Oracle Berkeley DB

Berkeley DB API Reference for TCL

11g Release 2

Library Version 11.2.5.2

Legal Notice

Published 9/14/2011

Table of Contents

Preface

Welcome to Berkeley DB (DB) 11*g* Release 2 (which provides library version 11.2.5.2). This book describes the DB Tcl API, including all classes, methods, and functions. As such, this document is intended for Tcl developers who are actively writing or maintaining applications that make use of DB databases.

Conventions Used in this Book

The following typographical conventions are used within in this manual:

Structure names are represented in `monospaced font`, as are `method names`. For example: "`berkdb open()` is a method on a `DB` handle."

Variable or non-literal text is presented in *italics*. For example: "Go to your *DB_INSTALL* directory."

Program examples are displayed in a `monospaced font` on a shaded background. For example:

```
db put
    -append
    [-partial {doff dlen}]
    [-txn txnid]
    data

db put
    [-nooverwrite]
    [-partial {doff dlen}]
    [-txn txnid]
    key data
```

Note

Finally, notes of interest are represented using a note block such as this.

For More Information

Beyond this manual, you may also find the following sources of information useful when building a DB application:

- Getting Started with Transaction Processing for C
- Berkeley DB Getting Started with Replicated Applications for C
- Berkeley DB Programmer's Reference Guide
- Berkeley DB Installation and Build Guide
- Berkeley DB Getting Started with the SQL APIs
- Berkeley DB C API Reference Guide

To download the latest Berkeley DB documentation along with white papers and other collateral, visit http://www.oracle.com/technetwork/indexes/documentation/index.html.

For the latest version of the Oracle Berkeley DB downloads, visit http://www.oracle.com/technetwork/database/berkeleydb/downloads/index.html.

Contact Us

You can post your comments and questions at the Oracle Technology (OTN) forum for Oracle Berkeley DB at: http://forums.oracle.com/forums/forum.jspa?forumID=271, or for Oracle Berkeley DB High Availability at: http://forums.oracle.com/forums/forum.jspa?forumID=272.

For sales or support information, email to: berkeleydb-info_us@oracle.com You can subscribe to a low-volume email announcement list for the Berkeley DB product family by sending email to: bdb-join@oss.oracle.com

Chapter 1. Berkeley DB Tcl APIs

This book documents the Tcl APIs that are available for working with Berkeley DB databases. This book assumes you have some familiarity with Berkeley DB.

Tcl Methods

Database Methods	Description
db close	Close a database
db count	Return a count of the key's data items
db del	Delete items from the database
db get	Get items from a database
db get_join	Get items from a database join
db get_type	Return the database type
db is_byteswapped	Return if the underlying database is in host order
berkdb open	Create and open a database handle
db put	Store items into a database
berkdb dbremove	Remove a database
berkdb dbrename	Rename a database
db stat	Return database statistics
db sync	Flush a database to stable storage
db truncate	Truncate a database
berkdb version	Return version information
Cursor Methods	
db cursor	Open a cursor in the database
db join	Perform a database join on cursors
dbc close	Close a cursor
dbc cmp	Compare two cursors
dbc del	Delete by cursor
dbc dup	Duplicate a cursor
dbc get	Retrieve by cursor
dbc put	Store by cursor
Environment Methods	
env close	Close an environment
env dbremove	Remove an environment
env dbrename	Rename a database
berkdb env	Create and open an environment handle
berkdb envremove	Remove an environment handle
Transaction Methods	
env txn	Begin a transaction

Database Methods	Description
txn abort	Abort a transaction
env txn_checkpoint	Checkpoint the environment
txn commit	Commit a transaction

db close

```
db close
 [-nosync]
```

The *db* **close** command flushes any cached database information to disk, closes any open cursors, frees any allocated resources, and closes any underlying files. Because key/data pairs are cached in memory, failing to sync the file with the *db* **close** or *db* **sync** command may result in inconsistent or lost information.

The options are as follows:

- **-nosync**

  Do not flush cached information to disk.

  The -nosync flag is a dangerous option. It should only be set if the application is doing logging (with transactions) so that the database is recoverable after a system or application crash, or if the database is always generated from scratch after any system or application crash.

  It is important to understand that flushing cached information to disk only minimizes the window of opportunity for corrupted data. Although unlikely, it is possible for database corruption to happen if a system or application crash occurs while writing data to the database. To ensure that database corruption never occurs, applications must either use transactions and logging with automatic recovery, use logging and application-specific recovery, or edit a copy of the database; and after all applications using the database have successfully called *db* **close**, atomically replace the original database with the updated copy.

After *db* **close** has been called, regardless of its return, the DB handle may not be accessed again.

The *db* **close** command returns 0 on success, and in the case of error, a Tcl error is thrown.

db count

```
db count key
```

The *db* **count** command returns a count of the number of duplicate data items for the key given. If the key does not exist, a value of 0 is returned. If there are no duplicates, or if the database does not support duplicates, but a key/data pair exists, a value of 1 is returned. If an error occurs, a Berkeley DB error message is returned or a Tcl error is thrown.

db cursor

```
db cursor
 [-txn txnid]
```

The *db* **cursor** command creates a database cursor. The returned cursor handle is bound to a Tcl command of the form **dbN.cX**, where X is an integer starting at 0 (for example, db0.c0 and db0.c1). It is through this Tcl command that the script accesses the cursor methods.

The options are as follows:

- **-txn txnid**

  If the operation is part of an application-specified transaction, the **txnid** parameter is a transaction handle returned from *env* **txn**. If no transaction handle is specified, but the operation occurs in a transactional database, the operation will be implicitly transaction protected.

In the case of error, a Tcl error is thrown.

db del

```
db del
 [-glob]
 [-txn txnid]
 key
```

The *db* **del** command removes key/data pairs from the database.

In the presence of duplicate key values, all records associated with the designated key will be discarded.

The options are as follows:

- **-glob**

  The specified key is a wildcard pattern, and all keys matching that pattern are discarded from the database. The pattern is a simple wildcard, any characters after the wildcard character are ignored. This option only works on databases using the Btree access method.

- **-txn txnid**

  If the operation is part of an application-specified transaction, the **txnid** parameter is a transaction handle returned from *env* **txn**. If no transaction handle is specified, but the operation occurs in a transactional database, the operation will be implicitly transaction protected.

The *db* **del** command returns 0 on success, and in the case of error, a Tcl error is thrown.

db get

```
db get
 [-consume]
 [-consume_wait]
 [-glob]
 [-partial {doff dlen}]
 [-recno]
 [-rmw]
 [-txn txnid]
 key

db get
 -get_both
 [-partial {doff dlen}]
 [-rmw]
 [-txn txnid]
 key data
```

The *db* **get** command returns key/data pairs from the database.

In the presence of duplicate key values, *db* **get** will return all duplicate items. Duplicates are sorted by insert order except where this order has been overridden by cursor operations.

The options are as follows:

- **-consume**

  Return the record number and data from the available record closest to the head of the queue, and delete the record. The cursor will be positioned on the deleted record. A record is available if it is not deleted and is not currently locked. The underlying database must be of type Queue for **-consume** to be specified.

