# Pragmatic Rules for Databases

12. September 2015

Ursula Gaedke[1], Katrin Tirok[1],[2] and Thomas Petzoldt[3]

## Aims of this guide

Empirical data are the fundamentals of science. Lots of human power and money is put into the collection of field data, in designing and running experiments and in their analysis. In short, measured data are extremely valuable. Therefore, it should be self-evident to invest some thoughts and knowledge in how to store data for a scientists own purpose, how to preserve them for the future and how to make sharing with colleagues a pleasure and not a pain.

This guide is written for empirical scientists and not for data base experts. Therefore, rules and suggestions are kept pragmatic and at the minimum. We are of course grateful for comments and suggestions.

Main emphasis is given to practical aspects of RDBMS (Relational Database Management Systems), the simplest and most common type of data bases. Data exchange between such relational data base management systems is supported by the (in principle) standardized *sequential query language* (SQL). Most database systems have SQL built-in (e.g. MS Access, MySQL) or interfaces to an SQL driver (e.g. MS Excel, R).

## Software

The rules presented here are mostly independent from the used software. They apply to spreadsheets (Excel, LibreOffice), text files (so-called CSV or ASCII), statistics packages (R, SAS, SPSS, Statistica) and of course to data base systems like Microsoft Access, Oracle, MySQL and so on. The proposed philosophy is even more important if the amount of data is high, e.g. for 3D simulation data, flow cytometers, DNA sequencers or complete organizations. However, additional considerations will then come into play.

## General rules

In the relational database model, data are organized in one or several **tables** (so-called relations) that have **rows** and **columns**. The rows are often called **records**, the columns **fields** or **attributes**. Tables can be connected by using **joins**.

1. Use clear **key fields** for all records. Each row of a data base table must be uniquely identifiable from the combination of its key fields.
2. The **resolution** of a table should match the resolution of the data.

[1]University of Potsdam, Germany
[2]University of KwaZulu-Natal, South Africa
[3]Technische Universität Dresden, Germany

- If resolution varies, split your data into several tables, e.g. one table for the general characteristics of your system, one table for the list of species (only name, family, genus, . . . , species traits and a code, but not yet the abundance) one table for everything that is measured once per sampling campaign (e.g. sunshine duration of the day), another table that is measured at several locations per sampling day (herein come the species abundances) and so on.
- As a general rule, a table has a wrong resolution if it contains lots of NA (not measured) entries or if the data base structure needs to be changed for additional measurements.
- There is nothing wrong with long tables. In fact, correct database tables have almost always many more records than columns. This may sound somehow inconvenient for typical Excel users, but computers are perfect in analyzing stupid information.

3. The key field(s) should be well designed, so that they are unique, but also not too complicated (or too long).
4. The names of the fields (columns, variables) should not be used to encode attributes like time or location.
    - wrong: `depth1`, `depth2`, `depth3` or `sample1`, `sample2`, `sample3`, `replicate 1`, `replicate 2`
    - correct: `site`, `depth`, `time`, `sample`, `replicate`, . . .
5. Avoid redundancy:
    - No column should be derived from others by simple transformations, e.g. by multiplication with a constant factor.
    - Except for the key fields, avoid to repeat the same information in different tables.
    - If date fields are used, decide whether you would like to have it in one **Date** field (our preference) or if you want to have different fields for Year, Month and Day, but never use both simultaneously. The same holds for Date and Time, either you have one variable Datetime including date and time, or separate variables for Date and Time.
    - If species names (or their codes) occur in a sample table, don't add their general species traits, systematic characterization etc. to the same table. Create a special species table (without the samples) for this purpose.
6. Always document measurement units and, when necessary, the applied measurement methods.
    - The measurement units can be put in a separate table or follow directly in a column. This depends on the data base structure.
    - You should also keep track of the detection limits in your samples and any changes in sample methods occurring over time, e.g. instrument changes, or changes in provider of specific analysis . . . . It is useful to have one table with a description of all measured variables including their units and sample method or/and links to references and sample protocols.
7. Use consistent identifiers for the tables, the column names (fields) and for the values of keys. Details about this can be found below.
8. In order to link (or join) two or more tables, it is wise to use consistent column names for the variables, e.g. `Date` and `Date`, not `Date` and `datetime` or `Species` and `Species`, but not `Species` and `Species_name`.

