

# Notes on MySQL

This html file is made as part of the documentation for SISAL version 3 as of the 11th of August 2023.

## How to install MySQL

Install MySQL community server and MySQL Workbench.

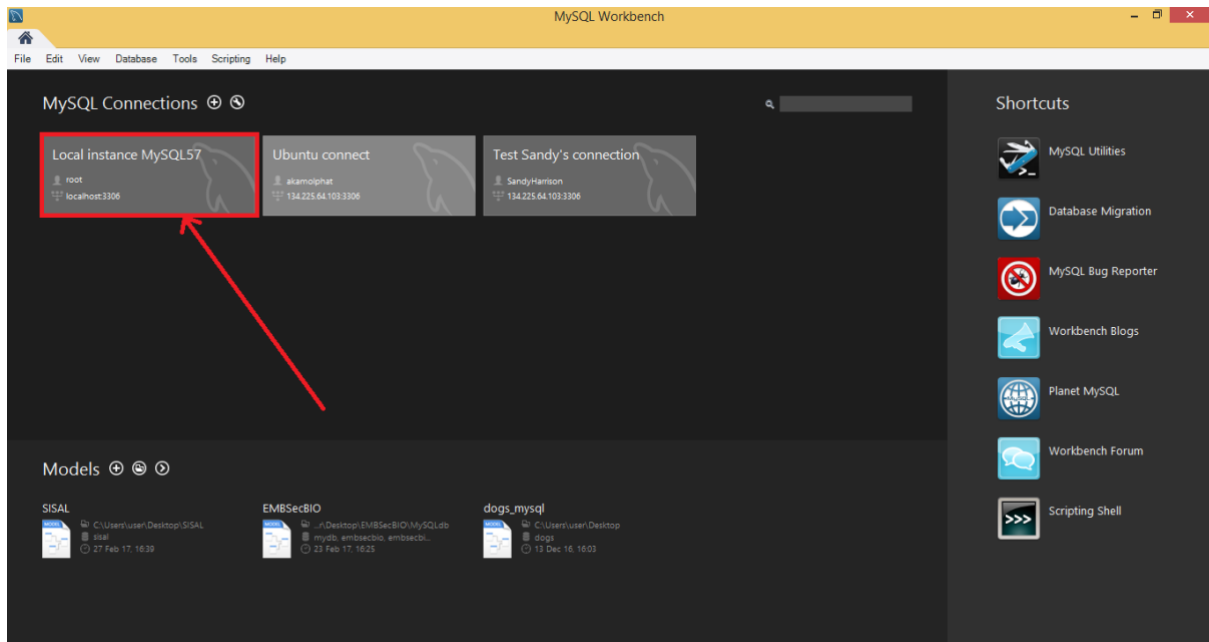
see: <https://dev.mysql.com/downloads/mysql/>

MySQL Workbench will help you visualise the database better.

