Natural File Server

In all supported TP-monitor environments, the Natural SQL Gateway provides an intermediate work file, referred to as the File Server, to prevent database selection results from being lost with each terminal I/O.

This section covers the following topics:

- Concept of the File Server
- Installing the File Server
- Logical Structure of the File Server

Concept of the File Server

To avoid reissuing the selection statement used and repositioning the cursors, Natural writes the results of a database selection to an intermediate file. The saved selected rows, which may be required later, are then managed by Natural as if the facilities for conversational processing were available. This is achieved by automatically scrolling the intermediate file for subsequent screens, maintaining position in the work file rather than in the SQL table.

All rows of all open cursors are rolled out to the file server before the first terminal I/O operation. Subsequently, all data is retrieved from this file if Natural refers to one of the cursors which were previously rolled out (see the description of roll out in *Logical Structure of File Server* below).

If a row is to be updated or deleted, the row is first checked to see if it has been updated in the meantime by some other process. This is done by reselecting and fetching the row from the SQL database, and then comparing it with the original version as retrieved from the file server. If the row is still unchanged, the update or delete operation can be executed. If not, a corresponding error message is returned. The reselection required when updating or deleting a row is possible in both dynamic mode and static mode.

Only the fields which are stored in the file server are checked for consistency against the record retrieved from the SQL table.

As the row must be uniquely identified, the Natural view must contain a field for which a unique row has been created. This field must be defined as a unique key in the SQL table. In a Natural DDM, it will then be indicated as a unique key via the corresponding Natural-specific short name.

Installing the File Server

The size of a row which can be written to the file server is limited to 32 KB or 32767 bytes. If a row is larger, a corresponding error message is returned.

The File Server can use either a VSAM RRDS file or the Software AG Editor buffer pool as the storage medium to save selected rows of SQL tables.

This section covers the following topics:

- Installing the File Server - VSAM
- Installing the File Server - Editor Buffer Pool

Installing the File Server - VSAM

The file server is installed via a batch job, which defines and formats the intermediate file. Samples of this batch job are supplied on the installation tape as described in the relevant section.

Defining the Size of the File Server

The file server is created by defining an RRDS VSAM file using AMS (Access Method Services). Its physical size and its name must be specified.

Formatting the File Server

The file server is formatted by a batch job, which requires five input parameters specified by the user, and which formats the file server according to these parameters. The parameters specify:

1. The number of blocks to be formatted (logical size of the VSAM file); this value is taken from the first parameter of the `RECORD` subcommand of the AMS `DEFINE CLUSTER` command.
2. The number of users that can log on to Natural concurrently.
3. The number of formatted blocks to be defined as primary allocation per user.
4. The number of formatted blocks to be used as secondary allocation per user.
5. The maximum number of file server blocks to be allocated by each user. If this number is exceeded, a corresponding Natural error message is returned.

Immediately before the first access to the file server, a file server directory entry is allocated to the Natural session and the amount of blocks specified as primary allocation is allocated to the Natural session.

The primary allocation is used as intermediate storage for the result of a database selection and should be large enough to accommodate all rows of an ordinary database selection. Should more space in the file server be required for a large database selection, the file server modules allocate a secondary allocation equal to the amount that was specified for secondary allocation when the file server was formatted.

Thus, a secondary area is allocated only when your current primary allocation is not large enough to contain all of the data which must be written to the intermediate file. The number of secondary allocations allowed depends upon the maximum number of blocks you are allowed to allocate. This parameter is also specified when formatting the file server.

The number of blocks defined as the secondary allocation is allocated repeatedly, until either all selected data has been written to the file or the maximum number of blocks you are allowed to allocate is exceeded. If so, a corresponding Natural error message is returned. When the blocks received as a secondary allocation are no longer needed (that is, once the Natural loop associated with this allocation is closed), they are returned to the free blocks pool of the file server.

Your primary allocation of blocks, however, is always allocated to you, until the end of your Natural session.

Changes Required for a Multi-Volume File Server

To minimize channel contention or bottlenecks that can be caused by placing a large and heavily used file server on a single DASD volume, you can create a file server that spans several DASD volumes.

To create and format such a file server, two changes are needed in the job that is used to define the VSAM cluster:

1. Change `VOLUME ()` to `VOLUMES (vol1,vol2,...)`.
2. Divide the total number of records required for the file (as specified with the first format job parameter) by the number of volumes specified above. The result of the calculation is used for the `RECORDS` parameter of the `DEFINE CLUSTER` command.

This means that in the file server format job, the value of the first parameter is the result of multiplying two parameters taken from the `DEFINE CLUSTER` command: `RECORDS` and `VOLUMES`.

