SISAL - documentation for database managers

This is a documentation for the database managers and addresses the main actions that are performed by the database manager. These are:

- How to upload a new workbook?
- How to replace an old entity?
- How to add a new SISAL chronology?
- How to add a missing hiatus?
- Miscallaneous

The documentation here assumes that the workbooks are cleaned and checked.

The uploading process require you to

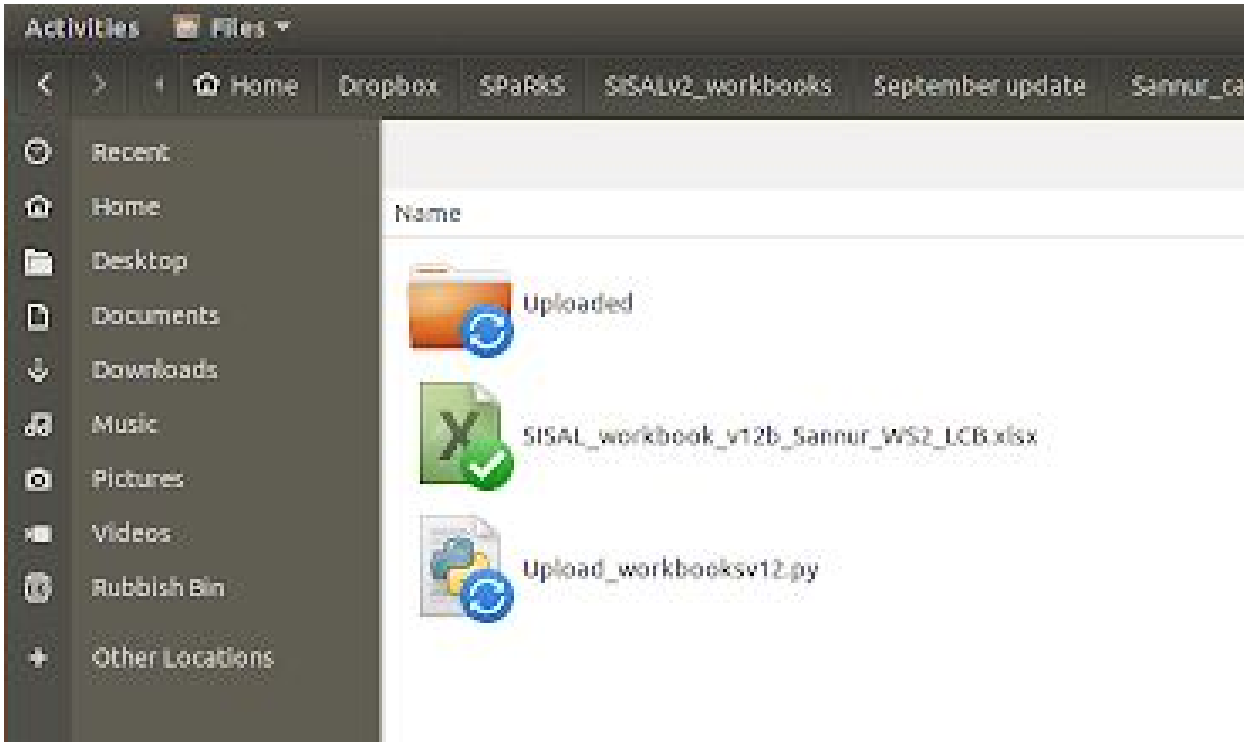
How to upload a workbook?

Setup of the uploading process

This can be done on your own copy of the SISAL database, but if you are editing the live database, you must have write access to the SISAL database. The requirements are:

- write access to the SISAL database
- able to access the SISAL database (require VPN into University of Reading network for live access)
- SISAL database to be running at the host computer (live database is almost always running with very few down time)
- have a version of the uploading script (i.e. Upload_workbooksv12.py)
    - This python script (Python 2.7) require the following modules
        - MySQLdb, numpy, pandas, sys, os, shutil (import XXX)
        - example for installing MySQLdb through anaconda:
            - conda install mysql-python
    - The script has been tested with Python 2 and 3. Please note that previous version used MySQLdb module which is not compatible with Python 3 and I have therefore moved to the mysql.connector module which is compatible with both versions of python.
        - mysql.connector, numpy, pandas, sys, os, shutil (import XXX)
        - example for installing mysql.connector through anaconda:
            - conda install -c anaconda mysql-connector-python
        - example for installing mysql.connector through pip:
            - pip install mysql-connector-python

Within the folder containing the workbooks, there must be a folder called “Uploaded” and the uploading script must be found here.