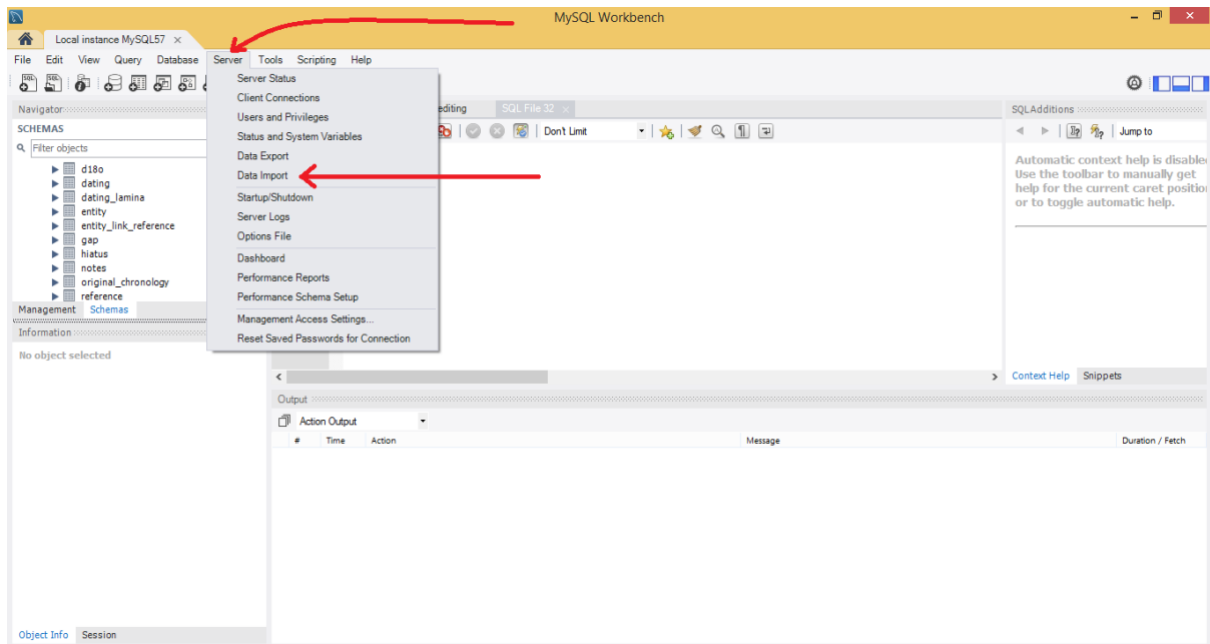
## How to import SISAL database into MySQL through MySQL workbench

Please note that these screenshots are from MySQL Workbench version 6.3.8. on Windows 7. Each version of MySQL Workbench in different operating system may appear slightly differently but the overall process is the same.

Step 1. Open MySQL Workbench and connect to MySQL server.



Step 2. Import database: Server>Data Import



### Step 3. Import database:

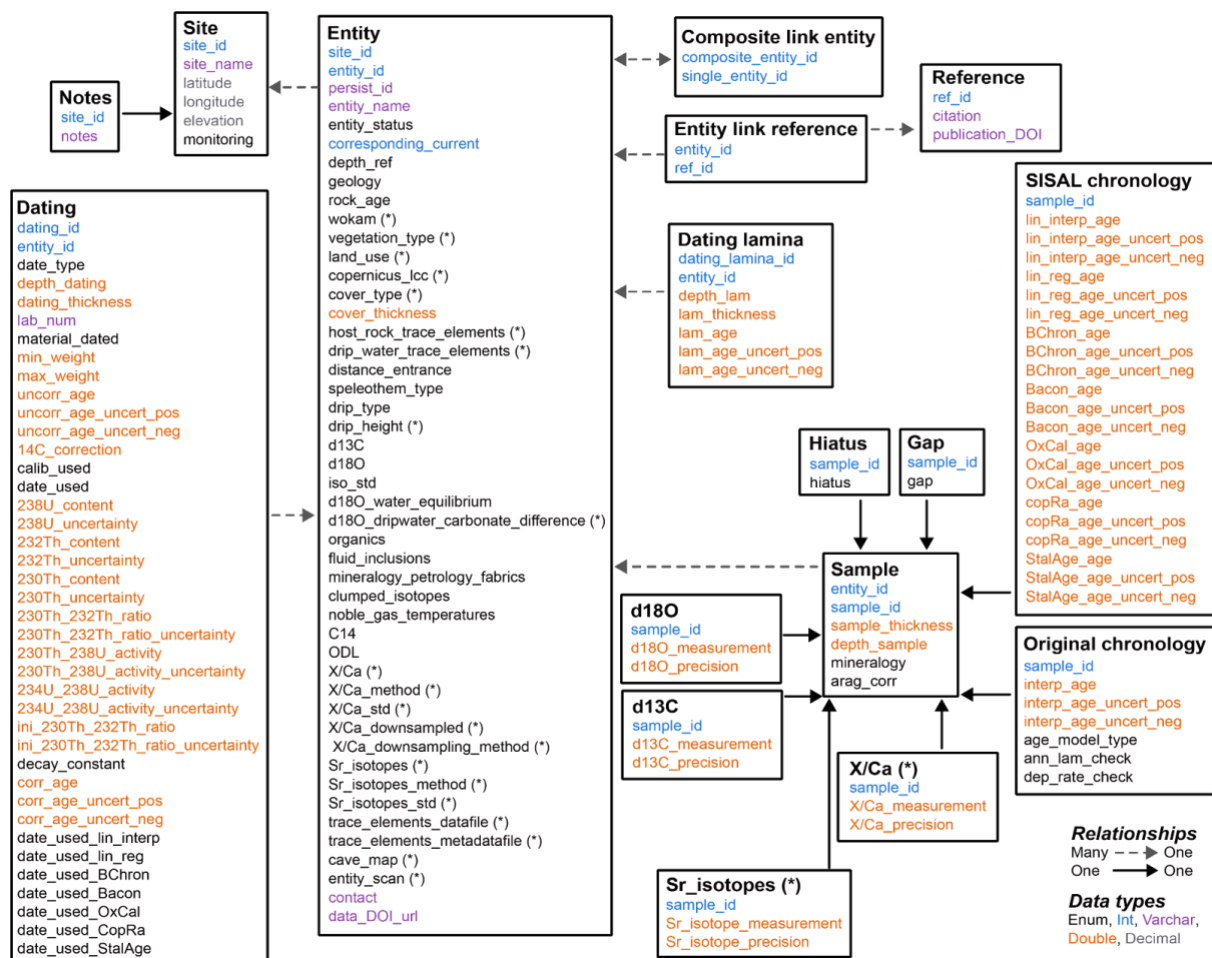
1. Select "Import from Self-Contained file"
2. Browse to find the SQL file you have downloaded.
3. Press New to import into a new 'schema'/database. Call this anything you want (i.e. sisalv3).
4. Start import

