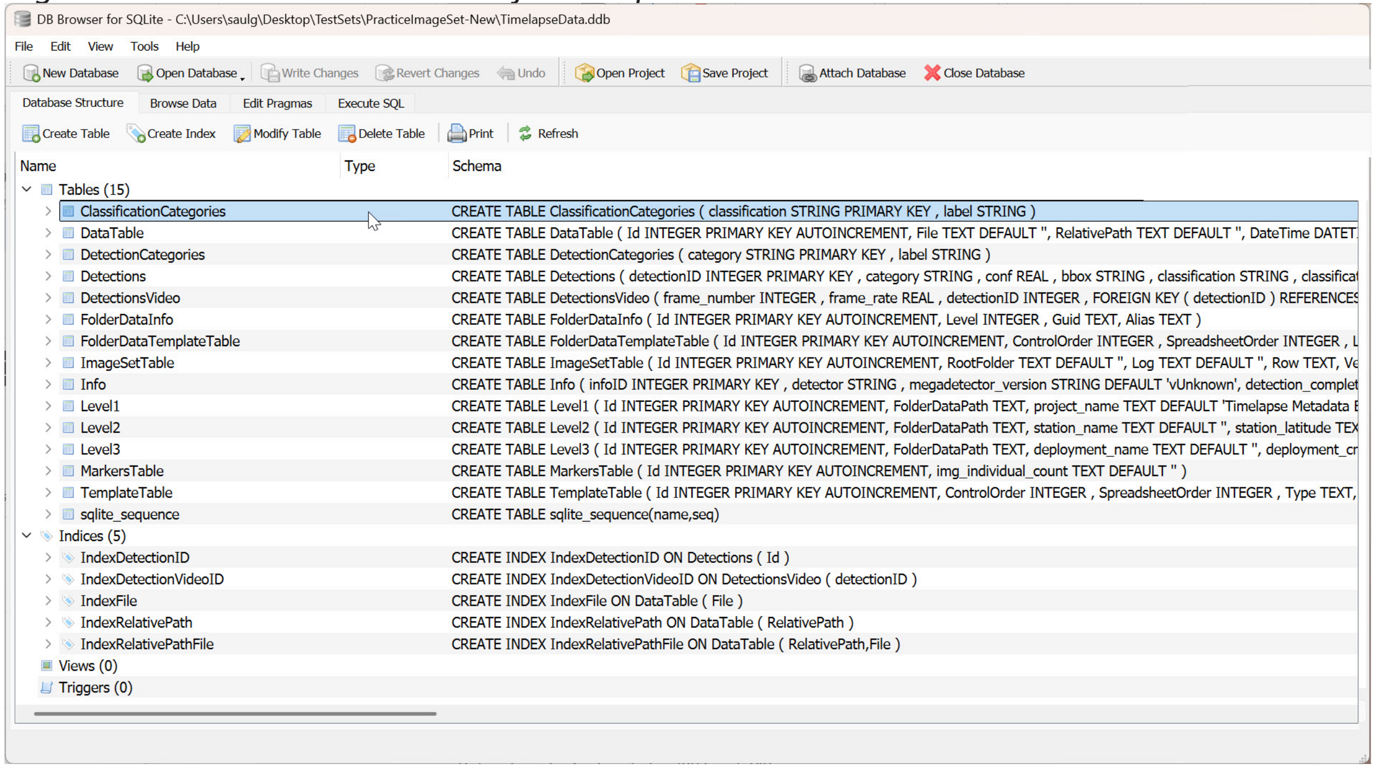
Timelapse Database Guide

A guide to the internal structure of Timelapse Database tables



Name	Type	Schema
Tables (15)		
ClassificationCategories		CREATE TABLE ClassificationCategories (classification STRING PRIMARY KEY , label STRING)
DataTable		CREATE TABLE DataTable (Id INTEGER PRIMARY KEY AUTOINCREMENT, File TEXT DEFAULT '', RelativePath TEXT DEFAULT '', DateTime DATET
DetectionCategories		CREATE TABLE DetectionCategories (category STRING PRIMARY KEY , label STRING)
Detections		CREATE TABLE Detections (detectionID INTEGER PRIMARY KEY , category STRING , conf REAL , bbox STRING , classification STRING , classificat
DetectionsVideo		CREATE TABLE DetectionsVideo (frame_number INTEGER , frame_rate REAL , detectionID INTEGER , FOREIGN KEY (detectionID) REFERENCES
FolderDataInfo		CREATE TABLE FolderDataInfo (Id INTEGER PRIMARY KEY AUTOINCREMENT, Level INTEGER , Guid TEXT, Alias TEXT)
FolderDataTemplateTable		CREATE TABLE FolderDataTemplateTable (Id INTEGER PRIMARY KEY AUTOINCREMENT, ControlOrder INTEGER , SpreadsheetOrder INTEGER , L
ImageSetTable		CREATE TABLE ImageSetTable (Id INTEGER PRIMARY KEY AUTOINCREMENT, RootFolder TEXT DEFAULT '', Log TEXT DEFAULT '', Row TEXT, Ve
Info		CREATE TABLE Info (infoID INTEGER PRIMARY KEY , detector STRING , megadetector_version STRING DEFAULT 'vUnknown', detection_complet
Level1		CREATE TABLE Level1 (Id INTEGER PRIMARY KEY AUTOINCREMENT, FolderDataPath TEXT, project_name TEXT DEFAULT 'Timelapse Metadata E
Level2		CREATE TABLE Level2 (Id INTEGER PRIMARY KEY AUTOINCREMENT, FolderDataPath TEXT, station_name TEXT DEFAULT '', station_latitude TEX
Level3		CREATE TABLE Level3 (Id INTEGER PRIMARY KEY AUTOINCREMENT, FolderDataPath TEXT, deployment_name TEXT DEFAULT '', deployment_cr
MarkersTable		CREATE TABLE MarkersTable (Id INTEGER PRIMARY KEY AUTOINCREMENT, img_individual_count TEXT DEFAULT '')
TemplateTable		CREATE TABLE TemplateTable (Id INTEGER PRIMARY KEY AUTOINCREMENT, ControlOrder INTEGER , SpreadsheetOrder INTEGER , Type TEXT,
sqlite_sequence		CREATE TABLE sqlite_sequence(name,seq)
Indices (5)		
IndexDetectionID		CREATE INDEX IndexDetectionID ON Detections (Id)
IndexDetectionVideoID		CREATE INDEX IndexDetectionVideoID ON DetectionsVideo (detectionID)
IndexFile		CREATE INDEX IndexFile ON DataTable (File)
IndexRelativePath		CREATE INDEX IndexRelativePath ON DataTable (RelativePath)
IndexRelativePathFile		CREATE INDEX IndexRelativePathFile ON DataTable (RelativePath,File)
Views (0)		
Triggers (0)		

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

Part 6 of the *Timelapse Manual Series*. Last updated May 1, 2025, Timelapse Version 2.3.3.0

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.

[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.

Table of Contents

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 ones the *DataTable* and the various *Levels* tables in the Timelapse database *.ddb* file, as it will contain all your image tag data and folder-level metadata (if used). Other tables, while described here, are of lesser or no interest, as they are mostly used internally by Timelapse to store or track secondary information.

Timelapse relies on two database files.

- ***The Timelapse database*** (*.ddb* suffix) contains all your tag data, image recognition data (if any), folder-level metadata, 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 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, where the template specifies your data fields and folder levels (if any). 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 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 data in these Timelapse database files through other software. For example:

- The *R statistical package* is often used by knowledgeable people to access SQLite data bases and 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/

Note. Altering database files can compromise Timelapse's ability to read those files if it deviates from Timelapse expectations. Problematic alterations include 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. No complex database installation is needed.

