Database tutorial

1. Introduction

In order to keep all our data, including information about sequences, samples, primers etc. organized, we have created a MySQL database that runs on the server of the zoological institute, evo-lutra. The database is intended as a central, permanent repository, from which data can be extracted in Excel format for temporary use. Being hosted on evo-lutra, the database benefits from this server's great backup system. In general, databases can be accessed and modified in a number of ways.
This could be using a command line tool such as the Terminal on Macintosh computers, or the Command shell in Windows. The syntax for communication with the database is SQL (Structured Query Language), which consists of a limited number of commands and is relatively easy to learn (see below).
Another option to access and modify databases would be through web interfaces written in HTML, but including code in PHP for dynamic presentation of content that is queried, again, using SQL that is also integrated in the webpage's source code. Many databases that you find on the web (e.g. Genbank) work that way. Your entries in the search fields are translated into SQL commands by the PHP code, and sent to the database. Query results are then again translated into HTML and presented in the browser.
Fortunately, there is a third possibility to access and modify databases, that is far easier and more user-friendly than the other two possibilities. Software such as Sequel Pro provides intuitive graphical user interfaces that allow easy access to the database. Behind the scenes, these programs also use SQL to communicate with the database, and Sequel Pro gives the user the possibility to enter SQL queries directly for more advanced queries.

2. Database design

MySQL databases are generally composed of a number of tables that are connected to each other by one-to-one or one-to-many links. Each table has a number of fields, and every entry in that table stores information for some or all of these fields. For example, a table named 'primer' is intended to store information about primers. It may contain fields for the primer sequence, the primer's fluorescent label, its direction, and so on. For every primer added to this table, this information may or may not be available (it may not have a fluorescent label at all), and correspondingly, these fields will be filled for that particular primer, or they will remain empty.
Every table contains a field that represents the primary key (PK) with which entries of that table can be referenced. Therefore, this field needs to uniquely identify a particular entry. One could assign the PK to the name of each entry (e.g. the primer name) if one would be absolutely sure that no two primers will ever have the same name, otherwise data retrieval could be confounded by the two identical names. However, a more secure and straight-forward way of assigning PKs is to us a separate field only for this purpose that is generally called 'id' and that is nothing but a number that is auto-incremented by the software.
Some tables also contain fields to which foreign keys (FK) are assigned. These represent links to the PKs of other tables. For example, if the id (the PK) of a specimen would be "123", then the field "specimen_id" (the FK) of a sequence isolated from this specimen should also contain the entry "123" to make clear that this sequence belongs to that particular specimen.
Fig. 1 shows the overall design of our database, and how the different tables are connected to each other. The database design is flexible to a certain degree, if you'd like to change something, for example add a table for quantitative data on RNA expression, etc, please let me know.

3. First steps

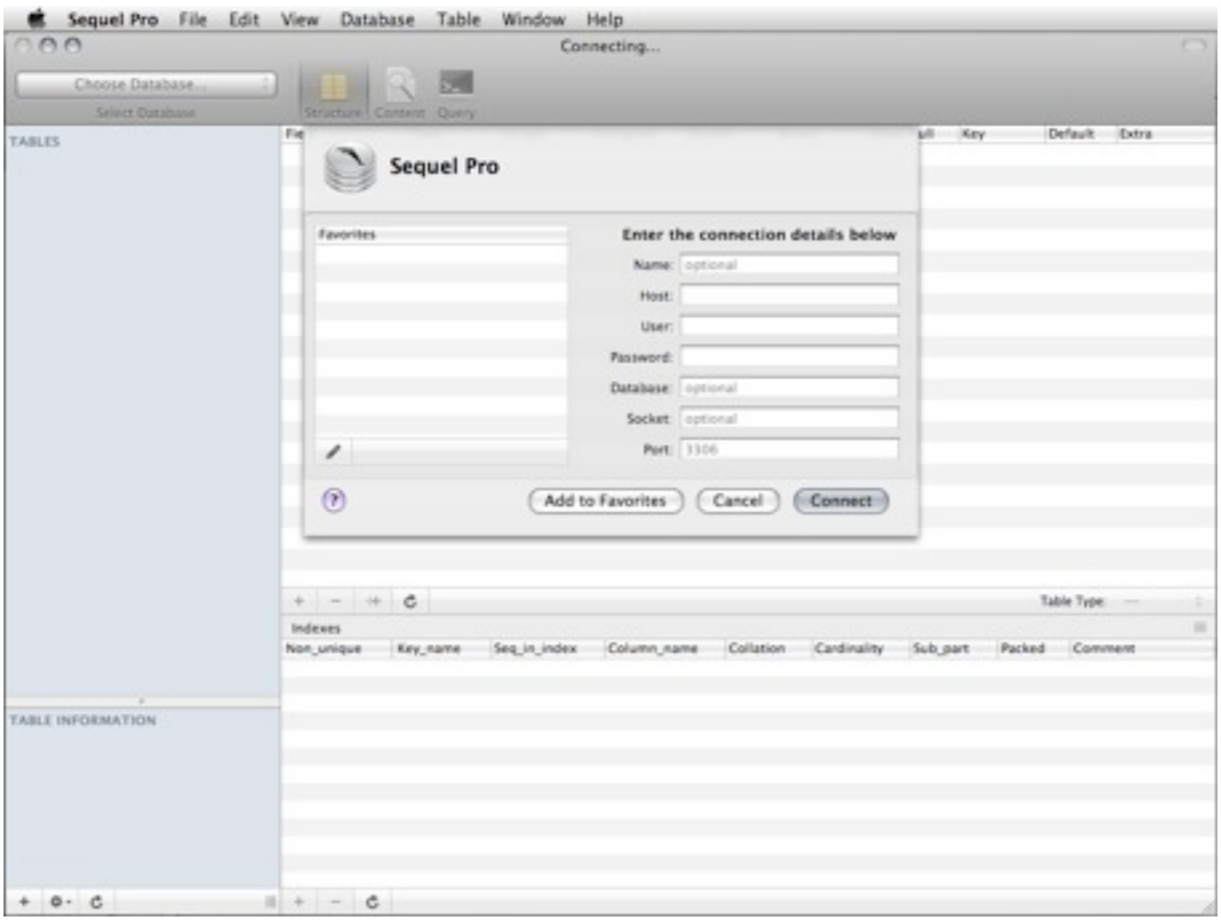
To start off, you should visit www.sequelpro.com to download the latest version of the free program Sequel Pro. This software provides a user-friendly graphical interface with which the database can be accessed and modified. Install Sequel Pro on your machine.

In order to access the database, you will also need an account on evo-lutra. If you don't have a login yet, contact Lukas Zimmermann (mailto:lukas.zimmermann@unibas.ch) to get one. If you've installed your backup system with Time Machine already, you do have such an account.