- **-consume_wait**

  The same as the **-consume** flag except that if the Queue database is empty, the thread of control will wait until there is data in the queue before returning. The underlying database must be of type Queue for **-consume_wait** to be specified.

- **-get_both key data**

  Retrieve the key/data pair only if both the key and data match the arguments.

- **-glob**

  Return all keys matching the given key, where the key is a simple wildcard pattern. Where it is used, it replaces the use of the key with the given pattern of a set of keys. Any characters after the wildcard character are ignored. For example, in a database of last names, the command "db0 get Jones" will return all occurrences of "Jones" in the database, and the command "db0 get -glob Jo*" will return both "Jones" and "Johnson" from the database. The command "db0 get -glob *" will return all of the key/data pairs in the database. This option only works on databases using the Btree access method.

- **-partial {doff dlen}**

  The **dlen** bytes starting **doff** bytes from the beginning of the retrieved data record are returned as if they comprised the entire record. If any or all of the specified bytes do not exist in the record, the command is successful and any existing bytes are returned.

- **-recno**

  Retrieve the specified numbered key/data pair from a database. For **-recno** to be specified, the specified key must be a record number; and the underlying database must be of type Recno or Queue, or of type Btree that was created with the **-recnum** option.

- **-rmw**

  Acquire write locks instead of read locks when doing the retrieval. Setting this flag may decrease the likelihood of deadlock during a read-modify-write cycle by immediately acquiring the write lock during the read part of the cycle so that another thread of control acquiring a read lock for the same item, in its own read-modify-write cycle, will not result in deadlock.

  Because the *db* **get** command will not hold locks across Berkeley DB interface calls in nontransactional environments, the **-rmw** argument to the *db* **get** call is only meaningful in the presence of transactions.

- **-txn txnid**

  If the operation is part of an application-specified transaction, the **txnid** parameter is a transaction handle returned from *env* **txn**. If no transaction handle is specified, but the operation occurs in a transactional database, the operation will be implicitly transaction protected.

If the underlying database is a Queue or Recno database, the given key will be interpreted by Tcl as an integer. For all other database types, the key is interpreted by Tcl as a byte array, unless indicated by a given option.

A list of key/data pairs is returned. In the error case that no matching key exists, an empty list is returned. In all other cases, a Tcl error is thrown.

db get_join

```
db get_join
 [-txn txnid]
 {db key}
 {db key}
 ...
```

The *db* **get_join** command performs the cursor operations required to join the specified keys and returns a list of joined {key data} pairs. See Equality Join in the *Berkeley DB Programmer's Reference Guide* for more information on the underlying requirements for joining.

The options are as follows:

- **-txn txnid**

  If the operation is part of an application-specified transaction, the **txnid** parameter is a transaction handle returned from *env* **txn**. If no transaction handle is specified, but the operation occurs in a transactional database, the operation will be implicitly transaction protected.

In the case of error, a Tcl error is thrown.

db get_type

```
db get_type
```

The *db* **get_type** command returns the underlying database type, returning one of "btree", "hash", "queue" or "recno".

In the case of error, a Tcl error is thrown.

db is_byteswapped

```
db is_byteswapped
```

The *db* **is_byteswapped** command returns 0 if the underlying database files were created on an architecture of the same byte order as the current one, and 1 if they were not (that is, big-endian on a little-endian machine, or vice versa). This information may be used to determine if application data needs to be adjusted for this architecture or not.

In the case of error, a Tcl error is thrown.

db join

```
db join
 db.cX
 db.cY
 db.cZ
 ...
```

The *db* **join** command joins the specified cursors and returns a cursor handle that can be used to iterate through the joined {key data} pairs. The returned cursor handle is bound to a Tcl command of the form **dbN.cX**, where X is an integer starting at 0 (for example, db0.c0 and db0.c1). It is through this Tcl command that the script accesses the cursor methods.

The returned join cursor has limited cursor functionality, and only the *dbc* **get** and *dbc* **close** commands will succeed.

See Equality Join in the *Berkeley DB Programmer's Reference Guide* for more information on the underlying requirements for joining.

In a transaction-protected environment, all the cursors listed must have been created within the same transaction.

In the case of error, a Tcl error is thrown.

berkdb open

```
berkdb open
 [-auto_commit]
 [-btree | -hash | -recno | -queue | -unknown]
 [-cachesize {gbytes bytes ncache}]
 [-create]
 [-delim delim]
 [-dup]
 [-dupsort]
 [-encrypt]
 [-encryptaes passwd]
 [-encryptany passwd]
 [-env env]
 [-errfile filename]
 [-excl]
 [-extent size]
 [-ffactor density]
 [-len len]
 [-mode mode]
 [-nelem size]
 [-pad pad]
 [-pagesize pagesize]
 [-rdonly]
 [-recnum]
 [-renumber]
 [-snapshot]
 [-source file]
 [-truncate]
 [-txn txnid]
 [--]
 [file [database]]
```

The **berkdb open** command opens and optionally creates a database. The returned database handle is bound to a Tcl command of the form **dbN**, where N is an integer starting at 0 (for example, db0 and db1). It is through this Tcl command that the script accesses the database methods.

The options are as follows:

- **-auto_commit**

  Enclose the call within an implicit transaction (you do not need to provide a transaction handle as a transaction is internally created and commited for you). If the call succeeds, the open operation will be recoverable and all subsequent database modification operations based on this handle will be transactionally protected. If the call fails, no database will have been created.

- **-btree**

Open/create a database of type Btree. The Btree format is a representation of a sorted, balanced tree structure.

- **-hash**

  Open/create a database of type Hash. The Hash format is an extensible, dynamic hashing scheme.

- **-queue**

  Open/create a database of type Queue. The Queue format supports fast access to fixed-length records accessed by sequentially or logical record number.

- **-recno**

  Open/create a database of type Recno. The Recno format supports fixed- or variable-length records, accessed sequentially or by logical record number, and optionally retrieved from a flat text file.

- **-unknown**

  The database is of an unknown type, and must already exist.

- **-cachesize {gbytes bytes ncache}**

  Set the size of the database's shared memory buffer pool (that is, the cache), to **gbytes** gigabytes plus **bytes**. The cache should be the size of the normal working data set of the application, with some small amount of additional memory for unusual situations. (Note: The working set is not the same as the number of simultaneously referenced pages, and should be quite a bit larger!)

  The default cache size is 256KB, and may not be specified as less than 20KB. Any cache size less than 500MB is automatically increased by 25% to account for buffer pool overhead; cache sizes larger than 500MB are used as specified.

  It is possible to specify caches to Berkeley DB that are large enough so that they cannot be allocated contiguously on some architectures; for example, some releases of Solaris limit the amount of memory that may be allocated contiguously by a process. If **ncache** is 0 or 1, the cache will be allocated contiguously in memory. If it is greater than 1, the cache will be broken up into **ncache** equally sized separate pieces of memory.

  For information on tuning the Berkeley DB cache size, see Selecting a Cache Size in the *Berkeley DB Programmer's Reference Guide*.

  Because databases opened within Berkeley DB environments use the cache specified to the environment, it is an error to attempt to set a cache in a database created within an environment.

- **-create**

  Create any underlying files, as necessary. If the files do not already exist and the **-create** argument is not specified, the call will fail.