### Step 4. Run queries:

1. This is where you type your queries.
2. This is how you run the queries. You can also press **ctrl+enter**.
3. You can save these queries in an SQL file which you can view later (SQL file is essentially a text file).

## Database structure

The structure of the database can be found below:



Please note that in different operating systems, table names may appear slightly differently. In UNIX systems, uppercases in table names is shown as uppercases, but not in Windows system (table d18O will be seen as d18o). However, this does not effect the queries at all as the queries are case insensitive.

## Example queries

These set of queries assume that the database has been downloaded and is in MySQL under the name: **sisalv3**

1. Tells MySQL that we are using the 'sisalv3' database. By doing this at the start, you will be able to skip the database name in individual queries.

```
USE sisalv3;
```

2. Select everything from any given table.

```
SELECT * from sisalv3.site;
```

This query translates to 'select everything from 'sisalv3' database, 'site' table. Try this same query but replace 'site' with whichever table you want.

### 3. Count how many entities are in the database.

```
SELECT count(*) from sisalv3.entity;
```

### 4. Select two variables from one table (ie site name and coordinates).

```
SELECT site.site_name, site.latitude, site.longitude  
FROM sisalv3.site;
```

### 5. Select two variables from two different tables (ie table with site name and the corresponding entity names).

```
SELECT entity.entity_name, site.site_name FROM entity, site WHERE entity.si  
te_id = site.site_id;  
  
SELECT entity.entity_name, site.site_name FROM site LEFT JOIN entity USING(  
site_id);
```

Note that there are different types of 'JOIN' statement. If not specified (only 'JOIN', the default 'INNER JOIN' is used). see: [https://www.w3schools.com/sql/sql\\_join.asp](https://www.w3schools.com/sql/sql_join.asp)

### 6. Rename columns.

```
SELECT site_name as `site name`, longitude, latitude, elevation as `masl` F  
ROM site;
```

### 7. Extract note linked to site name (instead of site\_id).

```
SELECT site.site_name, notes.notes from site LEFT JOIN notes USING(site_id)
;
```