Once you've got both, use the Terminal to login to evo-lutra and create an SSH tunnel. This tunnel will connect the port on which MySQL runs on evo-lutra to a specified port on your machine. The command to do this is:

```
ssh -NC your_username_on_evo-lutra@evo-lutra.zoo.unibas.ch -L 8888:localhost:3306
```

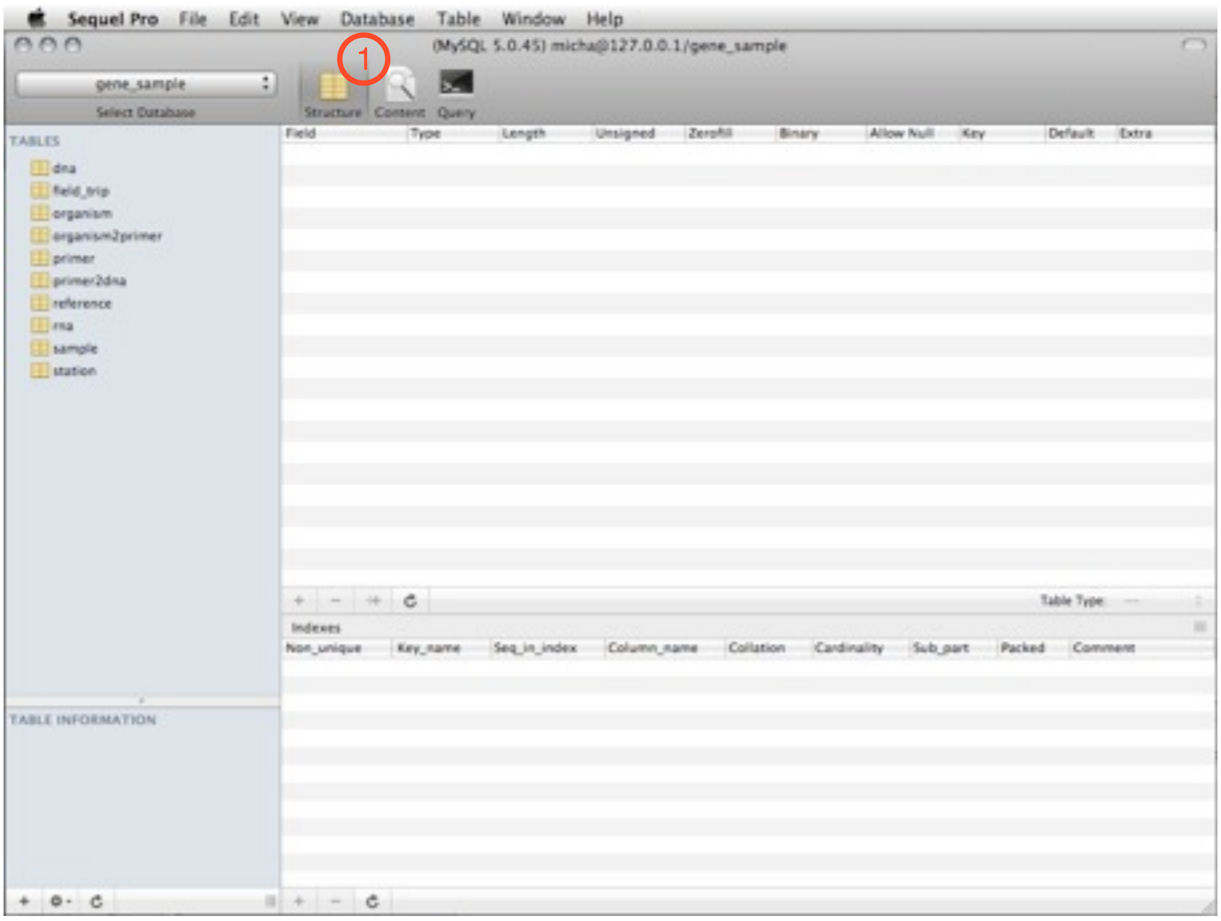
where you replace "your_username_on_evo-lutra" with your actual username. 3306 is the MySQL port on evo-lutra, 8888 is just a random port on your machine that proved to work. You will be prompted for you password for evo-lutra. Type it. There will be no further response from the Terminal window, however experience shows that the tunnel usually works. Leave the Terminal window open in the background and start Sequel Pro. You'll see the following window:



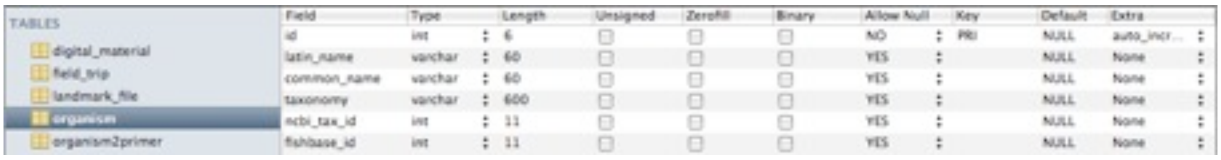
Enter the following connection details:

Name: leave this empty
Host: 127.0.0.1
User: walterlab
Password: arschgeweih
Database: gene_sample
Socket: leave this emtpy
Port: 8888
If you click 'Add to Favorites' you won't have to type this again.

The database tables will then appear in the left column (names may have changed):