- **-delim delim**

  Set the delimiting byte used to mark the end of a record in the backing source file for the Recno access method.

  This byte is used for variable length records if the **-source** argument file is specified. If the **-source** argument file is specified and no delimiting byte was specified, <newline> characters (that is, ASCII 0x0a) are interpreted as end-of-record markers.

- **-dup**

  Permit duplicate data items in the tree, that is, insertion when the key of the key/data pair being inserted already exists in the tree will be successful. The ordering of duplicates in the tree is determined by the order of insertion unless the ordering is otherwise specified by use of a cursor or a duplicate comparison function.

  It is an error to specify both **-dup** and **-recnum**.

- **-dupsort**

  Sort duplicates within a set of data items. A default lexical comparison will be used. Specifying that duplicates are to be sorted changes the behavior of the *db* **put** operation as well as the *dbc* **put** operation when the **-keyfirst**, **-keylast** and **-current** options are specified.

- **-encrypt**

  Specify the database in an environment should be encrypted with the same password that is being used in the environment.

- **-encryptaes passwd**

  Specify the database should be encrypted with the given password using the Rijndael/ AES (also known as the Advanced Encryption Standard and Federal Information Processing Standard (FIPS) 197) algorithm.

- **-encryptany passwd**

  Specify the already existing database should be opened with the given password. This option is used if the database is known to be encrypted, but the specific algorithm used is not known.

- **-env env**

  If no **-env** argument is given, the database is standalone; that is, it is not part of any Berkeley DB environment.

  If a **-env** argument is given, the database is created within the specified Berkeley DB environment. The database access methods automatically make calls to the other subsystems in Berkeley DB, based on the enclosing environment. For example, if the environment has been configured to use locking, the access methods will automatically acquire the correct locks when reading and writing pages of the database.

- **-errfile filename**

  When an error occurs in the Berkeley DB library, a Berkeley DB error or an error return value is returned by the function. In some cases, however, the errno value may be insufficient to completely describe the cause of the error especially during initial application debugging.

  The **-errfile** argument is used to enhance the mechanism for reporting error messages to the application by specifying a file to be used for displaying additional Berkeley DB error messages. In some cases, when an error occurs, Berkeley DB will output an additional error message to the specified file reference.

  The error message will consist of a Tcl command name and a colon (":"), an error string, and a trailing <newline> character. If the database was opened in an environment, the Tcl command name will be the environment name (for example, env0), otherwise it will be the database command name (for example, db0).

  This error-logging enhancement does not slow performance or significantly increase application size, and may be run during normal operation as well as during application debugging.

  For database handles opened inside of Berkeley DB environments, specifying the **-errfile** argument affects the entire environment and is equivalent to specifying the same argument to the **berkdb env** command.

- **-excl**

  Return an error if the database already exists.

- **-extent size**

  Set the size of the extents of the Queue database; the size is specified as the number of pages in an extent. Each extent is created as a separate physical file. If no extent size is set, the default behavior is to create only a single underlying database file.

  For information on tuning the extent size, see Selecting an Extent Size in the *Berkeley DB Programmer's Reference Guide*.

- **-ffactor density**

  Set the desired density within the hash table.

  The density is an approximation of the number of keys allowed to accumulate in any one bucket

- **-len len**

  For the Queue access method, specify that the records are of length **len**.

  For the Recno access method, specify that the records are fixed-length, not byte-delimited, and are of length **len**.

Any records added to the database that are less than **len** bytes long are automatically padded (see the **-pad** argument for more information).

Any attempt to insert records into the database that are greater than **len** bytes long will cause the call to fail immediately and return an error.

- **-mode mode**

  On UNIX systems, or in IEEE/ANSI Std 1003.1 (POSIX) environments, all files created by the access methods are created with mode **mode** (as described in **chmod**(2)) and modified by the process' umask value at the time of creation (see **umask**(2)). The group ownership of created files is based on the system and directory defaults, and is not further specified by Berkeley DB. If **mode** is 0, files are created readable and writable by both owner and group. On Windows systems, the mode argument is ignored.

- **-nelem size**

  Set an estimate of the final size of the hash table.

  If not set or set too low, hash tables will still expand gracefully as keys are entered, although a slight performance degradation may be noticed.

- **-pad pad**

  Set the padding character for short, fixed-length records for the Queue and Recno access methods.

  If no pad character is specified, <space> characters (that is, ASCII 0x20) are used for padding.

- **-pagesize pagesize**

  Set the size of the pages used to hold items in the database, in bytes. The minimum page size is 512 bytes, and the maximum page size is 64K bytes. If the page size is not explicitly set, one is selected based on the underlying filesystem I/O block size. The automatically selected size has a lower limit of 512 bytes and an upper limit of 16K bytes.

  For information on tuning the Berkeley DB page size, see Selecting a Page Size in the *Berkeley DB Programmer's Reference Guide*.

- **-rdonly**

  Open the database for reading only. Any attempt to modify items in the database will fail, regardless of the actual permissions of any underlying files.

- **-recnum**

  Support retrieval from the Btree using record numbers.

  Logical record numbers in Btree databases are mutable in the face of record insertion or deletion. See the **-renumber** argument for further discussion.

Maintaining record counts within a Btree introduces a serious point of contention, namely the page locations where the record counts are stored. In addition, the entire tree must be locked during both insertions and deletions, effectively single-threading the tree for those operations. Specifying **-recnum** can result in serious performance degradation for some applications and data sets.

It is an error to specify both **-dup** and **-recnum**.

- **-renumber**

  Specifying the **-renumber** argument causes the logical record numbers to be mutable, and change as records are added to and deleted from the database. For example, the deletion of record number 4 causes records numbered 5 and greater to be renumbered downward by one. If a cursor was positioned to record number 4 before the deletion, it will refer to the new record number 4, if any such record exists, after the deletion. If a cursor was positioned after record number 4 before the deletion, it will be shifted downward one logical record, continuing to refer to the same record as it did before.

  Using the *db* **put** or *dbc* **put** interfaces to create new records will cause the creation of multiple records if the record number is more than one greater than the largest record currently in the database. For example, creating record 28 when record 25 was previously the last record in the database, will create records 26 and 27 as well as 28.

  If a created record is not at the end of the database, all records following the new record will be automatically renumbered upward by one. For example, the creation of a new record numbered 8 causes records numbered 8 and greater to be renumbered upward by one. If a cursor was positioned to record number 8 or greater before the insertion, it will be shifted upward one logical record, continuing to refer to the same record as it did before.

  For these reasons, concurrent access to a Recno database with the **-renumber** flag specified may be largely meaningless, although it is supported.

- **-snapshot**

  This argument specifies that any specified **-source** file be read in its entirety when the database is opened. If this argument is not specified, the **-source** file may be read lazily.

- **-source file**

  Set the underlying source file for the Recno access method. The purpose of the **-source** file is to provide fast access and modification to databases that are normally stored as flat text files.

  If the **-source** argument is give, it specifies an underlying flat text database file that is read to initialize a transient record number index. In the case of variable length records, the records are separated as specified by **-delim**. For example, standard UNIX byte stream files can be interpreted as a sequence of variable length records separated by <newline> characters.

