Database

Table of contents

[I. Structure 2](#_Toc280098937)

[II. Accessors and Objects 3](#_Toc280098938)

[III. Fields Created and modified 3](#_Toc280098939)

[IV. Uniqueness 3](#_Toc280098940)

[V. Abstraction 3](#_Toc280098941)

[VI. Triggers 4](#_Toc280098942)

[1. Modification 4](#_Toc280098943)

[2. Delete 4](#_Toc280098944)

[3. Foreign keys 4](#_Toc280098945)

[VII. Tables 5](#_Toc280098946)

[1. Deleted 5](#_Toc280098947)

[2. Accounts 5](#_Toc280098948)

[3. Groups 5](#_Toc280098949)

[4. Limits 5](#_Toc280098950)

[5. Events 5](#_Toc280098951)

[6. Permissions 6](#_Toc280098952)

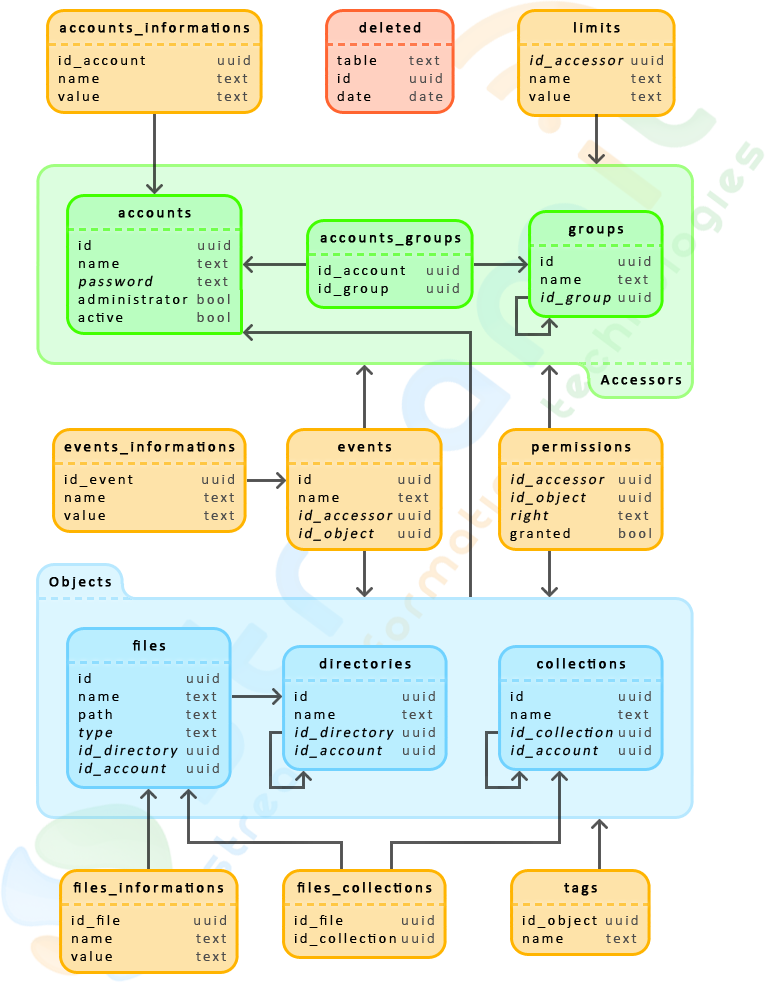
[7. Files 6](#_Toc280098953)

[8. Directories 7](#_Toc280098954)

[9. Collections 7](#_Toc280098955)

[10. Tags 7](#_Toc280098956)

# Structure

The diagram below shows the database structure:

To simplify the diagram, we don’t represent fields “Created” and “modified” available on every table except the table “deleted”. Similarly, all tables except “deleted” contains a field “id”, but only the id used by other tables, that is to say, which are used by foreign keys are shown in this diagram.  
The table names and fields are in lowercase. Field values in italics can be empty, but any field value can be NULL.

# Accessors and Objects

The term "**accessor**" includes tables “**accounts”** and “**directories”**. Similarly, the term "**object**" includes tables “**files**”, “**directories”**, and “**collections”**. The general idea of this representation is that accessor can perform operations on an object.

This system is made possible by the systematic use of Universal Unique Identifier (UUID) to identify record. Each accessor and object has a unique identifier in the database.

All objects have a field “**id\_account”** representing the owner if not empty.

# Fields Created and modified

Fields “Created” and “modified” present in every table are usefull to know when a data is created and modified. Their date format is yyyy-MM-dd hh: mm: ss. These fields are automatically initialized when adding a data, and the modified field is updated with each change by a trigger.

# Uniqueness

Some tables are subject to unique constraints more or less restrictive, on one or more fields. For example, the field “name” of the “accounts “ table is unique, which means that all users have a different name.  
Another example is the “files” table, in which two inputs can not have the same name and id\_directory in the same time. This means that two files cannot have the same name if they are in the same folder.

# Abstraction

The Server API provides an abstraction of the database, which allows easy access to fields of a record, and then to make request more or less complex.

# Triggers

## Modification

When a record is modified by the UPDATE clause, the field “**modified**” is automatically updated to the current date.

A trigger prohibits to change the **id** of a record. The **uuid** is finally defined when creating the record, thus avoiding to have foreign keys that point to nothing.