Upload process

The python script is to be executed from the command line, terminal or the anaconda prompt from the folder as follows:

```
python Upload_workbooksv12.py SISAL_workbook_v12b_Sannur_WS2_LCB.xlsx
root password localhost sisalv2
```

The arguments are in the following order:

- path to workbook (e.g. SISAL_workbook_v12b_Sannur_WS2_LCB.xlsx)
- MySQL username (e.g. root)
- MySQL password (e.g. password)
- MySQL host name (e.g. localhost)
- database name (e.g. sisalv2)

If the excel file is not left open, and there are no errors, the script automatically moves the specified. In the case with the workbooks with no age models, the original_chronology table in the database is supposed to be empty. However, this is not always the case and therefore the

user must check this (the user must look for the last entity_id; in this case entity_id = 691). The following line can be executed in MySQL to check for this:

```
SELECT original_chronology.* FROM sample JOIN original_chronology
USING (sample_id) WHERE entity_id = 691;
```

If this returns a table, this table should be deleted for the cleanliness of the database. This can be done as follows:

SET SQL_SAFE_UPDATES=0;

DELETE FROM original_chronology WHERE sample_id IN (SELECT sample_id FROM (SELECT sample_id FROM sample JOIN original_chronology USING (sample_id) WHERE entity_id = 691) t);

```
SET SQL_SAFE_UPDATES=1;
```

It is very important to turn on SQL_SAFE_UPDATES after this update. Then check again that the data for the entity in the original_chronology table has been removed.

```
SELECT original_chronology.* FROM sample JOIN original_chronology
USING (sample_id) WHERE entity_id = 691;
```

Clean up

Final clean up is to be done manually. These include:

- adding/editing notes in notes table
- clean up the citation(s) and publication_DOI(s) in the reference table
    - make sure that they all follow the same format. citation follows the copernicus pulblication style and the DOI follows “10.1000/xyz123” where possible.
- check that there are no repeated citations. One quick way to find out these repeated citations is to perform the following query:
    - `SELECT publication_DOI, group_concat(ref_id separator', ') as ref_ids FROM reference GROUP BY (publication_DOI) HAVING COUNT(*) > 1;`
    - Mistakes are easy to make here and therefore I suggest backing up the database before doing any changes here. Always recheck that they are really the same citation and if they are, update entity_link_reference to link to the most correct citation and remove the reference from the the reference table. Example(ref_id = 206 and 424 are the same but the formatting in 206 is more correct):
        - `UPDATE entity_link_reference SET ref_id = 206 WHERE ref_id = 424;`

        - `DELETE FROM reference WHERE ref_id = 424;`
- clean up the contact column in the entity table so that the same contact has their names spelt in the same way
    - `SELECT distinct(contact) FROM entity;`
    - sort out the contact column by alphabetical order and look for outliers. If there are:
    - `SELECT * FROM entity;`
    - sort out the contact column by alphabetical order clean them accordingly. Once cleaned, run:
    - `SELECT distinct(contact) FROM entity;`
- clean up the entity_status and corresponding current if need be
    - `SELECT * FROM entity WHERE entity_status IS NULL;`
    - `SELECT * FROM entity WHERE entity_status != 'current' AND corresponding_current IS NULL;`
    - consult the workbooks of the entities with missing entity_status for details.
- check that all references are attached to an entity
    - SELECT * FROM reference WHERE ref_id NOT IN (SELECT ref_id FROM entity_link_reference);

How to upload a new entity from a new site?

This is done as described in the section “How to upload a workbook?”. However, always check that the site is not already in the database.

How to upload a new entity from an existing site?

- Make sure that the site metadata in the workbook is identical to that in the database.
- Follow the procedure as described in the section “How to upload a workbook?”.

How to upload a new record of an old entity (superseding records)

In the case of uploading a new record of an entity which is already in the database, the user must ensure the following:

- entity names of the entity in the database and in the workbook follows the same nomenclature. This must be entityA_year (e.g. entityA_2019 and entityA_2010).
- site metadata in the database and the workbook is the same

You follow the same steps as described in the section “How to upload a workbook?”.

- Once the file is uploaded, check that the entity_status and corresponding_current columns in the entity table and make sure this is updated accordingly. This is most easily done in MySQL Workbench

How to replace an old entity?