# Examples

## 1. Use of key fields for each row

**Tab. 1:** Correct table where records are uniquely identifiable by their key fields `Location`, `Date`, `Depth` and `Species`.

| Location | Date | Depth | SpeciesNo | Abundance |
|---|---|---|---|---|
| RL111 | 2002-10-10 | 1 | 1 | 100 |
| RL111 | 2002-10-10 | 1 | 2 | 190 |
| RL111 | 2002-10-10 | 1 | 3 | 10 |
| RL111 | 2002-10-10 | 3 | 1 | 80 |
| RL111 | 2002-10-10 | 3 | 2 | 0 |
| RL111 | 2002-10-10 | 3 | 3 | 20 |
| RL117 | 2002-10-10 | 1 | 1 | 1000 |
| RL117 | 2002-10-10 | ... | ... | ... |
| ... | ... | ... | ... | ... |
| RL111 | 2002-10-17 | 1 | 1 | ... |
| ... | ... | ... | ... | ... |

## 2. The keys of a table should match the resolution of the data

In the previous example, the resolution of the data is defined by the spatial and temporal categories Location, Date and Depth.

However, the species themselves do not depend on time or space and deserve their own table. In the following tables (Tab. 2 and Tab. 3), each species number has an associated species name and further characteristics like cell size, systematic categories or ecological traits.

As both tables contain a common key (`SpeciesNo`) these tables can easily be combined by using data base operations (**merge** or **join**). Such operations are available in all data base programs (Microsoft Access, Oracle or MySQL), and also statistics software like SPSS, SAS or R.

Spreadsheet programs like MS Excel and LibreOffice have database functionality as well, and it makes sense to use it, even if it is relatively limited. Several functions are available for this, e.g. `Lookup`, `Vlookup`, `Hlookup` or `Indirect`. More about this can be found in the online help, the internet or a good Excel book.

**Note:** Sometimes, it looks impractical to have one column (e.g. abundance) directly in the table while another but similar one (e.g. biomass) needs to be calculated from two tables. That's why some of us save resulting data (e.g. the biomass) as a working copy directly to the original table. Here, we must clearly say that this approach does not follow the data base theory and may lead to inconsistencies, e.g if the standard cell biovolume is changed.

Therefore, queries should be used instead of modifying raw data tables if the software supports this (e.g. Access), otherwise we should always be aware what are original data and what are only working copies.

**Tab. 2:** Species table containing the species number (or a code identifying a species), the full name of the species and further characteristics, e.g. species-specific cell size (in $\mu m^3$), systematic categories or traits.

| SpeciesNo | Species | CellSize |
|---|---|---|
| 1 | Rhodomonas minuta | 10 |
| 2 | Cryptomonas ovata | 20 |
| 3 | Cryptomonas marsonii | 1000 |

**Tab. 3:** Species table where genus and species are separate fields, which can be advantageous, e.g. if we want to sum up over the genus.

| SpeciesNo | Genus | Species | CellSize |
|---|---|---|---|
| 1 | Rhodomonas | minuta | 10 |
| 2 | Cryptomonas | ovata | 20 |
| 3 | Cryptomonas | marsonii | 1000 |

### A bad example

The following data format is an intentionally bad example of redundant information, because abundance is measured in several depths each day (Location x Date x Depth resolution) while air temperature is only measured once per day (resolution: Date or Location x Date, cf. Tab. 5a and 5b). That's why it makes sense to spend a separate table for air temperature, and maybe also other measures that are measured once per day, e.g. secchi depth.

