enclosed in single quotes.

- Old and new names are separated by a comma or a blank:

```
***RENAME-COL SALARY,'Salary paid in 1983'↵
```

6.5.3 HEADING

Format:

```
***HEADING '<text>'↵
```

- This command provides a heading to a table when the table is written out with the command ***PRINT.

- The text is left justified. The text, as it appears in single quotes, is placed to the far left in the column where you place the heading. You can move the text to the right on the screen by placing blanks at the start of the text field.

  ```
  ***HEADING '     This is a heading'
  ```

6.5.4 CLEAR-HEADING

If you want to delete a previously defined heading, give the command:

```
***CLEAR-HEADING↵
```

6.6 A REPORT EXAMPLE USING NOTIS-WP

Define a query as follows:

EMPLOYEES	NAME	MANAGER	SALARY	DEPARTMENT
→ _L1			>10000	Marketing
→ _L1	print.		pri.asc.	

COMMAND-BOX
→ print 'marksalary:text',60

We assume that the file MARKSALARY exists.

Press the ⇥ key to execute the query,

or enter command position (HOME twice from the command box) give the command: **RUN-QUERY↵

When the query has been executed, ACCESS reports this, writes the result to the file MARKSALARY and displays it on the screen. To leave ACCESS, press the EXIT key three times (or type EXIT three times).

You are now out of ACCESS and can call NOTIS-WP. In NOTIS-WP you can write the report text and make use of the macro 'report'(see below). The table of salaries which you made in ACCESS is called to the final report by writing:

```
^in,MARKSALARY;
```

and the file which ACCESS created will be put in where you have specified.

The whole file written in NOTIS-WP may look as follows:

```
^report/Salaries in Marketing/MHA
/Salary/20 May 1984
/The matter was discussed on May 19.
In reply to the question put forward, the following summary covers
all employees in Marketing earning over 10000 per annum.
Comments from those concerned are requested by the undersigned.
/OP,KJ,DK,TS;
^in,MARKSALARY;
```

Then give the command:

```
WP:W "salaryreport"↵
```

The file is stored and the formatter NOTIS-TF is activated by typing 'J'.

The file is automatically formatted and displayed on the screen when ready. The whole file SALARYREPORT is displayed, including MARKSALARY.

To print the result on a printer, type:

```
WPI:C <printer name>
```

and the complete report is printed on the specified printer. Give the command:

```
WP:E    (EXIT)
```

to leave NOTIS-WP.

The search and the name list were the work of ACCESS, and the report was produced by means of NOTIS-WP.

The procedure was carried out on the basis of the example data base. You may, perhaps, find that 85 people earn more than 10000 in Marketing and the report may look like the following. (Note that with very small changes you can make similar reports for other departments and for other purposes.)

REPORT

Salaries in Marketing

Author:	MHA
To:	OP,KJ,DK,TS
Reference:	Salary
Written:	20 May 1984
Last updated:	30 May 1984

Abstract:

The matter was discussed on May 19. In reply to the question put forward, the following summary covers all employees in Marketing earning over 10000 per annum. Comments from those concerned are requested by the undersigned.

Name	Salary
Fred Atkins	11100
Margaret Benson	11200
Sid Stevens	11900
Christopher Eilifsen	12900
Jim Dale	13800
Mike Small	14100
Roger Mills	15100
Alex Sykes	17800
Ann Mills	17800
Christine Hall	18100
Dave Eilifsen	18300
Larry Offman	18700
Pam Offman	19400
Thad Ritchie	19800
John Rosier	20200
Richard Platt	22200
Joan Briggs	22300
Fred Mills	22400
Jodie Ritchie	23000

etc.

Fig. 17. Salary report

6.7 MOVING UP TO QUERY DEFINITION LEVEL

To get out of the printing of result level, press the EXIT key (or type EXIT in the command position). You then come back to the level where the execution of the query started. If this was the query definition level, you may edit the last query description. The background picture with the table frame and command box appears on the screen, and you automatically enter table position.