## 8. Extract dating table as in the workbook for a specific entity\_id.

```
SELECT dating.* FROM dating, entity WHERE entity.entity_id = dating.entity_id AND entity.entity_id =254;
```

9a. Extract site.site\_name, entity.entity\_name only for composites and entity\_name of entities forming the comp. Note that data on the individual records composing the Soreq\_composite is not available. 'compe' is the name given to the entity table when linking entity to the composite\_link\_entity table (for composite entities) while 'single\_ent' is the name given to the entity table when linking entity and composite\_link\_entity (for single entities). This is done because there are only the column names will be repeated and we will not be able to distinguish which entity name are the composites or the individual entities.

```
SELECT site.site_name, compe.entity_name as 'Composite entity name', single_ent.entity_name as 'Single entity name' FROM site LEFT JOIN entity compe USING(site_id) LEFT JOIN composite_link_entity ON compe.entity_id = composite_entity_id LEFT JOIN entity single_ent on composite_link_entity.single_entity_id = single_ent.entity_id WHERE compe.speleothem_type = 'composite';
```

9b. Extract site.site\_name, entity.entity\_name only for composites and entity\_name of entities forming the comp, but one row per composite entity. Note that data on the individual records composing the Soreq\_composite is not available.

```
SELECT site.site_name, compe.entity_name as 'Composite entity name', GROUP_CONCAT(single_ent.entity_name ORDER BY single_ent.entity_name SEPARATOR ',') as 'Single entity names' FROM site LEFT JOIN entity compe USING(site_id) LEFT JOIN composite_link_entity ON compe.entity_id = composite_entity_id LEFT JOIN entity single_ent on composite_link_entity.single_entity_id = single_ent.entity_id WHERE compe.speleothem_type = 'composite' GROUP BY compe.entity_id;
```

10. Create huge table with the following variables: site.site\_name, site.latitude, site.longitude, entity.entity\_name, sample.sample\_id, d18O.d18O\_measurement, hiatuses and original\_chronology.interp\_age with its uncertainties and remove from this table rows for which original\_chronology.interp\_age is NULL.

```
SELECT site.site_name, site.latitude, site.longitude, entity.entity_name, sample.sample_id, d18O.d18O_measurement, hiatus.hiatus, original_chronology.interp_age, original_chronology.interp_age_uncert_pos, original_chronology.interp_age_uncert_neg FROM site LEFT JOIN entity USING(site_id) LEFT JOIN sample USING(entity_id) LEFT JOIN original_chronology USING(sample_id) LEFT JOIN hiatus USING(sample_id) LEFT JOIN d18O USING(sample_id) WHERE interp_age IS NOT NULL;
```

11. Extract d18O\_measurements, hiatuses, interpolated ages and its uncertainties for all entities from "Liang Luar cave".

```
SELECT entity.entity_name, hiatus.hiatus, original_chronology.interp_age, original_chronology.interp_age_uncert_pos, original_chronology.interp_age_uncert_neg, d18O.d18O_measurement FROM site LEFT JOIN entity USING(site_id) LEFT JOIN sample USING(entity_id) LEFT JOIN hiatus USING(sample_id) LEFT JOIN d18O USING(sample_id) LEFT JOIN original_chronology USING(sample_id) WHERE site.site_name = 'Liang Luar cave';
```

12. Extract d18O\_measurements, hiatuses, interpolated ages and its uncertainties for all calcite samples which d18O water is in equilibrium and has interpolated ages.

```
SELECT entity.entity_name, hiatus.hiatus, original_chronology.interp_age, original_chronology.interp_age_uncert_pos, original_chronology.interp_age_uncert_neg, d18O.d18O_measurement FROM site LEFT JOIN entity USING(site_id) LEFT JOIN sample USING(entity_id) LEFT JOIN hiatus USING(sample_id) JOIN d18O USING(sample_id) JOIN original_chronology USING(sample_id) WHERE sample.mineralogy = 'calcite' AND entity.d18O_water_equilibrium = 'yes' AND original_chronology.interp_age IS NOT NULL;
```

