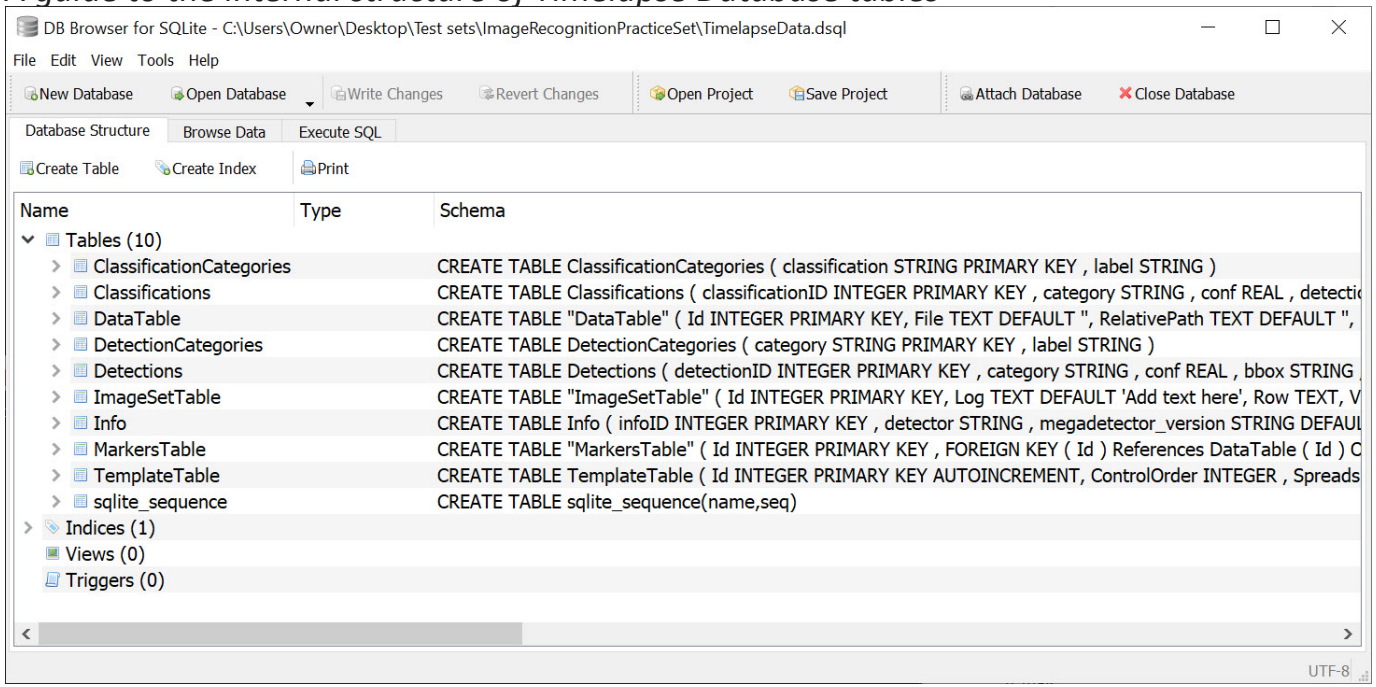
Timelapse Database Guide

A guide to the internal structure of Timelapse Database tables



Saul Greenberg
Greenberg Consulting Inc. / University of Calgary
saul@ucalgary.ca

Part 5 of the *Timelapse Manual Series*. Last updated December 23, 2023, Timelapse Version 2.3.0.9

Timelapse Database Guide

A guide to the internal structure of the Timelapse database tables[1]

This guide explains the internal structure of the various database tables found in the SQLite database files created by Timelapse.

This guide is only of interest if you want to access the data directly from the database rather than from an exported .csv file, and that you have the knowledge to do so. For example, the ***R statistical package*** has libraries that can be used to easily query SQLite databases, as explained in the final section of this guide.

Table of Contents

[1]What you see when you run Timelapse or other software to examine the database may not exactly match the screen images in this guide, due to software updates made after these screen images were taken. These differences should not affect your general understanding.

