Lock Statement

Description

Sets locks to coordinate access to files and records on a network system.

Syntax

Lock *filevar* | Cursor *cursorvar* [, *key [, locktype*]] Then | Else *statements*

Parameters

The Lock statement has the following parameters.

<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>Filevar</td><td>filevar must have been set by a previous Open statement.</td></tr>
<tr><td>Cursorvar</td><td>Contains a cursor variable. Cursor variables are initialized with a Select...By statement.</td></tr>
<tr><td>Key</td><td>Specifies which record in a table to lock if a record is to be locked. To lock an entire table, use the keyword All (no quotes), or pass a null as the key expression. Only one key can be specified with each call to Lock.</td></tr>
<tr><td>Locktype</td><td>Specifies the type of locking that is to take place. The locktype can be passed as a mnemonic code (not a literal). Valid locktype mnemonic codes are:
<table>
<tr><th>Code</th><th>Meaning</th></tr>
<tr><td>X</td><td>Exclusive lock.</td></tr>
<tr><td>S</td><td>Shared lock.</td></tr>
<tr><td>XC</td><td>Exclusive coordinated lock (record lock only if supported by network).</td></tr>
<tr><td>SC</td><td>Shared coordinated lock (record lock only if supported by network).</td></tr>
</table>
A locktype can also be passed as a two-digit numeric code (not a literal). Valid numeric codes for locktype are:
<table>
<tr><th>First Digit Code</th><th>Meaning</th></tr>
<tr><td>0</td><td>Exclusive lock.</td></tr>
<tr><td>1</td><td>Shared lock.</td></tr>
<tr><td>2</td><td>Exclusive coordinated lock (record lock only).</td></tr>
<tr><td>3</td><td>Shared coordinated lock (record lock only.</td></tr>
</table>
<table>
<tr><th>Second Digit Code</th><th>Meaning</th></tr>
<tr><td>0</td><td>All (UnLock All only).</td></tr>
<tr><td>1</td><td>File lock.</td></tr>
<tr><td>2</td><td>Record lock.</td></tr>
</table>
The default value for locktype is an exclusive lock. If key is not null, then the exclusive lock is a record lock. Coordinated locking is controlled through the locking related environment attributes.</td></tr>
<tr><td>Then</td><td>The statement(s) following Then are executed if a file or record is locked successfully or if a lock is attempted against a file type that does not support locking.</td></tr>
<tr><td>Else</td><td>The statement(s) following Else are executed if the file or record cannot be locked. The Status() function indicates the severity of the error (refer to Status, below), and the system variable @FILE_ERROR contains detail about the nature of the error.</td></tr>
</table>

Status()	The definition of the return value of Status() on the Else branch of the Lock statement is as follows. **Value** / **Meaning** 0 — Unsuccessful lock on a network; not the station's own lock. 1 — Unsuccessful lock; current station holds the lock.

Usage

You can lock either before or after you read a row (to be modified), but remember that if you lock after you have read, then you cannot guarantee the integrity of the row you are about to modify.

If you just want to read rows, don't lock. Also, use ReadO, rather than Read.

If you need to read rows before finding the one to modify, don't lock until you find the row you want, but read again after you lock it.

See also

UnLock, Open, Select...By

Remarks

Lock alerts the network system that the program wishes to establish a lock on a table or row. If the program attempts to set a lock that has been previously set by another user, the current program will execute the Else statements.

Used properly, Lock allows only one user in a network to access and modify any specific table or row. When a table or row is locked by a user in a network, no other user in the network can lock that row until it has been unlocked using UnLock.

Lock has no effect unless used on a local area network and with a table type that supports locking. Therefore, Lock and UnLock can be built into applications whether or not they will be used on network systems. Applications not running on a network will simply ignore the Lock and UnLock commands.

Termination of the program does not unlock any locks that have been set. Use UnLock to release each Lock, or UnLock All to release all locks currently set by this workstation.

Example

```
/* The following function fragment locks a row, then reads it, does some unspecified processing, and then
unlocks the row, before concluding. */
Equate TRUE$ To 1
Equate FALSE$ To 0
table = "SAMPLE_CUSTOMERS"
key = 8
status = ""
Open table To tablevar Then
  locked = FALSE$
  Loop
    Lock tablevar,key Then
      locked = TRUE$
    End Else
      /* set up delay for network server access */
      For ctr = 1 To 1000
      Next
    End
  Until Locked Repeat
  * we're locked, now read
  Read @RECORD From tablevar,key Then
    /* do row processing here */
  End
  * now, unlock the row we've written back
  UnLock tablevar,key Else
    Status = Set_FSError()
    /* A row should always unlock first time out. */
  End
End Else  ;* table fails to open
  status = Set_FSError()
End  ;* table taken care of
Return status
```

Note: Locking a table or row is not a guarantee that it will not be changed by another user. All programs must contain Lock logic to ensure that tables or rows are not simultaneously updated.