Database Access

You will now write a short program which reads specific data from a database file and displays the corresponding output.

When you have completed the exercises below, your sample application will consist of just one module (the data fields that are used by the program are defined within the program):

Program
PGM01
Local
Data

This chapter contains the following exercises:

- Saving Your Program Under a New Name
- Defining the Required Data Using a View
- Reading Data from a Database
- Reading Selected Data from a Database

Saving Your Program Under a New Name

You will now create a new program which will be used in the remainder of this tutorial. It will be created by saving your Hello World program under a new name.

To save the program under a new name

1. In the program editor's command line, enter one of the following:

```
SAVE PGM01
```

```
SA PGM01
```

   The current program is saved with the new name `PGM01`. The program named `HELLO` is still shown in the program editor.

2. Read the newly created program into the program editor by entering the following in the program editor's command line:

   ```
   READ PGM01
   ```

   The program name which is displayed in the program editor changes to `PGM01`.

3. Delete all code in the program editor. To do so, enter the following line command at the beginning of each line to be deleted and press ENTER:

   ```
   D
   ```

   Example:

   ```
   >> -----------Columns 001 072 <<  Program PGM01    Lines  4      User SAG
   Command ===>                                       Mode   Struct Lib  TUTORIAL
   ****** ****************************** top of data ****************************
   D00010 * The "Hello world!" example in Natural.
   D00020 *
   D00030 DISPLAY "Hello world!"
   D00040 END /* End of program
   ****** **************************** bottom of data ***************************
   ```

   Or:
   Enter the following line command at the beginning of the first line and press ENTER:

   ```
   D4
   ```

   where the number after the command indicates the number of lines to be deleted.

   Or:
   Enter the following line command in the command line, move the cursor to the first line to be deleted and press ENTER:

   ```
   :D4
   ```

   Line commands can also be entered in the command line of the editor screen. In this case, the command must be preceded by a colon (:). It always applies to the line marked by the cursor.

Defining the Required Data Using a View

The database file and the fields that are to be used by your program have to be specified between `DEFINE DATA` and `END-DEFINE` at the top of the program.

For Natural to be able to access a database file, a logical definition of the physical database file is required. Such a logical file definition is called a data definition module (DDM). The DDM contains information about the individual fields of the file. DDMs are usually defined by the Natural administrator.

To be able to use the database fields in a Natural program, you must specify the fields from the DDM in a view. For this tutorial, we will use the DDM for the `EMPLOYEES` database file.

Since you have deleted all lines of your previous Hello World program, the program editor currently looks as follows:

```
>> -----------Columns 001 072 <<  Program PGM01    Lines         User SAG
Command ===>                                       Mode   Struct Lib  TUTORIAL
****** ****************************** top of data *****************************
****** **************************** bottom of data ****************************
```

Before you can enter the code for your program, you have to insert blank lines.

To insert blank lines

- Enter the following line command at the beginning of the line which contains the words "top of data" and press ENTER:

  ```
  I
  ```

  This inserts one blank line and the editor switches to insert mode. You can now enter your program code (see below). When you press ENTER, a new line is inserted. If you do not enter any data in a newly inserted line (which has a number of apostrophes instead of a line number) and press ENTER, the editor leaves insert mode and the blank line is deleted.

  Tip:
  To insert a blank line below `LOCAL` (as indicated below), enter an asterisk (*) in this line. When insert mode is no longer active (after you have entered the last line of the program and have pressed ENTER twice), you can remove the asterisk. The resulting blank line will not be deleted in this case.

To specify the `DEFINE DATA` block

- Enter the following code in the program editor:

  ```
  DEFINE DATA
  LOCAL

  END-DEFINE
  *
  END
  ```

  `LOCAL` means that the variables that you will define with the next step are local variables which apply only to this program.

To display the data fields from the DDM in a split screen

1. In the program editor's command line, enter the following:

```
SPLIT VIEW EMPLOYEES SHORT
```

   SHORT indicates that the data fields are to be listed in short form (that is, only the Adabas short names and corresponding Natural field names are displayed).

