Design, Implementation and Testing of the Temporal Swiss Database

Yannick Widmer
Facharbeit 3P.

April 19, 2011

Contents

1 Introduction

This work is about the feed database called «Schweizerischee Futtermitteldatenbank» of the «Bundesamt für Landwirtschaft (BWL)».
For optimal feeding of domestic animals, farmers and agriculture companies of Switzerland need information about the content of nutrients and minerals in different feed types. This information is measured and then stored in the database. However, the database stores only averaged value of multiple measures and that is not enough for enhanced statistical analysis.

The first task was to make an ER model of the current database. The difficulty of this task was that the only information available was an SQL scheme of the database that was designed just to fit the assignment and not to be easy to maintain, extend or to understand. Also this scheme wasn't in the form, as one may be used to in SQL so things as relations and aggregation where hard to see and understand. So after a process of understanding the first result was an ER Diagram of the database.

After that, the second task was to change things of the Diagram to enhance the current database design and fit a little more the standards of data modelling. And then to make the mentioned statistical analysis possible in future, temporal information had to be added.

The lat task was to implement a part of the database, to fill in dummy data and to test some query's on it to estimate the efficiency of the new design.

2 Deriving current database design.

The first Task was to derive an ER-Diagramm from the current database, the only available information was an SQL scheme. From this the task was to find out what are the entity set the relation among them and for what each stands for in the real world.

This part was hard because there was very few information. Further it was hard to understand the meaning of entity sets and attributes because of the complicated names they had. As an example consider 'DI_Range' of table 'tbl-datainput'. The text segment presented below is an example of how tables in SQL schema of the feed database are defined:

```
/*Table structure for table `tbl_datainput` */
DROP TABLE IF EXISTS `tbl_datainput`;
CREATE TABLE `tbl_datainput` (
  `ID_tbl_DataInput` int(10) unsigned NOT NULL auto_increment,
  `ref_FeedSpecNr` int(10) unsigned NOT NULL,
  `DI_SourceType` int(2) unsigned NOT NULL,
  `ref_tbl_Formulas` varchar(50) character set latin1 collate latin1_german1_ci default NULL
  `ref_Components` int(10) unsigned default NULL,
  `ref_Units` int(10) unsigned default NULL,
  `DI_Value` varchar(20) character set latin1 collate latin1_german1_ci default NULL,
  `DI_Range` varchar(50) character set latin1 collate latin1_german1_ci default NULL,
  `DI_decimalplace` int(1) unsigned default NULL,
  `DI_SD` float default NULL,
  `DI_NumberOfValues` int(10) default NULL,
  `ref_terms` int(10) unsigned default NULL,
  `DI_timestamp` timestamp NULL default NULL,
  PRIMARY KEY  (`ID_tbl_DataInput`),
  KEY `tbl_DataInput_FKIndex2` (`ref_tbl_Formulas`),
  KEY `tbl_DataInput_FKIndex1` (`ref_FeedSpecNr`)
) ENGINE=MyISAM AUTO_INCREMENT=244772 DEFAULT CHARSET=latin1 COLLATE=latin1_general_ci;
```

Then the key challenge was to correctly identify relationships among the entity sets. The relations are made by index, for example a table has an ID 'F_FeedSpecNr' Then another table, which is in a relation with this, has an index 'ref_FeedSpecNr' called 'tbl_DataInput_FKIndex1'.

In total the SQL scheme of the feed database has 12 tables and 85 attributes. Each table has an artificial primary key 'ID'. Also, all names and tokens are stored in the tree different languages, French, German and English, so they're multivalued in the ER-Diagramm. The result of the investigation is the following data requirements description and the scheme.

Data requirements and ER diagram. After a precise analysis, we get the following ER diagram that is illustrated in the next two Figures.