Introduction

Timelapse saves your data and other information in ***SQLite*** database files. If you want to access the database directly (rather than the exported spreadsheet), read on. Otherwise you can ignore this guide.

This guide explains the tables found in the two database files. Of those tables, the most important is the *DataTable* in the Timelapse database *.ddb* file, as it will contain all your tagging data. Other tables, while described here, are likely of little or no interest, although they are valuable for debugging. They are included only for completeness.

Timelapse relies on two database files.

- ***The Timelapse database*** (*.ddb* suffix) contains all your tag data, image recognition data (if any), as well as other data used internally by Timelapse. It is created when Timelapse loads a template for the first time. By default, that file is called *TimelapseData.ddb*, but it can be renamed to anything as long as it maintains the *.ddb* suffix. (The *ddb* suffix stands for *data database file*).
- ***The Timelapse Template database*** (*.tdb suffix*) contains the data defining the template. It is created using the ***Timelapse Template Editor***, and is read in by ***Timelapse***. By default, that file is called *TimelapseTemplate.tdb*, but it can be renamed to anything as long as it maintains the *.tdb* suffix. (The *tdb* suffix stands for *template database file*).

Most people export and process their data via a CSV file. However, you can directly access the Timelapse database files (and the data within it) using software of your choice. For example:

- The ***R statistical package*** is often used by knowledgeable people to access SQLite data bases to perform statistical analysis of that data.
- ***Popular programming languages*** often include extensions or libraries that can access SQLite databases. If you are code-savvy, this gives you flexibility to do whatever you want.
- ***SQLite database viewers***. There are myriads of free tools available that will let you view SQLite database files, query them, and even edit their structure and contents. These are handy for inspecting and modifying the database table structure and the values contained within them. Examples include:
  - » DB Browser http://sqlitebrowser.org/
  - » SQLite Administrator http://sqliteadmin.orbmu2k.de/

Be aware that altering the database files can compromise Timelapse's ability to read those files if it deviates from Timelapse expectations. Problematic alterations cover changing table schema, adding or deleting columns, and changing data to unexpected formats. Make sure to back up your database files before you do any modifications.

Why SQLite?

SQLite is a small, fast, self-contained, high-reliability, full-featured, SQL database engine. Its web site says it is the most used database engine in the world, where it is built into all mobile phones, comes bundled inside countless other applications that people use every day, and is often the engine behind many web sites. Of particular value is that SQLite can be embedded into other software.

Timelapse includes the SQLite database engine, where everything is self-contained in the Timelapse software folder. This means that when you download Timelapse, you are also downloading SQLite onto your machine.

Positives

- As SQLite is installed as part of Timelapse on your local machine, you can run Timelapse (and the database) without an Internet connection. This is particularly valuable when working in the field.
- Everything is portable. You can move Timelapse software (and your images) from machine to machine, and it should all work. No extra configuration is needed
- Unlike most other databases, you don't need a systems person to install or configure SQLite.
- In most cases, the software will run fine even on locked down machines.
- SQLite architecture is a good fit for the data requirements of most tagging needs.

Negatives

- In practice, the SQLite database is reasonably fast when storing and accessing data for up to approximately a million or so images. It does slow down somewhat above that, but is still workable. For extremely large image sets, you may want to divide your work into smaller chunks, each defining its own TImelapse database. You can always merge these databases afterwards using the Timelapse ***File | Merge databases...*** facility.
- The Timelapse / SQLite architecture is not configured to run as a central server, e.g., as a database accessed through the cloud. However, there are options to make this work, including:
  - » locating your database files and images on network server,
  - » running virtual machines, where users log onto them to do their work.
- Even when located on a central system, SQLite is less suited for multiple people simultaneously tagging overlapping sets of images. Essentially, SQLite is not as robust as industrial database engines at handling conflicts that can occur when people simultaneously write to the database. It can still work, but you have to be somewhat more disciplined. A better strategy is to create independent subsets of images and database files, and assign those to different people to minimize overlap. See the Timelapse Reference Guide for suggestions.