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[1]. 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 as explained in the Timelapse Reference Guide.
- SQLite is less suited for multiple people simultaneously tagging overlapping sets of images, even when its located on a central system. 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.
- Local vs. cloud based vs. virtual machines. The Timelapse / SQLite architecture normally runs locally rather than on a central server, e.g., as a database accessed through the cloud. See FAQ: Timelapse file management on local disks, network drives, and the cloud if you are are using these options when using Timelapse:
  - » locating your database files and images on network server or cloud drives,
  - » running virtual machines, where users log onto them to do their work.

1 The slowdown is mostly due to how Timelapse managesSQLite queries, where Timelapse reads the entire database into memory after every selection rather than on demand. This inefficiency may be fixed in the future.

Tables in the Template .tdb file

Four tables are created and maintained by the Timelapse *Template Editor* whenever a project manager creates or updates a *.tdb* template file. These tables mostly hold the data field specifications and folder levels (if any) that will be used by Timelapse to create its user interface and to manage the data entered by the analyst, although some also store information, such as version and state values.

For all but the ***TemplateInfo*** table, Timelapse will also copy, maintain and update these table whenever the Timelapse *.ddb* file is created or used. In particular:

- The various tables are created or modified in the *.tdb* file through the Timelapse Template Editor.
- When an image set is loaded into Timelapse for the first time, Timelapse creates its own copy of these tables and stores it in the *.ddb* file. It then uses those tables as a specification for the data fields and folder levels present in the user interface, and to define its own tables that will hold the data entered by the analyst.
- During subsequent loads of that image set, Timelapse compares the *.tdb* tables with the *.ddb* copy for differences, and tries to resolve those differences by displaying a dialog to the user.

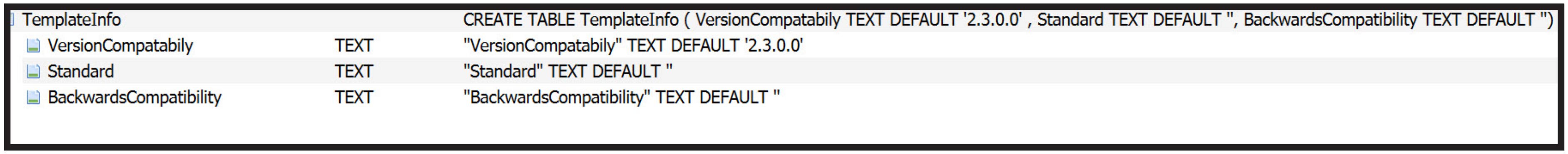
The TemplateInfo Table

The ***TemplateInfo Table*** is found in only the Timelapse Template *.tdb* file. Its purpose is to hold information that: a) helps Timelapse maintain version compatability between itself and the database files, and b) that names the metadata standard (if any) used to create this template.

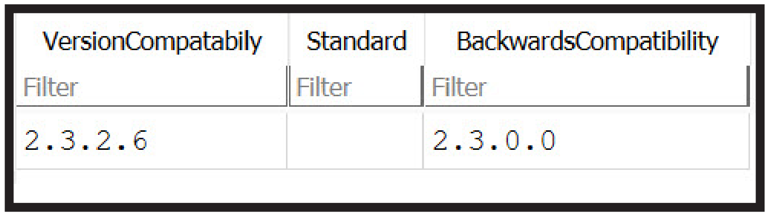
It contains three rows.

- *VersionCompatability* records the last version of Timelapse used to open this template.
- *Standard* records the name of the metadata standard being used, if any. If the template was not based on an existing standard, that field is left empty.
- *BackwardsCompatability* records the earliest version of Timelapse that can safely load this database. Timelapse will check this field to see if its Timelapse version is at least as current as the one recorded here. If not, Timelapse will generate a warning.

The TemplateInfo table schema is shown below, along with an example. In this example, the template was not created upon a standard, which is why the Standard field is empty.



TemplateInfo		CREATE TABLE TemplateInfo (VersionCompatabily TEXT DEFAULT '2.3.0.0' , Standard TEXT DEFAULT '', BackwardsCompatibility TEXT DEFAULT '')
VersionCompatabily	TEXT	"VersionCompatabily" TEXT DEFAULT '2.3.0.0'
Standard	TEXT	"Standard" TEXT DEFAULT ''
BackwardsCompatibility	TEXT	"BackwardsCompatibility" TEXT DEFAULT ''



VersionCompatabily	Standard	BackwardsCompatibility
Filter	Filter	Filter
2.3.2.6		2.3.0.0

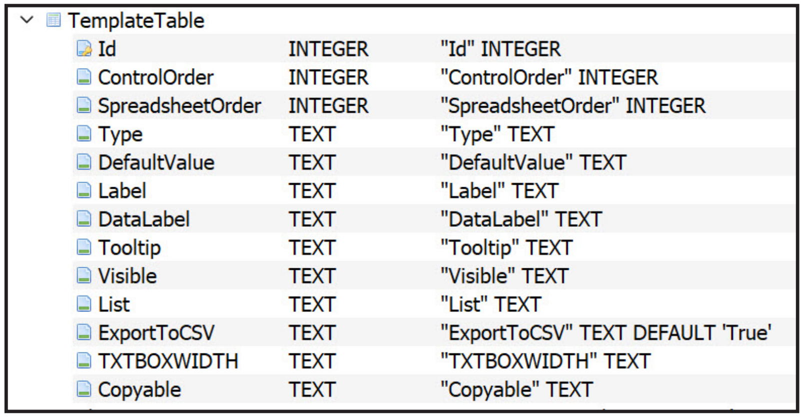
Template Table

The *TemplateTable* table is responsible for storing all the image-level data field specifications as those fields are created and updated by a project manager using the *Template Editor.* While you will not normally access this table, it can be of interest if you want to look up (or modify) the information associated with each data field.

An example *TemplateTable* is illustrated below, and is accompanied by its database schema. Its contents is similar to the template created in the *Timelapse QuickStart* guide, which in turn was included in the *PracticeImageSet.*

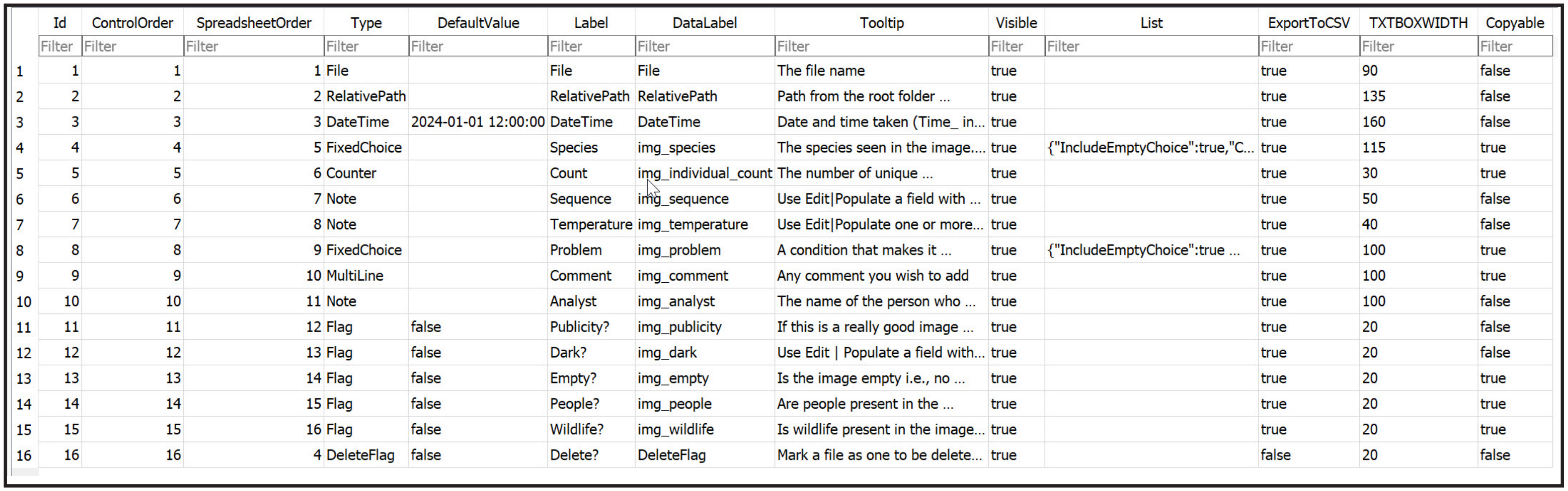
Several columns in the *TemplateTable* are used internally by Timelapse.