   The screen is divided into two sections. The data fields from the DDM displayed in the lower half of the screen. It is not possible to edit the data in the lower half of the screen.

```
>> ----------Columns 001 072  <<  Program PGM01    Lines  6      User SAG
Command ===>                                      Mode   Struct Lib  TUTORIAL
****** ****************************** top of data ****************************
000010 DEFINE DATA
000020 LOCAL
000030
000040 END-DEFINE
000050 *
000060 END
****** **************************** bottom of data ***************************

>> ----------Columns 001 072  <<     View EMPLOYEE Lines  38     User SAG
Command ===>                                                    Lib  SYSTEM
****** ****************************** top of data ****************************
000001 DB: 020 FILE: 014  - EMPLOYEES                      DEFAULT SEQUENCE:
000002   1 AA PERSONNEL-ID                      A    8    D
000003 G 1 AB FULL-NAME
000004   2 AC FIRST-NAME                        A   20  N
000005   2 AD MIDDLE-I                          A    1  N
000006   2 AE NAME                              A   20    D
000007   1 AD MIDDLE-NAME                       A   20  N
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help  Save  Exit  Run   Rfind Stow  -     +     Check Home  Undo  Canc
```

2. You can now page through the view to see which data fields are used and how they have been defined. To do so, use the following commands or keys:

Command or Key	Description
SWAP	Move the cursor from the command line of the edit area to the command line of the display area (view) and vice versa. It is also possible to simply move the cursor to the required command line.
+ or PF8	Page forward in the view. The cursor must be located in the command line of the display area (view).
- or PF7	Page backward in the view. The cursor must be located in the command line of the display area (view).
SPLIT END	Terminate split-screen mode. The cursor must be located in the command line of the edit area (program).

The next step assumes that split-screen mode has been terminated.

3. Place the cursor in the first position of the line containing `LOCAL` and enter the following:

   ```
   I9
   ```

   9 blank lines are inserted.

4. Enter the following code below `LOCAL`:

   ```
   1 EMPLOYEES-VIEW VIEW OF EMPLOYEES
     2 FULL-NAME
       3 NAME (A20)
     2 DEPT (A6)
     2 LEAVE-DATA
       3 LEAVE-DUE (N2)
   ```

5. Press ENTER.

   The remaining blank lines are eliminated. However, one blank line remains with the cursor in it (the editor stays in insert mode). When you now press ENTER, you will leave insert mode.

The first line contains the name of your view and the name of the database file from which the fields have been taken. This is always defined on level 1. The level is indicated at the beginning of the line. The names of the database fields from the DDM are defined at levels 2 and 3.

Levels are used in conjunction with field grouping. Fields assigned a level number of 2 or greater are considered to be a part of the immediately preceding group which has been assigned a lower level number. The definition of a group enables reference to a series of fields (this may also be only one field) by using the group name. This is a convenient and efficient method of referencing a series of consecutive fields.

Format and length of each field is indicated in parentheses. "A" stands for alphanumeric, and "N" stands for numeric.

Reading Data from a Database

Now that you have defined the required data, you will add a `READ` loop. This reads the data from the database file using the defined view. With each loop, one employee is read from the database file. Name, department and remaining days of vacation for this employee are displayed. Data are read until all employees have been displayed.

Note:
It may happen that an error message is displayed indicating that the transaction has been aborted. This usually happens when the non-activity time limit which is determined by Adabas has been exceeded. When such an error occurs, you should simply repeat your last action (for example, issue the `RUN` command once more).

To read data from a database

1. Insert the following below `END-DEFINE` (use the `I` command as described above to insert blank lines):

   ```
   READ EMPLOYEES-VIEW BY NAME
   *
     DISPLAY NAME 3X DEPT 3X LEAVE-DUE
   *
   END-READ
   ```

   `BY NAME` indicates that the data which is read from the database is to be sorted alphabetically by name.