The DataTable

The ***DataTable***, found in the Timelapse *.ddb* file, contains all the data entered by the analyzer. This likely makes this table the most important, and perhaps the only, database table of interest to a Timelapse user who wishes to directly access data.

The figure below illustrates an example DataTable as held by the database. Each row is a record, uniquely identified by an integer ***Id***. The ***Id*** is set by the database engine, where its value is incremented and assigned when images are loaded into Timelapse for the very first time. If the analyst deleted an image and its data, that row would no longer appear (i.e., the Id column would appear to skip a number).

Remaining columns correspond to the DataLabels specified in the Timelapse Template file. Columns corresponding to the required data fields listed in the Timelapse Template Editor are always present, even if they have their visibility set to invisible: ***File***, ***RelativePath***, ***DateTime*** and ***DeleteFlag***. All other columns are custom fields defined by the project manager when using the Timelapse Template Editor.

The example Data Table below illustrates its structure and contents after a user completed the exercises in the ***Timelapse QuickStart Guide***. The columns reflect the contents of the template provided in the *PracticeImageSet.*

The DataTable's schema is shown at the right. Even though most schema types are TEXT, Timelapse expects certain column data to be limited to specific values.

- ***Id*** values, set by the database engine, are positive integers.
- ***File*** and ***RelativePath*** values are combined to locate the file. ***File*** should be the file name of the image or video. ***RelativePath*** values should be the path from the root folder (which contains the template to the image). Looking at the first row of the example data table, the image file IMG_001.jpg is located relative to a root folder in the subfolder Station1\Fetched-2015-06. Files located directly in the root folder would have and empty RelativePath.
- ***DateTime*** values can only contain a date formated as yyyy-mm-dd hh:mm:ss (for example, 2015-05-27 18:01:53)
- ***Flag controls*** can only contain case-insensitive true or false values (e.g., the columns *Dark, Empty*, *Publicity*, and *DeleteFlag)*.
- ***Choice control*** values should match a ***Choice*** menu item as defined in the template (e.g., the ***Species*** column data must match bear, deer, etc.).
- ***Count control*** values are blank, 0 or a positive integer. (e.g., the Count column)
- ***Text control*** columns can contain any text (e.g., ***Analyst*** can contain any name).

DataTable		
Id	INTEGER	"Id" INTEGER
File	TEXT	"File" TEXT DEFAULT ''
RelativePath	TEXT	"RelativePath" TEXT DEFAULT ''
DateTime	DATETIME	"DateTime" DATETIME DEFAULT '1900-01-01 12:00:00'
Dark	TEXT	"Dark" TEXT DEFAULT 'false'
Empty	TEXT	"Empty" TEXT DEFAULT 'false'
Species	TEXT	"Species" TEXT DEFAULT ''
Count	TEXT	"Count" TEXT DEFAULT '0'
Sequence	TEXT	"Sequence" TEXT DEFAULT ''
Temperature	TEXT	"Temperature" TEXT DEFAULT ''
Problem	TEXT	"Problem" TEXT DEFAULT ''
Comment	TEXT	"Comment" TEXT DEFAULT 'None'
Analyst	TEXT	"Analyst" TEXT DEFAULT ''
Publicity	TEXT	"Publicity" TEXT DEFAULT 'false'
DeleteFlag	TEXT	"DeleteFlag" TEXT DEFAULT 'false'