**Tab. 4:** Redundant structure that requires to repeat air temperature, even if it was measured only once per day.

| Location | Date | Depth | SpeciesNo | Abundance | AirTemp |
|---|---|---|---|---|---|
| RL111 | 10.10.2002 | 1 | 1 | 100 | 10 |
| RL111 | 10.10.2002 | 1 | 2 | 190 | 10 |
| RL111 | 10.10.2002 | 1 | 3 | 10 | 10 |
| RL111 | 10.10.2002 | 3 | 1 | 80 | 10 |
| RL111 | 10.10.2002 | 3 | 2 | 0 | 10 |
| RL111 | 10.10.2002 | 3 | 3 | 20 | 10 |
| RL117 | 10.10.2002 | 1 | 1 | 1000 | 10 |
| RL117 | 10.10.2002 | ... | ... | ... | 10 |
| ... | ... | ... | ... | ... | ... |
| RL111 | 17.10.2002 | 1 | 1 | ... | 8 |
| ... | ... | ... | ... | ... | 8 |

**Tab. 5a:** Air temperature if measured once per day.

| Date | AirTemp |
|---|---|
| 2002-10-10 | 10 |
| 2002-10-17 | 8 |
| .... | .... |

**Tab. 5b:** Air temperature if measured once per sampling site (location) and day.

| Date | Location | AirTemp |
|---|---|---|
| 2002-10-10 | RL111 | 10 |
| 2002-10-10 | RL117 | 11 |
| 2002-10-17 | RL111 | 8 |
| 2002-10-17 | RL117 | 7 |
| ... | ... | ... |

The following table shows another example of a wrong data structure. Here, an abiotic variable (irradiation) was measured with a much higher temporal resolution. If abundance is recorded in the same table, it looks like if the abundance was measured once per hour, which is clearly not correct. As an alternative, some people would leave abundance blank (not measured) except at the time point of measurements. This would be scientifically correct, but from a data base perspective it would be still redundant, especially as the irradiation is independent of the species and would have to be repeated for all of them.

Suppose, you want to correct the irradiation in a table with 100 species at several locations …

The solution is simple: don't mix up information that has different spatial, temporal, (or whatever) resolution. Create separate tables and let the database operations make the repeated assignment automatically upon request.

**Tab. 6:** Example for a wrong table mixing up information with different spatial and temporal resolution. See Tab. 7 for a correct approach.

| Location | Date | Time | Depth | SpeciesNo | Abundance | Irradiation |
|---|---|---|---|---|---|---|
| RL111 | 2002-10-10 | 06:00 | 1 | 1 | 100 | 0 |
| RL111 | 2002-10-10 | 07:00 | 1 | 1 | 100 | 11 |
| RL111 | 2002-10-10 | 08:00 | 1 | 1 | 100 | 118 |
| RL111 | 2002-10-10 | 09:00 | 1 | 1 | 100 | 356 |
| RL111 | 2002-10-10 | 10:00 | 1 | 1 | 100 | 459 |
| RL111 | 2002-10-10 | 11:00 | 1 | 1 | 100 | 788 |
| RL111 | 2002-10-10 | 12:00 | 1 | 1 | 100 | 890 |
| … | … | … | … | … | … | … |

**Tab. 7:** Separate table for hourly measured meteorological data.

| Date | Time | Irradiation |
|---|---|---|
| 2002-10-10 | 06:00 | 0 |
| 2002-10-10 | 07:00 | 11 |
| 2002-10-10 | 08:00 | 118 |
| 2002-10-10 | 09:00 | 356 |
| … | … | … |

## 3. Variable names should not encode potentially varying information

There is often a tendency to use multiple additional columns for one variable in cases when in fact an additional key should be used, for example `Abundance1`, `Abundance2`, `Abundance3`, `...` or `Depth1` `Depth2` `Depth3`, `Replicate1, 2, 3`, and so on.

Such tables, called cross tables or `wide format` are very practical at a first look and make it easy to make a first analysis for an originally intended purpose. However, such tables are extremely unflexible and we would call it **the most common database mistake ever**.

Let's assume that Abundance 1, 2, 3, 11 are the abundances of 11 species. If now, a 12th species occurs, a new column has to be added, and a 13th and so on.

Or let's assume, Depth was measured in 1m steps with `depth1=0m`, `depth2=1m`, …, but the measurement interval has to be changed to 2m steps. Then every second column remains empty. Or we add a new depth of 3.5m. Where should we add the new column?

It is obvious that such things are very difficult in Table 8, while they are already built-in in Table 1. On the other hand, a crosstable like Table 8 can be easily derived from Table 1 by the software, either a "crosstable query" in a database system or a so-called pivot table in Excel or LibreOffice.