13. Extract d18O\_measurements, hiatuses, interpolated ages and its uncertainties for all aragonite samples which have been corrected for and have interpolated ages.

```
SELECT entity.entity_name, hiatus.hiatus, original_chronology.interp_age, original_chronology.interp_age_uncert_pos, original_chronology.interp_age_un
```

```
cert_neg, d18O.d18O_measurement FROM site LEFT JOIN entity USING(site_id) LEFT JOIN sample USING(entity_id) LEFT JOIN hiatus USING(sample_id) JOIN d18O USING(sample_id) JOIN original_chronology USING(sample_id) WHERE sample.mineralogy = 'aragonite' AND sample.arag_corr = 'yes' AND original_chronology.interp_age IS NOT NULL;
```

14. Extract d18O\_measurements, hiatuses, interpolated ages and its uncertainties from "LR07-E1", from "Liang Luar cave". Note that we should always specify the site name as there could be entities with the same name from multiple sites.

```
SELECT entity.entity_name, hiatus.hiatus, original_chronology.interp_age, original_chronology.interp_age_uncert_pos, original_chronology.interp_age_uncert_neg, d18O.d18O_measurement FROM site LEFT JOIN entity USING(site_id) LEFT JOIN sample USING(entity_id) LEFT JOIN hiatus USING(sample_id) LEFT JOIN d18O USING(sample_id) LEFT JOIN original_chronology USING(sample_id) WHERE site.site_name = 'Liang Luar cave' AND entity.entity_name = 'LR07-E1';
```

15. Extract details of entities with more than one date type but not including the 'Events' or unknown.

```
SELECT entity.* FROM entity WHERE entity_id IN (SELECT entity_id FROM (SELECT * FROM dating WHERE date_type NOT LIKE '%Event%' AND date_type != 'unknown') AS T GROUP BY entity_id HAVING count(distinct(T.date_type)) > 2);
```

16. Extract details of entities and dating information which were used in author's age model reconstruction and the date type is of U/Th methods ('MC-ICP-MS U/Th', 'ICP-MS U/Th Other', 'Alpha U/Th', 'TIMS' and 'U/Th unspecified') and the speleothem type is not flowstone. Note that this query is relatively long as I queried for the hiatuses as well. If we only query this once, we will only get the dates but miss out the 'Events' which may be useful for age model reconstruction.

```
SELECT entity.site_id, entity.entity_name, dating.date_type, dating.depth_dating, dating.corr_age, dating.corr_age_uncert_pos, dating.corr_age_uncert_neg FROM entity LEFT JOIN dating USING(entity_id) WHERE entity_id IN (SELECT entity_id FROM entity LEFT JOIN dating USING(entity_id) WHERE dating.date_used = 'yes' AND (dating.date_type LIKE '%U/Th%' OR dating.date_type = 'TIMS')) AND entity.speleothem_type != 'flowstone') AND dating.date_used = 'yes' AND (dating.date_type LIKE '%U/Th%' OR dating.date_type = 'TIMS' OR dating.date_type LIKE '%Event%');
```

17. Extract list of entities for which dating\_lamina exists and original\_chronology.annual\_lam\_check is not "unknown".

```
SELECT entity.* FROM entity WHERE entity_id in (SELECT distinct(entity_id)
FROM entity RIGHT JOIN dating_lamina USING(entity_id) LEFT JOIN sample USING(
entity_id) LEFT JOIN original_chronology USING(sample_id) WHERE ann_lam_c
heck != 'unknown');
```

18. Extract list of all entities for which interpolated ages are not available (partially or totally, omitting samples that only provide information on hiatuses).

```
SELECT * FROM entity WHERE entity_id in (SELECT distinct(entity_id) FROM sa
mple LEFT JOIN original_chronology USING(sample_id) LEFT JOIN hiatus USING(
sample_id) WHERE interp_age IS NULL AND hiatus IS NULL);
```