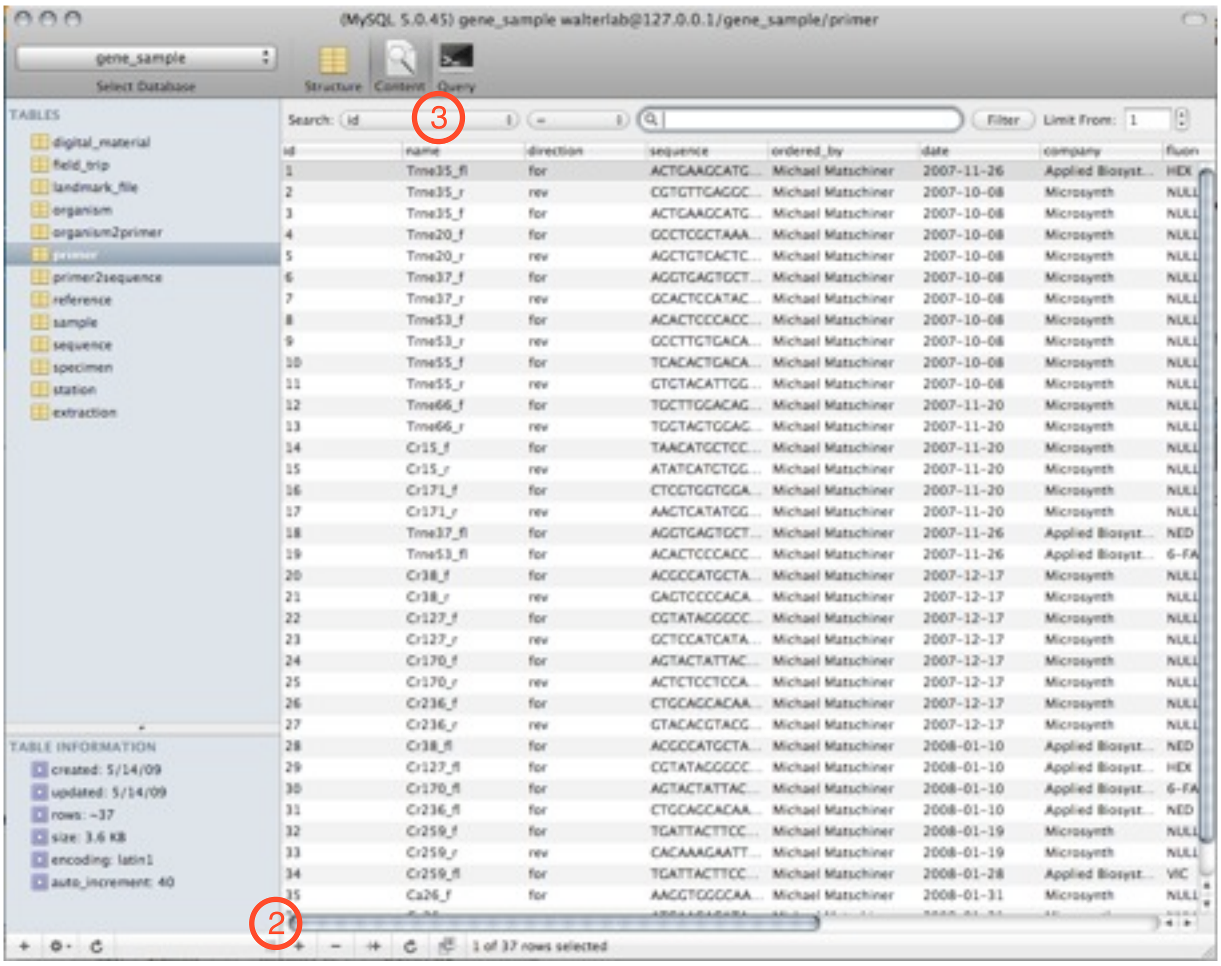
If you click on one of the tables, while the “Structure” symbol (marked with a red circled 1 in this screen shot) is still activated, you will see the structure of the respective table in the main window. The “field” column contains the field names of the individual tables, just as I entered them. The “type” column specifies the data types contained in each field. Possible data types are whole numbers (int), text



Field	Type	Length	Unsigned	Zerofill	Binary	Allow Null	Key	Default	Extra
id	int	6	☐	☐	☐	NO	PRI	NULL	auto_incr...
latin_name	varchar	60	☐	☐	☐	YES		NULL	None
common_name	varchar	60	☐	☐	☐	YES		NULL	None
taxonomy	varchar	600	☐	☐	☐	YES		NULL	None
ncbi_tax_id	int	11	☐	☐	☐	YES		NULL	None
fishbase_id	int	11	☐	☐	☐	YES		NULL	None

(varchar), floating point numbers (float; *e.g.* 12.345), or dates (date; *e.g.* 2009-05-14). For int and varchar, maximum lengths are specified in the third column. Don’t bother about the other columns to the right.
Never modify table structures without discussion with the other database users!

If you click on the “Content” icon to the right of the “Structure” symbol, you’ll see a window like the following, depending on which table you had chosen.



id	name	direction	sequence	ordered_by	date	company	fluor
1	Trne35_fl	for	ACTGAAGCATG...	Michael Matschiner	2007-11-26	Applied Biosyst...	HEX
2	Trne35_r	rev	CGTGTTCAGGC...	Michael Matschiner	2007-10-08	Microsynth	NULL
3	Trne35_f	for	ACTGAAGCATG...	Michael Matschiner	2007-10-08	Microsynth	NULL
4	Trne20_f	for	GCCTCGCTAAA...	Michael Matschiner	2007-10-08	Microsynth	NULL
5	Trne20_r	rev	AGCTGTCACTC...	Michael Matschiner	2007-10-08	Microsynth	NULL
6	Trne37_f	for	AGGTGAGTGCT...	Michael Matschiner	2007-10-08	Microsynth	NULL
7	Trne37_r	rev	GCACTCCATAC...	Michael Matschiner	2007-10-08	Microsynth	NULL
8	Trne53_f	for	ACACTCCCACC...	Michael Matschiner	2007-10-08	Microsynth	NULL
9	Trne53_r	rev	GCCTTGTGACA...	Michael Matschiner	2007-10-08	Microsynth	NULL
10	Trne55_f	for	TCACACTGACA...	Michael Matschiner	2007-10-08	Microsynth	NULL
11	Trne55_r	rev	GTGTACATTGG...	Michael Matschiner	2007-10-08	Microsynth	NULL
12	Trne66_f	for	TGCTTGCACAG...	Michael Matschiner	2007-11-20	Microsynth	NULL
13	Trne66_r	rev	TGCTACTGCAG...	Michael Matschiner	2007-11-20	Microsynth	NULL
14	Cr15_f	for	TAACATGCTCC...	Michael Matschiner	2007-11-20	Microsynth	NULL
15	Cr15_r	rev	ATATCATCTGG...	Michael Matschiner	2007-11-20	Microsynth	NULL
16	Cr171_f	for	CTCGTGGTGGA...	Michael Matschiner	2007-11-20	Microsynth	NULL
17	Cr171_r	rev	AACTCATATGG...	Michael Matschiner	2007-11-20	Microsynth	NULL
18	Trne37_fl	for	AGGTGAGTGCT...	Michael Matschiner	2007-11-26	Applied Biosyst...	NED
19	Trne53_fl	for	ACACTCCCACC...	Michael Matschiner	2007-11-26	Applied Biosyst...	6-FA
20	Cr38_f	for	ACGCCATGCTA...	Michael Matschiner	2007-12-17	Microsynth	NULL
21	Cr38_r	rev	GAGTCCCCACA...	Michael Matschiner	2007-12-17	Microsynth	NULL
22	Cr127_f	for	CGTATAGGGCC...	Michael Matschiner	2007-12-17	Microsynth	NULL
23	Cr127_r	rev	GCTCCATCATA...	Michael Matschiner	2007-12-17	Microsynth	NULL
24	Cr170_f	for	AGTACTATTAC...	Michael Matschiner	2007-12-17	Microsynth	NULL
25	Cr170_r	rev	ACTCTCCTCCA...	Michael Matschiner	2007-12-17	Microsynth	NULL
26	Cr236_f	for	CTGCAGCACAA...	Michael Matschiner	2007-12-17	Microsynth	NULL
27	Cr236_r	rev	GTACACGTACG...	Michael Matschiner	2007-12-17	Microsynth	NULL
28	Cr38_fl	for	ACGCCATGCTA...	Michael Matschiner	2008-01-10	Applied Biosyst...	NED
29	Cr127_fl	for	CGTATAGGGCC...	Michael Matschiner	2008-01-10	Applied Biosyst...	HEX
30	Cr170_fl	for	AGTACTATTAC...	Michael Matschiner	2008-01-10	Applied Biosyst...	6-FA
31	Cr236_fl	for	CTGCAGCACAA...	Michael Matschiner	2008-01-10	Applied Biosyst...	NED
32	Cr259_f	for	TGATTACTTCC...	Michael Matschiner	2008-01-19	Microsynth	NULL
33	Cr259_r	rev	CACAAAGAATT...	Michael Matschiner	2008-01-19	Microsynth	NULL
34	Cr259_fl	for	TGATTACTTCC...	Michael Matschiner	2008-01-28	Applied Biosyst...	VIC
35	Ca26_f	for	AAGGTGGCCAA...	Michael Matschiner	2008-01-31	Microsynth	NULL