- *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.
- The *List* column is 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.
- Other fields are as described in the Timelapse Template Guide.



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
Visible	TEXT	"Visible" TEXT
List	TEXT	"List" TEXT
ExportToCSV	TEXT	"ExportToCSV" TEXT DEFAULT 'True'
TXTBOXWIDTH	TEXT	"TXTBOXWIDTH" TEXT
Copyable	TEXT	"Copyable" TEXT

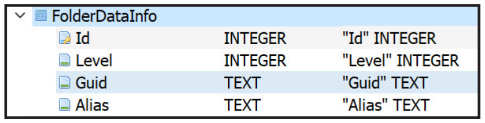


	Id	ControlOrder	SpreadsheetOrder	Type	DefaultValue	Label	DataLabel	Tooltip	Visible	List	ExportToCSV	TXTBOXWIDTH	Copyable
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	1	1	File		File	File	The file name	true		true	90	false
2	2	2	2	RelativePath		RelativePath	RelativePath	Path from the root folder ...	true		true	135	false
3	3	3	3	DateTime	2024-01-01 12:00:00	DateTime	DateTime	Date and time taken (Time_ in...	true		true	160	false
4	4	4	5	FixedChoice		Species	img_species	The species seen in the image....	true	{"IncludeEmptyChoice":true,"C...	true	115	true
5	5	5	6	Counter		Count	img_individual_count	The number of unique ...	true		true	30	true
6	6	6	7	Note		Sequence	img_sequence	Use Edit|Populate a field with ...	true		true	50	false
7	7	7	8	Note		Temperature	img_temperature	Use Edit|Populate one or more...	true		true	40	false
8	8	8	9	FixedChoice		Problem	img_problem	A condition that makes it ...	true	{"IncludeEmptyChoice":true ...	true	100	true
9	9	9	10	MultiLine		Comment	img_comment	Any comment you wish to add	true		true	100	true
10	10	10	11	Note		Analyst	img_analyst	The name of the person who ...	true		true	100	false
11	11	11	12	Flag	false	Publicity?	img_publicity	If this is a really good image ...	true		true	20	false
12	12	12	13	Flag	false	Dark?	img_dark	Use Edit | Populate a field with...	true		true	20	false
13	13	13	14	Flag	false	Empty?	img_empty	Is the image empty i.e., no ...	true		true	20	true
14	14	14	15	Flag	false	People?	img_people	Are people present in the ...	true		true	20	true
15	15	15	16	Flag	false	Wildlife?	img_wildlife	Is wildlife present in the image...	true		true	20	true
16	16	16	4	DeleteFlag	false	Delete?	DeleteFlag	Mark a file as one to be delete...	true		false	20	false

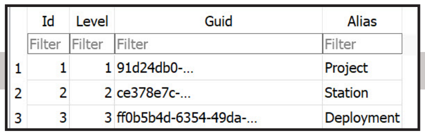
FolderDataInfo Table

The ***FolderDataInfo*** table is created by the *Template Editor* and stored in the *.tdb* file. This table tracks folders levels as they are created by the project manager, and how they were named. Similar to the ***Template*** table, it is also copied to the .ddb file and used by Timelapse to generate the user interface and look for differences.

This table is just a lookup table, where one can find the correspondence between the a level number and its alias used to name it (e.g., looking up 1 in the *Levels* column gives the name *Project* in the *Alias* column). The *Guid* column is used internally by Timelapse to associate a globally unique id with each level.



FolderDataInfo		
Id	INTEGER	"Id" INTEGER
Level	INTEGER	"Level" INTEGER
Guid	TEXT	"Guid" TEXT
Alias	TEXT	"Alias" TEXT

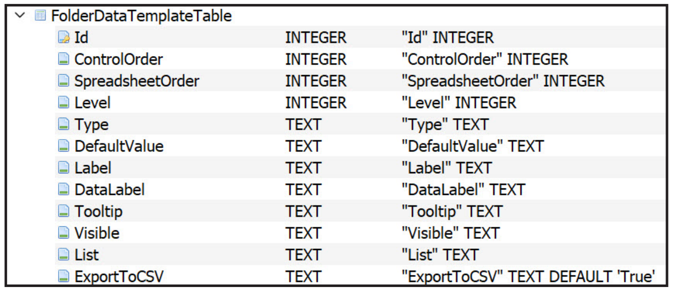


	Id	Level	Guid	Alias
	Filter	Filter	Filter	Filter
1	1	1	91d24db0-...	Project
2	2	2	ce378e7c-...	Station
3	3	3	ff0b5b4d-6354-49da-...	Deployment

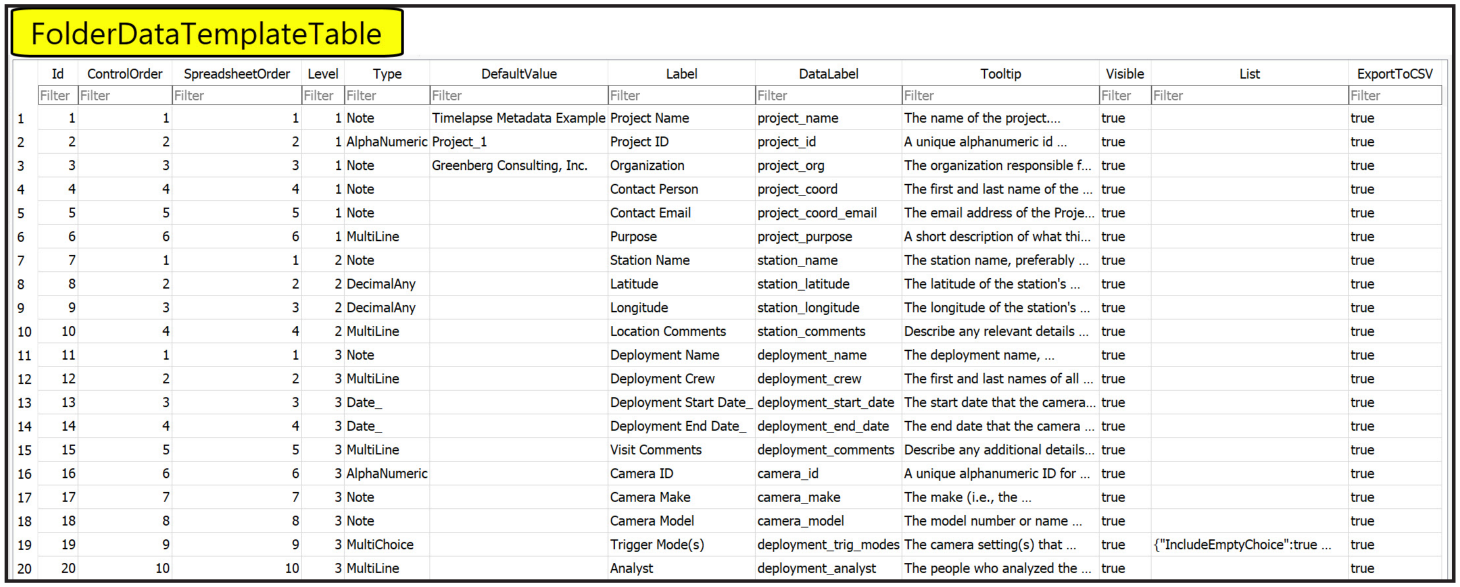
FolderDataTemplateTable

The ***FolderDataTemplateTable*** table is also created by the *Template Editor* and stored in the *.tdb* file. This table tracks the data fields for each folder level as they are created and modified by the project manager. Its columns are similar to the ***Template*** table. It is also copied to the .ddb file and used by Timelapse to generate the user interface and look for differences.