If you want to execute several queries in succession, you can use the same table frame for all of them by editing it between each query.

6.8 ERROR MESSAGES AND CORRECTIONS

If you try to execute a special query which is not permitted, or if you make some other mistake, an error message is given and the query will be interrupted.

The error message describes what is wrong and ACCESS goes into table position.

You must then correct the error before you proceed with the RUN-QUERY command.

CHAPTER 7

DEFINING MENUS

7 DEFINING MENUS

7.1 THE MENU COMMAND

In this version of ACCESS, experienced users can define menu pictures, which can be used by inexperienced users and novices.

This is done with the MENU command in the COMMAND-BOX. Each menu command sets up one item in a menu picture. Up to 9 menu entries can be defined per picture. Each item has information about which command to run, the query name and the text to be displayed in the menu picture.

We have the following two ways of using the menu:

1) Use only a COMMAND-BOX with menu commands. The menu picture will be displayed on the screen when the query is started.

2) Use the command box with menu commands in an ordinary query. The menu picture will not be displayed until the query is finished. This feature is useful for making a "what do you want to do with the result" type of decision.

The menu command has the following description:

```
MENU <command> <query name or SINTRAN command> <leading text>
```

command	- RUN	- Run a prestored query.
	FETCH	- Fetch a query description.
	@	- Exit ACCESS and execute a SINTRAN command <1>.
query name	- The query to be executed or fetched or the SINTRAN command to be executed.	
leading text	- The text to be displayed in the menu picture.	

Two more commands can be used with the MENU command:

MENU-HEADING <Heading text>

MENU-FOOTING <Line no><Footing text>

<Line no> is a number from 1 to 3. These denote the last three lines in the menu picture. The very last line is no. 3.

<1> This function can be used to start another subsystem directly from ACCESS. This, however, will not work on the ND-500.

EXAMPLE

We have the following query to define a menu picture:

```
 COMMAND-BOX

→ menu run     empl-menu 'Display employee menu'
→ menu run     query-1   'Execute query no. 1'
→ menu run     list-empl 'List all employees'
→ menu fetch   query-1   'Modify query no. 1'
→ menu @notis-wp  'NOTIS-WP'
→ menu-heading 'M A I N   M E N U'
→ menu-footing 1 'Use the up and down arrows to move bet
  ween menu items.'
→ menu-footing 2 'Use the "carriage return" key to selec
  t.'
```

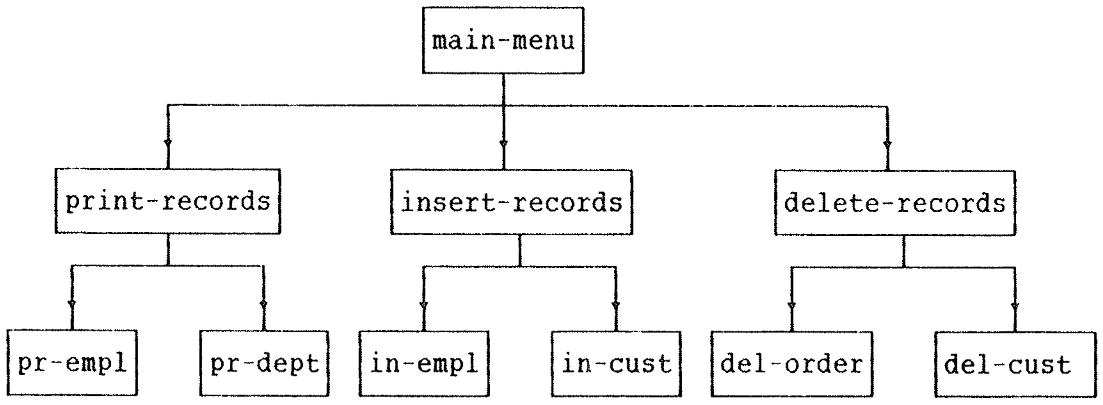
When this query is run, the following picture is displayed on the screen:

```
                    M A I N   M E N U

  1 Display employee menu
  2 Execute query no. 1
  3 List all employees
  4 Modify query no. 1
  5 NOTIS-WP


  Use the up and down arrows to move between menu items.
  Use the "carriage return" key to select.
```

The current menu item is displayed in inverse video on the screen. The user can move between the menu entries with the up and down arrows.