	Id	File	RelativePath	DateTime	Dark	Empty	Species	Count	Sequence	Temperature	Problem	Comment	Analyst	Publicity	DeleteFlag
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	IMG_001.jpg	Station1\Fetched-2015-06	2015-05-27 18:01:53	false	false	bear	1	1:1|9	14			Saul	false	false
2	2	IMG_002.jpg	Station1\Fetched-2015-06	2015-05-27 18:01:54	false	false	bear	1	1:2|9	14			Saul	false	false
3	3	IMG_003.jpg	Station1\Fetched-2015-06	2015-05-27 18:01:55	false	false	bear	1	1:3|9	14			Saul	false	false
4	4	IMG_004.jpg	Station1\Fetched-2015-06	2015-05-27 18:01:58	false	false	bear	1	1:4|9	13			Saul	false	false
5	5	IMG_005.jpg	Station1\Fetched-2015-06	2015-05-27 18:01:59	false	false	bear	1	1:5|9	13			Saul	false	false
6	6	IMG_006.jpg	Station1\Fetched-2015-06	2015-05-27 18:02:00	false	false	bear	1	1:6|9	13			Saul	false	false
7	7	IMG_007.jpg	Station1\Fetched-2015-06	2015-05-27 18:02:02	false	false	bear	1	1:7|9	13			Saul	false	false
8	8	IMG_008.jpg	Station1\Fetched-2015-06	2015-05-27 18:02:03	false	false	bear	1	1:8|9	13			Saul	false	false
9	9	IMG_009.jpg	Station1\Fetched-2015-06	2015-05-27 18:02:04	false	false	bear	1	1:9|9	13			Saul	false	false
10	10	IMG_010.jpg	Station1\Fetched-2015-06	2015-05-30 18:38:15	false	true		0	2:1|3	20	wind triggered		Saul	false	false
11	11	IMG_011.jpg	Station1\Fetched-2015-06	2015-05-30 18:38:17	false	true		0	2:2|3	20	wind triggered		Saul	false	false
12	12	IMG_012.jpg	Station1\Fetched-2015-06	2015-05-30 18:38:18	false	true		0	2:3|3	20	wind triggered		Saul	false	false
13	13	IMG_013.jpg	Station1\Fetched-2015-06	2015-06-01 17:23:46	false	false	deer	1	3:1|6	19			Saul	false	false
14	14	IMG_014.jpg	Station1\Fetched-2015-06	2015-06-01 17:23:47	false	false	deer	1	3:2|6	19			Saul	false	false
15	15	IMG_015.jpg	Station1\Fetched-2015-06	2015-06-01 17:23:48	false	false	deer	1	3:3|6	19			Saul	false	false
16	16	IMG_016.jpg	Station1\Fetched-2015-06	2015-06-01 17:23:51	false	false	deer	1	3:4|6	19			Saul	false	false
17	17	IMG_017.jpg	Station1\Fetched-2015-06	2015-06-01 17:23:52	false	false	deer	1	3:5|6	19			Saul	false	false
18	18	IMG_018.jpg	Station1\Fetched-2015-06	2015-06-01 17:23:53	false	false	deer	1	3:6|6	19			Saul	false	false

The TemplateTable

The ***TemplateTable*** is found in both the Timelapse Template *.tdb* and the Timelapse *.ddb* file.

- The TemplateTable in the *.tdb* file is created or modified through the Timelapse Template Editor.
- When an image set is loaded into Timelapse for the first time, Timelapse creates its own copy of that template in the *.ddb* file. It then uses the TemplateTable as a specification for the data fields present in the user interface, and to define the DataTable schema.
- During subsequent loads of that image set, Timelapse compares the *.tdb* TemplateTable with the *.ddb* copy for differences, and tries to resolve those differences by displaying a dialog to the user.

While you will not normally access this table, it can be of interest if you want to retrieve (or modify) the information associated with each data field.

The example TemplateTable below illustrates the structure and contents of the template used in the ***Timelapse QuickStart*** guide, which in turn was included in the *PracticeImageSet*.

- ***ControlOrder*** specifies the order of controls in the Timelapse user interface.
- ***SpreadSheetOrder*** column specifies the order of columns when data is exported to a *.CSV* file, which in turn specifies how those columns appear when displayed in a spreadsheet.
- Other fields are as described in the Timelapse Template Guide.
- Of particular note is the ***List*** column, a *JSON* structure that specifies the contents of the ***Choice*** menu item. The structure contains a *IncludeEmptyChoice* boolean field indicating whether an empty item should be included in the menu, and ChoiceListNonEmpty list field containing text describing its menu items.