In addition, when cached data would normally be written back to the underlying database file (for example, when the *db* **close** or *db* **sync** commands are called), the in-memory copy of the database will be written back to the **-source** file.

By default, the backing source file is read lazily, that is, records are not read from the file until they are requested by the application. **If multiple processes (not threads) are accessing a Recno database concurrently and either inserting or deleting records, the backing source file must be read in its entirety before more than a single process accesses the database, and only that process should specify the backing source argument as part of the berkdb open call. See the -snapshot argument for more information.**

Reading and writing the backing source file specified by -source cannot be transaction protected because it involves filesystem operations that are not part of the Berkeley DB transaction methodology. For this reason, if a temporary database is used to hold the records, it is possible to lose the contents of the **-source** file, for example, if the system crashes at the right instant. If a file is used to hold the database, that is, a filename was specified as the **file** argument to **berkdb open**, normal database recovery on that file can be used to prevent information loss, although it is still possible that the contents of **-source** file will be lost if the system crashes.

The **-source** file must already exist (but may be zero-length) when **berkdb open** is called.

It is not an error to specify a read-only **-source** file when creating a database, nor is it an error to modify the resulting database. However, any attempt to write the changes to the backing source file using either the *db* **close** or *db* **sync** commands will fail, of course. Specifying the **-nosync** argument to the *db* **close** command will stop it from attempting to write the changes to the backing file; instead, they will be silently discarded.

For all of the previous reasons, the **-source** file is generally used to specify databases that are read-only for Berkeley DB applications, and that are either generated on the fly by software tools, or modified using a different mechanism such as a text editor.

- **-truncate**

  Physically truncate the underlying file, discarding all previous databases it might have held. Underlying filesystem primitives are used to implement this flag. For this reason, it is only applicable to the physical file and cannot be used to discard databases within a file.

  The **-truncate** argument cannot be transaction-protected, and it is an error to specify it in a transaction-protected environment.

- **-txn txnid**

  If the operation is part of an application-specified transaction, the **txnid** parameter is a transaction handle returned from *env* **txn**. If no transaction handle is specified, but the -auto_commit flag is specified, the operation will be implicitly transaction protected.

- **--**

Mark the end of the command arguments.

- **file**

  The name of a single physical file on disk that will be used to back the database.

  An in-memory database never intended to be preserved on disk may be created by not specifying a file name. For example:

  ```
  berkdb open -create -btree
  ```

  creates an in-memory database.

- **database**

  The **database** argument allows applications to have multiple databases inside of a single physical file. This is useful when the databases are both numerous and reasonably small, in order to avoid creating a large number of underlying files. It is an error to attempt to open a second database file that was not initially created using a **database** name.

  Applications opening multiple databases in a single file will almost certainly need to create a shared database environment. See Opening multiple databases in a single file in the *Berkeley DB Programmer's Reference Guide* for more information.

  If more than one in-memory database is desired, it is necessary to specify an empty string as the database name. For example:

  ```
  berkdb open -create -btree "" foo
  berkdb open -create -btree "" bar
  ```

  will create two databases, neither of which will appear on disk.

The **berkdb open** command returns a database handle on success.

In the case of error, a Tcl error is thrown.

db put

```
db put
 -append
 [-partial {doff dlen}]
 [-txn txnid]
 data

db put
 [-nooverwrite]
 [-partial {doff dlen}]
 [-txn txnid]
 key data
```

The *db* **put** command stores the specified key/data pair into the database.

The options are as follows:

- **-append**

  Append the data item to the end of the database. For the **-append** option to be specified, the underlying database must be a Queue or Recno database. The record number allocated to the record is returned on success.

- **-nooverwrite**

  Enter the new key/data pair only if the key does not already appear in the database.

- **-partial {doff dlen}**

  The **dlen** bytes starting **doff** bytes from the beginning of the specified key's data record are replaced by the data specified by the data and size structure elements. If **dlen** is smaller than the length of the supplied data, the record will grow; if **dlen** is larger than the length of the supplied data, the record will shrink. If the specified bytes do not exist, the record will be extended using nul bytes as necessary, and the *db* **put** call will succeed.

  It is an error to attempt a partial put using the *db* **put** command in a database that supports duplicate records. Partial puts in databases supporting duplicate records must be done using a *dbc* **put** command.

  It is an error to attempt a partial put with differing **dlen** and supplied data length values in Queue or Recno databases with fixed-length records.

- **-txn txnid**

  If the operation is part of an application-specified transaction, the **txnid** parameter is a transaction handle returned from *env* **txn**. If no transaction handle is specified, but the operation occurs in a transactional database, the operation will be implicitly transaction protected.

The *db* **put** command returns either 0 or a record number for success (the record number is returned if the **-append** option was specified). If an error occurs, a Berkeley DB error message is returned or a Tcl error is thrown.

If the underlying database is a Queue or Recno database, then the given key will be interpreted by Tcl as an integer. For all other database types, the key is interpreted by Tcl as a byte array.

berkdb dbremove

```
berkdb dbremove
        [-encrypt]
 [-encryptaes passwd]
 [-encryptany passwd]
 [-env env]
 [--]
 file
 [database]
```

Remove the Berkeley DB database specified by the database name **file** and [database] name arguments. If no **database** is specified, the physical file represented by **file** is removed, incidentally removing all databases that it contained.

No reference count of database use is maintained by Berkeley DB. Applications should not remove databases that are currently in use.

The options are as follows:

- **-encrypt**

  Specify the database in an environment is encrypted with the same password that is being used in the environment.

- **-encryptaes passwd**

  Specify the database is encrypted with the given password using the Rijndael/AES (also known as the Advanced Encryption Standard and Federal Information Processing Standard (FIPS) 197) algorithm.

- **-encryptany passwd**

  Specify the already existing database is encrypted with the given password. This option is used if the database is known to be encrypted, but the specific algorithm used is not known.

- **-env env**

  If a **-env** argument is given, the database in the specified Berkeley DB environment is removed.

- **--**

  Mark the end of the command arguments.

The **berkdb dbremove** command returns 0 on success, and in the case of error, a Tcl error is thrown.

berkdb dbrename

```
berkdb dbrename
 [-encrypt]
 [-encryptaes passwd]
 [-encryptany passwd]
 [-env env]
 [--]
 file
 [database
 newname]
```

Renames the Berkeley DB database specified by the database name **file** and [database] name arguments to the new name given. If no **database** is specified, the physical file represented by **file** is renamed.

No reference count of database use is maintained by Berkeley DB. Applications should not rename databases that are currently in use.

The options are as follows:

- **-encrypt**

  Specify the database in an environment is encrypted with the same password that is being used in the environment.

- **-encryptaes passwd**

  Specify the database is encrypted with the given password using the Rijndael/AES (also known as the Advanced Encryption Standard and Federal Information Processing Standard (FIPS) 197) algorithm.

- **-encryptany passwd**

  Specify the already existing database is encrypted with the given password. This option is used if the database is known to be encrypted, but the specific algorithm used is not known.

- **-env env**

  If a **-env** argument is given, the database in the specified Berkeley DB environment is renamed.

- **--**

  Mark the end of the command arguments.