7.2 THE MENU TREE

The query "empl-menu" also consists of a command box with menu commands. Therefore, when "Display employee menu" is selected, the employee menu is displayed. This feature makes it possible to create a "tree" of menus:



When the user leaves a menu picture, control is given to the previous menu picture. When the first menu picture is terminated, the control is given to the first level of ACCESS.

In addition, the ACCESS supervisor can define a default query name in the ACCESS command file. How this is done is described in the DBA manual.

When this has been done, ACCESS automatically starts executing the default query when a user enters ACCESS. If the query consists of a menu picture, it is possible to run ACCESS entirely from a menu system, particularly if the user only needs to run predefined queries. It is also possible to give different users different command files, so that some users may have such a system, while others do not (see the DBA manual).

CHAPTER 8

MISCELLANEOUS COMMANDS

8 MISCELLANEOUS COMMANDS

8.1 SINTRAN COMMANDS

SINTRAN commands may be given on all command levels in ACCESS. You type @ followed by the SINTRAN command, and the command will be executed in SINTRAN. An example:

```
    **@WHO↵
```

As a rule, you will come back to the position in ACCESS where you gave the command. If you call another program, for example NOTIS-WP, or give the wrong SINTRAN command, you will not get back to ACCESS, but will have to call ACCESS again. SINTRAN commands should therefore be used with care.

Not all SINTRAN commands can be executed from ACCESS on the ND-500.

8.2 HELP

The HELP command or the HELP key can be used at all levels in ACCESS. It gives a picture on the screen showing available commands. Each level in the system has its own set of available commands and they are displayed together with a brief description of how they are used.

The HELP key can also be used in the table frame (see section 3.6.1), in the command/condition box, and in the input/output form (see section 4.5). SHIFT+HELP can be used in the table frame to find out on what file the data is stored (use SHIFT+HELP in the table column), and to get information on data types and key types (use SHIFT+HELP in the field columns).

8.3 OPEN-LIBRARY

Whereas earlier versions of ACCESS stored each query on a separate file, the present version stores many queries on one file. Such a file is called a query library. Normally all queries will be stored in a default query library. However, you may want to keep queries stored in more than one library. In this case, you can use the following start level command to switch between different library files:

```
    *OPEN-LIBRARY <library name>
```

ACCESS then uses this library when STORE-QUERY or FETCH-QUERY is executed.

The default type of the query library is :TRAN.

8.4 LIBRARY

The command displays the name of the current query library.

The call sequence is:

```
    *LIBRARY↵
```

The output looks like this:

```
    The query library name is: LIBRARY:TRAN
```

8.5 LIST-QUERIES

The command will list the names of all queries on the currently opened query library.

The command is given in this form:

```
    *LIST-QUERIES <query name>↵
```

If the query name is not given, all queries on the query library are displayed.

If the query name is given, ACCESS displays all queries of which the given name is an abbreviation.

If the command is given as *LIST-QUERIES INS, the result could look like this:

```
           L I S T   O F   S T O R E D   Q U E R I E S

   Query name                        Query size in bytes

   INSERT-EMPLOYEES                       12345
   INSERT-DEPARTMENTS                      9807
   INSERT-CUSTOMERS                        1032
```

It is possible to move from page to page with the arrow keys.

↓	gives the next page
↑	gives the previous page
EXIT or ↖	terminates the command

8.6 CONVERT-QUERIES

The command enables the user to convert queries defined in the B and C releases of ACCESS into the new format (several queries on one file).

