

READ Statement

The READ statement is used to read records from a database. The records can be retrieved from the database:

- in the order in which they are physically stored in the database (READ IN PHYSICAL SEQUENCE), or
- in the order of Adabas Internal Sequence Numbers (READ BY ISN), or
- in the order of the values of a descriptor field (READ IN LOGICAL SEQUENCE).

In this documentation, only READ IN LOGICAL SEQUENCE is discussed, as it is the most frequently used form of the READ statement; for information on the other two options, please refer to the description of the READ statement in the Natural Statements documentation.

The following topics are covered below:

- Syntax
- Limiting the Number of Records to be Read
- STARTING/ENDING Clauses
- WHERE Clause

Syntax

The basic syntax of the READ statement is:

READ *view* **IN LOGICAL SEQUENCE BY** *descriptor*

or shorter:

READ *view* **LOGICAL BY** *descriptor*

view is the name of a view defined in the DEFINE DATA statement (as explained earlier in this section).

descriptor is the name of a database field defined in that view. The values of this field determine the order in which the records are read from the database.

If you specify a descriptor, you need not specify the keyword "LOGICAL":

READ *view* **BY** *descriptor*

If you do not specify a descriptor, the records will be read in the order of values of the field defined as default descriptor (under "Default Sequence") in the DDM. However, if you specify no descriptor, you must specify the keyword "LOGICAL":

READ *view* **LOGICAL**

Example:

```
** Example Program 'READX01'  
DEFINE DATA LOCAL  
1 MYVIEW VIEW OF EMPLOYEES  
  2 PERSONNEL-ID  
  2 NAME  
  2 JOB-TITLE  
END-DEFINE  
READ (6) MYVIEW BY NAME  
  DISPLAY NAME PERSONNEL-ID JOB-TITLE  
END-READ  
END
```

With the READ statement in the above example, records from the EMPLOYEES file are read in alphabetical order of their last names.

The above program will produce the following output, displaying the information of each employee in alphabetical order of the employees' last names:

Page	1		99-08-19 13:16:04
	NAME	PERSONNEL ID	CURRENT POSITION
	-----	-----	-----
	ABELLAN	60008339	MAQUINISTA
	ACHIESON	30000231	DATA BASE ADMINISTRATOR
	ADAM	50005800	CHEF DE SERVICE
	ADKINSON	20008800	PROGRAMMER
	ADKINSON	20009800	DBA
	ADKINSON	2001100	

If you wanted to read the records to create a report with the employees listed in sequential order by date of birth, the appropriate READ statement would be:

```
READ MYVIEW BY BIRTH
```

You can only specify a field which is defined as a "descriptor" in the underlying DDM (it can also be a subdescriptor, superdescriptor or hyperdescriptor).

Limiting the Number of Records to be Read

As shown in the previous example program, you can limit the number of records to be read by specifying a number in parentheses after the keyword READ:

```
READ (6) MYVIEW BY NAME
```

In that example, the READ statement would read no more than 6 records.

Without the limit notation, the above READ statement would read *all* records from the EMPLOYEES file in the order of last names from A to Z.

STARTING/ENDING Clauses

The READ statement also allows you to qualify the selection of records based on the **value** of a descriptor field. With an EQUAL TO/STARTING from option in the BY or WITH clause, you can specify the value at which reading should begin. By adding a THRU/ENDING AT option, you can also specify the value in the logical sequence at which reading should end.

For example, if you wanted a list of those employees in the order of job titles starting with "TRAINEE" and continuing on to "Z", you would use one of the following statements:

```
READ MYVIEW WITH JOB-TITLE = 'TRAINEE'
  READ MYVIEW WITH JOB-TITLE STARTING from 'TRAINEE'
  READ MYVIEW BY JOB-TITLE = 'TRAINEE'
  READ MYVIEW BY JOB-TITLE STARTING from 'TRAINEE'
```

Note that the value to the right of the equal sign (=) or STARTING from option must be enclosed in apostrophes. If the value is numeric, this *text notation* is not required.

If a BY option is used, a WITH option cannot be used and vice versa.

The sequence of records to be read can be even more closely specified by adding an end limit with a THRU or ENDING AT clause.

To read just the records with the job title "TRAINEE", you would specify:

```
READ MYVIEW BY JOB-TITLE STARTING from 'TRAINEE' THRU 'TRAINEE'
  READ MYVIEW WITH JOB-TITLE EQUAL TO 'TRAINEE'
  ENDING AT 'TRAINEE'
```

To read just the records with job titles that begin with "A" or "B", you would specify:

```
READ MYVIEW BY JOB-TITLE = 'A' THRU 'C'
  READ MYVIEW WITH JOB-TITLE STARTING from 'A' ENDING AT 'C'
```

The values are read up to and including the value specified after THRU/ENDING AT. In the two examples above, all records with job titles that begin with "A" or "B" are read; if there were a job title "C", this would also be read, but not the next higher value "CA".

WHERE Clause

The WHERE clause may be used to further qualify which records are to be read.

For instance, if you wanted only those employees with job titles starting from "TRAINEE" who are paid in US currency, you would specify:

```
READ MYVIEW WITH JOB-TITLE = 'TRAINEE'
  WHERE CURR-CODE = 'USD'
```

The WHERE clause can also be used with the BY clause as follows:

```
READ MYVIEW BY NAME
  WHERE SALARY = 20000
```

The WHERE clause differs from the WITH/BY clause in two respects:

- The field specified in the WHERE clause need not be a descriptor.
- The expression following the WHERE option is a logical condition. The following logical operators are possible in a WHERE clause:

EQUAL	EQ	=
NOT EQUAL TO	NE	≠
LESS THAN	LT	<
LESS THAN OR EQUAL TO	LE	<=
GREATER THAN	GT	>
GREATER THAN OR EQUAL TO	GE	>=

The following program illustrates the use of the STARTING from, ENDING AT and WHERE clauses:

```

** Example Program 'READX02'
DEFINE DATA LOCAL
1 MYEMP VIEW OF EMPLOYEES
  2 NAME
  2 JOB-TITLE
  2 INCOME (1:2)
  3 CURR-CODE
  3 SALARY
  3 BONUS (1:1)
END-DEFINE
*
READ (3) MYVIEW WITH JOB-TITLE = 'TRAINEE' THRU 'TRAINEE'
      WHERE CURR-CODE (*) = 'USD'
      DISPLAY NOTITLE NAME / JOB-TITLE 5X INCOME (1:2)
      SKIP 1
END-READ
END
    
```

It produces the following output:

NAME CURRENT POSITION	INCOME		
	CURRENCY CODE	ANNUAL SALARY	BONUS
-----	-----	-----	-----
SENKO	USD	23000	0
TRAINEE	USD	21800	0
BANGART	USD	25000	0
TRAINEE	USD	23000	0
LINCOLN	USD	24000	0
TRAINEE	USD	22000	0

Further Example of READ Statement:

See program READX03 in library SYSEXPG.