In this particular case, you are hoping to replace an old record within the database with a new record.

1. First, you take down the entity_id of the old entity in the database.
2. Secondly, you delete the entity from the database.
3. Third, you follow the instructions as described under “How to upload a new entity from an existing site?”.
4. Once the new entity is uploaded, it will be uploaded to the latest entity_id. You can then query the database for the entity table and change this to the entity_id you have taken note in the first step.

How to add a new SISAL chronology?

This require the user having the R codes to upload the sisal chronology. The functions can be found in **upload_SISAL_agemodels.R.**

The script requires the following libraries to be installed:

- **RMariaDB, openxlsx**

Adding a new type of age model

The steps below assume that the entities have already been uploaded into the SISAL database. If it has not been uploaded, this must be uploaded first.

The example below here will walk you through how to add a new type of age model. As an example here, we are adding a SISAL chronology based on the Bchron age model
The steps are as follows:

1. Create new columns to store this data in the dating table
2. Create new columns to store this data in sisal_chronology table.
3. Uploading SISAL ages into sisal_chronology
4. Fill in the date_used_agemodels in dating table

1. Add new columns to dating table

The columns to be added here would be date_used_Bchron. This column is added to indicate which of these dates were used in constructing the Bchron age model. The basic options in this column are: “yes” or “no”. In the case of more elaborate age models, it may be appropriate to also add the “cannot be performed” option in cases where it is not possible due to the quality of the dates.

```
ALTER TABLE `sisalv2`.`dating`
ADD COLUMN `date_used_Bchron` ENUM('yes', 'no', 'cannot be
performed') NULL AFTER `date_used_linear_regress`;
```

2. Add new columns to sisal_chronology table

The columns to be added here would be Bchron_age, Bchron_age_uncert_post and Bchron_age_uncert_neg. These columns are added to store the ages output from the age models. The Bchron_age columns would be of a DOUBLE (numeric) type and the uncertainty columns would be an unsigned DOUBLE (positive numbers) type.

```
ALTER TABLE `sisalv2`.`sisal_chronology`
ADD COLUMN `Bchron_age` DOUBLE NULL DEFAULT NULL AFTER
`linear_regress_age_uncert_neg`,
ADD COLUMN `Bchron_age_uncert_pos` DOUBLE UNSIGNED NULL DEFAULT NULL
AFTER `Bchron_age`,
ADD COLUMN `Bchron_age_uncert_neg` DOUBLE UNSIGNED NULL DEFAULT NULL
AFTER `Bchron_age_uncert_pos`;
```

3. Uploading SISAL ages into sisal_chronology

Input table with only depths

This is currently done through the `input_sisal_chronology_depth` function defined in upload_SISAL_agemodels.R. Here the entity must have been uploaded with the corresponding sample data/table already in the SISAL database, with the same depth_samples. The entity_id must also be noted down (in this case entity_id = 691).

In R:

```
# Load prerequisite libraries and source upload_SISAL_agemodels.R
library(RMariaDB)
library(openxlsx)
```

```
source("upload_SISAL_agemodels.R")
# "upload_SISAL_agemodels.R" can be any path to the file

# Connect to SISAL database
mydb = dbConnect(MariaDB(), user='root',
                 password='password',
                 dbname='sisalv2',
                 host='localhost')

# Read the table with the Bchron ages
# Note that every row must be filled in
input_sisal_chronology_depth(cnx = mydb,
                      entity_id = 691,
                      agemodeltype = 'Bchron', # Prefix of column
name
                      tb = tb_bchron, # dataframe name
                      tb_depth_col = 'depth_sample', # name of depth
column in dataframe
                      tb_age_col = 'bchron_age', # name of age column
in dataframe
                      tb_age_uncert_pos_col =
'bchron_age_uncert_pos',
                      tb_age_uncert_neg_col =
'bchron_age_uncert_neg',
                      outputcsv =
'bchron_chronology_withsample_id.csv',
                      execution = F)

# Check the outputcsv file to see if this is good

input_sisal_chronology_depth(cnx = mydb,
                      entity_id = 691,
                      agemodeltype = 'Bchron',
                      tb = tb_bchron,
                      tb_depth_col = 'depth_sample',
                      tb_age_col = 'bchron_age',
                      tb_age_uncert_pos_col =
'bchron_age_uncert_pos',
                      tb_age_uncert_neg_col =
'bchron_age_uncert_neg',
                      outputcsv =
'bchron_chronology_withsample_id.csv',
                      execution = T)
```

```
# Close connection
dbDisconnect(mydb)
```

Input table with sample_id(s)

This is currently done through the input_sisal_chronology_sampleid function defined in upload_SISAL_agemodels.R. Here the entity must have been uploaded with the corresponding sample data/table already in the SISAL database with the same sample_ID. The advantage here is that you can upload data from multiple entities which are stored in one table all in one go.