The **berkdb dbrename** command returns 0 on success, and in the case of error, a Tcl error is thrown.

db stat

```
db stat
 [-faststat]
```

The *db* **stat** command returns a list of name/value pairs comprising the statistics of the database.

The options are as follows:

- **-faststat**

  Return only that information which does not require a traversal of the database.

In the case of error, a Tcl error is thrown.

db sync

```
db sync
```

The *db* **sync** command function flushes any database cached information to disk.

See *db* **close** for a discussion of Berkeley DB and cached data.

The *db* **sync** command returns 0 on success, and in the case of error, a Tcl error is thrown.

db truncate

```
db truncate
 [-txn txnid]
```

Empties the database, discarding all records it contains.

The options are as follows:

- **-txn txnid**

  If the operation is part of an application-specified transaction, the **txnid** parameter is a transaction handle returned from *env* **txn**. If no transaction handle is specified, but the operation occurs in a transactional database, the operation will be implicitly transaction protected.

The *db* **truncate** command returns the number of records discarded from the database on success.

In the case of error, a Tcl error is thrown.

dbc close

```
dbc close
```

The *dbc* **close** command discards the cursor.

After *dbc* **close** has been called, regardless of its return, the cursor handle may not be used again.

The *dbc* **close** command returns 0 on success, and in the case of error, a Tcl error is thrown.

dbc cmp

```
dbc cmp cursor
```

The *dbc* **cmp** method compares two cursors for equality. Two cursors are equal if and only if they are positioned on the same item in the same database.

The *dbc* **cmp** command returns 0 if the cursors are equal, and a non-zero value if they are not equal.

dbc del

```
dbc del
```

The *dbc* **del** command deletes the key/data pair to which the cursor currently refers.

The cursor position is unchanged after a delete, and subsequent calls to cursor commands expecting the cursor to refer to an existing key will fail.

The *dbc* **del** command returns 0 on success, and in the case of error, a Tcl error is thrown.

dbc dup

```
dbc dup
 [-position]
```

The *dbc* **dup** command duplicates the cursor, creates a new cursor that uses the same transaction and locker ID as the original cursor. This is useful when an application is using locking and requires two or more cursors in the same thread of control.

The options are as follows:

- **-position**

  The newly created cursor is initialized to refer to the same position in the database as the original cursor and hold the same locks. If the **-position** flag is not specified, the created cursor is uninitialized and will behave like a cursor newly created using the *db* **cursor** command.

The *dbc* **dup** command returns 0 on success, and in the case of error, a Tcl error is thrown.

dbc get

```
dbc get
 [-current]
 [-first]
 [-get_recno]
 [-join_item]
 [-last]
 [-next]
 [-nextdup]
 [-nextnodup]
 [-partial {offset length}]
 [-prev]
 [-prevnodup]
 [-rmw]

dbc get
 [-partial {offset length}]
 [-rmw]
 [-set]
 [-set_range]
 [-set_recno]
 key

dbc get
 -get_both
 [-partial {offset length}]
 [-rmw]
 key data
```

The *dbc* **get** command returns a list of {key value} pairs, except in the case of the **-get_recno** and **-join_item** options. In the case of the **-get_recno** option, *dbc* **get** returns a list of the record number. In the case of the **-join_item** option, *dbc* **get** returns a list containing the joined key.

The options follow, and are grouped by the action they perform.

The first group affects the position of the cursor in the database without regard for the key or data item.

- **-current**

  Return the key/data pair to which the cursor currently refers.

  If the cursor key/data pair was deleted, *dbc* **get** will return an empty list.

- **-first**

  The cursor is set to refer to the first key/data pair of the database, and that pair is returned. In the presence of duplicate key values, the first data item in the set of duplicates is returned.

If the database is a Queue or Recno database, *dbc* **get** using the **-first** option will skip any keys that exist but were never explicitly created by the application, or were created and later deleted.

If the database is empty, *dbc* **get** will return an empty list.

- **-last**

  The cursor is set to refer to the last key/data pair of the database, and that pair is returned. In the presence of duplicate key values, the last data item in the set of duplicates is returned.

  If the database is a Queue or Recno database, *dbc* **get** using the **-last** option will skip any keys that exist but were never explicitly created by the application, or were created and later deleted.

  If the database is empty, *dbc* **get** will return an empty list.

- **-next**

  If the cursor is not yet initialized, the **-next** option is identical to **-first**.

  Otherwise, the cursor is moved to the next key/data pair of the database, and that pair is returned. In the presence of duplicate key values, the value of the key may not change.

  If the database is a Queue or Recno database, *dbc* **get** using the **-next** option will skip any keys that exist but were never explicitly created by the application, or were created and later deleted.

  If the cursor is already on the last record in the database, *dbc* **get** will return an empty list.

- **-nextdup**

  If the next key/data pair of the database is a duplicate record for the current key/data pair, the cursor is moved to the next key/data pair of the database, and that pair is returned. Otherwise, *dbc* **get** will return an empty list.

- **-nextnodup**

  If the cursor is not yet initialized, the **-nextnodup** option is identical to **-first**.

  Otherwise, the cursor is moved to the next non-duplicate key/data pair of the database, and that pair is returned.

  If no non-duplicate key/data pairs occur after the cursor position in the database, *dbc* **get** will return an empty list.

- **-prev**

  If the cursor is not yet initialized, **-prev** is identical to **-last**.

Otherwise, the cursor is moved to the previous key/data pair of the database, and that pair is returned. In the presence of duplicate key values, the value of the key may not change.

If the database is a Queue or Recno database, *dbc* **get** using the **-prev** flag will skip any keys that exist but were never explicitly created by the application, or were created and later deleted.

If the cursor is already on the first record in the database, *dbc* **get** will return an empty list.

- **-prevnodup**

  If the cursor is not yet initialized, the **-prevnodup** option is identical to **-last**.

  Otherwise, the cursor is moved to the previous non-duplicate key/data pair of the database, and that pair is returned.

  If no non-duplicate key/data pairs occur before the cursor position in the database, *dbc* **get** will return an empty list.

The next group of options are used to move the cursor to a location in the database based on characteristics of the key and/or data items.

- **-set**

  Move the cursor to the specified key/data pair of the database, and return the datum associated with the given key.

  In the presence of duplicate key values, *dbc* **get** will return the first data item for the given key.

  If the database is a Queue or Recno database and the requested key exists, but was never explicitly created by the application or was later deleted, *dbc* **get** will return an empty list.

  If no matching keys are found, *dbc* **get** will return an empty list.

- **-set_range**

  The **-set_range** option is identical to the **-set** option, except that the key is returned as well as the data item, and, in the case of the Btree access method, the returned key/data pair is the smallest key greater than or equal to the specified key (as determined by the comparison function), permitting partial key matches and range searches.

- **-get_both**

  The **-get_both** option is identical to the **-set** option, except that both the key and the data arguments must be matched by the key and data item in the database.

  For **-get_both** to be specified, the underlying database must be of type Btree or Hash.

The last group of options is a general collection of operations. Some of these involve positioning the cursor to a location in the database based in information other than what can

be found in the key and/or data items. Others of these have to do with the cursor's behavior upon retrieving information.