The opposite, transforming a crosstable to a correct data base table needs more effort, either manual error-prone copy and paste, advanced tools like indirect references in Excel or package `reshape2` in R, or even programming.

**Tab. 8:** Crosstables are very inflexible and not suitable for raw data. When needed, they can always be derived from correct tables by the software.

| Location | Date | Abundance1 | Abundance2 | ... |
|---|---|---|---|---|
| RL111 | 2002-10-10 | 100 | 80 | ... |
| RL111 | 2002-10-11 | 190 | 0 | ... |
| RL111 | 2002-10-12 | 10 | 20 | ... |
| RL117 | 2002-10-10 | 1000 | ... | ... |
| RL117 | 2002-10-11 | ... | ... | ... |
| ... | ... | ... | ... | ... |

## 5. Avoid redundancy: columns should not be derived from others by simple transformations

To make a decision whether the following table (Tab. 9) is correct or not, we need to consider if Biomass was an independent *original* measure, e.g. was directly measured with a balance, or if it is just a derived measure, calculated from Abundance.

If it is only calculated, it should not be entered in the raw data table. The reason is not the small amount of additional memory used, the reason is data consistency. Let's assume that it turns out that the conversion factors were wrong. Then the biomass needs to be changed at all places. On the other hand, if biomass is calculated by a formula in a query (or a formula column in excel), only the conversion factors in the species table have to be changed and everything is automatically updated.

**Tab. 9:** Table with potentially redundant column

| Location | Date | Depth | SpeciesNo | Abundance | Biomass |
|---|---|---|---|---|---|
| RL111 | 2002-10-10 | 1 | 1 | 100 | 1000 |
| RL111 | 2002-10-10 | 1 | 2 | 190 | 380 |
| RL111 | 2002-10-10 | 1 | 3 | 10 | 10000 |
| RL111 | 2002-10-10 | 3 | 1 | 80 | 800 |
| RL111 | 2002-10-10 | 3 | 2 | 0 | 0 |
| RL111 | 2002-10-10 | 3 | 3 | 20 | 40 |
| RL117 | 2002-10-10 | 1 | 1 | 1000 | 10000 |
| RL117 | 2002-10-10 | ... | ... | ... | ... |
| ... | ... | ... | ... | ... | ... |

Data base experts may use a generic table format like the following, that write almost everything in a few rows. Here we have again one or several key fields, e.g. Date, Time or Location and then only two important columns, `Variable` (or Parameter) and `Value`. An additional comment field may also be useful, and (as a pragmatic approach) also the measurement units.

**Tab. 10:** A generic data format. Please note that in most cases it would be much better to store the `Unit` together with other criteria like detection limit and measurement method in a separate table.

| Date | Variable | Value | Unit | Comment |
|---|---|---|---|---|
| 04.08.2014 | PO4-P | 0.1 | mg/L | ... |
| 04.08.2014 | NO3-N | 3 | mg/L | ... |
| 04.08.2014 | NH4-N | 0.1 | mg/L | ... |
| 04.08.2014 | Temp | 22 | deg C | ... |
| 04.08.2014 | pH | 7.8 | - | ... |
| 04.08.2014 | Chl-a | 5.4 | mu g /L | ... |
| 04.08.2014 | ... | | | |
| 12.08.2015 | PO4-P | 0.012 | mg/L | ... |
| 12.08.2015 | NO3-N | 2.5 | mg/L | ... |
| 12.08.2015 | NH4-N | 0.01 | mg/L | ... |
| 12.08.2015 | Temp | 23 | deg C | ... |
| 12.08.2015 | ... | ... | ... | ... |

## Things to remember

The following seemingly little details can be very important and should carefully be considered:

