t

Datafile Manager Primitives Descriptions *494*

cluster-delete

494

Input names: Table ID; Cluster ID; [Key Values]

Input types: integer; integer; [list]

Output types: integer

Description: Flags a cluster as deleted. The cluster is then inaccessible by other Prograph datafile primitives. If db-compact is performed, the space for the cluster is recovered. If a cluster is already deleted or the cluster ID is invalid, then a “cluster does not exist” error is returned. Clusters may not be deleted in query mode else a mode (4) error is returned.

 If a Key Values list is provided, the appropriate key-cluster associations are deleted. Key values are specified in sorted order of key names. If any of the keys do not exist, a “cluster does not exist” error is returned. If NULL is specified as a key value it will be ignored.

cluster-first

495

Input types: integer; boolean

Output types: integer; integer

Description: Returns the Cluster ID of the first cluster in the datafile.   If Active is TRUE the ID of the first active cluster is returned, otherwise the ID of the first deleted cluster is returned.

cluster-lock

495

Input types: integer; integer

Output types: integer

Description: Prevents other applications using the data file from reading, replacing, or deleting the specified cluster. Clusters may only be locked in share mode else a mode (4) error is returned.

cluster-next

496

Input types: integer; integer; boolean

Output types: integer; integer

Description: Returns the Cluster ID of the next cluster in the datafile after StartingClusterID.   If Active is TRUE the ID of the next active cluster is returned, otherwise the ID of the next deleted cluster is returned.

cluster-read

496

Input types: integer; integer

Output types: integer; any

Description: Reads the cluster with the specified cluster ID from the table. If the cluster has been deleted or if the cluster ID is invalid, then a “cluster does not exist” (10) error is returned. “Name not in class map” (3) error may be returned.

cluster-read-db

496

Input types: integer; integer

Output types:
integer; any

Description: Reads the cluster specified by ClusterId.

cluster-replace

497

Input names: Table ID; Old Cluster ID; Value; [Old Key Values; New Key Values]

Input types: integer; integer; any; [list; list]

Output types: integer; integer

Description: Replaces a cluster in a table. If the new cluster size is equal to or smaller than the original cluster size, the cluster is written in the same location in the data file. The cluster keeps the same cluster ID. If the new cluster size is greater than the old cluster size, the cluster is written in the next available space. The cluster will then receive a new cluster ID and the original cluster flagged as deleted. If the old cluster is deleted or the cluster ID is invalid,a “cluster does not exist” error is returned. Clusters may not be replaced in query mode else a mode (4) error is returned.

If an Old Key Values list is provided, the appropriate key-cluster associations are deleted. If any of the keys do not exist, a “cluster does not exist” (10) error is returned.

If a New Key Values list is provided, the appropriate key-cluster associations are created. Key values are specified in sorted order of key names. If a key is defined as unique and a key value already exists for that key, a “cluster already exists” error is returned. If NULL is specified as a key value it will be ignored. If an error is returned, the datafile will not be changed.

---

cluster-undelete

497

Input names: Table ID; Cluster ID; [Key Values]

Input types: integer; integer; [list]

Output types: integer

Description: Flags a deleted cluster as valid. The cluster is then accessible to other Prograph datafile primitives. If a cluster is already valid, then a “cluster already exists” error is returned. If the cluster ID does not point to a cluster, then a “cluster does not exist” (10) error is returned. Clusters may not be undeleted in query mode else a mode (4) error is returned.

If a Key Values list is provided, the appropriate key-cluster associations are restored. Key values are specified in sorted order of key names. If a key is defined as unique and a key value already exists for that key, a “cluster already exists” error is returned. If NULL is specified as a key value it is ignored.

Note: cluster-undelete is only guaranteed to undelete everything, including keys, if it is used immediately after   a call to the cluster-delete primitive.

cluster-unlock

498

Input types: integer; integer

Output types: integer

Description: Allows other applications using the data file to read, replace, or delete a previously locked cluster. Clusters may only be unlocked in share mode else a mode (4) error is returned.

Note: If unlocking a replaced cluster, unlock using the old cluster ID—not the new cluster ID.

cluster-write

498

Input names: Table ID; Cluster Value; [Key Values]

Input types: integer; any; [list]

Output types: integer; integer

Description: Writes a cluster into a table. The cluster is written at the next available space in the file. A mode error (4) will be returned if an attempt is made to write a cluster in query mode. If a Key Values list is provided, the appropriate key-cluster associations are created. Key values are specified in sorted order of key names. If a key is defined as unique and a key value already exists for that key, a “cluster already exists” error is returned. If NULL is specified as a key value it will be ignored. If an error is returned, the datafile will not be changed.

db-backup

499

Input names: Name; [Volume]

Input types: string; [integer]

Output types: integer

Description: Copies the specified datafile’s files. These copies will be prefixed with ‘Copy of’ and truncated to 31 characters. The volume reference number is optional. All paths to the datafile must be closed.

db-close

500

Input types: integer

Output types: integer

Description: Closes an open datafile. db-close will also close any table or key opened using the specified Database ID. If the specified datafile path is the last path open on the datafile, the file will also be closed.

db-compact

500

Input types: integer

Output types: integer

Description: Removes space occupied by deleted clusters. Because compaction relocates clusters, it may change cluster IDs. Compaction should never be performed on datafiles that use cluster IDs for long term cluster access (as in a hard-link datafile).