The individual columns now correspond to the fields of the selected table, and each row contains a table entry that may or may not contain information for all fields. The first column shows the IDs of all entries that are the PKs, and therefore used to refence this entry from entries of other tables. IDs are auto-incremented by the database, which means that if you add a new entry (by clicking “+” where the red 2 is), you won’t have to fill in this field, the software does this for you (well you could choose an ID yourself as long as it doesn’t exist yet in the database, but it really makes no sense - leave that field empty). You can modify existing entries by double-clicking on a field, but be sure to know what you’re doing.

If you’ld like to filter table entries you can use the Search fields next to the red 3. For example, set Search: ‘direction = for’, click “Filter”, and only forward primers will be shown. Alternatively, you could search for ‘name contains Trne’, and only those primers will be shown that were isolated from *Trematomus newnesi*. Just play around a bit with it, this filter only changes what is shown on screen while the database table remains untouched - you can’t break anything.

If you want to export the currently selected table entries in Excel format, you can do that using the menu: File > Export > Current Browse View > CSV File...

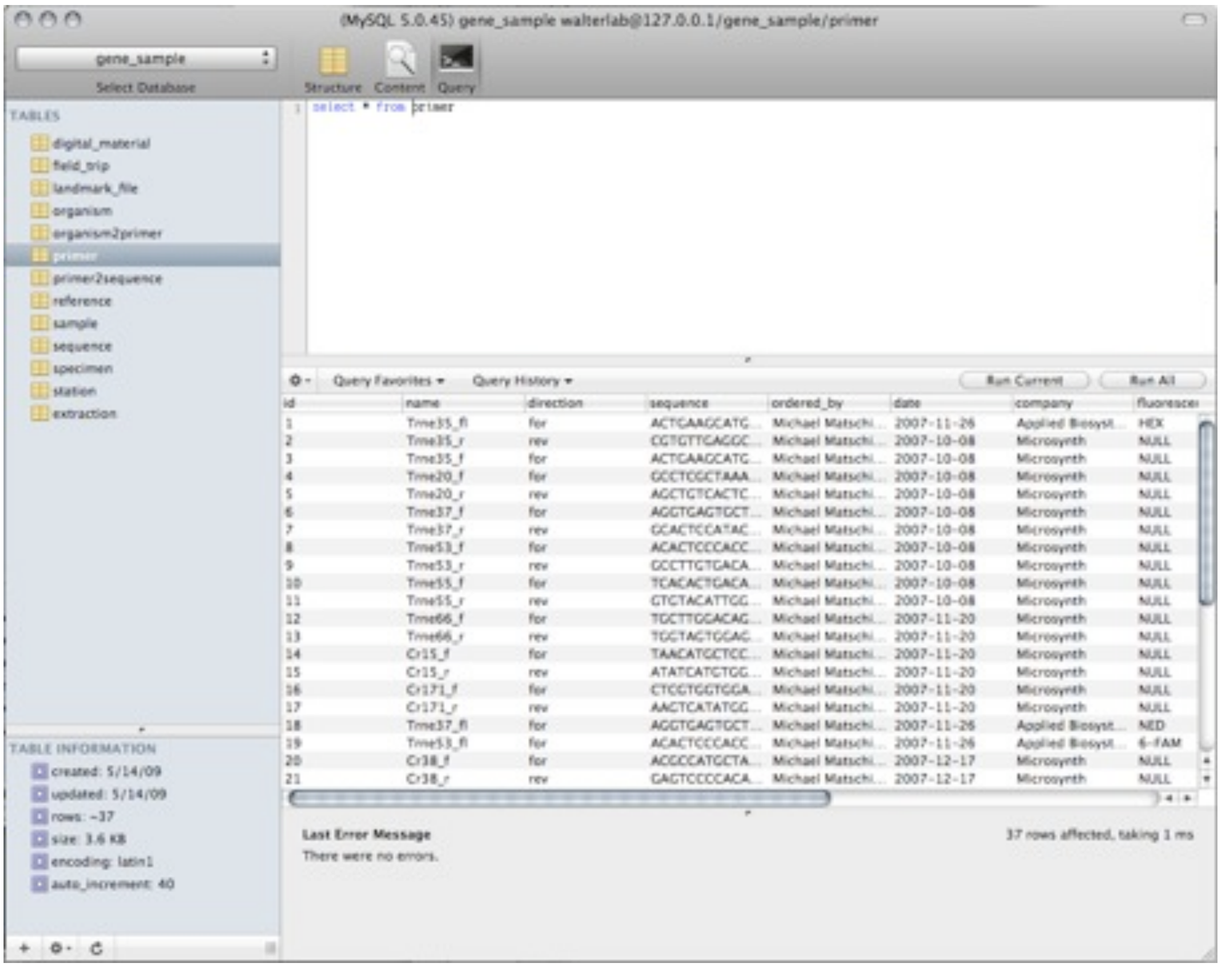
4. Some basic SQL

a) Data selection

If you click the “Query” symbol (above the red 3 in the above image), you will see that even the graphical user interface of Sequel Pro still supports commands in SQL syntax. In fact this is not only helpful, but essential for batch processing of data. It will save a lot of time. However to start off, you could type

```
SELECT * FROM `primer`
```

(from now on, SQL syntax will be highlighted in blue) into the text window and click ‘Run Current’.You will see the following window:



```
select * from primer
```

