CLOUD/INTERNET

tcat_import

catalogId	
id	
state	// pending-create, pending-upload, pending-repair, complete
md5	// md5 of the file when created (not updated)
sourceServer	// root for the path (server or c:\)
sourcePath	// relatative to the source server
uploadData	// the date it was uploaded to the server
source	// name of the client that imported this file

populated when the media is initially imported (either through migration, media import workflow -- either from a specific location, or in-place from the workgroup cache), or when a new version item is created.

every item in the catalog should have a corresponding import item. if not present, repair will synthesize one using the workgroup cache as the source of the import (as if it was an in place media import)

tcat_media

catalogId	
id	
virtualPath	// non-unique relative path from the import. this is used as a hint during // caching to determine where in the cache to store the file. items can be repathed.
mimeType	
state	// pending, active, unknown
md5	// pending: this is is the md5 from the client that owns uploading it // active: this is the md5 for the item in Azure storage.

-memberName

tcat_workgroups

catalog_id	
id	
name	
serverPath	// the root path to the workgroup (either UNC path // or X:\ for a single user workgroup)
cacheRoot	// path to the root of the workgroup cache

this table defines every workgroup connected with a catalog. if 2 clients are on the same LAN then they should share the same workgroup (so they can share the same workgroup cache)

its also possible for a client to have a "single user" workgroup that is rooted to a local fixed drive (e.g. c:\)

SQL Catalog Database

SQL Database accessible to all clients, usually in Azure SQL

Blob storage storing master copy of media. Each media item has a property storing the MD5.

(The MD5 from Azure storage can't be trusted since it isn't calculate on blob items larger than a threshold, hence we store our own MD5 calculation).

Each item is stored simply as GUID (no extension)

AZURE Storage

WORKGROUP/LOCAL LAN

tcat_workgroup_clients

id	// unique ID for the workgroup client
name	// name for the client
vectorClock	// the vector clock this entry is current as of

this is local to every workgroup and defines each client in the workgroup. this is used to differentiate the ownership of different workgroup tasks (like importing media).

the vector clock value represents the vector clock last used when a change was made to workgroup media FROM THIS CLIENT (when the workgroup media is updated, the vector clock is updated on the workgroup client as well).

tcat_workgroup_vectorclock

clock	// the clock name (workgroup-clock is the only clock)
value	// the monotonically increasing vector clock for the entire workgroup

The workgroup vector clock allows sequencing of operations on the workgroup database. Every change to the workgroup media table will have a vector clock value, and that value is guaranteed to not be shared between two different clients. In this way, we always know which operation "happened first".

tcat_workgroup_media

media	// id for the media item

path	// the path to the full fidelity cache of the media item. relative to the
	// cache root
	// (for items that haven't been uploaded yet, or that are marked "don't
	// upload to cloud" this might be the only source of truth)

cacheBy	// id of the workgroup client that cached this value (either downloaded // from azure, or pre-cached during the import operation
cachedDate	
vectorClock	// the workgroup vector clock when the database was updated for this item
md5	// the md5 for the item in the workgroup cache. this might be different // from the media item (see local changes and/or server changes)

this table has information for every media item the local workgroup knows about. It might be incomplete if the local client needs to download new media items that were added to the catalog

Staking a claim: when a workgroup first notices items missing in the workgroup database, it will add them to the workgroup database marked with the client name and then it will try to download the media. (DETAILS??) once the media is downloaded, the workgroup DB is again updated to reflect the completion. (DETAILS?? how do we know its downloaded and not still pending). We know which client owns in-progress downloads (even after a restart) because the client name is in the cachedBy field.

Local changes: if we detect that the local version of a file has changed (because the MD5 value in the workgroup matches the media server, but doesn't match the local file), we have to upload a new version of the file to the server. See SCAN FOR CHANGES -- need to detail here.

Remote changes: if we detect that the local MD5 of the file matches the workgroup MD5, but those values don't match the catalog's MD5, then we have a new version of the media to download from the server. Treat this like a pending download -- stake the claim, download the file, and mark as downloaded

Workgroup Database

Each workgroup has its own database, stored at the root of the workgroup cache location.
Every workgroup client on the same LAN shares the same Workgroup database, so extra care must be taken for concurrency situations.

LOCAL CLIENT

tcat_md5cache

path	// full path to file (lowercase)
lastModified	// UTC date-time of the last modification date/time
size	// size in bytes
md5	// base64 encoded string representing the md5 hash

caches MD5 hash values for files that the local client has seen. They should always be validated against the last modified time and size in the filesystem. (NOTE: the filesystem has higher precision than Date/Time in SQLite, so be sure to discount fractional seconds when compairing)

NOTE there is no media ID in this table. This cache is global across all catalogs/workgroups that the local client knows about

tcat_derivatives

media	// ID for the media item

mimeType	// the mimeType for this derivative
	// (might not match the media mimeType,
	// especially during transcoding)

scaleFactor	// double precision floating number representing
	// the scale factor from the original media's
	// dimensions

transformationsKey	// "-" separated set of tranformations of the form // GUID@VALUE

every client caches "derivatives" that can be used while running the client. The workgroup cache has to truth about the full-fidelity media, but that might not be in a format that is easy to display or is performant (like JP2)

as each client encounters media in the workgroup, it will determine if it needs a derivate -- for example, if we are previewing the media in the explorer window, then we will ask for a resampled derivative that will be much smaller and faster to load. Likewise, if the media is not in a convenient format (like PSD or JP2), then we will call a costlier resample function to get a full fidelity resampled version.

All of these derivatives are store in the local clients temp path and this table stores all of these paths.

NOTE that each item has an MD5 that represents the MD5 of the media at the point the derivative was created. If the MD5 doesn't match, then a new derivative will be created and the table will be updated with the new MD5 value.

Client Database

Each client has its own database, stored at %appdata%\thetacat\client.db