

**Collaboard Architecture**

DATABASE OVERVIEW

|  |  |
| --- | --- |
| Title: | Collaboard Database Overview |
| Abstract: | This document describes the database, tables, and relations in the Collaboard project |
| Status\* | Approved |
| Document Owner: | IBV |
| Document Author: |  |
| \*Draft, In progress, Reviewed, Authorized, Approved, Completed | |

|  |  |  |
| --- | --- | --- |
| **Document Control** | | |
| Document Distribution: | Internal: | All dev Teams |
|  | External: |  |
| Document Authorization: | Authorizer(s): | GP Santopaolo |

|  |  |  |  |
| --- | --- | --- | --- |
| **Change History** | | | |
| Date | Ver | Changed By | Details |
| 18/05/2021 | 1.0 | Dennis Vroegop | Initial draft |
| 19/05/2021 | 1.1 | GP Santopaolo | Review and approval |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Index

[Introduction 3](#_Toc72335381)

[Structure 3](#_Toc72335382)

[Main system 3](#_Toc72335383)

[Projects 4](#_Toc72335384)

[Database overview 4](#_Toc72335385)

[Description 4](#_Toc72335386)

[Tilestatus 5](#_Toc72335387)

[Database overview 5](#_Toc72335388)

[Description 5](#_Toc72335389)

[Templates 6](#_Toc72335390)

[Database overview 6](#_Toc72335391)

[Description 6](#_Toc72335392)

[Users in projects 7](#_Toc72335393)

[Database overview 7](#_Toc72335394)

[Description 7](#_Toc72335395)

[Conversions 8](#_Toc72335396)

[Database overview 8](#_Toc72335397)

[Description 8](#_Toc72335398)

[Authentication 10](#_Toc72335399)

[Database overview 10](#_Toc72335400)

[Description 10](#_Toc72335401)

[Licensing 11](#_Toc72335402)

[Database overview 11](#_Toc72335403)

[Description 12](#_Toc72335404)

[State Machines 13](#_Toc72335405)

[Database overview 13](#_Toc72335406)

[Description 13](#_Toc72335407)

[MFT 15](#_Toc72335408)

[Database overview 15](#_Toc72335409)

[Description 15](#_Toc72335410)

# Introduction

Collaboard runs on a SQL Server database. This database is the heart of the system; it holds all user data and control data.

This document describes the general overview and the architecture of the system.

For detailed information about the objects in the database and their structure, we have an additional document that holds all the information about that.

# Structure

We divided the system into 10 subsystems. Some of these subsystems have a seperate schema, and others share the main dbo schema.

The subsystems are:

* Main system
* Projects
* TileStatus
* Templates
* Users in projects
* Conversions
* Authentication
* Licensing
* State Machines
* MFT

# Main system

The main system is located in the DBO schema. Main is where all the main components are, and all the helper tables, stored procedures, and functions are.

The most important table not mentioned in any of the following parts, is the t\_configuration table.

This contains all configurable parameters, such as the location of the blobs, the number of processors to use for each background processes, and so on.

This configuration data is cached in the system. We use the t\_cache table to keep track of the age of the data so we can determine when to refresh.

# Projects

A project is a collection of tiles. One user creates a project, but there can be multiple collaborators on a project.

## Database overview



## Description

A project has all the information about a project in Collaboard. This contains in the t\_project table information about the user who created it, the location where all blobs are stored, information about the zoom factor, the thumbnail, the backgroundcolor, and some housekeeping data such as the current system version (currently set to version 2).

Related to t\_project is information about all users currently active in that project (t\_online\_user). Also, a project can have tags (t\_project\_tag) and quicklinks (t\_quicklink). Next to that, there is also a way to store all user id’s for all users who have access to this project, including their rights (t\_project\_participant).

The t\_tilestatus contains all information about all tiles in this project.

A project can be transformed into a template. This template will then not be useable as a project to the user anymore but can be instantiated. Under the hood, it is still a project, so that information is stored in the same table.

All projects with projecttypeid 0 are user projects. All other types are templates.

# Tilestatus

The tilestatus contains all tiles for all projects.

## Database overview



## Description

T\_TileStatus is the heart of the system. All user-generated content is stored here. We have several types of tiles, as indicitated by the table t\_tile\_type. Currently we support:

* Undefined
* Striky
* Image
* PDF
* Group
* Video
* Audio
* Stack
* YouTubeVideo
* Word
* Excel
* PowerPoint
* Text
* Shape
* Thumbnail
* Ink
* Embed

Ink tiles have strokes and that is stored in a separate table (t\_tilestatus\_stroke). Tiles can have relations to one and another. These relations (t\_tile\_relation) have startpoints and endpoints, and these can be drawn in different styles. The same goes for the linetype. These are all available in the t\_relation\_\* tables.

The BlobStatus determines if a tile has blobs, if the tile is deleted and if the blob has already been uploaded and processed. The available types are to be found in t\_blob\_status.

# Templates

Templates are pre-defined projects a user can use to quickly start a project.