id	name	direction	sequence	ordered_by	date	company	fluorescei
1	Tme35_fl	for	ACTGAAGCATG...	Michael Matschi...	2007-11-26	Applied Biosyst...	HEX
2	Tme35_r	rev	CGTGTTGAGGC...	Michael Matschi...	2007-10-08	Microsynth	NULL
3	Tme35_f	for	ACTGAAGCATG...	Michael Matschi...	2007-10-08	Microsynth	NULL
4	Tme20_f	for	GCCTCGCTAAA...	Michael Matschi...	2007-10-08	Microsynth	NULL
5	Tme20_r	rev	AGCTGTCACTC...	Michael Matschi...	2007-10-08	Microsynth	NULL
6	Tme37_f	for	AGGTGAGTGCT...	Michael Matschi...	2007-10-08	Microsynth	NULL
7	Tme37_r	rev	GCACTCCATAC...	Michael Matschi...	2007-10-08	Microsynth	NULL
8	Tme53_f	for	ACACTCCCACC...	Michael Matschi...	2007-10-08	Microsynth	NULL
9	Tme53_r	rev	GCCTTGTGACA...	Michael Matschi...	2007-10-08	Microsynth	NULL
10	Tme55_f	for	TCACACTGACA...	Michael Matschi...	2007-10-08	Microsynth	NULL
11	Tme55_r	rev	GTCTACATTGG...	Michael Matschi...	2007-10-08	Microsynth	NULL
12	Tme66_f	for	TGCTTGGACAG...	Michael Matschi...	2007-11-20	Microsynth	NULL
13	Tme66_r	rev	TGCTAGTGCAC...	Michael Matschi...	2007-11-20	Microsynth	NULL
14	Cr15_f	for	TAACATGCTCC...	Michael Matschi...	2007-11-20	Microsynth	NULL
15	Cr15_r	rev	ATATCATGTGG...	Michael Matschi...	2007-11-20	Microsynth	NULL
16	Cr171_f	for	CTCGTGGTGGA...	Michael Matschi...	2007-11-20	Microsynth	NULL
17	Cr171_r	rev	AACTCATATGG...	Michael Matschi...	2007-11-20	Microsynth	NULL
18	Tme37_fl	for	AGGTGAGTGCT...	Michael Matschi...	2007-11-26	Applied Biosyst...	NED
19	Tme53_fl	for	ACACTCCCACC...	Michael Matschi...	2007-11-26	Applied Biosyst...	6-FAM
20	Cr38_f	for	ACGCCATGCTA...	Michael Matschi...	2007-12-17	Microsynth	NULL
21	Cr38_r	rev	GAGTCCCCACA...	Michael Matschi...	2007-12-17	Microsynth	NULL

Last Error Message
There were no errors.

37 rows affected, taking 1 ms

All the entries and all their fields are shown in the results window. If you’ld like to see only a limited number of columns, you could type

```
SELECT id, name, direction FROM `primer`
```

to see only these three columns. Be sure to use the diagonal dashes, not the vertical ones, they won’t work. You could also set id, name and direction within dashes. For some reasons, dashes are sometimes optional, sometimes not. If something doesn’t work, try it again with dashes.

If you'ld like to limit the rows instead (which means the shown table entries) while seeing all columns for these selected rows, you could type

```
SELECT * FROM `primer` WHERE `direction` = "for"
```

to see only forward primers.

```
SELECT * FROM `primer` WHERE `direction` LIKE "for"
```

does the same job.

You could also select for primers with names that contain “Trne”:

```
SELECT * FROM `primer` WHERE `name` LIKE "%Trne%"
```

(note the percentage signs within the quotes). This will show not only primers named “Trne”, but also those named “Trne35”, “Trne53”, “abcTrne”, and so on.

If you'ld like to see only the ten first primers, you could type

```
SELECT * FROM `primer` WHERE `id` < 11
```

Naturally, you could limit the number of rows and columns at the same time by combining both methods of selection.

```
SELECT name,sequence FROM `primer` WHERE `id` < 11
```

shows only two columns, and ten rows. You could also select table entries by more than one search criteria. If we'ld like to see only forward primers isolated from *Trematomus newnesi*, we could

```
SELECT * FROM `primer` WHERE `name` LIKE "%Trne%" AND `direction` = "for"
```

and so on. Sometimes you may wish to have your query results ordered. This can be achieved by adding an ‘order by‘ clause to the selection statement:

```
SELECT * FROM `primer` WHERE `id` < 11 ORDER BY `name`
```

The general selection syntax is

```
SELECT <field_names_separated_by_commas> FROM <table_name_in_dashes> WHERE
<conditional_statement> ORDER BY <a_field_name>
```

It is always possible to export the current selection in Excel format via the menu.

b) Advanced data selection

This is where it gets interesting. And complicated, cause queries can be very long. Don't worry, you can always save queries to your favorites in Sequel Pro, then you won't have to type them again. The point is that very often you will need to select data not only based on information in the currently selected table, but using information from other tables as well. For example, I may want to select only those primers (entries from the primer table) that link to a reference (an entry from the reference table) where Papetti is among the authors. I could type

```
SELECT primer.name, reference.title FROM primer, reference WHERE primer.reference_id = reference.id
AND reference.authors LIKE "%Papetti%"
```

Let me walk you through this. We now select fields from two different tables, thats why we need to specify tables AND fields in the query using the format

```
SELECT <table1_name>.<table1_field>, <table2_name>.<table2_field> ...
```

Note the period between table names and table fields. The comma sign separates fields just as it did when both fields were from the same table. The query goes on...

```
...FROM primer, reference...
```

again, we have to specify both tables here.

```
...WHERE primer.reference_id = reference.id...
```

this may be the most important part of the query that ensures that only those entries are shown, where the FKs and the PKs fit. As you can see from Fig. 1, the primer table and the reference table are connected via a one-to-many link, whereby the FK of the primer table (which has been assigned to the field "reference_id") links to the PK of the reference table (which has been assigned to the field "id", as in every table). With our query, we intend to select only for those combinations of primer and reference, where the FK of the primer table actually match the PK of the reference table.

So far all we did was to ensure that only fitting combinations of primer names and reference titles are shown, but there's no selection done yet. This comes with the last part of the query

```
...AND reference.authors LIKE "%Papetti%"
```

where we filter results for those that contain "Papetti" in the authors field of the corresponding reference entry.

However, there's a better way to do this that keeps the linking of both tables (WHERE primer.reference_id = reference.id) and the actual selection (AND reference.authors LIKE "%Papetti%") separate. For this, we need a new command named INNER JOIN ... ON.

```
SELECT primer.name, reference.title FROM primer INNER JOIN reference ON primer.reference_id =
reference.id WHERE reference.authors LIKE "%Papetti%"
```

There is also a possibility to use shortcuts:

```
SELECT p.name, r.title FROM primer p INNER JOIN reference r ON p.reference_id = r.id WHERE
r.authors LIKE "%Papetti%"
```

Here, shortcuts were defined by FROM primer p... and by INNER JOIN reference r... , and they were utilized in p.name, r.title..., in ON p.reference_id = r.id..., and in WHERE r.authors...

But if you think that's as difficult as it get, you're wrong. Have a look at Fig. 1. What if we would like to find the paths of all landmark files for Tropheus moorii? Or more precisely, the paths of all the landmark files that link to digital material that links to a sample that links to a specimen that links to an organism that has "Tropheus moorii" as its latin name?
Using the same logic as above, we can type the following query:

```
SELECT l.path
FROM landmark_file l INNER JOIN digital_material d ON l.digital_material_id = d.id
                     INNER JOIN sample sa ON d.sample_id = sa.id
                     INNER JOIN specimen sp ON sa.specimen_id = sp.id
                     INNER JOIN organism o ON sp.organism_id = o.id
WHERE o.latin_name = "Tropheus moorii"
```

We could also join these five tables starting at the other end (organism to landmark_file):

```
SELECT l.path
FROM organism o INNER JOIN specimen sp ON sp.organism_id = o.id
                INNER JOIN sample sa ON sa.specimen_id = sp.id
                INNER JOIN digital_material d ON d.sample_id = sa.id
                INNER JOIN landmark_file l ON l.digital_material_id = d.id
WHERE o.latin_name = "Tropheus moorii"
```

Again, these statements could be combined with ORDER BY clauses, or WHERE statements could be extended to multiple conditions (WHERE... AND...).