The command is given as:

*CONVERT-QUERIES <library name>↵

The old queries are copied from their old files to the given library or, if no library name is given, to the current query library.

The following picture will be displayed on the screen:

```
            CONVERSION OF ACCESS-B AND -C QUERIES

 In this version of ACCESS (D version and later), many queries are
 stored on a  single  library file.

 This  command  converts  queries  from ACCESS B and C, where each
 query was placed on its own file,. to the new format.

 Enter  a  new  name  for  the query and the name of the old :TRAN
 file.

 Use  HOME  or  EXIT  when you have finished converting queries.

                 New query name : ..............................
                 Old query file : ..............................


 Afterwards you should FETCH all the converted queries, check their
 syntax (all text in single quotes; UPDATE. used alone should be
 changed to UPDATE.=), and STORE them.
```

WARNING:

If you forget to change UPDATE. used alone (for example, UPDATE.15) to UPDATE.=, you do not get an error message, but ACCESS interprets the query differently. In the B and C versions of ACCESS, UPDATE.15 had the same meaning as UPDATE.=15; in the D version, it is equivalent to UPDATE.+15.

8.8 DELETE-QUERY

This command deletes a stored query.

The command is given as:

```
    *DELETE-QUERY <query name>↵
```

8.9 RESERVE-DATABASE

This command makes ACCESS work faster.

Since the data base files can only be used by one user at a time, ACCESS must reserve a file when it is needed. Usually, it is released immediately afterwards. This process of reserving and releasing files is time consuming. The command *RESERVE-DATABASE makes ACCESS try to minimize this time.

Give the command a second time to disable it!

After you have given the *RESERVE-DATABASE command the first time, a reserved file is not released immediately after the work on it is finished. ACCESS keeps the file reserved until you get back to the start level or give a SINTRAN command. In this way the command reduces the number of 'reserve' and 'release' operations.

However, the files are unavailable for other users for quite a long time. The command should therefore not be used when other users work on the data base at the same time.

8.10 SET-PASSWORD AND CLEAR-PASSWORD

These commands make it possible for the user to have full control of his/her own password to ACCESS. Use them to set your password, clear your old password or replace it by a new one. Passwords are used to protect the data base from unauthorized use.

8.11 PRINT-QUERY

With this command, you can write a query description to a file or print it on a printer. This is useful if you want to have a record of stored query descriptions or just be able to remember a procedure for a later occasion.

It can also be useful to have a printout of the query description together with a printout of the result.

You do this by giving the command:

```
    **PRINT-QUERY <file name>↵
```

The parameter <file name> may be the name of a file or a printer.

If you write it to a file, note that this is not the same as storing the query description. ACCESS cannot use a printed query description as input. For this purpose you must use the STORE-QUERY command.

By using the **PRINT-QUERY command, only the pictures that describe the query are stored, the way they look on the screen.

If you use the command **STORE-QUERY, ACCESS's interpretation of the query is stored. You cannot get this out in a readable form on a printer, or read it into NOTIS-WP or PED.