Note : db-compact uses a scratch file during compaction. Compaction requires free space equal to the size of the key file.

db-delete

501

Input names: Name; [Volume]

Input types: string; [integer]

Output types: integer

Description: Deletes the files associated with the specified datafile. The volume reference number is optional. All paths to the datafile must be closed since they will no longer be valid.

db-flush

501

Input types: integer

Output types: integer

Description: Writes out all buffered data (data stored in RAM but not yet written out to disk) to an open datafile. Periodic flushing will improve the recoverability of a datafile after a system crash. (This is only necessary in update mode—share mode always flushes between operations.)

db-get-flush

501

Input types: integer

Output types: integer; boolean

Description: Returns the auto-flush setting of the datafile.

db-info

502

Input types: integer

Output types: integer; integer; integer; integer; integer; integer

Description: Returns statistical information about an open datafile. Size is the total number of bytes in the datafile. ValidCount is the number of valid clusters. ValidSize is the number of bytes used in valid clusters. DeleteCount is the number of deleted clusters. DeleteSize is the number of bytes used in deleted clusters. Since Size also includes header, map and key information, ValidSize plus DeleteSize will not equal Size.

db-list

502

Output types: integer; list

Description: Returns a list of names of all open datafiles for an application. (The order is the order of opening, with oldest first.)

db-new

503

Input names:
Name; [Volume; [Creator ; File Type]]

Input types: string; [integer; [integer; integer]]

Output types: integer; integer

Description: Creates a new datafile and opens it in update mode. The specified name (maximum of 26 characters) will be the name of the datafile (and of its associated data file). A volume reference number, creator type and file type may be provided. db-new returns a datafile path reference identifier to be used by other Prograph datafile primitives.

db-open

504

Input names:
Mode; Name; [Volume]

Input types: string; string; [integer]

Output types: integer; integer

Description: Opens an existing datafile. Mode must be either “query”, “update” or “share”. Name is the name of the datafile. Volume reference number is optional. Each new call will generate a different datafile path, so that an application can perform independent operations on the same datafile. Each open path on a datafile must be in the same mode.

db-rename

504

Input names:
OldName; NewName; [Volume]

Input types:
string; string; [integer]

Output types:
integer

Description: Renames the datafile from OldName to NewName. The volume reference number is optional. All paths to the datafile should be closed.

db-set-flush

505

Input types: integer; boolean

Output types: integer

Description: Turns auto-flush on and off for that datafile. When auto-flush is on, the datafile and its key file are written to disk after every change.

db-shutdown

505

Description: Performs a blanket close of all open keys, tables and datafiles. All space for internal tables is released until the next datafile is opened.

db-wait

505

Input types: integer

Description: Allows the user to specify how long to wait for a busy shared file to be released (5 second default). This is referred to as time out in some contexts.

key-close

505

Input types: integer

Output types: integer

Description: Closes an open key.

key-delete

506

Input types: integer; string

Output types: integer

Description: Deletes a key from a table. Key-cluster associations for the specified key are discarded. Keys may only be deleted in update mode. If the key does not exist, or the datafile is not opened in update mode, a non-zero error number will be returned.

NOTE: Key deletions do not take effect until the table is reopened.

key-find

506

Input types: integer; simple

Output types: integer

Description: Positions a key path at the first cluster greater than or equal to the key value. If the cluster is not equal to the key value, a “cluster not found” (10) error is returned.

key-first

506

Input types: integer

Output types: integer

Description: Positions the key path to the least cluster in the table. A “cluster not found” (10) error is returned if the table is empty.

key-info

507

Input types: integer

Output types: integer; list

Description: Returns the list of option strings associated with an open key.

key-last

507

Input types: integer

Output types: integer

Description: Positions the key path to the greatest cluster in the table. A “cluster not found” (10) error is returned if the table is empty.

key-list

507

Input types: integer

Output types: integer; list

Description: Returns a sorted list of names of all keys in the specified table.

key-new

508

Input names: Table ID; Name; [Options]

Input types: integer; string; [list]

Output types: integer; integer

Description: Creates and opens a new key. You supply an open table path and the name (maximum 127 characters) of the new key. Provide a list of option strings if you wish to use key options other than the defaults. key-new returns a key path reference identifier. Keys may only be created in update mode. If the key already exists, or the datafile is not opened in update mode, an   error (7 or 4 respectively) is returned.