The ***Level*** column in this table indicates the folder level the data field specification is associate with. For example, consider the first row containing a control labeled *Project Name*. It is associated with the first root folder level (Level = 1). From the table above, we know that level is named *Project*.



FolderDataTemplateTable		
Id	INTEGER	"Id" INTEGER
ControlOrder	INTEGER	"ControlOrder" INTEGER
SpreadsheetOrder	INTEGER	"SpreadsheetOrder" INTEGER
Level	INTEGER	"Level" INTEGER
Type	TEXT	"Type" TEXT
DefaultValue	TEXT	"DefaultValue" TEXT
Label	TEXT	"Label" TEXT
DataLabel	TEXT	"DataLabel" TEXT
Tooltip	TEXT	"Tooltip" TEXT
Visible	TEXT	"Visible" TEXT
List	TEXT	"List" TEXT
ExportToCSV	TEXT	"ExportToCSV" TEXT DEFAULT 'True'



FolderDataTemplateTable

	Id	ControlOrder	SpreadsheetOrder	Level	Type	DefaultValue	Label	DataLabel	Tooltip	Visible	List	ExportToCSV
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	1	1	1	Note	Timelapse Metadata Example	Project Name	project_name	The name of the project....	true		true
2	2	2	2	1	AlphaNumeric	Project_1	Project ID	project_id	A unique alphanumeric id ...	true		true
3	3	3	3	1	Note	Greenberg Consulting, Inc.	Organization	project_org	The organization responsible f...	true		true
4	4	4	4	1	Note		Contact Person	project_coord	The first and last name of the ...	true		true
5	5	5	5	1	Note		Contact Email	project_coord_email	The email address of the Proje...	true		true
6	6	6	6	1	MultiLine		Purpose	project_purpose	A short description of what thi...	true		true
7	7	1	1	2	Note		Station Name	station_name	The station name, preferably ...	true		true
8	8	2	2	2	DecimalAny		Latitude	station_latitude	The latitude of the station's ...	true		true
9	9	3	3	2	DecimalAny		Longitude	station_longitude	The longitude of the station's ...	true		true
10	10	4	4	2	MultiLine		Location Comments	station_comments	Describe any relevant details ...	true		true
11	11	1	1	3	Note		Deployment Name	deployment_name	The deployment name, ...	true		true
12	12	2	2	3	MultiLine		Deployment Crew	deployment_crew	The first and last names of all ...	true		true
13	13	3	3	3	Date_		Deployment Start Date_	deployment_start_date	The start date that the camera...	true		true
14	14	4	4	3	Date_		Deployment End Date_	deployment_end_date	The end date that the camera ...	true		true
15	15	5	5	3	MultiLine		Visit Comments	deployment_comments	Describe any additional details...	true		true
16	16	6	6	3	AlphaNumeric		Camera ID	camera_id	A unique alphanumeric ID for ...	true		true
17	17	7	7	3	Note		Camera Make	camera_make	The make (i.e., the ...	true		true
18	18	8	8	3	Note		Camera Model	camera_model	The model number or name ...	true		true
19	19	9	9	3	MultiChoice		Trigger Mode(s)	deployment_trig_modes	The camera setting(s) that ...	true	{"IncludeEmptyChoice":true ...	true
20	20	10	10	3	MultiLine		Analyst	deployment_analyst	The people who analyzed the ...	true		true

Tables in the Data .ddb file

The *DataTable* and the numbered *Level* tables (if any) are perhaps the only tables of interest to a Timelapse manager who wishes to directly access data. The other tables are either copies of the various template table, or contain information used by Timelapse, or contain recognition data imported from a recogntion file.

The DataTable

The *DataTable*, found in the Timelapse *.ddb* file, contains all the image-level tagging data entered by the analyzer.

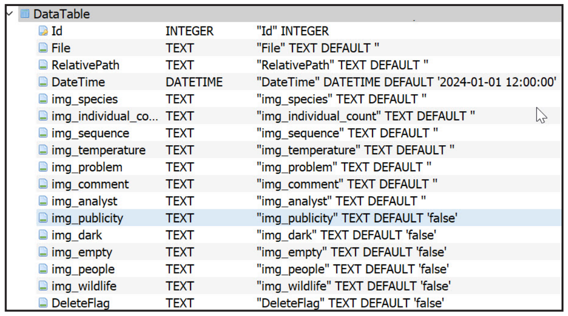
The figure below illustrates an example *DataTable* as held by the database. Each row is a record of data associated with an image. Each row is uniquely identified by an integer *Id*. The *Id* is set by the database engine, where its value is incremented and assigned when images are loaded into Timelapse 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).

Each column corresponds to the *DataLabels* as specified in the Timelapse Template file. Four represent the required data fields listed in the Timelapse Template Editor (*File*, *RelativePath*, *DateTime* and *DeleteFlag*). These are always present, even if their *Visibility* attribute is set to false. All other columns represent custom data fields defined by the project manager when using the Timelapse Template Editor, where each contains data entered into each image's data field.

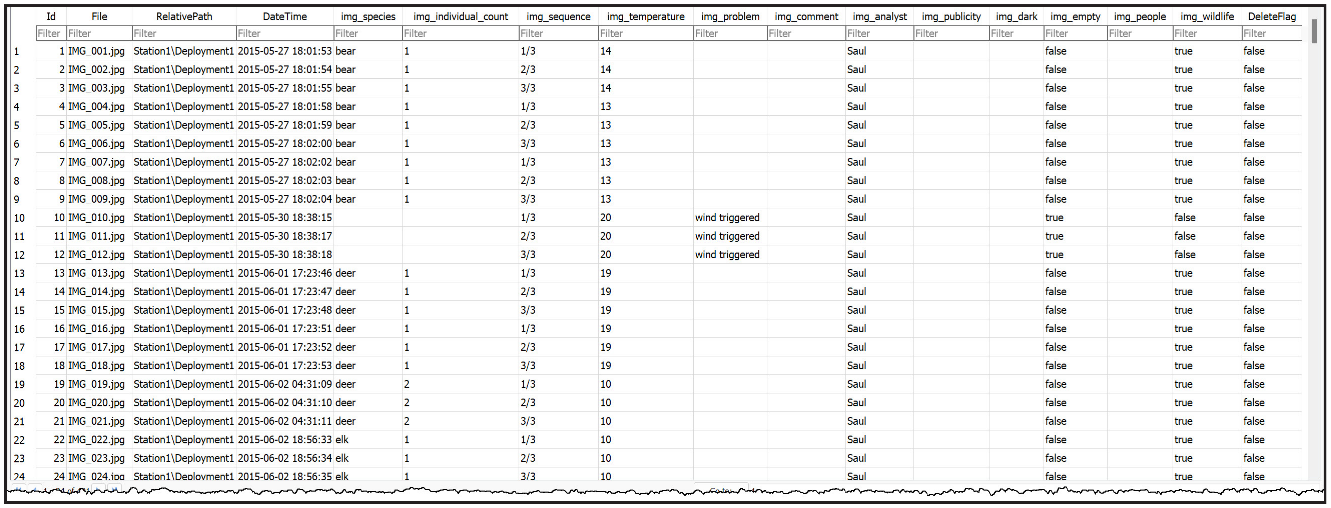
The example Data Table below illustrates its structure and contents after a user completed the exercises in the *Timelapse QuickStart Guide*. The columns reflect the contents of the template provided in the *PracticeImageSet.*