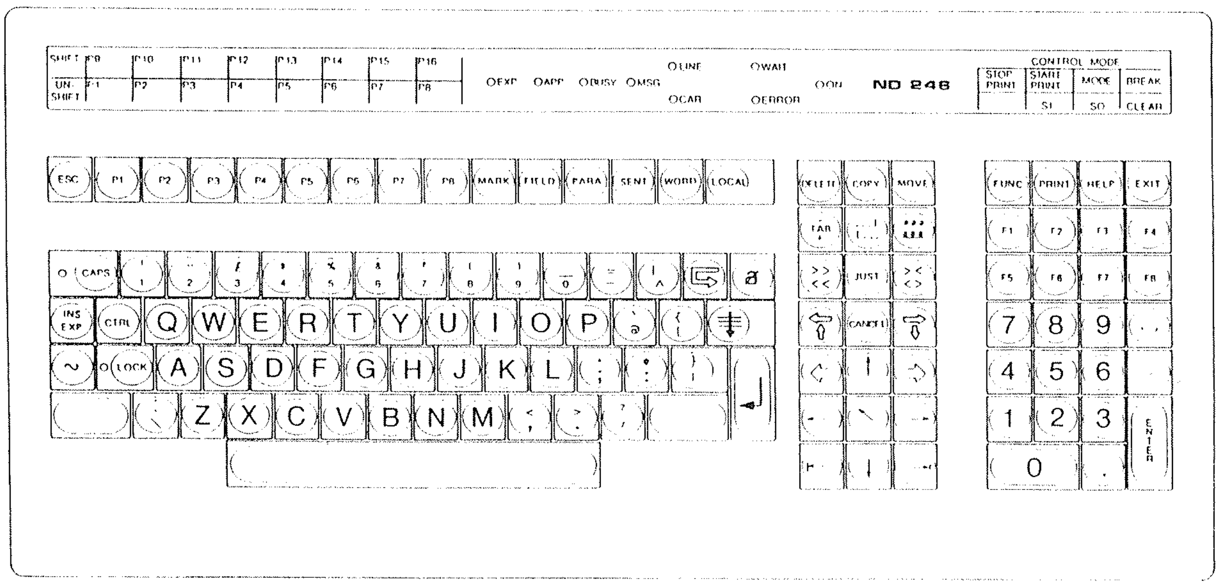
8.12 CLEAR-QUERY

```
    **CLEAR-QUERY↵
```

This command can be used when you have defined a query and want to start on a new one. It clears the query description you have on the screen, including all table frames, boxes and forms. It does not delete a query stored in the query library.

A P P E N D I X   A

THE KEYBOARD ON THE NOTIS TERMINAL (TANDBERG 2200/9)


SHIFT
UN-SHIFT
P9 P10 P11 P12 P13 P14 P15 P16
P1 P2 P3 P4 P5 P6 P7 P8
EXP APP BUSY MSG LINE CAR WAIT ERROR ON
ND 246
CONTROL MODE
STOP PRINT
START PRINT
SI
MODE
SO
BREAK
CLEAR
ESC P1 P2 P3 P4 P5 P6 P7 P8 MARK FIELD PARA SENT WORD LOCAL
DELETE COPY MOVE
TAB
JUST
CANCEL
FUNC PRINT HELP EXIT
F1 F2 F3 F4 F5 F6 F7 F8
CAPS
INS EXP
CTRL
LOCK
Q W E R T Y U I O P
A S D F G H J K L
Z X C V B N M
7 8 9 4 5 6 1 2 3 0
ENTER

A P P E N D I X   B

ERROR MESSAGES IN ACCESS

SINTRAN ERROR MESSAGES

Messages marked with an asterisk (*) will be used by ACCESS/DBA.

Code	Explanation
0	Not used
1	Not used
2	Bad file number
3	End of file
4	Card Reader Error (Card Read)
5	Device not reserved
6	Not used
7	Card Reader Error (Card not read)
8	Not used
9	Not used
10	End of device
11	Not used
12	Not used
13	Not used
14	Not used
15	Not used
16	Not used
*17	Illegal character in parameter
*18	No such page
19	Not decimal number
20	Not octal number
21	You are not authorized to do this
22	Directory not entered
23	Ambiguous directory name
24	No such device name
25	Ambiguous device name
26	Directory entered
27	No such logical unit
28	Unit occupied
29	Master block transfer error
30	Bit file transfer error
31	No more tracks available
32	Directory not on specified unit
33	Files opened on this directory
34	Main directory not last one released
35	No main directory
36	Too long parameter
*37	Ambiguous user name
38	No such user name
39	No such user name in main directory
40	Attempt to create too many users
41	User already exists
42	User has files
43	User is entered
44	Not so much space unreserved in directory
45	Reserved space already used