19. Extract site name, entity name, date type, depths, depth reference point, corrected age of dates used in the original age models for LR07-E1 from Liang Luar cave.

```
SELECT site.site_name, entity.entity_name, dating.date_type, dating.depth_d
ating, entity.depth_ref, dating.corr_age FROM site LEFT JOIN entity USING(s
ite_id) LEFT JOIN dating USING(entity_id) WHERE dating.date_used = 'yes' AN
D entity_name = 'LR07-E1' AND site_name = 'Liang Luar cave';
```

20. Extract site details and the average d18O measurements.

```
SELECT site.*, count(distinct(entity.entity_id)) as 'entity_counts', avg(d1
80.d18O_Measurement) as 'mean_d18O' FROM site LEFT JOIN entity USING (site_
id) LEFT JOIN sample USING (entity_id) LEFT JOIN d18O USING (sample_id) GRO
UP BY site.site_id;
```

21. Extract d18O samples which have ages younger than 22000 years old and older than 1000 years old (BP 1950). This type of query works with all columns which are numeric.



```
SELECT site.site_name, entity.entity_name, hiatus.hiatus, original_chronology.interp_age, original_chronology.interp_age_uncert_pos, original_chronology.interp_age_uncert_neg, d180.d180_measurement FROM site LEFT JOIN entity USING(site_id) LEFT JOIN sample USING(entity_id) LEFT JOIN hiatus USING(sample_id) LEFT JOIN d180 USING(sample_id) LEFT JOIN original_chronology USING(sample_id) WHERE original_chronology.interp_age <= 22000 AND original_chronology.interp_age >= 1000;
```

## Additional queries

### 22. Selecting specific columns from the site table in the database.

```
SELECT site_name, latitude from sisalv3.site;  
SELECT site.site_name, site.latitude from sisalv3.site;
```

These two queries translate to 'select site\_name and latitude from 'sisalv3' database, 'site' table. The second query specifies exactly what type table the column is in. This is not necessary when all the column names are unique, but this can be a problem when you join tables which contains the same column names.

### 23. Selecting information from a joint site and entity table.

```
SELECT * from sisalv3.site JOIN sisalv3.entity USING(site_id);  
SELECT site.site_name, entity.entity_name from sisalv3.site JOIN sisalv3.entity USING(site_id);
```

The first query simply translates to 'select everything from site table joined to entity table using the site\_id'.

The second query only selects the site name and the entity name.

### 24. Selecting entity information from a joint site and entity table where the sites are at elevations above 500 metres and contain carbon isotope data.

```
SELECT entity.* from sisalv3.site LEFT JOIN sisalv3.entity USING(site_id) WHERE site.elevation >= 500 AND entity.d13C = 'yes';
```

This query translates to "select all columns from entity table from 'site' table joined with 'entity' table using the 'site\_id' when the 'elevation' column in the 'site' table is greater than 500 metres and the 'd13C' column in the 'entity' table is 'yes'."

25. Selecting all the distinct contacts in the database.

```
SELECT distinct(contact) from sisalv3.entity WHERE entity.d13C = 'yes';
```

26. Selecting all dating information in the database where the dating methods are U/Th or TIMS.

```
SELECT * FROM dating WHERE (date_type LIKE '%U/Th%' OR date_type = 'TIMS');
```

Selecting all dating information in the database with lamination events.

```
SELECT * FROM dating WHERE (date_type LIKE '%Event%laminations');
```

## Additional details

There are several connectors to MySQL in different programming language. Some of these connectors may be installed during the installation process. Please

check: <https://dev.mysql.com/doc/refman/5.7/en/connectors-apis.html>

The table in the link above also list the 'APIs'/modules in different languages

Python: check the "MySQLdb" module

R: check the "RMySQL" module