The DataTable's schema is shown at the right. Even though most schema types are TEXT, Timelapse populates and 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\Deployment1*. Files located directly in the root folder would have and empty RelativePath.
- *DateTime, Date, Time* values can only contain a date formated as a full date yyyy-mm-dd hh:mm:ss (for example, 2015-05-27 18:01:53), or just its date or time portion.
- *Text controls* can contain any text, except for *Alphanumeric* which is limited to letters, numbers, dashes and underscores.
- *Integer, IntegerPostive, Decimal, DecimalPositive, Count* values are blank or a number of a particular type.
- *FixedChoice, MultiChoice* values should match its List menu item(s) as defined in the template (e.g., the *Species* column data must match bear, deer, etc.).
- *Flag controls* can only contain case-insensitive true or false values (e.g., the columns *Dark, Empty*, *Publicity*, and *DeleteFlag).*



DataTable		
Id	INTEGER	"Id" INTEGER
File	TEXT	"File" TEXT DEFAULT "
RelativePath	TEXT	"RelativePath" TEXT DEFAULT "
DateTime	DATETIME	"DateTime" DATETIME DEFAULT '2024-01-01 12:00:00'
img_species	TEXT	"img_species" TEXT DEFAULT "
img_individual_co...	TEXT	"img_individual_count" TEXT DEFAULT "
img_sequence	TEXT	"img_sequence" TEXT DEFAULT "
img_temperature	TEXT	"img_temperature" TEXT DEFAULT "
img_problem	TEXT	"img_problem" TEXT DEFAULT "
img_comment	TEXT	"img_comment" TEXT DEFAULT "
img_analyst	TEXT	"img_analyst" TEXT DEFAULT "
img_publicity	TEXT	"img_publicity" TEXT DEFAULT 'false'
img_dark	TEXT	"img_dark" TEXT DEFAULT 'false'
img_empty	TEXT	"img_empty" TEXT DEFAULT 'false'
img_people	TEXT	"img_people" TEXT DEFAULT 'false'
img_wildlife	TEXT	"img_wildlife" TEXT DEFAULT 'false'
DeleteFlag	TEXT	"DeleteFlag" TEXT DEFAULT 'false'

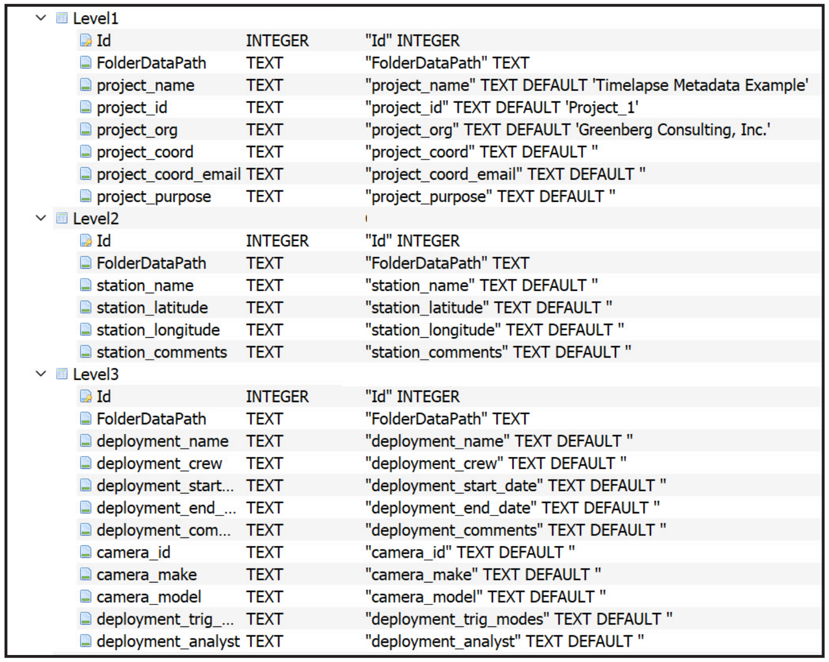


	Id	File	RelativePath	DateTime	img_species	img_individual_count	img_sequence	img_temperature	img_problem	img_comment	img_analyst	img_publicity	img_dark	img_empty	img_people	img_wildlife	DeleteFlag
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	IMG_001.jpg	Station1\Deployment1	2015-05-27 18:01:53	bear	1	1/3	14			Saul			false		true	false
2	2	IMG_002.jpg	Station1\Deployment1	2015-05-27 18:01:54	bear	1	2/3	14			Saul			false		true	false
3	3	IMG_003.jpg	Station1\Deployment1	2015-05-27 18:01:55	bear	1	3/3	14			Saul			false		true	false
4	4	IMG_004.jpg	Station1\Deployment1	2015-05-27 18:01:58	bear	1	1/3	13			Saul			false		true	false
5	5	IMG_005.jpg	Station1\Deployment1	2015-05-27 18:01:59	bear	1	2/3	13			Saul			false		true	false
6	6	IMG_006.jpg	Station1\Deployment1	2015-05-27 18:02:00	bear	1	3/3	13			Saul			false		true	false
7	7	IMG_007.jpg	Station1\Deployment1	2015-05-27 18:02:02	bear	1	1/3	13			Saul			false		true	false
8	8	IMG_008.jpg	Station1\Deployment1	2015-05-27 18:02:03	bear	1	2/3	13			Saul			false		true	false
9	9	IMG_009.jpg	Station1\Deployment1	2015-05-27 18:02:04	bear	1	3/3	13			Saul			false		true	false
10	10	IMG_010.jpg	Station1\Deployment1	2015-05-30 18:38:15			1/3	20	wind triggered		Saul			true		false	false
11	11	IMG_011.jpg	Station1\Deployment1	2015-05-30 18:38:17			2/3	20	wind triggered		Saul			true		false	false
12	12	IMG_012.jpg	Station1\Deployment1	2015-05-30 18:38:18			3/3	20	wind triggered		Saul			true		false	false
13	13	IMG_013.jpg	Station1\Deployment1	2015-06-01 17:23:46	deer	1	1/3	19			Saul			false		true	false
14	14	IMG_014.jpg	Station1\Deployment1	2015-06-01 17:23:47	deer	1	2/3	19			Saul			false		true	false
15	15	IMG_015.jpg	Station1\Deployment1	2015-06-01 17:23:48	deer	1	3/3	19			Saul			false		true	false
16	16	IMG_016.jpg	Station1\Deployment1	2015-06-01 17:23:51	deer	1	1/3	19			Saul			false		true	false
17	17	IMG_017.jpg	Station1\Deployment1	2015-06-01 17:23:52	deer	1	2/3	19			Saul			false		true	false
18	18	IMG_018.jpg	Station1\Deployment1	2015-06-01 17:23:53	deer	1	3/3	19			Saul			false		true	false
19	19	IMG_019.jpg	Station1\Deployment1	2015-06-02 04:31:09	deer	2	1/3	10			Saul			false		true	false
20	20	IMG_020.jpg	Station1\Deployment1	2015-06-02 04:31:10	deer	2	2/3	10			Saul			false		true	false
21	21	IMG_021.jpg	Station1\Deployment1	2015-06-02 04:31:11	deer	2	3/3	10			Saul			false		true	false
22	22	IMG_022.jpg	Station1\Deployment1	2015-06-02 18:56:33	elk	1	1/3	10			Saul			false		true	false
23	23	IMG_023.jpg	Station1\Deployment1	2015-06-02 18:56:34	elk	1	2/3	10			Saul			false		true	false
24	24	IMG_024.jpg	Station1\Deployment1	2015-06-02 18:56:35	elk	1	3/3	10			Saul			false		true	false

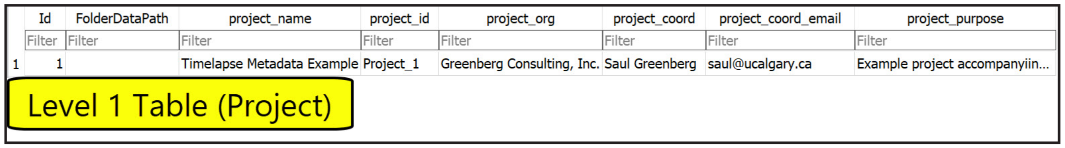
The Level Tables

If your template defined folder-levels and corresponding folder-level data fields, data tables representing these levels and their data will be found in the the Timelapse *.ddb* file. Tables, if any, are named *Level1*, *Level2*, *Level3* etc, where *Level1* corresponds to the root folder level. Their columns represent the folder-level data field specifictaion in the template, along with the data that was filled in by the analyst. These tables, along with the DataTable, are of interest to a Timelapse user who wishes to directly access data.

