NAME

db_deadlock – the DB database deadlock detector

SYNOPSIS

db_deadlock [**-vw**] [**-a m | o | y**] [**-h home**] [**-L file**] [**-t sec**]

DESCRIPTION

The *db_deadlock* utility traverses the database lock structures and aborts a transaction each time it detects a deadlock. This utility should be run as a background daemon whenever multiple threads or processes are using locking. By default, a random transaction involved in the deadlock is aborted.

The options are as follows:

–a When a deadlock is detected, abort the oldest (“o”) transaction, the youngest (“y”) transaction, or the transaction with the minimum number of locks (“m”).

–h Specify a home directory for the database.

–L Log the execution of the db_deadlock utility to the specified file in the following format, where “###” is the process ID, and the date is the time the utility starting running.

db_deadlock: ### Wed Jun 15 01:23:45 EDT 1995

This file will be removed if the db_deadlock utility exits gracefully.

–t Initiate a pass over the database locks at least every *sec* seconds.

–v Run in verbose mode, generating messages each time the detector runs.

–w Make a single pass over the database locks every time a process is forced to wait for a lock.

At least one of the **-t** and **-w** options must be specified.

The *db_deadlock* utility attaches to DB shared memory regions. In order to avoid region corruption, it should always be given the chance to detach and exit gracefully. To cause *db_deadlock* to clean up after itself and exit, send it an interrupt signal (SIGINT).

The *db_deadlock* utility exits 0 on success, and >0 if an error occurs.

ENVIRONMENT VARIABLES

The following environment variables affect the execution of *db_deadlock*:

DB_HOME

If the **–h** option is not specified and the environment variable *DB_HOME* is set, it is used as the path of the database home, as described in *db_appinit*(3).

SEE ALSO

The DB library is a family of groups of functions 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 functional groups (e.g., the file page caching functions) are useful independent of the other DB functions, although some functional groups are explicitly based on other functional groups (e.g., transactions and logging). For a general description of the DB package, see *db_intro*(3).

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_appinit*(3), *db_cursor*(3), *db_dbm*(3), *db_internal*(3), *db_lock*(3), *db_log*(3), *db_mpool*(3), *db_open*(3), *db_thread*(3), *db_txn*(3)