- **-set_recno**

  Move the cursor to the specific numbered record of the database, and return the associated key/data pair. The key must be a record number.

  For the **-set_recno** option to be specified, the underlying database must be of type Btree, and it must have been created with the **-recnum** option.

- **-get_recno**

  Return a list of the record number associated with the current cursor position. No key argument should be specified.

  For **-get_recno** to be specified, the underlying database must be of type Btree, and it must have been created with the **-recnum** option.

- **-join_item**

  Do not use the data value found in all the cursors as a lookup key for the primary database, but simply return it in the key parameter instead. The data parameter is left unchanged.

  For **-join_item** to be specified, the cursor must have been created by the *db* **join** command.

- **-partial {offset length}**

  The **dlen** bytes starting **doff** bytes from the beginning of the retrieved data record are returned as if they comprised the entire record. If any or all of the specified bytes do not exist in the record, the command is successful and any existing bytes are returned.

- **-rmw**

  Acquire write locks instead of read locks when doing the retrieval. Setting this flag may decrease the likelihood of deadlock during a read-modify-write cycle by immediately acquiring the write lock during the read part of the cycle so that another thread of control acquiring a read lock for the same item, in its own read-modify-write cycle, will not result in deadlock.

If a key is specified, and if the underlying database is a Queue or Recno database, the given key will be interpreted by Tcl as an integer. For all other database types, the key is interpreted by Tcl as a byte array, unless indicated by a given option.

In the normal error case of attempting to retrieve a key that does not exist an empty list is returned.

In the case of error, a Tcl error is thrown.

dbc put

```
dbc put
 [-after]
 [-before]
 [-current]
 [-partial {doff dlen}]
 data

dbc put
 [-keyfirst]
 [-keylast]
 [-partial {doff dlen}]
 key data
```

The *dbc* **put** command stores the specified key/data pair into the database. One of the following options must be specified:

- **-after**

  In the case of the Btree and Hash access methods, insert the data element as a duplicate element of the key to which the cursor refers. The new element appears immediately after the current cursor position. It is an error to specify **-after** if the underlying Btree or Hash database was not created with the **-dup** option. No key argument should be specified.

  In the case of the Recno access method, it is an error to specify the **-after** option if the underlying Recno database was not created with the **-renumber** option. If the **-renumber** option was specified, a new key is created, all records after the inserted item are automatically renumbered, and the key of the new record is returned in the structure to which the key argument refers. The initial value of the key parameter is ignored. See **berkdb open** for more information.

  In the case of the Queue access method, it is always an error to specify **-after**.

  If the current cursor record has already been deleted, and the underlying access method is Hash, *dbc* **put** will throw a Tcl error. If the underlying access method is Btree or Recno, the operation will succeed.

- **-before**

  In the case of the Btree and Hash access methods, insert the data element as a duplicate element of the key to which the cursor refers. The new element appears immediately before the current cursor position. It is an error to specify **-before** if the underlying Btree or Hash database was not created with the **-dup** option. No key argument should be specified.

  In the case of the Recno access method, it is an error to specify **-before** if the underlying Recno database was not created with the **-before** option. If the **-before** option was specified, a new key is created, the current record and all records after it are automatically renumbered, and the key of the new record is returned in the structure to which the key argument refers. The initial value of the key parameter is ignored. See **berkdb open** for more information.

In the case of the Queue access method, it is always an error to specify **-before**.

If the current cursor record has already been deleted and the underlying access method is Hash, *dbc* **put** will throw a Tcl error. If the underlying access method is Btree or Recno, the operation will succeed.

- **-current**

  Overwrite the data of the key/data pair to which the cursor refers with the specified data item. No key argument should be specified.

  If the **-dupsort** option was specified to **berkdb open** and the data item of the key/data pair to which the cursor refers does not compare equally to the data parameter, *dbc* **put** will throw a Tcl error.

  If the current cursor record has already been deleted and the underlying access method is Hash, *dbc* **put** will throw a Tcl error. If the underlying access method is Btree, Queue, or Recno, the operation will succeed.

- **-keyfirst**

  In the case of the Btree and Hash access methods, insert the specified key/data pair into the database.

  If the key already exists in the database, and the **-dupsort** option was specified to **berkdb open**, the inserted data item is added in its sorted location. If the key already exists in the database, and the **-dupsort** option was not specified, the inserted data item is added as the first of the data items for that key.

  The **-keyfirst** option may not be specified to the Queue or Recno access methods.

- **-keylast**

  In the case of the Btree and Hash access methods, insert the specified key/data pair into the database.

  If the key already exists in the database, and the **-dupsort** option was specified to **berkdb open**, the inserted data item is added in its sorted location. If the key already exists in the database, and the **-dupsort** option was not specified, the inserted data item is added as the last of the data items for that key.

  The **-keylast** option may not be specified to the Queue or Recno access methods.

- **-partial {doff dlen}**

  The **dlen** bytes starting **doff** bytes from the beginning of the specified key's data record are replaced by the data specified by the data and size structure elements. If **dlen** is smaller than the length of the supplied data, the record will grow; if **dlen** is larger than the length of the supplied data, the record will shrink. If the specified bytes do not exist, the record will be extended using nul bytes as necessary, and the *dbc* **put** call will succeed.

It is an error to attempt a partial put using the *dbc* **put** command in a database that supports duplicate records. Partial puts in databases supporting duplicate records must be done using a *dbc* **put** command.

It is an error to attempt a partial put with differing **dlen** and supplied data length values in Queue or Recno databases with fixed-length records.

If a key is specified, and if the underlying database is a Queue or Recno database, the given key will be interpreted by Tcl as an integer. For all other database types, the key is interpreted by Tcl as a byte array.

If *dbc* **put** fails for any reason, the state of the cursor will be unchanged. If *dbc* **put** succeeds and an item is inserted into the database, the cursor is always positioned to refer to the newly inserted item.

The *dbc* **put** command returns 0 on success, and in the case of error, a Tcl error is thrown.

env close

```
env close
```

Close the Berkeley DB environment, freeing any allocated resources and closing any underlying subsystems.

This does not imply closing any databases that were opened in the environment.

Where the environment was initialized with the **-txn** option, calling *env* **close** does not release any locks still held by the closing process, providing functionality for long-lived locks.

After *env* **close** has been called the **env** handle may not be accessed again.

The *env* **close** command returns 0 on success, and in the case of error, a Tcl error is thrown.

env dbremove

```
env dbremove
 [-auto_commit]
 [-txn txnid]
 [--]
 file
```

Remove the Berkeley DB database **file**.

The options are as follows:

- **-auto_commit**

  Enclose the call within an implicit transaction (you do not need to provide a transaction handle as a transaction is internally created and commited for you). If the call succeeds, changes made by the operation will be recoverable. If the call fails, the operation will have made no changes.

- **-txn txnid**

  If the operation is part of an application-specified transaction, the **txnid** parameter is a transaction handle returned from *env* **txn**. If no transaction handle is specified, but the -auto_commit flag is specified, the operation will be implicitly transaction protected.

The *env* **dbremove** command returns 0 on success, and in the case of error, a Tcl error is thrown.

env dbrename

```
env dbrename
 [-auto_commit]
 [-txn txnid]
 [--]
 file
 newname
```

Rename the Berkeley DB database **file** to **newname**.