TemplateTable

Id	INTEGER	"Id" INTEGER
ControlOrder	INTEGER	"ControlOrder" INTEGER
SpreadsheetOrder	INTEGER	"SpreadsheetOrder" INTEGER
Type	TEXT	"Type" TEXT
DefaultValue	TEXT	"DefaultValue" TEXT
Label	TEXT	"Label" TEXT
DataLabel	TEXT	"DataLabel" TEXT
Tooltip	TEXT	"Tooltip" TEXT
TXTBOXWIDTH	TEXT	"TXTBOXWIDTH" TEXT
Copyable	TEXT	"Copyable" TEXT
Visible	TEXT	"Visible" TEXT
List	TEXT	"List" TEXT
ExportToCSV	Flag	"ExportToCSV" Flag DEFAULT 'true'

	Id	ControlOrder	SpreadsheetOrder	Type	DefaultValue	Label	DataLabel	Tooltip	TXTBOXWIDTH	Copyable	Visible	List	ExportToCSV
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	1	1	File		File	File	The file name	100	false	true		true
2	2	2	2	RelativePath		RelativePath	RelativePath	Path from the folder containin...	100	false	true		true
3	4	3	3	DateTime	1900-01-01 12:00:00	DateTime	DateTime	Date and time taken (Year-...	160	false	true		true
4	8	14	13	Flag	false	Dark?	Dark	True if the image is dark, usua...	20	false	true		false
5	9	13	14	DeleteFlag	false	Delete?	DeleteFlag	Mark a file as one to be delete...	20	false	true		false
6	10	5	5	FixedChoice		Species	Species	The species seen in the image	90	true	true	{"IncludeEmptyChoice":true,"ChoiceListNonEmpty":["bear","bighorn ...	true
7	11	6	6	Counter	0	Count	Count	The number of each species ...	30	true	true		true
8	13	7	7	Note		Sequence	Sequence	Position of this image in a ...	40	false	true		true
9	14	9	9	FixedChoice		Problem	Problem	A condition that makes it ...	80	true	true	{"IncludeEmptyChoice":true,"ChoiceListNonEmpty":["wind triggered","le...	true
10	15	12	12	Flag	false	Publicity?	Publicity	A really good image useful for ...	20	false	true		true
11	16	8	8	Note		Temperature	Temperature	The temperature in Celcius ...	30	false	true		true
12	17	4	4	Flag	false	Empty?	Empty	If no wildlife is pesent	20	true	true		true
13	18	11	11	Note		Analyst	Analyst	Person who analyzed this image	70	false	true		true
14	19	10	10	Note		Comment	Comment	Any comment you wish to add	100	true	true		true

The TemplateInfo Table

The ***TemplateInfo Table*** is found in only the Timelapse Template *.tdb* file. It contains a single row with a single field called ***VersionCompatability***, which records the last version of Timelapse used to open this template.

	VersionCompatabily
	Filter
1	2.3.0.0

TemplateInfo	CREATE TABLE TemplateInfo (VersionCompatabily TEXT DEFAULT '2.3.0.0')
VersionCompatabily TEXT	"VersionCompatabily" TEXT DEFAULT '2.3.0.0'

The ImageSetTable

The ***ImageSetTable*** is found in the Timelapse *.ddb* file. It stores internal information as used by Timelapse, primarily to store a few settings about using a particular image set, that in turn are used to restore state between sessions. This table are likely of little to no interest to you.

- ***Log*** : contents of the notes added through the Timelapse ***Edit | Edit Notes for this Image Set*** menu item.
- ***Row***: Indicates the Id of a row in the DataTable corresponding to the last image the user was viewing.
- ***VersionCompatability***: The last version of Timelapse used to open this database.
- ***SortTerms***: The last used criteria used to sort the images (via the ***Sort*** menu), stored as a JSON structure.
- ***QuickPasteXML***: used internally by Timelapse to save/restore QuickPaste information, stored in XML format.
- ***Root folder***: the name of the root folder containing the template.
- ***SearchTerms:*** the last used criteria used to select images (via the ***Select*** menu), stored as a JSON structure.
- ***BBDisplayThreshold***: The confidence threshold for displaying bounding boxes when image recognition is used.