- How to save measurement units?
  - in some programs (e.g. Excel) it is possible to add the measurement units directly after the column names, e.g. Abundance ($ml^{-1}$). However this limits data exchange between different software programs, e.g. MS Access or R.
  - It is therefore better, to create a separate table with the measurement units or to use the format of Table 10.
- Which identifiers and codes should be used?
  - The use of consistent identifiers (for column names, tables and queries) is very important, both for understandability by humans and for avoiding software problems. Therefore, it is highly recommended to create consistent (or use existing) schemes, e.g. all lower case `speciesno` or capital first (`Speciesno`), so-called camel character style (`SpeciesNo`) or underscore notation `species_no`. All of this is o.k., but mixing should be avoided.
  - To be compatible with all kinds of software, identifiers should be composed of letters, numerals and the underscore. All other special characters (e.g. hyphen), accents, umlauts and spaces should be avoided, even if your favorite program supports it.
  - Similar rules apply for codes to be used in the keys of the tables. At a first look, the rules are less strict here (e.g. allow spaces, -, or comma), but sometimes we want to make a cross table and the codes become column names ...
  - If character values for **coding** of variables with many levels are used, e.g. you've sampled n different sites and want to name them `site1` to `siten`, you have to be careful. Depending on the total number of levels in your variable you should number using leading zeros for the one-digit numbers, e.g. `site01`, `site02`, ... `site11`,..., `site14`. If only `site1`, `site2` ... `site11` are used, then after ordering, `site11` ... `site14` will follow after `site1` and before `site2`, because the codes are ordered alphabetically and not as numbers. This may lead to confusion, and even more important, can cause serious problems in subsequent analyses, i.e. when the correct order is needed.

- Make a clear distinction between “not measured” (no information available) and “nothing measured”, i.e. value below the detection limit or species not found.
  - not measured is often encoded with an empty cell (Excel) or NA (not available), sometimes also with another numeric code like -1 or - 999.
  - A different code has to be used if a measurement was below the detection limit, e.g. $< 0.05$ or “nondetect” (n.d.) and sometimes simply 0 (zero) or a so-called *zero replacement value*.

  The way how to handle nondetects is a special topic, that is not (yet) covered in this text, but see Analytical Methods Committee (1987, 2001) for a short overview and recommendations.

## Exchange of data between work groups

Everyone has its favorite software. This is o.k. and has to be respected. In contrast to this, it is not self-evident that everyone has each program at hand or wants to download and learn a multiverse of tools. That’s why it is advisable to use open formats that can be imported by almost all programs, as long as the amount of data is of small to medium size.

- text format (often called ANSII or ASCII). Files of this have often the extension .txt, .csv or .dat.
  - make clear what is the decimal separator (English . preferred) and what the column separator (e.g. comma, semicolon or tab). Make sure that the column separator is not used elsewhere (e.g. in comments) or if this unavoidable, that such constructs are enclosed in quotes.
  - Be careful with “.csv”, which means “comma separated values” (decimal is dot, column separator is comma), but some languages (e.g. German) use comma for the decimal and semicolon for the columns :-(
  - for big data bases, special formats are in use, e.g. access data base files (.mdb, .accdb) or NetCDF, a special format for multidimensional data used by oceanographers and modelers. Please contact your collaborator, before sending emails with GB of data to her or his mobile phone.
- Date formats can create lots of confusion, so is the 2015/08/10 the tenths of August or the eighths of October or what comes first in alphabetical order 2.1.2014 or 12.02.2012?
  - All this can be avoided if the date format of the ISO 8601 standard is applied (https://en.wikipedia.org/wiki/ISO_8601), e.g. `2014-08-10`, or with time and time zone `2015-08-10 15:14:05 GMT` or `2015-08-10 15:14:05+00:00`.
  - Apropos time zone: please check yourself and your measurement device if “real time” or daylight saving time is used during the summer.
- It is a good idea to use compression for large text (or csv) files, but please use a common archiver. Here, ZIP should be a first choice. Other formats like .tar.gz may be fine for special groups of people, e.g. for programmers and Linux users. And yes, rar, 7Z, . . . etc. compress even better but please be nice to your addressee and avoid proprietary formats in your standard communication.

## Acknowledgments

This guide and especially the examples are based on a translation of the data exchange guide from Gaedke and Tirok (2004) of the Aquashift priority program of the DFG. We are grateful to Lothar Paul for comments on the first version.

## References

Analytical Methods Committee (1987) Recommendations for the definition, estimation and use of the detection limit. Royal Society of Chemistry, London. Analyst 112, 199 - 204.

Analytical Methods Committee (2001) Measurement of near zero concentration: recording and reporting results that fall close to or below the detection limit. Royal Society of Chemistry, London. The Analyst 126, 256-259. DOI 10.1039/b009590g.