The options are as follows:

- **-auto_commit**

  Enclose the call within an implicit transaction (you do not need to provide a transaction handle as a transaction is internally created and commited for you). If the call succeeds, changes made by the operation will be recoverable. If the call fails, the operation will have made no changes.

- **-txn txnid**

  If the operation is part of an application-specified transaction, the **txnid** parameter is a transaction handle returned from *env* **txn**. If no transaction handle is specified, but the -auto_commit flag is specified, the operation will be implicitly transaction protected.

The *env* **dbrename** command returns 0 on success, and in the case of error, a Tcl error is thrown.

berkdb env

```
berkdb env
 [-cachesize {gbytes bytes ncache}]
 [-create]
 [-data_dir dirname]
 [-encryptaes passwd]
 [-encryptany passwd]
 [-errfile filename]
 [-home directory]
 [-log_dir dirname]
 [-mode mode]
 [-private]
 [-recover]
 [-recover_fatal]
 [-shm_key shmid]
 [-system_mem]
 [-tmp_dir dirname]
 [-txn [nosync]]
 [-txn_max max]
 [-use_environ]
 [-use_environ_root]
```

The **berkdb env** command opens and optionally creates a database environment. The returned environment handle is bound to a Tcl command of the form **envN**, where N is an integer starting at 0 (for example, env0 and env1). It is through this Tcl command that the script accesses the environment methods. The command automatically initializes the Shared Memory Buffer Pool subsystem. This subsystem is used whenever the application is using any Berkeley DB access method.

The options are as follows:

- **-cachesize {gbytes bytes ncache}**

  Set the size of the database's shared memory buffer pool (that is, the cache), to **gbytes** gigabytes plus **bytes**. The cache should be the size of the normal working data set of the application, with some small amount of additional memory for unusual situations. (Note: The working set is not the same as the number of simultaneously referenced pages, and should be quite a bit larger!)

  The default cache size is 256KB, and may not be specified as less than 20KB. Any cache size less than 500MB is automatically increased by 25% to account for buffer pool overhead; cache sizes larger than 500MB are used as specified.

  It is possible to specify caches to Berkeley DB that are large enough so that they cannot be allocated contiguously on some architectures; for example, some releases of Solaris limit the amount of memory that may be allocated contiguously by a process. If **ncache** is 0 or 1, the cache will be allocated contiguously in memory. If it is greater than 1, the cache will be broken up into **ncache** equally sized separate pieces of memory.

For information on tuning the Berkeley DB cache size, see Selecting a Cache Size in the *Berkeley DB Programmer's Reference Guide*.

- **-create**

  Cause Berkeley DB subsystems to create any underlying files, as necessary.

- **-data_dir dirname**

  Specify the environment's data directory as described in Berkeley DB File Naming in the *Berkeley DB Programmer's Reference Guide*.

- **-encryptaes passwd**

  Specify the database should be encrypted with the given password using the Rijndael/AES (also known as the Advanced Encryption Standard and Federal Information Processing Standard (FIPS) 197) algorithm.

- **-encryptany passwd**

  Specify the already existing environment should be opened with the given password. This option is used if the environment is known to be encrypted, but the specific algorithm used is not known.

- **-errfile filename**

  When an error occurs in the Berkeley DB library, a Berkeley DB error or an error return value is returned by the function. In some cases, however, the errno value may be insufficient to completely describe the cause of the error especially during initial application debugging.

  The **-errfile** argument is used to enhance the mechanism for reporting error messages to the application by specifying a file to be used for displaying additional Berkeley DB error messages. In some cases, when an error occurs, Berkeley DB will output an additional error message to the specified file reference.

  consist of the environment command name (for example, env0) and a colon (":"), an error string, and a trailing <newline> character.

  This error-logging enhancement does not slow performance or significantly increase application size, and may be run during normal operation as well as during application debugging.

- **-home directory**

  The **-home** argument is described in Berkeley DB File Naming in the *Berkeley DB Programmer's Reference Guide*.

- **-log_dir dirname**

  Specify the environment's logging file directory as described in Berkeley DB File Naming in the *Berkeley DB Programmer's Reference Guide*.

- **-mode mode**

  On UNIX systems, or in IEEE/ANSI Std 1003.1 (POSIX) environments, all files created by Berkeley DB are created with mode **mode** (as described in **chmod**(2)) and modified by the process' umask value at the time of creation (see **umask**(2)). The group ownership of created files is based on the system and directory defaults, and is not further specified by Berkeley DB. If **mode** is 0, files are created readable and writable by both owner and group. On Windows systems, the mode argument is ignored.

- **-private**

  Specify that the environment will only be accessed by a single process (although that process may be multithreaded). This flag has two effects on the Berkeley DB environment. First, all underlying data structures are allocated from per-process memory instead of from shared memory that is potentially accessible to more than a single process. Second, mutexes are only configured to work between threads.

  This flag should not be specified if more than a single process is accessing the environment, as it is likely to cause database corruption and unpredictable behavior. For example, if both a server application and the Berkeley DB utility db_stat will access the environment, the **-private** option should not be specified.

- **-recover**

  Run normal recovery on this environment before opening it for normal use. If this flag is set, the **-create** option must also be set because the regions will be removed and re-created.

- **-recover_fatal**

  Run catastrophic recovery on this environment before opening it for normal use. If this flag is set, the **-create** option must also be set since the regions will be removed and re-created.

- **-shm_key key**

  Specify a base segment ID for Berkeley DB environment shared memory regions created in system memory on systems supporting X/Open-style shared memory interfaces, for example, UNIX systems supporting shmget(2) and related System V IPC interfaces. See Shared Memory Regions in the *Berkeley DB Programmer's Reference Guide* for more information.

- **-system_mem**

  Allocate memory from system shared memory instead of memory backed by the filesystem. See Shared Memory Regions in the *Berkeley DB Programmer's Reference Guide* for more information.

- **-tmp_dir dirname**

  Specify the environment's tmp directory, as described in Berkeley DB File Naming in the *Berkeley DB Programmer's Reference Guide*.

- **-txn [nosync]**

Initialize the Transaction subsystem. This subsystem is used when recovery and atomicity of multiple operations and recovery are important. The **-txn** option implies the initialization of the logging and locking subsystems as well.

If the optional **nosync** argument is specified, the log will not be synchronously flushed on transaction commit. This means that transactions exhibit the ACI (atomicity, consistency, and isolation) properties, but not D (durability); that is, database integrity will be maintained, but it is possible that some number of the most recently committed transactions may be undone during recovery instead of being redone.

The number of transactions that are potentially at risk is governed by how often the log is checkpointed (see db_checkpoint in the *Berkeley DB C API Reference Guide* for more information) and how many log updates can fit on a single log page.

- **-txn_max max**

  Set the maximum number of simultaneous transactions that are supported by the environment, which bounds the size of backing files. When there are more than the specified number of concurrent transactions, calls to *env* **txn** will fail (until some active transactions complete).