As a reminder, the previously discussed *FolderDataInfo* table maps the level number of each table to its actual name. For example, looking up Level1 in the *Levels* column of the *FolderDataInfo* table indicates that it is named *Project.*

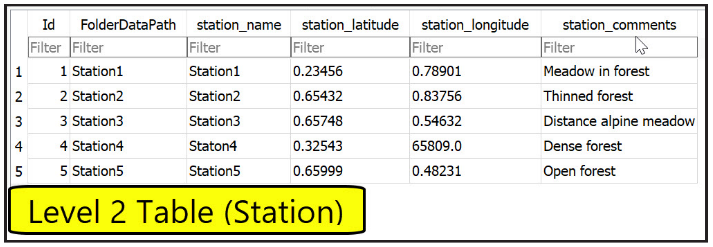


Level1		
Id	INTEGER	"Id" INTEGER
FolderDataPath	TEXT	"FolderDataPath" TEXT
project_name	TEXT	"project_name" TEXT DEFAULT 'Timelapse Metadata Example'
project_id	TEXT	"project_id" TEXT DEFAULT 'Project_1'
project_org	TEXT	"project_org" TEXT DEFAULT 'Greenberg Consulting, Inc.'
project_coord	TEXT	"project_coord" TEXT DEFAULT ''
project_coord_email	TEXT	"project_coord_email" TEXT DEFAULT ''
project_purpose	TEXT	"project_purpose" TEXT DEFAULT ''
Level2		
Id	INTEGER	"Id" INTEGER
FolderDataPath	TEXT	"FolderDataPath" TEXT
station_name	TEXT	"station_name" TEXT DEFAULT ''
station_latitude	TEXT	"station_latitude" TEXT DEFAULT ''
station_longitude	TEXT	"station_longitude" TEXT DEFAULT ''
station_comments	TEXT	"station_comments" TEXT DEFAULT ''
Level3		
Id	INTEGER	"Id" INTEGER
FolderDataPath	TEXT	"FolderDataPath" TEXT
deployment_name	TEXT	"deployment_name" TEXT DEFAULT ''
deployment_crew	TEXT	"deployment_crew" TEXT DEFAULT ''
deployment_start...	TEXT	"deployment_start_date" TEXT DEFAULT ''
deployment_end_...	TEXT	"deployment_end_date" TEXT DEFAULT ''
deployment_com...	TEXT	"deployment_comments" TEXT DEFAULT ''
camera_id	TEXT	"camera_id" TEXT DEFAULT ''
camera_make	TEXT	"camera_make" TEXT DEFAULT ''
camera_model	TEXT	"camera_model" TEXT DEFAULT ''
deployment_trig_...	TEXT	"deployment_trig_modes" TEXT DEFAULT ''
deployment_analyst	TEXT	"deployment_analyst" TEXT DEFAULT ''



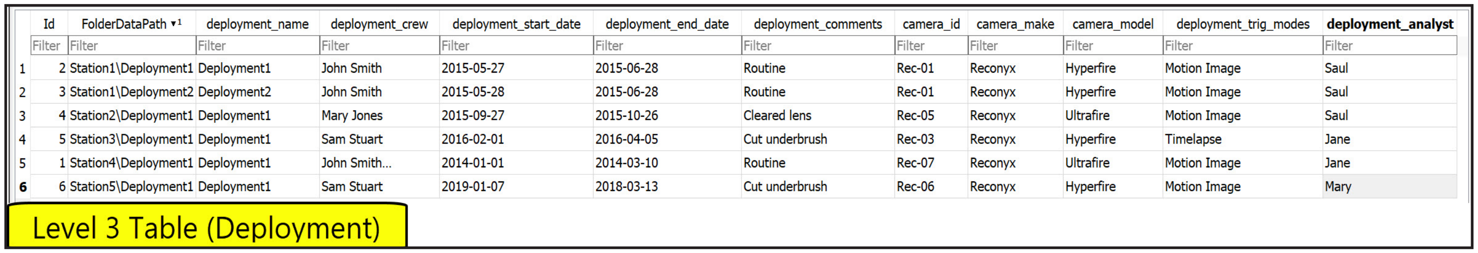
	Id	FolderDataPath	project_name	project_id	project_org	project_coord	project_coord_email	project_purpose
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1		Timelapse Metadata Example	Project_1	Greenberg Consulting, Inc.	Saul Greenberg	saul@ucalgary.ca	Example project accompanyiin...

Level 1 Table (Project)



	Id	FolderDataPath	station_name	station_latitude	station_longitude	station_comments
	Filter	Filter	Filter	Filter	Filter	Filter
1	1	Station1	Station1	0.23456	0.78901	Meadow in forest
2	2	Station2	Station2	0.65432	0.83756	Thinned forest
3	3	Station3	Station3	0.65748	0.54632	Distance alpine meadow
4	4	Station4	Staton4	0.32543	65809.0	Dense forest
5	5	Station5	Station5	0.65999	0.48231	Open forest

Level 2 Table (Station)



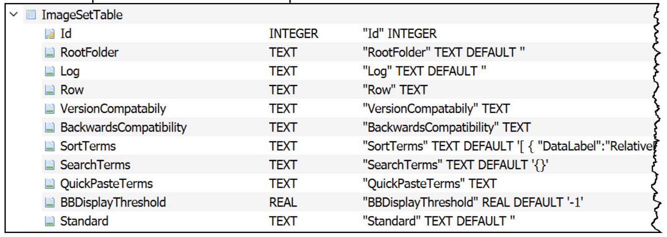
	Id	FolderDataPath ▾1	deployment_name	deployment_crew	deployment_start_date	deployment_end_date	deployment_comments	camera_id	camera_make	camera_model	deployment_trig_modes	**deployment_analyst**
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	2	Station1\Deployment1	Deployment1	John Smith	2015-05-27	2015-06-28	Routine	Rec-01	Reconyx	Hyperfire	Motion Image	Saul
2	3	Station1\Deployment2	Deployment2	John Smith	2015-05-28	2015-06-28	Routine	Rec-01	Reconyx	Hyperfire	Motion Image	Saul
3	4	Station2\Deployment1	Deployment1	Mary Jones	2015-09-27	2015-10-26	Cleared lens	Rec-05	Reconyx	Ultrafire	Motion Image	Saul
4	5	Station3\Deployment1	Deployment1	Sam Stuart	2016-02-01	2016-04-05	Cut underbrush	Rec-03	Reconyx	Hyperfire	Timelapse	Jane
5	1	Station4\Deployment1	Deployment1	John Smith...	2014-01-01	2014-03-10	Routine	Rec-07	Reconyx	Ultrafire	Motion Image	Jane
6	6	Station5\Deployment1	Deployment1	Sam Stuart	2019-01-07	2018-03-13	Cut underbrush	Rec-06	Reconyx	Hyperfire	Motion Image	Mary

Level 3 Table (Deployment)

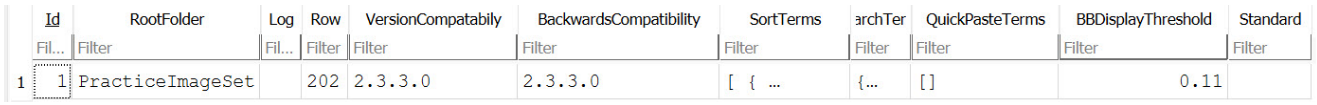
The ImageSetTable

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.

