Natural Statements and Transaction Logic with VSAM

This section describes special considerations on Natural statements and Natural transaction logic when used with VSAM.

The Natural statements used to access VSAM files are a subset of those provided with the Natural language. No new statements are needed to access a VSAM file, since each Natural statement performs the same function regardless of the database management system or access method used. Therefore, programs written for VSAM files can also be used to access Adabas databases.

The Natural interface to VSAM has no built-in transaction logic and uses the one of the environment it is running in. This leads to different results depending on the environment.

This section covers the following topics:

Natural Statements with VSAM

This section mainly consists of information also contained in the Natural Statements documentation, where each Natural statement is described in detail, including notes on VSAM usage where applicable. Summarized below are the particular points a programmer has to bear in mind when using Natural statements with VSAM.

Note:
Since the Natural compiler does not check if a program adheres to the restrictions imposed by the Natural interface to VSAM, VSAM-specific programming errors concerning the use of Natural statements only occur when the program is executed.

Any Natural statement not mentioned in this section can be used with VSAM without restrictions.

- BACKOUT TRANSACTION
- DELETE
- END TRANSACTION
- FIND
- GET
- GET SAME
- GET TRANSACTION DATA

- HISTOGRAM
- READ
- STORE
- UPDATE

BACKOUT TRANSACTION

The `BACKOUT TRANSACTION` statement is used to back out all database updates performed during the current user logical transaction. This statement also releases all records held during the transaction.

If used with Natural for VSAM, the `BACKOUT TRANSACTION` statement releases records held in the UPD table. It does not back out transactions unless Natural is running under a TP monitor or DFSMStvs which supports dynamic transaction backout (for example, CICS). In this case, a `ROLLBACK` to the last `SYNCPOINT` is issued.

DELETE

The `DELETE` statement is used to delete a record from a VSAM file.

The use of the `DELETE` statement places each record selected in the corresponding `FIND` or `READ` statement in hold status.

The `DELETE` statement is not valid for VSAM entry-sequenced data sets (ESDS).

END TRANSACTION

The `END TRANSACTION` statement is used to indicate the end of a logical transaction. A logical transaction is the smallest logical unit of work (as defined by the user) which must be performed in its entirety to ensure that the information contained in the VSAM file is logically consistent.

The `END TRANSACTION` statement also releases all records placed in hold status during the transaction.

An `END TRANSACTION` only releases records held in the UPD table unless Natural is running under a TP monitor or DFSMStvs which supports dynamic transaction backout (for example, CICS). In this case, an `END TRANSACTION` statement causes a `SYNCPOINT` to be issued.

FIND

The `FIND` statement is used to select a set of records from the VSAM file based on a search criterion consisting of fields defined as descriptors (keys).

The `WITH` clause is used to specify the search criterion consisting of key fields (descriptors) defined in the VSAM file.

Only VSAM key fields can be used.

The number of records to be selected as a result of a WITH clause can be limited by specifying the keyword LIMIT together with a limit value (*`operand 1`*) expressed as a numeric constant or a user-defined variable. The limit value is enclosed within parentheses. If the number of records selected exceeds the limit value, the program is terminated with an error message.

The descriptor must be defined in a VSAM file as a VSAM key field. In a DDM, it is marked with `P` for primary key, `S` for primary sub/superdescriptor, `X` for alternate sub/superdescriptor or `A` for alternate key (see *Edit DDM* in the section *Operation*, and the *SYSDDM Utility* as described in the Natural *Editors* documentation).

The formats of the descriptor and the search value must be compatible.

The following Natural system variables are available with the `FIND` statement:

Variable	**Content**
`*ISN`	The system variable `*ISN` contains the relative byte address of the record currently being processed (ESDS files only). This variable is not available for the `FIND NUMBER` and `FIND FIRST` statements.
`*NUMBER`	The system variable `*NUMBER` contains the number of records which satisfied the basic search criterion specified in the `WITH` clause, and before evaluation of any `WHERE` criterion. `*NUMBER` only contains a meaningful value if the `EQUAL TO` operator is used in the search criterion. With any other operator, `*NUMBER` will be `0` if no records have been found; any other value indicates that records have been found, but the value will have no relation to the number of records actually found. The same applies to `*NUMBER` with the `FIND NUMBER` statement.
`*COUNTER`	The system variable `*COUNTER` contains the number of times the processing loop has been entered. This system variable is not available for the `FIND NUMBER` statement.