ImageSetTable		
Id	INTEGER	"Id" INTEGER
Log	TEXT	"Log" TEXT DEFAULT 'Add text here'
Row	TEXT	"Row" TEXT
VersionCompatabily	TEXT	"VersionCompatabily" TEXT
SortTerms	TEXT	"SortTerms" TEXT
QuickPasteTerms	TEXT	"QuickPasteTerms" TEXT
RootFolder	TEXT	"RootFolder" TEXT DEFAULT "
SearchTerms	TEXT	"SearchTerms" TEXT DEFAULT '{}'
BBDisplayThreshold	REAL	"BBDisplayThreshold" REAL DEFAULT '-1'

	Id	Log	Row	VersionCompatabily	SortTerms	QuickPasteTerms	RootFolder	SearchTerms	BBDisplayThreshold
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	Add text here	4	2.3.0.0	[{ "DataLabel": "RelativePath",	[{ "Title": "Elk - 1", "Items": [{ "	PracticeImageSet-FilledIn	{ "SearchTerms": [{ "ControlTy	0.5

The MarkersTable

The ***MarkersTable*** is found in the Timelapse *.ddb* file. When Timelapse users use the ***Count*** visual marker capability, the positions of those markers are recorded within a JSON list structure as x,y ratios coordinate pairs that locate the marker relative to the image size. For example, a marker's position of 0.5, 0.5 would be in the center of the image. The Id is the Id of the record that has a marker associated with it, while the column names reflect the name of the Count's data label. A column exists for each Count data type included in the template. For example, if another template defined two counters with the data labels 'Counter1' and 'Counter2', we would see two columns with those names.

	Id	Count
	Filter	Filter
1	4	["0.4162,0.6134"]
2	5	["0.4903,0.5767"]
3	6	["0.5448,0.5767"]
4	7	["0.7571,0.5558"]
6	18	["0.7393,0.7663"]
7	19	["0.4379,0.6059","0.1365,0.39...
8	20	["0.454,0.5934","0.1003,0.384...
9	21	["0.4641,0.6059","0.0914,0.39...
10	22	["0.2056,0.5039"]

MarkersTable		
Id	INTEGER	"Id" INTEGER
Count	TEXT	"Count" TEXT DEFAULT "

Tables for Image Recognition

If image recognition is enabled and you have imported recognition data, Timelapse will create several additional tables to hold the recognition data, which in turn is used to select and display recognition data to the analyst. If you have not read in recognition data, these tables will be absent.

For the most part, the data in those tables mirrors what was read in from the JSON recognition file, albeit in a different format and with a few exceptions as indicated below. For specific information, you should review the Microsoft Metadetector specification for JSON files.

Although you could use these tables to access the recognition data, that data will likely be best exploited within the Timelapse software. Thus the various image recognition tables are likely of little interest to you.

DetectionCategories

The image recognition file contains an entry called ***detection_categories***, which broadly identifies what the recognizer thinks it has detected and assigns a unique integer to each category. Timelapse reads those values into the table (as illustrated below). Timelapse also adds a new category called 'Empty', which will be used to identify any images analyzed by the detector but which produced no detections. It is mostly used as a lookup table to correlate the category number with the human-readable label.

Table: DetectionCategories

	category	label
	Filter	Filter
1	0	Empty
2	2	person
3	1	animal

Detections

The image recognizer contains, for each image, a list of zero or more possible detections.

The ***Detections*** table holds each detection as a row. The ***detectionID*** column is the primary key. ***Id*** is the ID of the image in the ***DataTable***, and is used to link each detection to a single image i.e., it is a foreign key enabling a many to one relation between the ***Detections*** and the ***DataTable*** tables. Each detection identifies the detection category ***category*** used to look up the label in the ***DetectionsCategory*** table, a confidence value ***conf*** for that detection, a bounding box ***bbox*** of 4 coordinates identifying where in the image that detection is located (in relative terms).

