NAME

lock, unlock, query_lock, set_lock_cache_enable, lock_cache_enabled – Class ss_m Methods for Locking

SYNOPSIS

```
#include <sm_vas.h>  // which includes sm.h

static rc_t                 lock(
    const lvid_t&               lvid,
    lock_mode_t                 mode,
    lock_duration_t             duration = t_long,
    long                        timeout = WAIT_SPECIFIED_BY_XCT);

static rc_t                 lock(
    const lvid_t&               lvid,
    const serial_t&             serial,
    lock_mode_t                 mode,
    lock_duration_t             duration = t_long,
    long                        timeout = WAIT_SPECIFIED_BY_XCT);

static rc_t                 unlock(
    const lvid_t&               lvid,
    const serial_t&             serial);

static rc_t                 query_lock(
    const lvid_t&               lvid,
    const serial_t&             serial,
    lock_mode_t&                mode,
    bool                        implicit = FALSE);

static rc_t                 set_lock_cache_enable(bool enable);
static rc_t                 lock_cache_enabled(bool& enabled);

static rc_t                 set_escalation_thresholds(
    int4                        toPage,
    int4                        toStore,
    int4                        toVolume);

static rc_t                 get_escalation_thresholds(
    int4&                       toPage,
    int4&                       toStore,
    int4&                       toVolume);

static rc_t                 dont_escalate(
    const lockid_t&             n,
    bool                        passOnToDescendants = true);
static rc_t                 dont_escalate(
    const lvid_t&               lvid,
    const serial_t&             serial,
    bool                        passOnToDescendants = true);
static rc_t                 dont_escalate(
    const lvid_t&               lvid,
    bool                        passOnToDescendants = true);
```

DESCRIPTION

For a discussion of the locking done by the SSM is found in **An Overview of Shore.**

Locks are acquired implicitly by many **ss_m** methods. For those situations where more precise control of locking is desired, the following methods allow explicit locking and unlocking.

lock(lvid, mode, duration, timeout)

lock(lvid, serial, mode, duration, timeout)

The **lock** method is used to acquire a lock on volume, index, file or record. The first version of the method locks the volume specified by *lvid.* The second version locks the index, file or record specified by *lvid,serial.* The *mode* parameter specifies the lock mode to acquire. Valid lock_mode_t values are listed in
`basics.h`. The *duration* parameter specifies how long the lock will be held. Valid values (among those listed in `basics.h`) are: t_instant, t_short and t_long. The *timeout* parameter specifies how long to wait for a lock.

unlock(lvid, serial)

The **unlock** method releases the most recently acquired lock on the file, index, or record identified by *lvid,serial.* Note, that only locks with duration **t_short** can be released before end-of-transaction.

query_lock(lvid, serial, mode, implicit)

The **query_lock** method the mode of the lock held on *lvid,serial* by the current transaction. The lock mode is returned in *mode* and will be **NL** (no lock) if not locked. If *implicit* is **false** then only explicit locks on *lvid,serial* will be considered. For example, if file F is **SH** locked and a query is made about a record in F, the mode returned will be **NL. However, if** *implicit* is **true,** then **SH** would be returned for this example.

Lock Cache Control

Each transaction has a cache of recently acquired locks The following methods control the use of the cache. These are not supported methods and may be removed in later versions of the software. Note: that the methods only affect the transaction associated with the current thread.

set_lock_cache_enable(enable)

The **set_lock_cache_enable** method turns on the cache if *enable* is **true** and turns it off otherwise.

lock_cache_enabled(enabled)

The **lock_cache_enabled** method sets *enabled* to **true** if the lock cache is on.

Escalation

The lock manager will escalate from a record lock to a page lock, from a page lock to a store lock, and from a store lock to a volume lock, to reduce the number of locks in the table. You can control the thresholds for escalation throught the methods **get_escalation_thresholds** and **set_escalation_thresholds.** The default values are as follows:

record-to-page
5

page-to-store
25

store-to-volume
0

In all cases, a threshold of 0 prevents escalation.

When escalation is in use, it be prevented on selected volumes or other lock-able objects through the three **don_escalate** methods. If the argument *passOnToDescendants is false*, locks acquired on objects below the volume (or given lockid) in the lock hierarchy will still be escalated according to the thresholds.

ERRORS

TODO

EXAMPLES

TODO

VERSION

This manual page applies to Version 2.0 of the Shore Storage Manager.

SPONSORSHIP

The Shore project is sponsored by the Advanced Research Project Agency, ARPA order number 018 (formerly 8230), monitored by the U.S. Army Research Laboratory under contract DAAB07-91-C-Q518. Further funding for this work was provided by DARPA through Rome Research Laboratory Contract No. F30602-97-2-0247.

COPYRIGHT

SEE ALSO

An Overview of Shore , **transaction(ssm)** and **intro(ssm).**