The `FIND` statement is only valid for key-sequenced (KSDS) and entry-sequenced (ESDS) VSAM datasets. For ESDS, an alternate index or a path for an alternate index must be defined. Relative record datasets (RRDS) are not allowed, since they do not contain any key fields (descriptors).

GET

The `GET` statement is used to read a record with a given VSAM record number. For an ESDS file, the record number (ISN) would be the relative byte address (RBA); for RRDS and VRDS files, it would be the relative record number (RRN). The `GET` statement does not initiate a processing loop. As a result, a subsequent `UPDATE` or `DELETE` statement will not be processed and Natural returns a corresponding error message.

For ESDS, the RBA must be contained in a user-defined variable (numeric format) or specified as an integer constant. The same rules apply for RRDS and VRDS with the exception that the RRN must be provided instead of the RBA.

GET SAME

The `GET SAME` statement applies to VSAM ESDS, RRDS, and VRDS only (see also the `GET` statement above).

GET TRANSACTION DATA

The `GET TRANSACTION DATA` statement is not applicable to the Natural interface to VSAM.

HISTOGRAM

The `HISTOGRAM` statement is used to read the values of a field which is defined as a descriptor, subdescriptor, or superdescriptor.

The `HISTOGRAM` statement initiates a processing loop, but does not provide access to any fields other than the field specified in the statement.

Only VSAM key fields can be used as descriptors.

The following Natural system variable is available with the `HISTOGRAM` statement:

Variable	Content
`*NUMBER`	When used in conjunction with a KSDS primary key or a unique alternate index, `*NUMBER` is always 1.

Note:
The `*ISN` system variable is not available for the Natural interface to VSAM.

When used with VSAM, the `HISTOGRAM` statement is only valid for KSDS and ESDS datasets. For ESDS, an alternate index or a path for an alternate index must be defined.

The values are read directly from the VSAM index and are returned in ascending or descending value sequence.

READ

The `READ` statement is used to read records from a VSAM file. The records can be retrieved in the value sequence (ascending or descending) of a descriptor (key) field. The `READ` sequence initiates a processing loop.

`IN LOGICAL SEQUENCE` is used to read records in the order of the values of a descriptor (key). If `LOGICAL` is specified with a descriptor, the records are read in the value sequence of the descriptor. A descriptor can be used for sequence control. A descriptor within a periodic group cannot be used. If `LOGICAL` is specified without a descriptor, the records are read in the default descriptor sequence, as defined in the DDM.

`WITH REPOSITION` can be used for skip-sequential processing inside the active loop, the new position must be defined as the new start value for the loop and must reset the system variable `*COUNTER`.

`IN LOGICAL SEQUENCE` is only valid for KSDS with primary and alternate keys defined and ESDS with alternate keys defined. A subdescriptor or superdescriptor can be used for sequence control, too.

The following Natural system variables are available with the `READ` statement:

Variable	Content
*ISN	The system variable *ISN contains either the RRN (for RRDS or VRDS) or the RBA (for ESDS) of the current record.
*COUNTER	This system variable contains the number of times the processing loop has been entered.

Records can also be retrieved `IN PHYSICAL SEQUENCE`, which is used to read records in the order in which they are physically stored in a database. It is only valid for VSAM ESDS, RRDS and VRDS. This is the default sequence.

`STARTING WITH ISN` can be used as start value for the loop in ascending or descending physical sequence.

`BY ISN` is used to read records in RBA and RRN order for ESDS, RRDS and VRDS files, respectively.

STORE

The `STORE` statement is used to add a record to a database.

A unique value for the primary-key field or the alternate-index field must be provided if the dataset is defined with a primary key or a unique alternate index.

The `USING/GIVING NUMBER` clause is only valid for RRDS or VRDS, in which case the ISN corresponds to the relative record number.