Being, after all, a programming language, SQL allows even much more complicated queries. There's stuff like OUTER JOIN, LEFT JOIN, GROUP BY, HAVING and many more commands, but we might not need all this. The full reference manual for the SELECT syntax can be found here:
http://dev.mysql.com/doc/refman/5.1/en/select.html

One very useful example including a GROUP BY statement would be

```
SELECT organism.latin_name, COUNT(specimen.id)
FROM organism INNER JOIN specimen ON specimen.organism_id = organism.id
GROUP BY organism.latin_name
```

which gives you a list of organisms and the number of specimens of this organism that are present in the database.

Another useful example lists the minimum and maximum water depth at which organisms were found:

```
SELECT organism.latin_name, MIN(station.mean_water_depth), MAX(station.mean_water_depth)
FROM organism INNER JOIN specimen ON specimen.organism_id = organism.id
              INNER JOIN station ON specimen.station_id = station.id
GROUP BY organism.latin_name
```

There's one more thing I'll have to explain. The database contains two tables that do nothing more but link two other tables. These are organism2primer and primer2amplicon. The reason why we have these two extra tables instead of direct links is that without these tables we would have many-to-many links connecting organism and primer, and primer and amplicon. Naturally, multiple primers work (succeed to amplify something) for an organism, but any single primer may also work in more than one organism (cross-amplify something). It is similar for the amplicon-primer link: Every amplicon has been amplified with two primers (forward and reverse), while every primer may have been used in multiple amplifications (with multiple samples). If I would like to find the two primers with which one particular amplicon was amplified, I could use

```
SELECT p.name, p.direction
FROM primer p INNER JOIN primer2amplicon p2a ON p2a.primer_id = p.id
                    INNER JOIN amplicon a ON p2a.amplicon_id = a.id
WHERE a.id = “123”
```

Unfortunately, many of the examples given here cannot be tested yet, simply because these tables don’t contain any data yet. (violet text will be rewritten for later versions).

c) Adding data

Adding of individual entries is more easily done with Sequel Pro’s graphical user interface (see above, chapter 3) than with SQL syntax. For the sake of completeness, here’s a simple example of how you would add a new primer to the primer table in SQL:

```
INSERT INTO primer (name, direction, sequence, fluorescent_label) VALUES ("Trne99", "for",
"ACTGCAACTGACAGTAA", "ROX")
```

Hereby, the first pair of parentheses contains a list of field names, and the second pair contains the values to be added to these fields. The number and the order of values should match those of specified fields, *i.e.* the second value will always be added to the second specified field. Never specify the primary key field \`id\`, as this field will be auto-incremented by the database. Fields that are not included in the first pair of parentheses will be assigned the value \`NULL\` (with the exception of the primary key field \`id\`). You can verify that your new entry was added to the table by clicking the content symbol (close to the red 3 in the above screenshot), assuming that the primer table is still the selected one (which means it is highlighted in the left sidebar. You may have to refresh the table to see the new entry. This can be done via the menu (Database > Refresh Database), with the shortcut ⌘R, or with the button to the right of the red 2 in the screenshot above (the one with a circular arrow).

d) Batch import data

This will be far more important for us than adding single entries. Most often, for example upon return from the field, we will have whole Excel sheets full of data, that we’ld like to add all at once. In fact the database would be rather useless, if we couldn’t quickly do this. Note that Excel sheets should be in .csv format (csv = comma separated values). Thus, save your Excel sheet in this format before import. Import of Excel sheets could be done using the Sequel Pro menu (File > Import...), however I recommend doing so with SQL syntax, which gives you the possibility to specify fields included in the data sheet. Here’s some sample code:

```
LOAD DATA LOCAL INFILE "/Users/michaelmatschiner/Documents/addToPrimer.csv"
INTO TABLE primer
FIELDS TERMINATED BY "," ENCLOSED BY "\""
LINES TERMINATED BY "\r"
IGNORE 1 LINES
(name, direction, sequence, ordered_by, date, company, fluorescent_label, box_name, row, column,
reference_id)
```

Again, the primary key field \`id\` should not be included in the Excel sheet, as this field will be filled by the database. Fields that are present in the table, but not included in the SQL syntax will be filled with the value \`NULL\`. A few more things regarding this sample code: in FIELDS TERMINATED BY \`,\` we specify that values are separated by commas, as is true for .csv format. We could also save the Excel sheet in tab-delimited format if we use FIELDS TERMINATED BY \`\t\` instead. The ENCLOSED BY "\"" clause specifies that fields can be enclosed in quotes. This is important if we have field entries with commas. For example, if the location field of a field trip is ‘Zambia, Africa’, we need to use quotes around this field, otherwise it would be read as two different fields with comma separated format. LINES TERMINATED BY

"\r" specifies that 'carriage return' characters (symbolized by '\r') code for line endings. For some reason, Excel still uses this type of line terminator, even though the default on Macs has been the 'line feed' character (symbolized by '\n') since Mac OS10.0. On Windows computers, this would be a combination of the 'carriage return' (CR; '\r') and 'line feed' (LF; '\n') characters: '\r\'n'. Anyway, just make sure that the Excel sheet has been saved on a Mac computer. If you experience errors with data imports where only one, or no row at all seems to be recognized, the line endings are probably the cause of it. The IGNORE 1 LINES part allows the use of one header line in the Excel sheets that is not read as a database entry. Finally, if the file name contains blanks, they must be escaped with a backslash symbol as in "/Users/michaelmatschiner/Documents/add\ to\ primer. csv". On Mac computers file names always start with

'/Users/<your_user_name>/...". In case of doubt you can always find out the full file name by dragging the file icon from the Finder into a new Terminal window.