Installing the File Server - Editor Buffer Pool

The Software AG Editor buffer pool is used as the storage medium when `EBPFSRV=ON` is set in the `NDBPARM` module. In this case, the primary, secondary and maximum allocation amounts for the file server are specified by `EBPPRAL`, `EBPSEC`, `EBPMAX` parameters of the `NDBPRM` macro. Before Natural SQL Gateway tries to write data from a Natural user session to the file server for the first time, a Software AG Editor buffer pool logical file is allocated with the Natural terminal identifier as user name and the number 2240 as session number.

The operation of the file server is in this case depending on the definition of the Software AG Editor buffer pool as described in the Natural *Operations* documentation.

The number of logical files for the buffer pool limits the number of users concurrently accessing the file server. The number of work file blocks limits the amount of data to be saved at a specific moment. (You also have to consider that there are other users than Natural SQL Gateway of the Software AG Editor.)

However, using the Software AG Editor buffer pool as the storage medium for the file server enables Natural SQL Gateway to run in a Parallel Sysplex environment. In this case, your Natural session must use the auxiliary editor buffer pool. See also *Support of a z/OS Parallel Sysplex Environment* in the *Installation* documentation.

Logical Structure of the File Server

Immediately before a Natural user session accesses the file server, a file server directory entry (VSAM) or a logical file (Software AG Editor buffer pool) is allocated to the Natural user session and the number of blocks specified as primary allocation is reserved until the end of the session.

Generally, the file server is only used when a terminal I/O occurs within an active `READ`, `FIND`, or `SELECT` loop, where database selection results would be lost. Before each terminal I/O operation, Natural checks for any open cursors. For each non-scrollable cursor found, all remaining rows are retrieved from the SQL table and written to an intermediate file. In the Natural SQL Gateway documentation, this process is referred to as cursor roll out.

For each cursor roll out, a logical file is opened to hold all the rows fetched from this cursor. The space for the intermediate file is managed within the space allocated to your session. The logical file is then positioned on the row that was `CURRENT OF CURSOR` when the terminal I/O occurred.

Subsequent requests for data are then satisfied by reading the rows directly from the intermediate file. The database is no longer involved, and SQL is only used for update, delete or store operations.

Once the corresponding processing loop in the application has been closed, the file is no longer needed and the blocks it occupies are returned to your pool of free blocks. From here, the blocks are returned to the free blocks pool of the file server, so that you are left with your primary allocation only.

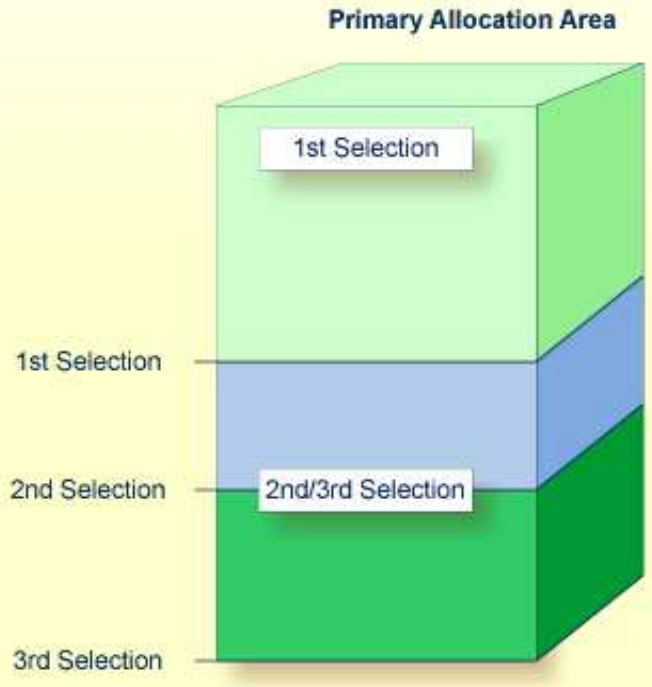
In the following example, the space allocated to the first selection is not released until all rows selected during the third selection have been retrieved. The same applies to the space allocated to the third selection.

The space allocated to the second selection, however, is released immediately after the last row of the corresponding selection result has been retrieved.

