Natural Buffer Pool

The shared-memory Natural buffer pool serves as a temporary storage area into which Natural cataloged object(s) (for example, programs) are loaded from a Natural system file for subsequent execution by the Natural runtime system and/or object compilation by the Natural compiler.

Since Natural generates reentrant Natural object code, it is possible for a single copy of a Natural object to be executed by more than one user at the same time. For this purpose, each object is loaded only once from a Natural system file into the Natural buffer pool, instead of being loaded by every caller of the object.

This section describes the basic principles of buffer pool operation and depicts the object loading procedure.

- Object Loading
- Object Removal
- Example of Object Loading and Execution
- Related Topics

Object Loading

Object loading into the buffer pool is initialized when an executable Natural object is requested for execution.

When object execution is requested, the respective cataloged object(s) are read from the appropriate Natural system file and loaded into the buffer pool where they can be executed by the Natural runtime system.

Apart from executable Natural objects, non-executable objects of the type data area (local, global, parameter) are also loaded into the buffer pool if a Natural object that references such a data area is cataloged or recataloged (compiled).

Related Topic:

- *Object Execution* - *Natural Runtime System, Natural Nucleus*

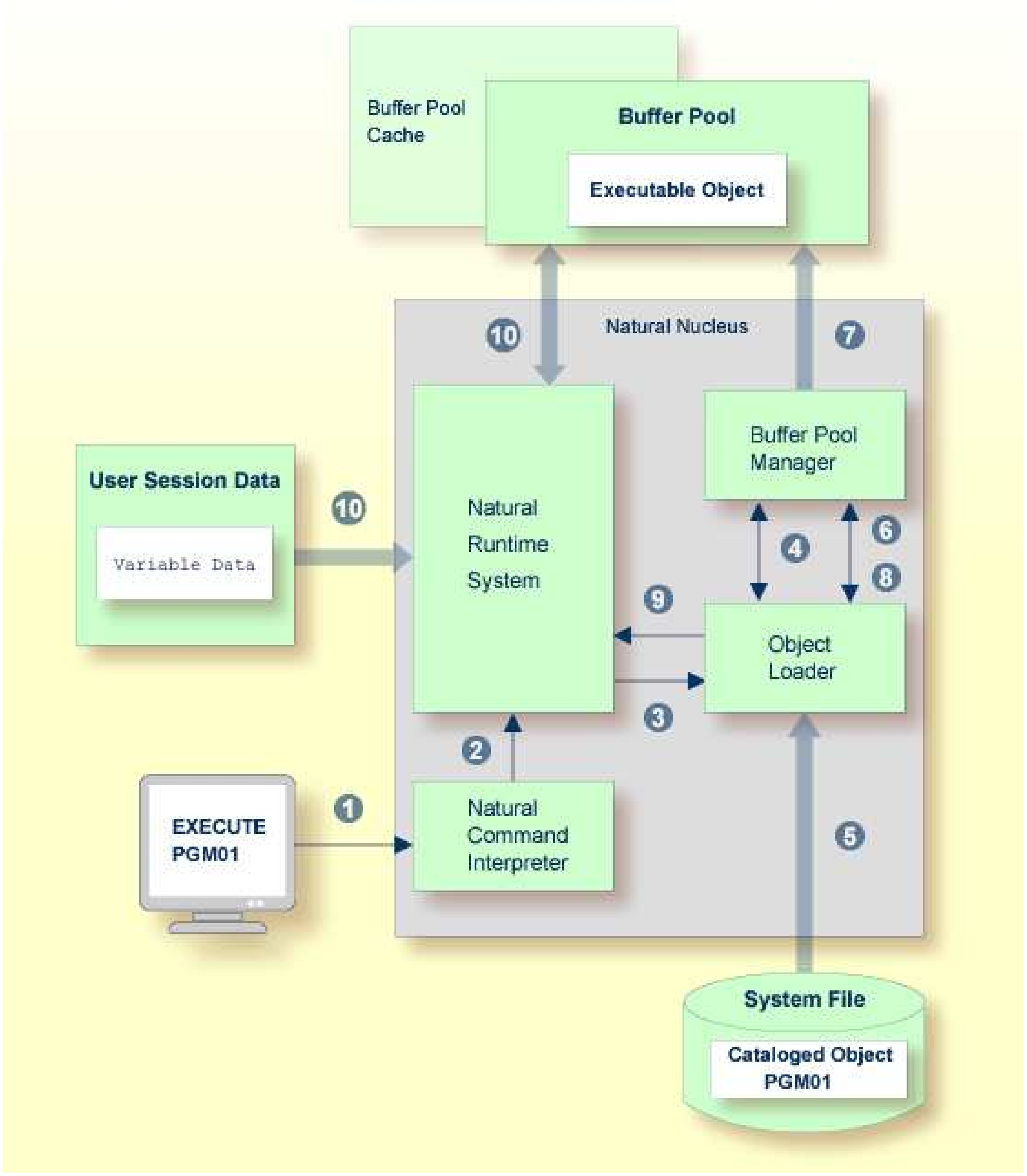
Object Removal

Objects are removed from the buffer pool when they are no longer referenced by a user and space is required for loading new objects. This guarantees maximum use of available storage space in the buffer pool.

If a buffer pool cache exists, unused objects are swapped from the buffer pool into the buffer pool cache. If an unused object is also removed from the buffer pool cache or if no buffer pool cache exists, the respective object is reloaded from the appropriate Natural system file when a user references it again.

Example of Object Loading and Execution

The diagram below illustrates the process flow when loading and executing a Natural program:



Legend

① A user issues the Natural system command `EXECUTE`, requesting execution of a Natural object of the type program with the name PGM01.

② The Natural command interpreter interprets the command and passes the request to the Natural runtime system.

③ The Natural runtime system passes the request to the object loader.

④ The object loader instructs the buffer pool manager to search the buffer pool (and the buffer pool cache, if available) for PGM01 and determine the address at which the object is stored in the buffer pool.

If the buffer pool manager finds the buffer pool address of PGM01, the buffer pool passes the address to the object loader and execution continues with ⑧ .
If the buffer pool manager does not find the buffer pool address of PGM01, the buffer pool manager returns the negative search result to the object loader and object execution continues with ⑤ .

⑤ The object loader retrieves the cataloged object of PGM01 from the appropriate Natural system file.

⑥ The object loader passes the cataloged object of PGM01 to the buffer pool manager.

⑦ The buffer pool manager loads PGM01 into the buffer pool.

⑧ The buffer pool manager returns the buffer pool address of PGM01 to the object loader.

⑨ The object loader passes the buffer pool address of PGM01 to the Natural runtime system.

⑩ The Natural runtime system interprets and executes the compiled code of the cataloged object of PGM01.

Related Topics

The following topics in the *System Architecture* documentation refer to the Natural buffer pool:

- *Natural System Files*
- *Natural Runtime System* - *Natural Nucleus*
- *Cataloged Objects* - *Natural Compiler*, *Natural Nucleus*

The following topics in other Natural documentation refer to the Natural buffer pool:

- *Natural Buffer Pool* - *Operations*
- *Natural Storage Management* - *Operations*