- *Root folder*: the name of the root folder containing the template.
- *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 before closing the image set.
- *VersionCompatability*: The last version of Timelapse used to open this database.
- *BackwardsCompatability*: The last version of Timelapse able to open this database; used to see if prior versions should try to open the current file.
- *SortTerms*: The last used criteria used to sort the images (via the *Sort* menu), stored as a JSON structure.
- *SearchTerms*: The last used criteria used to select the images (via the *Select* menu), stored as a JSON structure.
- *QuickPasteXML*: used internally by Timelapse to save/restore QuickPaste information, stored in XML format.
- *BBDisplayThreshold*: The confidence threshold for displaying bounding boxes when image recognition is used.
- *Standard*: records the name of the metadata standard being used, if any. If the template was not based on an existing standard, that field is left empty. Its value is copied from the TemplateInfo table when the .ddb file is first created.



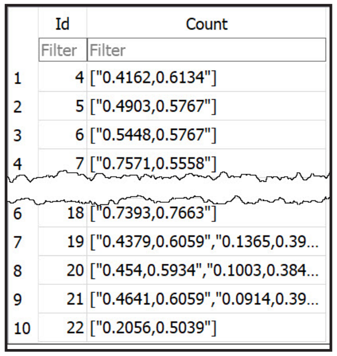
ImageSetTable		
Id	INTEGER	"Id" INTEGER
RootFolder	TEXT	"RootFolder" TEXT DEFAULT ''
Log	TEXT	"Log" TEXT DEFAULT ''
Row	TEXT	"Row" TEXT
VersionCompatabily	TEXT	"VersionCompatabily" TEXT
BackwardsCompatibility	TEXT	"BackwardsCompatibility" TEXT
SortTerms	TEXT	"SortTerms" TEXT DEFAULT '[{ "DataLabel":"Relative
SearchTerms	TEXT	"SearchTerms" TEXT DEFAULT '{}'
QuickPasteTerms	TEXT	"QuickPasteTerms" TEXT
BBDisplayThreshold	REAL	"BBDisplayThreshold" REAL DEFAULT '-1'
Standard	TEXT	"Standard" TEXT DEFAULT ''



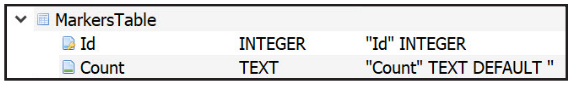
	Id	RootFolder	Log	Row	VersionCompatabily	BackwardsCompatibility	SortTerms	archTer	QuickPasteTerms	BBDisplayThreshold	Standard
	Fil...	Filter	Fil...	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	PracticeImageSet		202	2.3.3.0	2.3.3.0	[{ ...	{...	[]	0.11	

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 in the *.ddb* file 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 Megadetector 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 unless you want to do something with the recognition data that is not offered by Timelapse.

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.

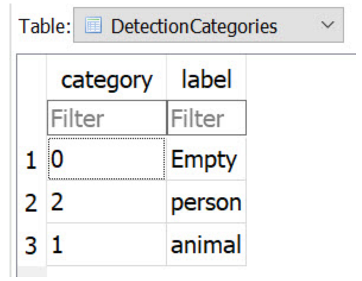
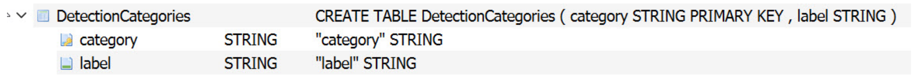


Table: DetectionCategories

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



DetectionCategories		CREATE TABLE DetectionCategories (category STRING PRIMARY KEY , label STRING)
category	STRING	"category" STRING
label	STRING	"label" STRING

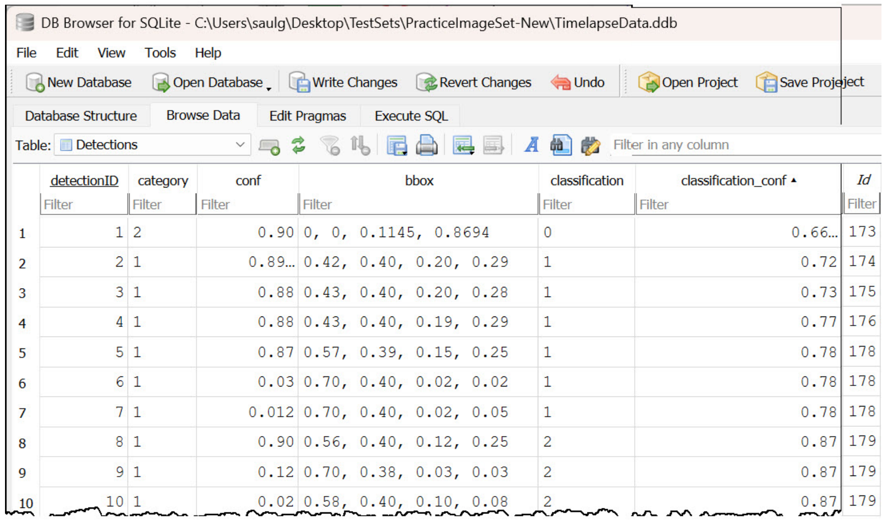
Detections

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), a classificaiton category ***classification*** used to look up the label in the ***ClassificationCategory*** table, and a cconfidence value ***classification_conf*** for that classification.

For example, in the table below:

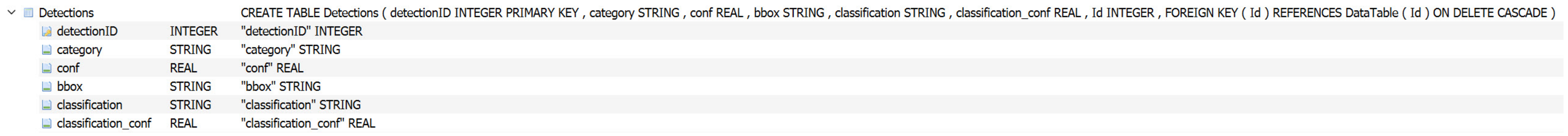
- detectionID 2 identifies a detection on image 174 in the ***DataTable***. Its category is 1 (Animal, as looked up on the ***DetectionsCategory*** table) with a detection confidence of .8. The coordinates are the bounding box around the animal. It classification of 2 identifies it as a Bear as indicated in the ***ClassificationCategory*** table, with a classification confidence of .72 .



DB Browser for SQLite - C:\Users\saulg\Desktop\TestSets\PracticeImageSet-New\TimelapseData.ddb

Table: Detections

	detectionID	category	conf	bbox	classification	classification_conf	Id
	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	2	0.90	0, 0, 0.1145, 0.8694	0	0.66…	173
2	2	1	0.89…	0.42, 0.40, 0.20, 0.29	1	0.72	174
3	3	1	0.88	0.43, 0.40, 0.20, 0.28	1	0.73	175
4	4	1	0.88	0.43, 0.40, 0.19, 0.29	1	0.77	176
5	5	1	0.87	0.57, 0.39, 0.15, 0.25	1	0.78	178
6	6	1	0.03	0.70, 0.40, 0.02, 0.02	1	0.78	178
7	7	1	0.012	0.70, 0.40, 0.02, 0.05	1	0.78	178
8	8	1	0.90	0.56, 0.40, 0.12, 0.25	2	0.87	179
9	9	1	0.12	0.70, 0.38, 0.03, 0.03	2	0.87	179
10	10	1	0.02	0.58, 0.40, 0.10, 0.08	2	0.87	179



Detections		CREATE TABLE Detections (detectionID INTEGER PRIMARY KEY , category STRING , conf REAL , bbox STRING , classification STRING , classification_conf REAL , Id INTEGER , FOREIGN KEY (Id) REFERENCES DataTable (Id) ON DELETE CASCADE)
detectionID	INTEGER	"detectionID" INTEGER
category	STRING	"category" STRING
conf	REAL	"conf" REAL
bbox	STRING	"bbox" STRING
classification	STRING	"classification" STRING
classification_conf	REAL	"classification_conf" REAL

ClassificationCategories

The classification table is mostly used as a lookup table to list classifications, and to correlate a classification number with a human-readable label and description.