For example, in the table below:

- detectionID 1 identifies a detection on image 1406 in the DataTable. As its category is 0 (Empty, as looked up on the DetectionsCategory table), it means that although that image was analyzed, no detections were identified for it. This is also why there are no bounding box coordinates.
- detectionID 5 identifies a detection on image 1409 in the DataTable. Its category is 1 (animal, as looked up on the DetectionsCategory table) with a confidence of 0.657. The coordinates are the bounding box around the animal. Each detection shows the confidence of that detection, and the coordinates of its bounding box.
- detectionID 6,7,8 identifies 3 other detections on image 1409,for a total of 4 detections (and boundingboxes) on that image. This reflects the many to one relationship.

Table: Detections

	detectionID	category	conf	bbox	Id
	Filter	Filter	Filter	Filter	Filter
1	1	0	0.0		1405
2	2	0	0.0		1406
3	3	0	0.0		1407
4	4	0	0.0		1408
5	5	1	0.657	0.9413, 0.3842, 0.0587, 0.2191	1409
6	6	1	0.158	0.9346, 0.3019, 0.0654, 0.2788	1409
7	7	1	0.157	0.9258, 0.3897, 0.0736, 0.3491	1409
8	8	1	0.113	0.2583, 0.9265, 0.172, 0.0504	1409
9	9	0	0.0		1410
10	10	0	0.0		1411

ClassificationCategories

The image recognition file contains an entry called ***classification_categories***, which produces zero or more possible classifications of what each detection could be. For example, while a detection may broadly identify something as an animal, a classification may further identify that as a deer with high confidence, an elk with lower confidence, and so on. The classification_categories list all possible entities that the recognizer will consider. Each classification_category comprises an identifying integer and label.

Timelapse reads those values into the table (as illustrated below). As with detection_categories, Timelapse adds an 'Empty' classification to identify images that do not contain any classifications. It is mostly used as a lookup table to correlate the classification number with the human-readable label.

Table: ClassificationCategories

	classification	label
	Filter	Filter
1	2	human
2	3	WTD
3	8	BlackBear
4	9	Wolf
5	4	deer
6	7	MD
7	5	Cattle
8	1	elk
9	11	prong
10	10	Lion
11	0	empty
12	6	Moose

Classifications

The image recognizer contains, for each detection in each image, a list of zero or more possible classifications. Each classification identifies the ***classification*** category, and a ***confidence value*** for that classification.

The Classifications table holds each classification as a row. The ***classifciationID*** column is the primary key. ***The detectionID*** is the ***ID*** of the detection in the ***Detections*** table, and is used to link each classification to a single detection i.e., it is a foreign key describing a many to one relation between ***Classifications*** and ***Detections***. For example, in the table below:

- classificationID 1-2 identifies 2 different possible classifications on detection 14 in the DetectionTable. In descending order of confidence (conf) these are 1 – elk and 4 – deer.

Table: Classifications

	classificationID	category	conf	detectionID
	Filter	Filter	Filter	Filter
1	1	1	0.6299	14
2	2	4	0.3179	14
3	3	4	0.9788	36
4	4	4	0.9012	37
5	5	5	0.6833	38
6	6	1	0.2121	38
7	7	4	0.9268	39
8	8	1	0.8459	40
9	9	4	0.8027	41
10	10	5	0.4333	42
11	11	4	0.4103	42
12	12	4	0.9433	43

Info

The Megadetector image recognition file includes extra information that Timelapse records in its Info table. This includes Megadetector version information, time taken to do the recognitions, and several values indicating suggested confidence value thresholds when using detections and classifications.

	infoID	detector	detection_completion_time	classifier	classification_completion_time	megadetector_version	typical_detection_threshold	conservative_detection_threshold	typical_classification_threshold
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	md_v5a.0.0.pt	2022-06-15 19:32:16	megaclassifier_v0.1_efficientn...	2022-06-15 00:00:00	vUnknown	0.8	0.3	0.75

Accessing the Database with R

R is a popular programming language used for statistical computing. *R* can import data from many sources, such as CSV files and SQLite databases. Many users rely on CSV files containing data exported by Timelapse, as it is simple. However, users familiar with the SQL query language can access the data directly from the database, where they can form more complex queries to retrieve subsets of data. The data held in the Timelapse datatable can also be updated via these queries, although one has to be careful to conform to the data formats expected by Timelapse.