S I N T R A N - III Error codes returned by A C C E S S

Code	Explanation
*46	No such file name
*47	Ambiguous file name
48	Wrong password
49	User already entered
50	No user entered
51	Friend already exists
52	No such friend
53	Attempt to create too many friends
54	Attempt to create yourself as friend
55	Continuous space not available
56	Not directory access
57	Space not available to expand file
58	Space already allocated
59	No space in default directory
60	No such file version
*61	No more pages available for this user
*62	File already exists
*63	Attempt to create too many files
64	Outside device limits
65	No previous version
*66	File not continuous
67	File type already defined
68	No such access code
*69	File already opened
*70	Not write access
*71	Attempt to open too many files
*72	Not write and append access
*73	Not read access
*74	Not read, write and common access
*75	Not read and write access
*76	Not read and common access
*77	File reserved by another user
78	File already opened for write by you
79	No such user index
*80	Not append access
*81	Attempt to create too many mass storage files
*82	Attempt to open too many files
83	Not opened for sequential write
84	Not opened for sequential read
84	Not opened for random write
86	Not opened for random read
*87	File number out of range
88	File number already used
89	No more buffer space
90	No file opened with this number
91	Not mass storage file
*92	File used for write
93	File used for read

S I N T R A N - III Error codes returned by A C C E S S

Code	Explanation
94	File only opened for sequential read or write
95	No scratch file opened
96	File not reserved by you
97	Transfer error
98	File already reserved
99	No such block
100	Source and destination equal
101	Illegal on Tape device
102	End of tape
103	Device unit reserved for special use
104	Not random access on tape files
105	Not last file on tape
106	Not tape device
107	Illegal address reference in monitor call
108	Source empty
*109	File already opened by another user
*110	File already opened for write by another user
111	Missing parameter
112	Two pages must be left unreserved
113	No answer from remote computer
114	Device cannot be reserved
115	Overflow in read
116	DMA error
117	Bad datablock
118	CONTROL/MODUS word error
119	Parity error
120	LRC error
121	Device error (read-last-status to get status)
122	No device buffer available
123	Illegal mass storage unit number
124	Illegal parameter
125	Write-protect violation
126	Error detected by read after write
127	No EOF mark found
128	Cassette not in position
129	Illegal function code
130	Time-out (no datablock found)
131	Paper fault
132	Device not ready
133	Device already reserved
134	Not peripheral file
135	No such queue entry
136	Not so much space left
137	No spooling for this device
138	No such queue
139	Queue empty
140	Queue full
141	Not last used by you

S I N T R A N - III Error codes returned by A C C E S S

Code	Explanation
142	No such channel name
143	No remote connection
144	Illegal channel
145	Channel already reserved on remote computer
146	No remote file
147	Formatting error
148	Incompatible device sizes
149	Remote processor not available
150	Tape format error
151	Block count error
152	Volume not on specified unit
153	Not deleted record
154	Device error
155	Error in object entry
156	Odd number of bytes (right byte in last word insignificant)
157	Error in backspace/forward space print
158	Block format error
159	Overflow in write
160	Illegal device type
161	Segment not contiguously fixed
162	Segment not fixed
163	Approaching end of accounting file
164	End of accounting file encountered
165	No more unused spooling files available
166	Inconsistent directory
	Reserved for special use

A P P E N D I X C

TABLES USED IN THE EXAMPLES

The tables used in the examples are:

EMPLOYEES	NAME	MANAGER	SALARY	DEPARTMENT
→ Data type:	Character-32	Character-32	Integer-4	Character-20
→ Key:	Primary	Secondary		Secondary

DEPARTMENTS	NAME	FLOOR
→ Data type:	Character-20	Numeric-3
→ Key:	Primary	

The table descriptions and some sample data will be delivered together with ACCESS.

Index

*************** 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 User Guide   Manual number: ND–60.153.03

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

Answer from Norsk Data

Answered by

Date

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

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