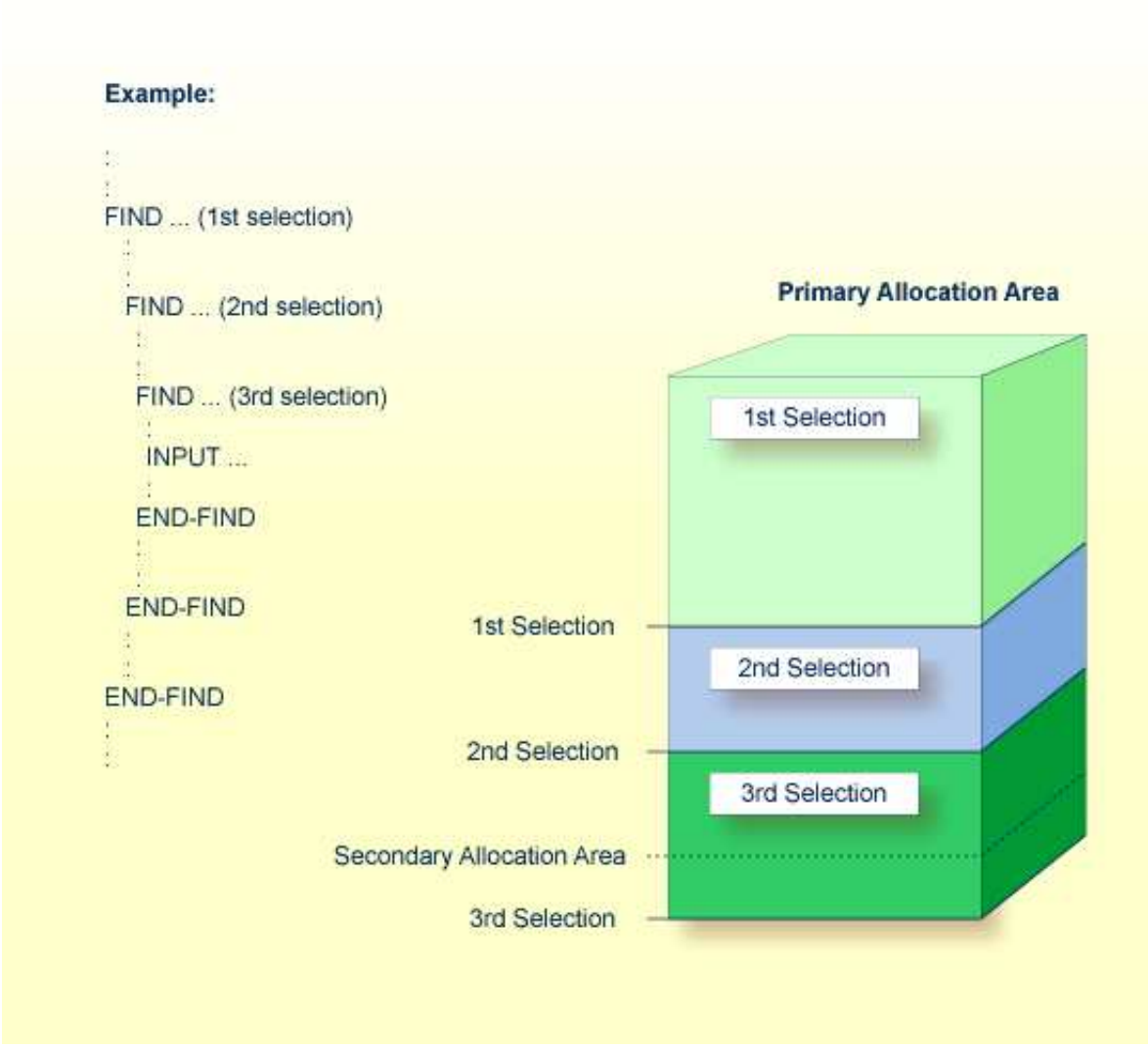
Therefore, the space allocated to the second selection can be used for the selection results of the third selection.

Example:

```
:
:
FIND ... (1st selection)
  :
  :
  FIND ... (2nd selection)
    :
    INPUT ...
    :
  END-FIND
  :
  :
  FIND ... (3rd selection)
    :
    INPUT ...
    :
  END-FIND
  :
  :
END-FIND
:
:
```



If the primary allocation area is not large enough, for example, if the third selection is nested within the second selection, the secondary allocation area is used.



When a session is terminated, all of a user's blocks are returned to the free blocks pool. If a session ends abnormally, Natural checks, where possible, whether a file server directory entry for the corresponding user exists. If so, all resources held by this user are released.

If Natural is unable to free the resources of an abnormally-ended user session, these resources are not released until the same user ID logs on from the same logical terminal again.

If the same user ID and/or logical terminal are not used again for Natural, the existing directory entry and the allocated space remain until the file server is formatted again. A new run of the formatting job deletes all existing data and recreates the directory.