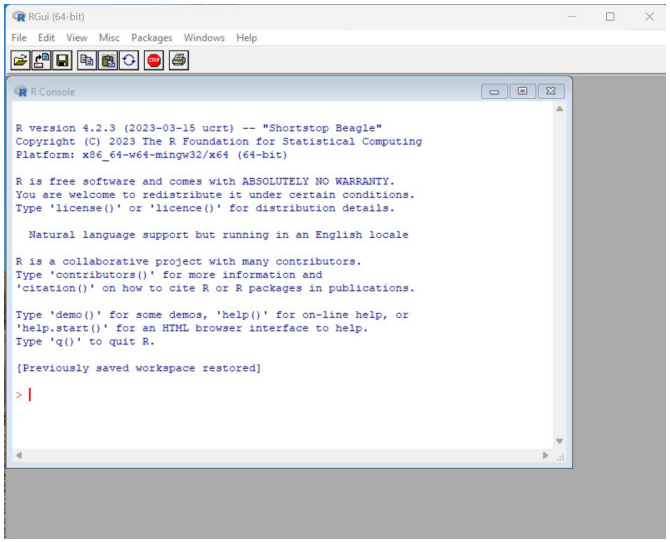
This brief tutorial describes how to open a Timelapse .ddb database file with *R*, and retrieve data from a particular table using SQL statements. We do not show how to analyze that data, as that would be something specific to the analyst's needs and can be done via routine *R* programming. Various other tutorials are available online that provide examples of how to use SQLite within *R* to query and manipulate a database.

Installing R and loading RSQLite

If not already on your system, the R programming environment needs to be installed and its RSQLite package loaded. This is very easy to do, and only needs to be done once.

Install R on windows

Various sites include the *R* download for Windows, such as https://cran.r-project.org/bin/windows/base/



Follow the instructions on that page for downloading and installing *R*. It should take just a few moments.

Running R

R should now be available as a new application, for example, under your Start menu. Run it as you would any other application. A window should appear, which includes a menu and an *R* Console window.

Installing and loading RSQLite

RSQLite needs to be installed on your machine, which is a one-time operation. From the *R* menu at the top of the window, select *Packages | **Install** Package(s)*. You will be asked for a preferable site to download it from (choose something from your counter). It wil lthen ask you which package you want to install. Select ***RSQLite*** from the scrollable list.

You then need to load the RSQLite package into *R*. From the *R* menu, select *Packages | **Load** Package(s)*, and select ***RSQLite*** from the scrollable list.

Using R and RSQLite

Connect to the Timelapse database

Lets assume a database file called TimelapseData.ddb is available that contains tag data. To access this database, we have to connect to it. This is done through the following command, where the full path to the database file is supplied. The connection is assigned to the variable ***conn***, whichis then used to access that database.
Note: '\' is a special character, written as '\\'

```
conn <- dbConnect(RSQLite::SQLite(),
                  "C:\\Users\\saulg\\Desktop\\PracticeImageSet\\TimelapseData.ddb")
```

Query the Timelapse database

SQL queries can now be easily generated and the results collected. In this example, we collect the file names of images in the *Station1\Fetched-2015-09* folder that contain bobcat in the Species field.

```
# Collect the query result in the variable bobcatFiles
bobcatFiles <- dbGetQuery(conn,
      "SELECT RelativePath, File FROM DataTable
       WHERE RelativePath= ' Station1\\Fetched-2015-09'
       AND Species = 'bobcat'")

# List the contents of bobcatFiles
> bobcatFiles
          RelativePath          File
1 Station1\\Fetched-2015-09   IMG_031.jpg
2 Station1\\Fetched-2015-09   IMG_032.jpg
3 Station1\\Fetched-2015-09   IMG_033.jpg
```