In R:

```
# Load required libraries
-----------------------------------------------#####
library(RMariaDB)
library(openxlsx)
source('codes/upload_SISAL_agemodels.R')

# Connect to SISAL database
mydb = dbConnect(MariaDB(), user='root',
                 password='password',
                 dbname='test',
                 host='localhost')


# Read in the csv file
tb_bchron <- read.csv( 'bchron_chronology_withsample_id.csv')

input_sisal_chronology_sampleid(cnx = mydb,
                            agemodeltype = 'Bchron', # Prefix of
column name
                            tb = tb_bchron, # dataframe name
                            tb_sample_id_col = 'sample_id', # name of
depth column in dataframe
                            tb_age_col = 'Bchron_age', # name of age
column in dataframe
                            tb_age_uncert_pos_col =
'Bchron_age_uncert_pos',
                            tb_age_uncert_neg_col =
'Bchron_age_uncert_neg')
# Close connection
dbDisconnect(mydb)
```

Final check

Check that this is correct by running the following query in MySQL (i.e. through MySQL workbench)

```
SELECT * FROM sample JOIN sisal_chronology USING (sample_id) WHERE
entity_id = 691;
```

4. Fill in the date_used_agemodels in dating table

This is currently done through the update_date_used_agemodel function defined in upload_SISAL_agemodels.R. Here the entity must have already been uploaded with the corresponding dating data/table already in the SISAL database. The input table must contain dating_id from the dating table in the SISAL database.

In R:

```
# libraries
library(RMariaDB)
source('upload_SISAL_agemodels.R')
# Connect to SISAL database
mydb = dbConnect(MariaDB(), user='root',
                 password='password',
                 dbname='sisalv2',
                 host='localhost')
# Read in the table where date_used_agemodels are to be updated
tb_dateused <- read.csv('sannur_date_used_bchron_bacon.csv')
# Run the function
update_date_used_agemodel(cnx = mydb, # connection
                          tb = tb_dateused, # dataframe containing
date_used information
                          dating_id_col = 'dating_id', #name of
dating_id column in dataframe
                          tbdate_used_col = 'date_used_Bchron', #
name of date_used_agemodel column in dataframe
                          dbdate_use_col = 'date_used_Bchron') #
name of date_used_agemodel column in database
# close connection
dbDisconnect(mydb)
```

If one row is filled in for that age model for that entity, the other rows of that entity cannot be NULL. Run query below for date_used_Bchron:

```
SELECT distinct(entity_id) FROM dating WHERE entity_id IN (SELECT
distinct(entity_id) FROM dating WHERE date_used_Bchron IS NOT NULL)
AND date_used_Bchron IS NULL;
```

5. Check for integrity
Run the following SQL queries to see if date_used is filled in when the age model is filled in the sisal_chronology table

```
# COPRA
SELECT * FROM dating WHERE date_used_COPRA IS NULL AND entity_id IN
(SELECT distinct(entity_id) FROM sample JOIN sisal_chronology USING
(sample_id) WHERE COPRA_age IS NOT NULL);
# Bchron
SELECT * FROM dating WHERE date_used_Bchron IS NULL AND entity_id IN
(SELECT distinct(entity_id) FROM sample JOIN sisal_chronology USING
(sample_id) WHERE Bchron_age IS NOT NULL);
# Bacon
SELECT * FROM dating WHERE date_used_Bacon IS NULL AND entity_id IN
(SELECT distinct(entity_id) FROM sample JOIN sisal_chronology USING
(sample_id) WHERE Bacon_age IS NOT NULL);
# linear
SELECT * FROM dating WHERE date_used_linear IS NULL AND entity_id IN
(SELECT distinct(entity_id) FROM sample JOIN sisal_chronology USING
(sample_id) WHERE linear_age IS NOT NULL);
# linear
SELECT * FROM dating WHERE date_used_linear_regress IS NULL AND
entity_id IN (SELECT distinct(entity_id) FROM sample JOIN
sisal_chronology USING (sample_id) WHERE linear_regress_age IS NOT
NULL);
```

Adding SISAL chronology where the age model type exists

This is identical to the section “Adding a new type of age model” but only subsections “3. Uploading SISAL ages into sisal_chronology” and “4. Fill in the date_used_agemodels in dating table” are to be performed.