The image recognition file contains an entry called ***classification_categories***, which produces zero or more possible classifications of what each detection could be along with an optional description that typically provides further information about that classification (e.g., SpeciesNet includes the species taxa). For example, while a detection may broadly identify something as an animal, a classification may further identify that as a elk with high confidence and perhaps describe it as mammalia;cetartiodactla;cervidae;cervus;canadensis. The classification_categories list all possible entities that the recognizer will consider. Each classification_category comprises an identifying integer, a label and a (perhaps empty) description.

Timelapse reads those values into the table (as illustrated below). As with detection_categories, Timelapse's 'Empty' classification used to identify images that do not contain any classifications is calculated on the fly, rather than included in the table.

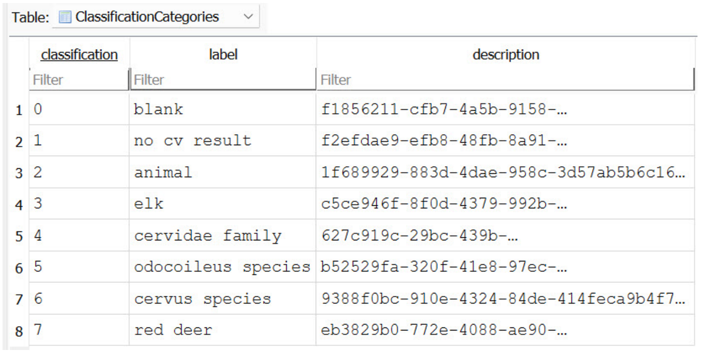
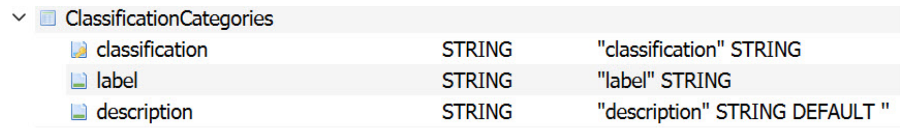


Table: ClassificationCategories

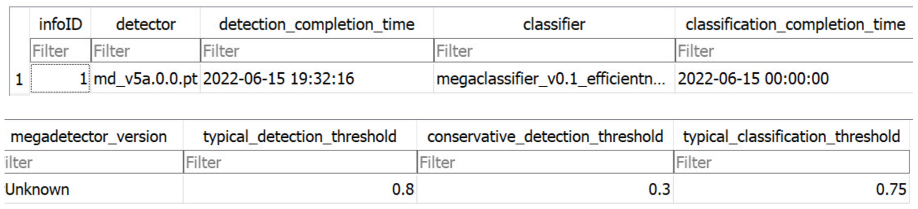
	classification	label	description
	Filter	Filter	Filter
1	0	blank	f1856211-cfb7-4a5b-9158-…
2	1	no cv result	f2efdae9-efb8-48fb-8a91-…
3	2	animal	1f689929-883d-4dae-958c-3d57ab5b6c16…
4	3	elk	c5ce946f-8f0d-4379-992b-…
5	4	cervidae family	627c919c-29bc-439b-…
6	5	odocoileus species	b52529fa-320f-41e8-97ec-…
7	6	cervus species	9388f0bc-910e-4324-84de-414feca9b4f7…
8	7	red deer	eb3829b0-772e-4088-ae90-…



ClassificationCategories		
classification	STRING	"classification" STRING
label	STRING	"label" STRING
description	STRING	"description" STRING DEFAULT ''

Info

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



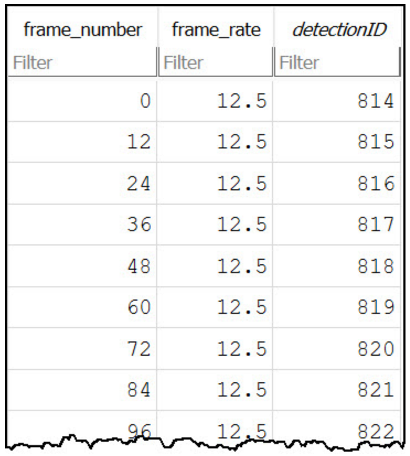
	infoID	detector	detection_completion_time	classifier	classification_completion_time
	Filter	Filter	Filter	Filter	Filter
1	1	md_v5a.0.0.pt	2022-06-15 19:32:16	megaclassifier_v0.1_efficientn…	2022-06-15 00:00:00

megadetector_version	typical_detection_threshold	conservative_detection_threshold	typical_classification_threshold
ilter	Filter	Filter	Filter
Unknown	0.8	0.3	0.75

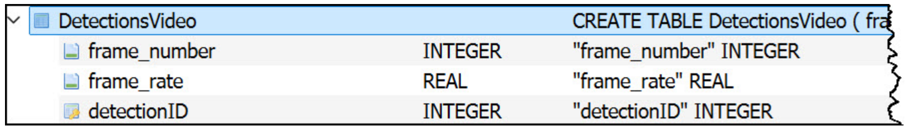
DetectionsVideo Table

The *DetectionsVideo* table is found in the Timelapse *.ddb* file. Recognitions for video files are associated wth a particular frame number (a video can thus have many detections), where each video has a particular frame rate. Timelapse stores these value in this table, where the detectionID (a foreign key) associates these values with a particular detection.

For example, the first entry below shows that a detection with ID 814 is for the 0th frame in the video. The frame_rate is used to calculate the time that frame is presented in the video.



frame_number	frame_rate	*detectionID*
Filter	Filter	Filter
0	12.5	814
12	12.5	815
24	12.5	816
36	12.5	817
48	12.5	818
60	12.5	819
72	12.5	820
84	12.5	821
96	12.5	822



DetectionsVideo		CREATE TABLE DetectionsVideo (fra
frame_number	INTEGER	"frame_number" INTEGER
frame_rate	REAL	"frame_rate" REAL
detectionID	INTEGER	"detectionID" INTEGER

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 '\\'

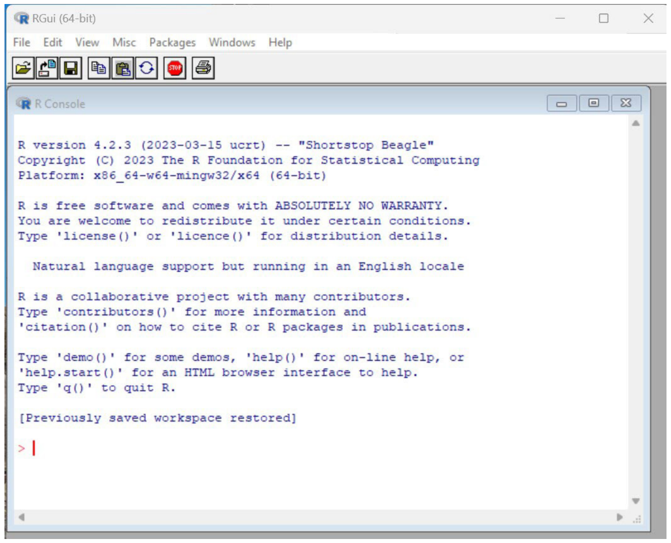
```
conn <- dbConnect(RSQLite::SQLite(),
                  "C:\\Users\\saulg\\Desktop\\PracticeImageSet\\TimelapseData.ddb")
```

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\Deployment1a* 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\\Deployment1a'
       AND Species = 'bobcat'")

# List the contents of bobcatFiles
> bobcatFiles
        RelativePath              File
1 Station1\\Deployment1a  IMG_031.jpg
2 Station1\\Deployment1a   IMG_032.jpg
3 Station1\\Deployment1a   IMG_033.jpg
```



```
R version 4.2.3 (2023-03-15 ucrt) -- "Shortstop Beagle"
Copyright (C) 2023 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Previously saved workspace restored]

> 
```