`USING/GIVING NUMBER` is used to store a record with a user-supplied RRN. If a record with the specified RRN already exists, an error message is returned and the execution of the program is terminated, unless `ON ERROR` processing was specified.

The Natural system variable `*ISN` contains the RRN assigned to the new record as a result of the `STORE`

statement execution. A subsequent reference to `*ISN` must include the statement number of the related `STORE` statement. `*ISN` is available for RRDS or VRDS files only.

UPDATE

The `UPDATE` statement is used to update one or more fields of a record in a database. The record to be updated must have been previously selected using a `FIND` or `READ` statement.

The primary key cannot be updated.

Natural Transaction Logic with VSAM

Natural for VSAM uses the transaction logic of the environment it is running in. Thus, the results of the Natural `END TRANSACTION` and `BACKOUT TRANSACTION` statements (see also the relevant sections in Natural Statements with VSAM) differ depending on the actual environment:

- With Native VSAM

- Under CICS
- Under DFSMStvs

With Native VSAM

Since VSAM itself has no transaction logic, there is no transaction logic available if Natural is working in a native VSAM environment. This is the case under Com-plete, TSO, and in batch mode, which means when `NVSMISC` is the I/O module in use.

With `NVSMISC`, `END TRANSACTION` and `BACKOUT TRANSACTION` statements do not return any error messages, but are ignored by the Natural interface to VSAM.

Under CICS

Under CICS, VSAM files can be defined as "recoverable resources" or for RLS as "recoverable sphere", all of which are synchronized by CICS using the concept of "logical units of work" (LUWs). An LUW ends if a `SYNCPOINT` command is issued or if the CICS task is terminated. For details, refer to the relevant IBM literature on CICS.

Below is information on:

- NVSCICS Module
- Conversational Tasks
- Pseudo-Conversational Tasks

NVSCICS Module

For CICS, the I/O module `NVSCICS` is a normal command-level application program. It transfers `END TRANSACTION` and `BACKOUT TRANSACTION` statements to the `NATCICS` driver which issues the `EXEC CICS SYNCPOINT` and `EXEC CICS ROLLBACK` commands. If an error occurs in a Natural session with uncommitted updates and no error transaction is supplied, Natural itself triggers the interface to VSAM to issue a `ROLLBACK` command.

If a `SYNCPOINT` or `ROLLBACK` command fails (for example, when CICS answers with a `ROLLEDBACK` condition to a `SYNCPOINT` request), error messages NAT3544 or NAT3545 are returned.

Conversational Tasks

If the Natural session runs in CICS conversational mode, the LUW is not ended by a terminal I/O. Natural runs in conversational mode if either the Natural parameter `PSEUDO=OFF` has been specified or Natural itself has determined that pseudo-conversational processing is not possible.

Since terminal I/Os do not disturb the transaction logic of an application as long as Natural is running in conversational mode, a program like the following one would work without problems:

Example:

```
READ vsam-file
 UPDATE
 INPUT ...
END-READ
BACKOUT TRANSACTION
```

Pseudo-Conversational Tasks

If the Natural session is running in pseudo-conversational mode, each terminal I/O terminates the CICS task, thus implicitly performing a `SYNCPOINT`. Therefore, the impact of a `BACKOUT TRANSACTION` statement, that is of an `EXEC CICS SYNCPOINT ROLLBACK` command, only goes back to the most recent terminal I/O. The example program above would, therefore, end with error message NAT3548, because it is not possible to roll back all the updates.

Note:
Keep in mind that all messages of the Natural interface to VSAM are issued at runtime only, since the Natural compiler is not able to detect this kind of logical error.

Under DFSMStvs

DFSMS Transactional VSAM Services (DFSMStvs) provides the same features CICS provides: forward and backward recovery logging, backout processing and a two-phase commit process. An LUW ends if the RRS (Resource Recovery Service) call `SRRCMIT` or `SRRBACK` is issued (`END TRANSACTION` or `BACKOUT TRANSACTION`). For details, refer to the relevant IBM literature on DFSMStvs and RRS.