How to add a missing actively growing event?

Always back up the database before performing this query (A prefers if you do that manually by querying the dating table and adding a new row):

```
INSERT INTO `sisalv2`.`dating` (`entity_id`, `date_type`,
`depth_dating`, `date_used`, `corr_age`, `corr_age_uncert_pos`,
`corr_age_uncert_neg`) VALUES ('348', 'Event; actively forming', '0',
'yes', '-60', '0', '0');
```

Query the database so that there is only one 'Event; actively forming' per entity:

```
SELECT entity_id, count(*) as activelyforming_counter FROM dating
WHERE date_type = 'Event; actively forming' GROUP BY (entity_id)
HAVING COUNT(*) > 1;
SELECT * FROM dating WHERE entity_id = 179;
```

How to add a missing hiatus?

Currently, the addition of missing hiatuses are being done manually. For example:

```
# entity_id = 71, hiatus at depth = 168.9
INSERT INTO `sisalv2`.`dating` (`entity_id`, `date_type`,
`depth_dating`, `date_used`) VALUES ('71', 'Event; hiatus', '168.9',
'yes');
INSERT INTO `sisalv2`.`sample` (`entity_id`, `depth_sample`) VALUES
('71', '168.9');
# sample_id = 331567
INSERT INTO `sisalv2`.`hiatus` (`sample_id`, `hiatus`) VALUES
('331567', 'H');
```

Make sure all date_used is filled in

```
SELECT * FROM dating WHERE date_used IS NULL;
```

Make sure that all hiatuses in the dating table is also present in the sample table. I check for this in R:

```
# Load prerequisite libraries
library(RMariaDB)
# Connect to SISAL database
cnx = dbConnect(MariaDB(), user='root',
                password='password',
                dbname='sisalv2',
                host='localhost')
# query to get all the entities
query <- "SELECT * FROM entity;"
```

```
dt_ent <- dbGetQuery(cnx, query)
entity_id <- dt_ent$entity_id

# Loop through each entity to look for hiatuses that do not match
for (i in entity_id){
  # print(i)
  query1 <- paste("SELECT * FROM dating WHERE date_type = 'Event;
hiatus' AND depth_dating NOT IN (SELECT depth_sample FROM sample JOIN
hiatus USING (sample_id) WHERE entity_id = '",
                i,"') AND entity_id = '",i,"';", sep = '')
  # print(query)
  if (dim(dbGetQuery(cnx, query1))[1] > 0){
     print(paste('entity_id = ', i, ' has hiatus(es) in dating table
which are not found in the respective sample table. This is possible
if there hiatuses outside of the sample range.'))
  }
  query2 <- paste("SELECT * FROM sample JOIN hiatus USING (sample_id)
WHERE depth_sample NOT IN (SELECT depth_dating FROM dating WHERE
date_type = 'Event; hiatus' AND entity_id = '",
                i,"') AND entity_id = '",i,"';", sep = '')
  # print(query)
  if (dim(dbGetQuery(cnx, query2))[1] > 0){
     print(paste('entity_id = ', i, ' has hiatus(es) in sample table
which are not found in the respective entity table.'))
  }
}
```

Miscellaneous

Minor edits

Minor edits can be done within MySQL workbench. These minor edits include edits such as:

- Correcting information in the site metadata table
- Correcting information in the entity table

Custom codes

Custom codes can be written to query the database. For example, see the R codes under the “How to add a missing hiatus?” section.

Post upload checklist

1. Check that site names are correct. If site name contains the word ‘cave’ in it, it must not be capitalised (‘cave’ not ‘Cave’)
2. Check that hiatuses in sample table is found in dating table of every entity
3. Check that citations in reference table are cleaned
4. Check that publication_DOI(s) are cleaned
5. Check that contact names are cleaned
6. Check that data_DOI_URL(s) are cleaned (if link is from NOAA, make sure it is in the “https://www.ncdc.noaa.gov/paleo/study/10450” format if possible)
7. Check that entity_status is updated
8. Check that corresponding current is correct
9. Check that composite_link_entity is correct
10. Check that sisal_chronology is uploaded correctly
11. Check that date_used and date_used_agemodels are filled in accordingly
12. Check that notes have been updated