   The `DISPLAY` statement arranges the output in column format. A column is created for each specified field and a header is placed over the column. `3X` means that 3 spaces are to be inserted between the columns.

2. Run the program.

   The following output appears.

   ```
   MORE
   Page      1                                                 09-06-30  16:06:49

           NAME            DEPARTMENT   LEAVE
                              CODE       DUE
   --------------------   ----------   -----

   ABELLAN                PROD04        20
   ACHIESON               COMP02        25
   ADAM                   VENT59        19
   ADKINSON               TECH10        38
   ADKINSON               TECH10        18
   ADKINSON               TECH05        17
   ADKINSON               MGMT10        28
   ADKINSON               TECH10        26
   ADKINSON               SALE30        36
   ADKINSON               SALE20        37
   ADKINSON               SALE20        30
   AECKERLE               SALE47        31
   AFANASSIEV             MGMT30        26
   AFANASSIEV             TECH10        35
   AHL                    MARK09        30
   AKROYD                 COMP03        20
   ALEMAN                 FINA03        20
   ```

   As a result of the `DISPLAY` statement, the column headers (which are taken from the DDM) are underlined and one blank line is inserted between the underlining and the data. Each column has the same width as defined in the `DEFINE DATA` block (that is: as defined in the view).

   The title at the top of each page, which contains the page number, date and time, is also caused by the `DISPLAY` statement.

3. Press ENTER repeatedly to display all pages.

   You will return to the program editor when all employees have been displayed.

Tip:
If you want to return to the program editor before all employees have been displayed, enter `EDIT` or its abbreviation `E` at the `MORE` prompt. It is also possible to enter the terminal command `%.`, which interrupts the current Natural operation, at the `MORE` prompt. By default, each terminal command starts with the control character `%`. Your administrator, however, may have defined another control character.

Reading Selected Data from a Database

Since the previous output was very long, you will now restrict it. Only the data for a range of names is to be displayed, starting with "Adkinson" and ending with "Bennett". These names are defined in the demo database.

To restrict the output to a range of data

1. Before you can use new variables, you have to define them. Therefore, insert the following below `LOCAL`:

   ```
   1 #NAME-START        (A20) INIT <"ADKINSON">
   1 #NAME-END          (A20) INIT <"BENNETT">
   ```

   These are user-defined variables; they are not defined in demo database. The hash (#) at the beginning of the name is used to distinguish the user-defined variables from the fields defined in the demo database; however, it is not a required character.

   `INIT` defines the default value for the field. The default value must be specified in pointed brackets and quotation marks.

2. Insert the following below the `READ` statement:

   ```
   STARTING FROM #NAME-START
   ENDING AT #NAME-END
   ```

   Your program should now look as follows:

   ```
   DEFINE DATA
   LOCAL
     1 #NAME-START        (A20) INIT <"ADKINSON">
     1 #NAME-END          (A20) INIT <"BENNETT">
     1 EMPLOYEES-VIEW VIEW OF EMPLOYEES
       2 FULL-NAME
         3 NAME (A20)
       2 DEPT (A6)
       2 LEAVE-DATA
         3 LEAVE-DUE (N2)
   END-DEFINE
   *
   READ EMPLOYEES-VIEW BY NAME
     STARTING FROM #NAME-START
     ENDING AT #NAME-END
   *
     DISPLAY NAME 3X DEPT 3X LEAVE-DUE
   *
   END-READ
   *
   END
   ```

Your program code now exceeds one screen page. To navigate in the program source, you can use the following commands or keys:

Command	**Description**
`BOT`	Go to the end of the program.
`TOP`	Return to the beginning of the program.
Key	**Description**
PF8	Scroll down one page in the program.
PF7	Scroll up one page in the program.

3. Run the program.

   The output is shown. When you press ENTER repeatedly, you will notice that you will return to the program editor after a couple of pages (that is: when the data for the last employee named Bennett has been displayed).

4. Stow the program.

You can now proceed with the next exercises: *User Input*.