NAME

DbLock − lock class

SYNOPSIS

```
#include <db_cxx.h>

int
DbLock::put(DbLockTab *lt);

unsigned int
DbLock::get_lock_id();

void
DbLock::set_lock_id(unsigned int);

DBLock::DbLock(unsigned int);
DBLock::DbLock();
DBLock::DbLock(const DbLock &);
DbLock &DBLock::operator = (const DbLock &);
```

DESCRIPTION

The DB library is a family of classes that provides a modular programming interface to transactions and record-oriented file access. The library includes support for transactions, locking, logging and file page caching, as well as various indexed access methods. Many of the classes (e.g., the file page caching class) are useful independent of the other DB classes, although some classes are explicitly based on other classes (e.g., transactions and logging). For a general description of the DB package, see *db_intro*(3).

This manual page describes the specific details of the locking interface. The DbLock class is used in conjunction with *DbLockTab*(3) to provide general-purpose locking. While designed to work with the other Db classes, these classes are also useful for more general locking purposes. Locks can be shared between processes. In most cases, when multiple threads or processes are using locking, the deadlock detector, *db_deadlock*(1), should be run.

DbLock::put

The *DbLock::put* method is a simple interface to the *DbLockTab::vec* functionality, and is equivalent to calling the *DbLockTab::vec* method with a single element *list* array, for which the *op* field is DB_LOCK_PUT and the *lock* field is represented by the DbLock object. The *DbLock::put* method returns success and failure as described for the *DbLockTab::vec* method.

DbLock::get_lock_id

The *DbLock::get_lock_id* method gets the underlying representation of the lock id. Direct manipulation of the underlying representation is not recommended.

DbLock::set_lock_id

The *DbLock::set_lock_id* method sets the underlying representation of the lock id. Direct manipulation of the underlying representation is not recommended. Rather, *DbLockTab::get* should be used to initialize locks (see *DbLockTab*(3)), and the copy constructor and assignment operators can be used to assign to other DbLock objects.

ERRORS

Methods marked as returning *errno* will, by default, throw an exception that encapsulates the error information. The default error behavior can be changed, see *DbException*(3).

The *DbLock::put* method may fail and throw a *DbException*(3)

or return *errno* for any of the errors specified for the following DB and library functions:
DbLock::detect(3), fcntl(2), fflush(3), lseek(2), memcpy(3), memset(3), mmap(2), munmap(2), strerror(3), and write(2).

In addition, the *DbLock::put* method may fail and throw a *DbException*(3) or return *errno* for the following conditions:

[EACCES]
An attempt was made to release lock held by another locker.

[EINVAL]
An invalid flag value or parameter was specified.

BUGS

If a process dies while holding locks, those locks remain held and are **never** released. In this case, all processes should exit as quickly as possible, so that *db_recover* can be run.

SEE ALSO

db_archive(1), *db_checkpoint*(1), *db_deadlock*(1), *db_dump*(1), *db_load*(1), *db_recover*(1), *db_stat*(1), *db_intro*(3), *db_internal*(3), *db_thread*(3), *Db*(3), *Dbc*(3), *DbEnv*(3), *DbException*(3), *DbInfo*(3), *DbLock*(3), *DbLockTab*(3), *DbLog*(3), *DbLsn*(3), *DbMpool*(3), *DbMpoolFile*(3), *Dbt*(3), *DbTxn*(3), *DbTxnMgr*(3)