NOTE: New keys do not take effect until the table is reopened.

key-next

508

Input types: integer

Output types: integer

Description: Positions the key path to the next cluster in the table. A “cluster not found” (10) error is returned if the end of the table is reached.

key-open

509

Input types: integer; string

Output types: integer; integer

Description: Opens an existing key, given an open table path and the name of the existing key. key-open returns a key path reference identifier. Each call will return a new path. The key path is initially positioned at the least cluster (first cluster in key’s ordering) in the table. If the key doesn’t exist an error (8) is returned.

key-previous

509

Input types: integer

Output types: integer

Description: Positions the key path to the previous cluster in the table. A “cluster not found” (10) error is returned if the beginning of the table is reached. The key can not be positioned prior to the first cluster of the table.

key-read

510

Input names: Key ID; [Key Value]

Input types: integer; [simple]

Output types: integer; any

Description: If key value is not specified, the cluster pointed to by the key path is returned. If key value is specified, key-read reads the first cluster having a key equal to the key value. A “no cluster exists” error is returned if there is no such cluster. The key path is positioned at the first cluster greater than or equal to the key value. “Name in class map” error may be returned.

key-rename

510

Input types:
integer; string; string

Output types: integer

Description: Changes the name of a key. Keys may only be renamed in update mode. “Key does not exist”, “key already exists”, and mode errors may be returned.
NOTE: A renamed key does not take effect until the table is reopened.

key-value

511

Input types: integer

Output types: integer; simple; integer

Description: Returns the current key value and cluster ID of the cluster pointed to by the specified key path. A “cluster not found” (10) error is returned if the key points off the end of the table.

table-close

511

Input types: integer

Output types: integer

Description: Closes an open table. table-close will also close any keys opened using the specified Table ID.

table-delete

512

Input types:
integer; string

Output types: integer

Description: Deletes a table from a datafile. table-delete also deletes all clusters in the table and the set of keys associated with the table. Tables may only be deleted in update mode. All open paths to the table must be closed. If the table does not exist or the datafile mode is not update, a non-zero error number will be returned.

table-export

512

Input names: Key ID; Name; [Volume]

Input types: integer; string; [integer]

Output types: integer; integer; integer

Description: Exports data to a text file so it can be used by a non-Prograph application. Name (with optional volume reference number) is the name of a text file containing records separated by carriage returns. Only lists and instances can be written as records. Each record will contain several fields separated by tabs. The values of attributes can only be boolean, integer, real or string. The records are written out in the order of the specified key.

GoodCount is the number of acceptable records. BadCount is the number of rejected records.

table-import

513

Input names: Table ID; ColumnList; Name; [Volume]

Input types: integer; list; string; [integer]

Output types: integer; integer; integer

Description: Imports data from a non-Prograph application into a Prograph datafile table. Name (with optional volume reference number) is the name of a text file containing records separated by carriage returns, with each record containing several fields separated by tabs. The values of fields can only be boolean, integer, real or string. The field size is limited to 255 characters.

If ColumnList is NULL, each record is read in as an instance of the class whose name is the same as the table. The keys are determined by matching the key names with the names of attributes of the class.

If ColumnList is non-NULL, it must be a list of integers corresponding to columns (fields) containing the key values (keys are given in sorted order of key names). Each record is read as a list. For example, if there are four fields in the records being read in, and fields 1 and 3 have keys “ID” and “Artist” respectively associated with them, the value of ColumnList should be (3 1)

GoodCount is the number of records that were acceptable. BadCount is the number of records that were rejected.

Importing into a table such that each cluster will be an instance of class videos:

Importing into a table such that each cluster will be a list, with elements of the list corresponding to fields of the table being imported:

table-info

514

Input types: integer

Output types: integer; integer

Description: Returns the number of valid clusters stored in the specified table.

table-list

514

Input types: integer

Output types: integer; list

Description: Returns a list of names of all the tables in the specified datafile, in sorted order.

table-new

515

Input types: integer; string

Output types: integer; integer

Description: Creates and opens a new table of clusters. The user supplies an open datafile path and the name (maximum of 127 characters) of the new table. table-new returns a table path identifier. Tables may only be created in update mode. If the table already exists, or if the datafile open mode is not update, a non-zero error number will be returned.

table-open

515

Input types: integer; string

Output types: integer; integer

Description: Opens an existing table of clusters. The user supplies an open datafile path and the name of the existing table. table-open returns a table path identifier. Each call will open a new path. If a table of the given name does not exist, a non-zero error number will be returned.

table-rename

516

Input types: integer; string; string

Output types: integer

Description: Changes the name of a table. Tables may only be renamed in update mode. All open paths to the table must be closed. If the table does not exist or the database mode is not update, a non-zero error number will be returned.