- **-use_environ**

  The Berkeley DB process' environment may be permitted to specify information to be used when naming files; see Berkeley DB File Naming in the *Berkeley DB Programmer's Reference Guide*. Because permitting users to specify which files are used can create security problems, environment information will be used in file naming for all users only if the **-use_environ** flag is set.

- **-use_environ_root**

  The Berkeley DB process' environment may be permitted to specify information to be used when naming files; see Berkeley DB File Naming in the *Berkeley DB Programmer's Reference Guide*. As permitting users to specify which files are used can create security problems, if the **-use_environ_root** flag is set, environment information will be used for file naming only for users with appropriate permissions (for example, users with a user-ID of 0 on IEEE/ANSI Std 1003.1 (POSIX) systems).

The **berkdb env** command returns an environment handle on success.

In the case of error, a Tcl error is thrown.

berkdb envremove

```
berkdb envremove
 [-data_dir directory]
 [-force]
 [-home directory]
 [-log_dir directory]
 [-tmp_dir directory]
 [-use_environ]
 [-use_environ_root]
```

Remove a Berkeley DB environment.

The options are as follows:

- **-data_dir dirname**

  Specify the environment's data directory, as described in Berkeley DB File Naming in the *Berkeley DB Programmer's Reference Guide*.

- **-force**

  If there are processes that have called **berkdb env** without calling *env* **close** (that is, there are processes currently using the environment), **berkdb envremove** will fail without further action, unless the **-force** flag is set, in which case **berkdb envremove** will attempt to remove the environment regardless of any processes still using it.

- **-home directory**

  The **-home** argument is described in Berkeley DB File Naming in the *Berkeley DB Programmer's Reference Guide*.

- **-log_dir dirname**

  Specify the environment's log directory, as described in Berkeley DB File Naming in the *Berkeley DB Programmer's Reference Guide*.

- **-tmp_dir dirname**

  Specify the environment's tmp directory, as described in Berkeley DB File Naming in the *Berkeley DB Programmer's Reference Guide*.

- **-use_environ**

  The Berkeley DB process' environment may be permitted to specify information to be used when naming files; see Berkeley DB File Naming in the *Berkeley DB Programmer's Reference Guide*. Because permitting users to specify which files are used can create security problems, environment information will be used in file naming for all users only if the **-use_environ** flag is set.

- **-use_environ_root**

The Berkeley DB process' environment may be permitted to specify information to be used when naming files; see Berkeley DB File Naming in the *Berkeley DB Programmer's Reference Guide*. As permitting users to specify which files are used can create security problems, if the **-use_environ_root** flag is set, environment information will be used for file naming only for users with appropriate permissions (for example, users with a user-ID of 0 on IEEE/ANSI Std 1003.1 (POSIX) systems).

The **berkdb envremove** command returns 0 on success, and in the case of error, a Tcl error is thrown.

env txn

```
env txn
 [-nosync]
 [-nowait]
 [-parent txnid]
 [-sync]
```

The *env* **txn** command begins a transaction. The returned transaction handle is bound to a Tcl command of the form **env.txnX**, where X is an integer starting at 0 (for example, env0.txn0 and env0.txn1). It is through this Tcl command that the script accesses the transaction methods.

The options are as follows:

- **-nosync**

  Do not synchronously flush the log when this transaction commits or prepares. This means the transaction will exhibit the ACI (atomicity, consistency, and isolation) properties, but not D (durability); that is, database integrity will be maintained, but it is possible that this transaction may be undone during recovery instead of being redone.

  This behavior may be set for an entire Berkeley DB environment as part of the **berkdb env** call.

- **-nowait**

  If a lock is unavailable for any Berkeley DB operation performed in the context of this transaction, throw a Tcl error immediately instead of blocking on the lock.

- **-parent txnid**

  Create the new transaction as a nested transaction, with the specified transaction indicated as its parent. Transactions may be nested to any level.

- **-sync**

  Synchronously flush the log when this transaction commits or prepares. This means the transaction will exhibit all of the ACID (atomicity, consistency, isolation, and durability) properties.

  This behavior is the default for Berkeley DB environments unless the **-nosync** option was specified to the **berkdb env** call.

The *env* **txn** command returns a transaction handle on success.

In the case of error, a Tcl error is thrown.

txn abort

```
txn abort
```

The *txn* **abort** command causes an abnormal termination of the transaction.

The log is played backward, and any necessary recovery operations are performed. After recovery is completed, all locks held by the transaction are acquired by the parent transaction in the case of a nested transaction, or released in the case of a non-nested transaction. As is the case for *txn* **commit**, applications that require strict two-phase locking should not explicitly release any locks.

In the case of nested transactions, aborting the parent transaction causes all children of that transaction to be aborted.

After *txn* **abort** has been called, regardless of its return, the **txn** handle may not be accessed again.

The *txn* **abort** command returns 0 on success, and in the case of error, a Tcl error is thrown.

env txn_checkpoint

```
env txn_checkpoint
 [-force]
 [-kbyte kb]
 [-min minutes]
```

The *env* **txn_checkpoint** command writes a checkpoint.

The options are as follows:

- **-force**

  The checkpoint will occur regardless of activity level.

- **-kbyte kb**

  The checkpoint will occur only if at least the specified number of kilobytes of log data has been written since the last checkpoint.

- **-min minutes**

  The checkpoint will occur only if at least the specified number of minutes has passed since the last checkpoint.

In the case of error, a Tcl error is thrown.

txn commit

```
txn commit
 [-nosync]
 [-sync]
```

The *txn* **commit** command ends the transaction.

In the case of nested transactions, if the transaction is a parent transaction with unresolved (neither committed or aborted) child transactions, the child transactions are aborted and the commit of the parent will succeed.

In the case of nested transactions, if the transaction is a child transaction, its locks are not released, but are acquired by its parent. Although the commit of the child transaction will succeed, the actual resolution of the child transaction is postponed until the parent transaction is committed or aborted; that is, if its parent transaction commits, it will be committed, and if its parent transaction aborts, it will be aborted.

If the **-nosync** option is not specified, a commit log record is written and flushed to disk, as are all previously written log records.

The options are as follows:

- **-nosync**

  Do not synchronously flush the log. This means the transaction will exhibit the ACI (atomicity, consistency, and isolation) properties, but not D (durability); that is, database integrity will be maintained, but it is possible that this transaction may be undone during recovery instead of being redone.

  This behavior may be set for an entire Berkeley DB environment as part of the **berkdb env** call.

- **-sync**

  Synchronously flush the log. This means the transaction will exhibit all of the ACID (atomicity, consistency, isolation and durability) properties.

  This behavior is the default for Berkeley DB environments unless the **-nosync** option was specified to the **berkdb env** or *env* **txn** calls.

After *txn* **commit** has been called, regardless of its return, the **txn** handle may not be accessed again. If *txn* **commit** encounters an error, this transaction and all child transactions of this transaction are aborted.

The *txn* **commit** command returns 0 on success, and in the case of error, a Tcl error is thrown.

berkdb version

```
berkdb version
 [-string]
```

Return a list of the form {major minor patch} for the major, minor and patch levels of the underlying Berkeley DB release.

The options are as follows:

- **-string**

  Return a string with formatted Berkeley DB version information.

In the case of error, a Tcl error is thrown.