There's one major drawback to this import method when you import data into tables with foreign keys: Oftentimes, you do not know the ID of the entry to which the foreign key points to, or it may be tedious to find it. This should be alright when adding primers to the database, cause primers only link to references, whereby the reference table is an optional table anyway, and primers are usually added one by one, rarely in batch mode. I assume it would be alright to look up the respective reference IDs when adding a limited number of primers, but finding the right organism IDs for every specimen upon return from field trips, with hundreds or thousands of entries to the specimen table may be tough. It would be much more convenient to type for example the latin name into the corresponding column of the Excel table, that should then be automatically translated into the right organism ID upon import to the database. The problem may be even worse when importing samples, whereby every sample has to point to the right specimen (the `specimen_id` field of the sample table has to point to the `id` field of the specimen table. It would be very error-prone and tedious to find the right IDs manually and include them in the Excel sheets before import.
This is how you replace latin names of organisms by the corresponding organism ID during import of data into the specimen table:

```
LOAD DATA LOCAL INFILE "/Users/michaelmatschiner/Documents/addToSpecimen.csv"
INTO TABLE specimen
FIELDS TERMINATED BY "," ENCLOSED BY "\""
LINES TERMINATED BY "\r"
IGNORE 1 LINES
(name, collector, sex, birthday, day_of_death, standard_length, terminal_length, weight,
n_pharyngeal_teeth, n_upper_oral_teeth, n_lower_oral_teeth, @latin_name, station_id)
SET organism_id = (SELECT id FROM organism WHERE latin_name = @latin_name)
```

Note how the SET clause translates the variable @latin_name, that was specified in the import file's second-last column, into the correct organism_id. If we furthermore prefer to specify the station name instead of the station id in the import file's last column (cause it's easier to memorize 'Wonzye point' than it's corresponding ID), we can extend the SET clause:

```
LOAD DATA LOCAL INFILE "/Users/michaelmatschiner/Documents/addToSpecimen.csv"
INTO TABLE specimen
FIELDS TERMINATED BY "," ENCLOSED BY "\""
LINES TERMINATED BY "\r"
IGNORE 1 LINES
(name, collector, sex, birthday, day_of_death, standard_length, terminal_length, weight,
n_pharyngeal_teeth, n_upper_oral_teeth, n_lower_oral_teeth, @latin_name, @station_name)
SET organism_id = (SELECT id FROM organism WHERE latin_name = @latin_name),
     station_id = (SELECT id FROM station WHERE name = @station_name)
```

Note the change from 'station_id' to '@station_name' in line 7. In order for this method to work, station names must be unique (no two stations should have the same name). Of course, latin names are unique anyway. The code for import of landmark_file, digital_material, primer, sample, extraction, and amplicon data is similar. All codes can be found online at www.evolution.unibas.ch/salzburger/protocols.htm and are adapted to the Excel import file at that you can find at the same website. This Excel file is made up of a number of sheets for every individual database table. The code pieces can be copy-pasted into the Sequel Pro Query window, and then executed with ‘Run All’. You’ll have to change the specified file name first.

Take care to import ‘upstream‘ data first! For example, you should first import field trips, then see what IDs they are assigned, then use these IDs for the field ‘field_trip_id‘ when importing data into the ‘station‘ table.

e) Modifying data

Fortunately, this is one of the more straight-forward exercises. For example, if you just found out that you misspelled the box name in a whole series of entries to the ‘sample‘ table, you could easily change this:

```
UPDATE sample
SET box_name = "icefish_samples"
WHERE box_name = "icefish_smples"
```

If you want to change the box names of all entries to the sample table, you would simply type

```
UPDATE sample
SET box_name = "icefish_samples"
```

However be careful with these sorts of modifications! If you want to change single entries only, you could also do so right in the Sequel Pro Content window.

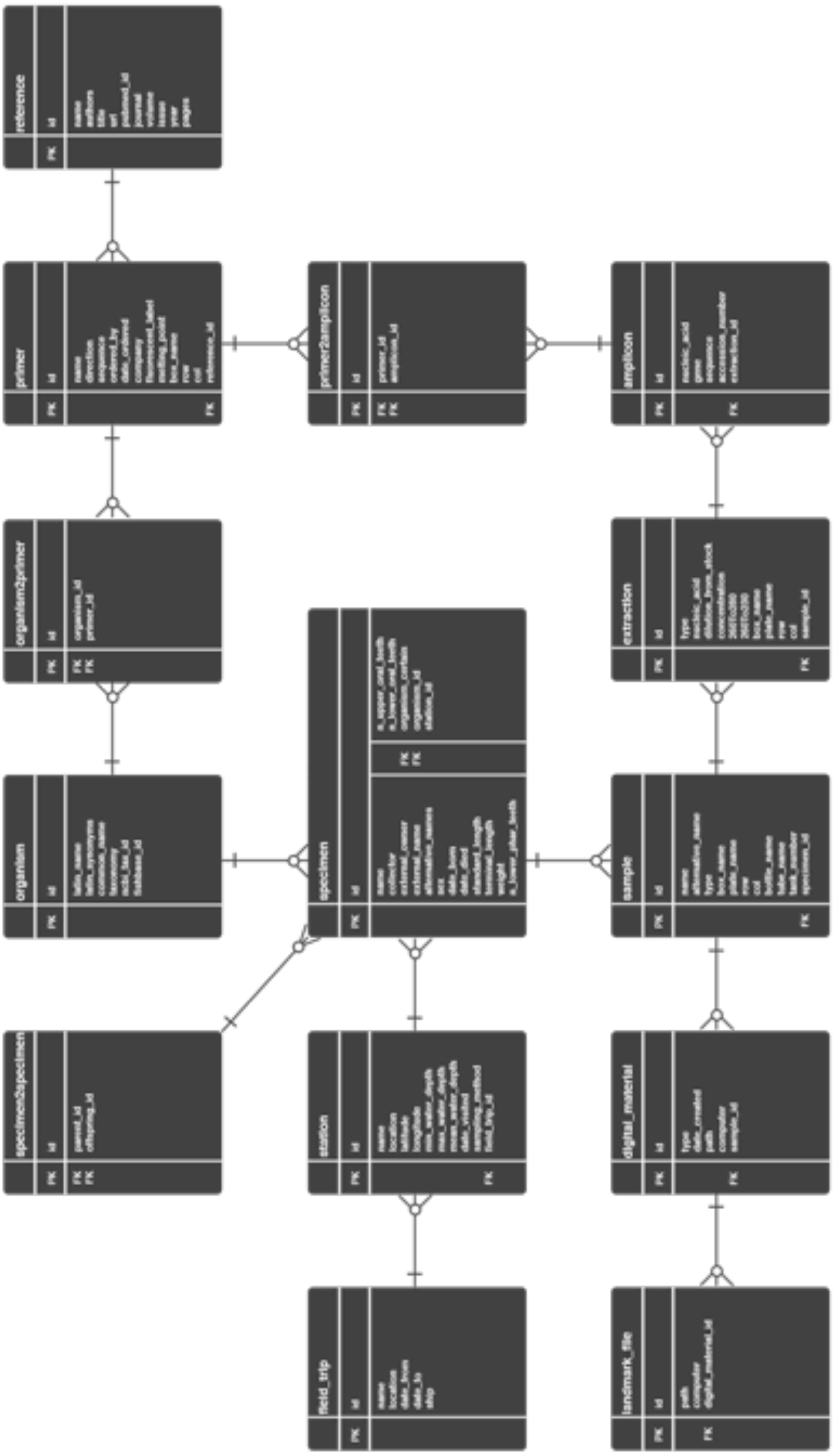


Fig. 1: The database layout. Black boxes represent tables that are connected by one-to-many links, linking foreign keys (FK) to primary keys (PK).