## Database overview



## Description

Templates are special types of projects. They can be found alongside normal projects in the t\_project table. Their projecttype id must be higher than 0 to be seen as a template.

The projecttypeid is a foreign key to the t\_project\_type table. This holds all categories under which a template can reside. These descriptions and names can be translated into other languages, hence the t\_project\_type\_translation table. The same goes for the description of the project: for a template the translated description is stored in t\_template\_translation.

# Users in projects

We store users in two locations: one in the authentication/licensing system, and one in the main dbo schema. The last one is used in the system itself to show who is online and what the user is doing. This is what we store in “users in projects”.

## Database overview



## Description

When a user registers their details are stored in t\_user.

Users can be part of a project because they created it, or because they were invited. Either way, their data is stored in t\_project\_participant. We also store their permissions here (read, write, readwrite, admin).

Whenever a user enters a project we store that information in t\_online\_user. This is used to show other users who is online in that project. This is also used to determine if a user is disconnected without logging off: there will be an entry in this table without a corresponding active connection to the client.

# Conversions

Collaboard supports the automatic conversion of files to images, and also the automatic resizing of images. The resizing happens to both user-uploaded images as the generated images from the document conversion.

All data for the conversion state machine is in the [conversion] schema.

## Database overview



## Description

When a user uploads a document or an image, an entry is made in either the t\_conversion\_booking table or in the t\_image\_resizing\_booking table. This is picked up by our statemachines (more on the working of that can be found at State Machines )

These two tables contain all the information the engine needs to do it’s job.

All supported documents we can convert are stored in t\_convertable\_file\_type. Currently we support:

* Word document (.doc, .docx)
* Excel document (.xls, .xlsx)
* PowerPoint document (.ppt, .pptx)
* PDF file (.pdf)

The current status of the state machine is stored in the table, and referenced by t\_state\_machine\_status. This can be:

* Not ready
* Ready
* Working
* Finished
* Error

After a pre-determined amount of time, the data in the booking tables are moved to the history tables so we can have a smaller working table but still maintain a history of what has happened.

# Authentication

When users log in, their credentials are validated against our authentication system. This is all happening in the [auth] schema.

## Database overview



## Description

The t\_user table contains all information we have about the user. We store this information here and we use this to validate the log in.

Users can have roles, currently we support:

* LicensingManager
* UserManager
* CollaboardManager

If a user does not have one of these roles, they are considered to be a regular user.

# Licensing

Licensing controls all license related work. This is using the [licensing] schema.

## Database overview



## Description

We have separate ways of licensing our product. Most of the work is done around the t\_subscription tables, but we also support locally installed licensing files.

All information about pricing (t\_price, t\_pricelist, and t\_promo) is here as well.

All orders, paymens and payment methods are handled through the t\_order, t\_orderline, t\_payment and t\_payment\_method tables.

Users can have a subscription based on user or on device. This last one is only used in the original “on device” version of our system and doesn’t apply to the online version. The data structures are still here though.

# State Machines

Most of the work is done in background worker jobs. We do this to offload the burden of the webserver and this enables us to scale up the system if needed.

We have processes running that poll the different \*\_booking tables. If an entry is made there, the status will change from ready to working. The background process now gets all the data from that table and does it’s job.

In case of a failure the status is set back to “ready” and the retry is increased. If the retry is exceeding the maximum number allowed for this job, the entry in \_booking is marked as failed.

We copy most data from the \_booking tables to a \_booking\_history table to keep the working tables sizes small enough to ensure fast handling of the work.

## Database overview



## Description

We currently have the following state machines in our system

* Tilestatus\_copy. This takes care of copying tiles and if necessary, the associated blobs. This can be done in a project and between projects.
* Project\_Copy. This handles copying projects. This also takes care of instantiating a template. Internally this calls tilestatus\_copy
* CanvasShotter\_copy. This is the mechanism that generated the thumbnails for the projects and templates that can be seen on the project overview screen in Collaboard.
* Conversion. This is also a statemachine, but is explained in Conversions
* MFT. MFT is what takes care of the blobs in non-Azure based installations. This is also based on statemachines, but for details we refer to the section about MFT

# MFT

MFT is the system that stores and retrieves files. Examples of this are uploaded Office files and all images in the project.

For Azure based installations, we do not use this. For all other systems, MFT forms the backbone of all image and file contents.

## Database overview



## Description

MFT is based on statemachines. We currently have three:

* Upload based
* Download based
* File Operations

All files that are being uploaded, are being uploaded in chunks. The size of these chunks is stored in the configuration. Once all chunks have been uploaded , the t\_mergemachine statemachine kicks in and merges all these chunks into one file.

For downloads, the reverse happens. When a request comes in, the system puts an entry in the t\_chunkingmachine table. MFT will then chop up the requested file and all parts will then be transmitted

We also have the storage operations state machine. This is what is used to copy a file, rename a file, create a folder for a project and reads a list of available files for a project.

We keep a history of all activities in the \_history tables, to make sure we can have the working table as small and fast as possible.