Then, a trigger checks that fields that are not in italics in the diagram are not empty.

## Delete

When a record is deleted, it’s id, it’s table, and the deletion time are saved in the “**deleted”** table.  
Then, the trigger makes a delete cascade, that’s mean all record depending on the first record deleted will be deleted. For example, deleting a file will also delete entries **files\_informations, files\_collections, tags, permissions, and events** of the according file.

## Foreign keys

Foreign keys are simulated by using triggers that check that keys are still pointing to valid i. For example, when a record is inserted or updated in the table” **tag”s**, a trigger checks the value of the field “ **id\_object**” is pointing to a existing id object, that’s means the id of a file , folder, or collection. If this is not the case, an error is generated, and the request is cancelled.

# Tables

## Deleted

The table “deleted” tracks each deletion on the database. When a record is deleted, its id is inserted into the table. This operation is make by using specific triggers.

|  |  |
| --- | --- |
| **Fields** | |
| **Table** | Contains the name of the table whose record has been deleted. |
| **Id** | Id of the deleted record. |
| **Date** | Date of the deletion record. |

## Accounts

This table contains accounts stored on the server. The table contains “**accounts\_informations”** contains personal information (address, phone, language…).

|  |  |
| --- | --- |
| **Fields** | |
| **Name** | Account name |
| **Password** | The password is stored as a SHA1. If empty, it means that the account is public, so anyone can connect to it. |
| **Administrator** | Define if the account is an administrator. |
| **Active** | Define if the account is enable. |

## Groups

A group contains multiple accounts to facilitate their administration. Accounts are linked to groups through the table “**accounts\_groups”**. An account can belong to several groups, but also to any one, and a group can be contained in another group.

|  |  |
| --- | --- |
| **Fields** | |
| **Name** | Group name |
| **Id\_group** | Id parent group. May be empty if the group is at the root, that is to say if it has no parent. |

## Limits

This table is used to set limits to a group or an account. For example, it can be used to set a limit of downloading file that accessor can download in a day / month.

## Events

This table contains happening event on the server. Field “**created**”is used to know the date of the event occurred.Table “**events\_informations”** contains additional information about the event.

|  |  |
| --- | --- |
| **Fields** | |
| **Name** | Event name |
| **Id\_accessor** | Accessor id associated to the event. Can be empty if the event has no accessor. |
| **Id\_object** | Object id associated to the event. Can be empty if the event has no object. |

## Permissions

This table manages the system server rights. The general concept is to link an accessor to object, and associate a right that may be granted or denied.

|  |  |
| --- | --- |
| **Fields** | |
| **Id\_accessor** | Accessor on which the related permission. If the field is not filled, the permission will be apply on everybody. |
| **Id\_object** | Object referred by permission. If the object is not specified, permission will focus on the root server. Object visé par la permission. |
| **Right** | Type of right. If the right is not defined, all rights are concerned. |
| **Granted** | Define if the permission is granted (**true**) or denied (**false**). |

The permissions system is described more specifically in the server documentation.

## Files

This table represents a file on the server. Table “**Files\_informations »**  contains information about a file.

|  |  |
| --- | --- |
| **Fields** | |
| **Name** | Nom du fichier tel qu’il est affiché à l’utilisateur. Ce nom peut être différent de son nom réel. |
| **Path** | Chemin vers l’emplacement du fichier sur le disque dur. Peut être relatif ou absolu. Si le chemin est relatif, son point de départ est le dossier « **filesPath** » du serveur. |
| **Type** | Le type du fichier. Les valeurs possibles sont **image**, **video**, **audio**, **document**, ou **other**. Si ce champ vaut une autre valeur ou qu’il est vide, cela correspond à **other**. |
| **Id\_directory** | Dossier dans lequel se trouve le fichier. S’il est vide, le fichier se trouve à la racine du serveur. |
| **Id\_account** | Identifiant du propriétaire du fichier. Le propriétaire est le compte qui a ajouté le fichier. |

Examples of data that can be contained in your table “**files\_informations”**. Their presence and their number depends on the type of file, and plugins installed on the server.

|  |  |
| --- | --- |
|  | |
| **Size** | File size in bytes. |
| **Mime** | The MIME type of the file. |
| **Extension** | It  extension. |
| **Width** | The width of the image or video. |
| **Height** | The height of the image or video. |
| **Format** | File format |
| **Duration** | The duration of video or audio in seconds. |
| **Title** | Title of the music or film. |

## Directories

This table represents a folder. Folder doesn’t correspond to real folders present on the hard drive of the server. They can store files or other folders.

|  |  |
| --- | --- |
| **Fields** | |
| **Name** | Folder name |
| **Id\_directory** | Id parent folder. Can be empty if it has no parent. In this case, the file is at the root of the server. |
| **Id\_accessor** | Id folder owner. The owner is the account that has created the file. |

## Collections

Collections are lists of files. They can be compared to playlists. The table “**files\_collections”** link files to collections. A file can be associated to several collections.

|  |  |
| --- | --- |
| **Fields** | |
| **Name** | Name of the collection. Must be unique within its parent's son. |
| **Id\_collection** | Id of the parent collection. Can be NULL if it has no parent. |
| **Id\_accessor** | Id folder owner. The owner is the account that created the file. |

## Tags

A tag is information that may be associated to an object, this allows easy search. For example, a photo of forest can have tags like "forest", "nature", or “wood".