Table 1: The following gives examples and instructions on how to enter information into the individual fields of every table. Table names are written in **bold**, and field names are written below table names in the left column. The right column contains either examples of field entries in double quotes, or an exhaustive list of possible field entries (without quotes). In the latter case, you're supposed to use one of the list items strictly the way it is given, case-sensitive and using underscores where given. This is to facilitate database searches. If some of us would use "CT_scan", while others would type "ctscan", "CT scan", and so on, a user searching for this type of information could only retrieve part of the relevant data. Comments are given in *grey italics*.

amplicon

nucleic_acid	DNA, cDNA
gene	"CytB", "D-loop"
sequence	"ACGTGGACTCTGAACTATAGCCG"
accession_number	"AF145428"

digital_material

type	image, CT_scan, video *if CT_scan, the path refers to the folder containing all CT images*
date_created	"2008-12-17" *use YYYY-MM-DD format*
path	"User/michaelmatschiner/Pictures/Fish Pictures/ab1234.jpg" *if type = CT_scan, specify foldername*
computer	"evo-pagrus" *use the evo-... name*

extraction

type	working, stock, backup
nucleic_acid	DNA, RNA
dilution_from_stock	"1:20" *use 1:xxx*
concentration	"2.34" *a floating point number, in units of ng/ml*
260To280	"2.34" *a floating point number*
260To230	"2.34" *a floating point number*
box_name	"extracted icefish DNA 3"
plate_name	"extracted icefish DNA 4"
row	"B", "7"
col	"10"

field_trip

name	"Africa_2008", "ANT-XXIII/8"
location	"Zambia, Africa", "Antarctica"
date_from	"2008-08-28" *use YYYY-MM-DD format*
date_to	"2009-09-22" *use YYYY-MM-DD format*
ship	"Polarstern" *leave empty when different ships have been used during one field trip*

landmark_file

path	"Users/michaelmatschiner/Pictures/Fish Pictures/ab1234.jpg"
computer	"evo-pagrus" *use the evo-... name*

organism	*please make sure the entry doesn't exist yet before creating a new one. To do so, click 'latin_name' in the Sequel Pro Content window to sort entries alphabetically by their latin names. The table includes a few cichlid species already!*
latin_name:	"Gobionotothen gibberifrons"
latin_synonyms:	"Notothenia gibberifrons" *if more than one, separate with commas*
common_name:	"Humped rockcod" *the common names according to FishBase*
taxonomy:	"Eukaryota; Fungi/Metazoa group; Metazoa; Eumetazoa; Bilateria; Coelomata; Deuterostomia; Chordata; Craniata; Vertebrata; Gnathostomata; Teleostomi; Euteleostomi; Actinopterygii; Actinopteri; Neopterygii; Teleostei; Elopocephala; Clupeocephala; Euteleostei; Neognathi; Neoteleostei; Eurypterygii; Ctenosquamata; Acanthomorpha; Euacanthomorpha; Holacanthopterygii; Acanthopterygii; Euacanthopterygii; Percomorpha; Perciformes; Notothenioidei; Nototheniidae; Gobionotothen" *copy from Genbank's Taxonomy Browser (www.ncbi.nlm.nih.gov/Taxonomy/Browser/wwwtax.cgi?) starting with "Eukaryota"*
ncbi_tax_id:	"36202"
fishbase_id:	"7042" *find it in the Fishbase URL (...speciesSummary.php?ID=**7068**&genusname…)*

primer	
name	"Trne35f"
direction	for, rev
sequence	"GTGCCGATGATCTGAGA"
ordered_by	"Michael Matschiner"
date_ordered	"2008-12-17" *use YYYY-MM-DD format*
company	Applied Biosystems, Microsynth
fluorescent_label	VIC, HEX, 6-FAM, NED, ROX, PET, LIZ, JOE, TAMRA, dTAMRA
melting_point	"63.5"
box_name	"Icefish primers"
row	"B", "7"
col	"10"

reference	*entries to this table are rather optional*
name	"Papetti_2007a"
authors	"Papetti C, Zane L, Patarnello T"
title	"Isolation and characterization of microsatellite loci in the icefish Chionodraco rastrospinosus (Perciformes, Notothenioidea, Channichthyidae)"
url	"http://www3.interscience.wiley.com/journal/118613643/abstract?CRETRY=1&SRETRY=0"
pubmed_id	"19333432"
journal	"Molecular Ecology Notes"
volume	"6"
issue	"1"
year	"2007"
pages	"652-655"

sample

type	muscle_tissue, fin_clip, liver, brain, pharyngeal_jaw, upper_oral_jaw, lower_oral_jaw, jaw_tissue, egg, whole_specimen, live_specimen
name	"GG-670-31_mt", "GG-670-31_fc" *for consistency, please use the specimen name, underscore, and either mt (muscle_tissue), fc (fin_clip), mt/fc (muscle_tissue/fin_clip), lv (liver), br (brain), pj (pharyngeal jaw), uoj (upper oral jaw), loj (lower oral jaw), jt (jaw tissue), ws (whole specimen), ls (live specimen), or xx (unknown). This will facilitate links from the extraction and the digital_material table. Add a number if more than one sample of the same type are taken from one specimen.*
alternative_name	"Tromoo20B8" *our earlier labelling system*
box_name	"Notothenioid dating", "14"
plate_name	"Tmoor jaws" *I assume that every sample is either in a box, a plate, a tube or a tank*
row	"B", "8"
col	"10" *rows and columns specify samples that are either in boxes or plates. leave empty if the sample is in a bottle, a tube or a tank*
bottle_name	"14", "AB3"
tube_name	"14", "AB3"
tank_number	"20"

specimen

name	"Gobgib_12", "Tromoo_12" *I propose this as a uniform labelling system. In any case, names must be unique, no two specimens can have the same name*
collector	"Michael Matschiner" *use first and last name*
external_owner	"Reinhold Hanel", "Erik Verheyen" *use first and last name, only if the specimen is in external possession*
external_name	"1234" *use only if the specimen is in external possession and you did not adopt the external labelling system*
alternative_names	"GG-640-12" *use if you used a different labelling system in the past, or if two different labels are still in use. Use commas if you use more than one.*
sex	m, f *one character only*
date_born	"2009-05-18" *use YYYY-MM-DD format*
date_died	"2009-05-19" *use YYYY-MM-DD format*
standard_length	"23.5" *always in cm*
terminal_length	"25.5" *always in cm*
weight	"108.3" *always in g*
n_lower_phar_teeth	"24" *a whole number*
n_upper_oral_teeth	"24" *a whole number*
n_lower_oral_teeth	"24" *a whole number*
organism_certain	yes, no *if the organism is identified with something like* Hippocampus sp. aff. reidi, *declare 'no'. In most cases, however, it is 'yes'*

station

name	"670", "Wonzye point 12"
location	"South Sandwich Islands"
latitude	"-55.25413" *negative numbers for southern latitudes*
longitude	"-13.62145" *negative numbers for western longitudes*
water_depth	"6" *in meters*
date_visited	"2004-06-18" *use YYYY-MM-DD format*
sampling_method	"gill net", "bottom trawl", "angling"