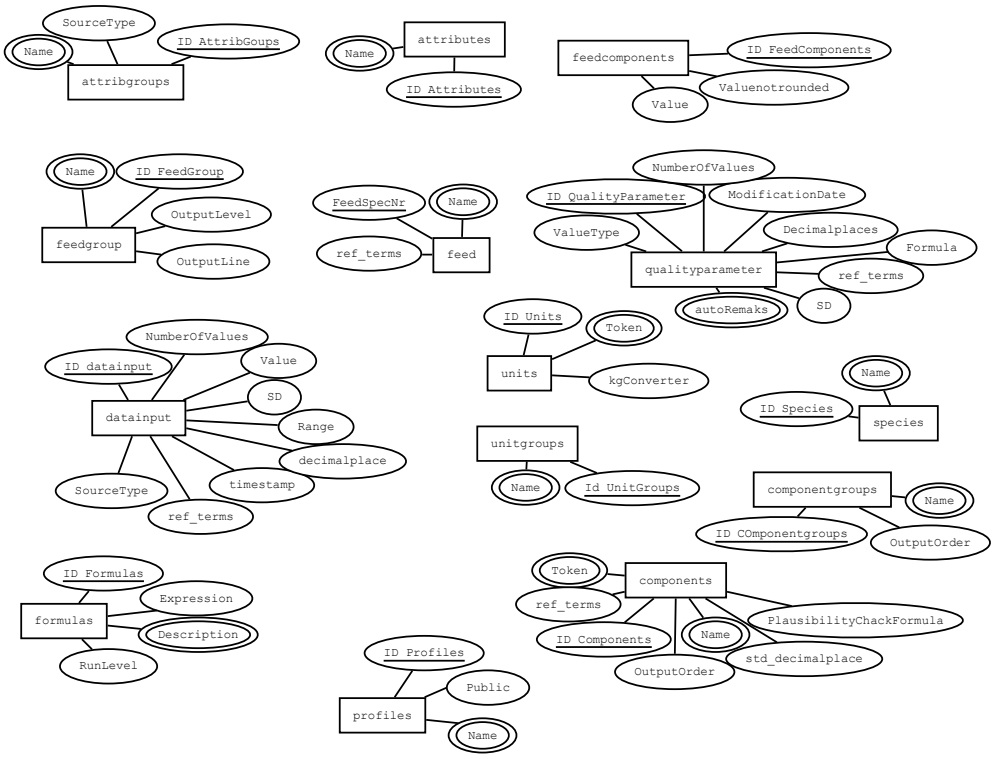

Figure 1: Entity sets and their attributes

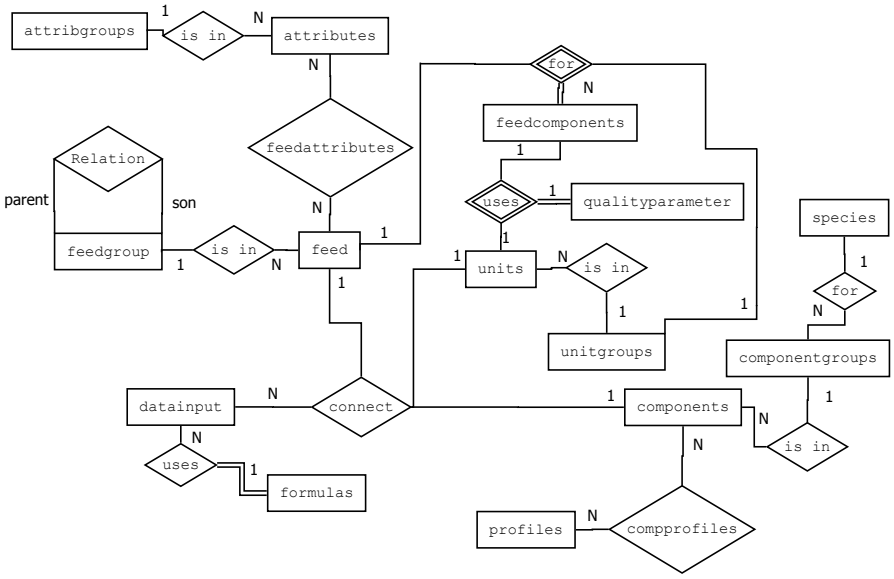

Figure 2: Relationships among the entity sets

In total, there are 14 entity sets and 10 relationships among them.

FeedComponents entity set stores the aggregated measures of different components which compose different feed types. First they make some chemical investigation, then they take the mean values and store them as an entity of this set. Each measure has a Value and Valuenotrounded, of components amount measured in the feed, it is uniquely identified by its ID.

'Feed' entity set represents different types of food used to feed domestic animals as horses, cows and pigs. These are the feed which are analyzed, they take some sample at different places for the investigation of the different types of components. Some examples of entities are wheat, grass and corn. They all have name and are uniquely identified by 'FeedSpecNr'.

FeedGroup. Different feeds types are organized into different groups which are modeled by 'FeedGroup' entity set. Additionally to that, an entity of 'FeedGroup' can also group different feed groups. Each 'FeedGroup' has a name and is uniquely identified by its 'ID'. In the web layout, position of each feed group is characterized by 'OutputLine' and 'OutputLevel'.

Attributes. Each feed has certain characteristics or 'attributes'. For example, location and date of harvesting are attributes of wheat. Different feed types may have different attributes. For example, for grass it is important to store information about how and for how long it was dried (in an open air, in a closed environment, ...). However, this information is irrelevant for wheat. These attributes are modeled as the entity set 'Attributes', each entity has a name and is uniquely identified by its 'ID'.

AttribuGroups. Similarly to feed types, attributes are organized into groups, which are represented by 'AttribuGroups' entity set. Each entity of it has a name and is uniquely identified by it's 'ID'. This time no group groups other groups.

Components entity set is uniquely identified by it's 'ID', has a name and a token. Individual entity is a kind of nutrient or mineral which composes some kind of animal feed. Further each has a 'std-decimalplace' to reduce the volume of stored information, Outputorder for the Layout on the page and a PlausibilityCheckFormula, which is a mathematical equation used to check if measures of this component is correct. An incorrect measure may occur in case of a bad chemical investigation. Then, for the relations, which are implemented by links, each has the attribute ref_terms.

ComponentGroups. Different components are also organized into groups, which are represented by entity set 'ComponentGroups'. Each entity is uniquely identified by it's 'ID' and has a name. In the web layout, position of each group is determined by 'outputorder'.

Species entity set represents different animal species like for example cow or horse. Each entity has a name and is uniquely identified by it's 'ID'.

Profiles. For users the entity set 'Profiles' stores feed assortment for species and special diet, like «Wiederkuer Aminosure-Profil» which helps them to chose the feed's, each entity has a name and is uniquely identified by it's 'ID' and has an attribute public.

Units entity set represents different measure units which are used to quantify containment of components in an animal feed. Each 'unit' has a token and is identified by it's 'ID'. Attribute kgConverter is a float number, which converts given unit into kilograms.

UnitGroups represents groups of units, each group has a name and is uniquely identified by it's 'ID'.

'QualityParameter'. For each measure in FeedComponents an entity in 'QualityParameter' is assigned. Each entity is uniquely identified by it's 'ID'. Further there are the entries: SD (Standard deviation), NumberOfValues which represents how many values have been used to take the mean, Decimalplaces, which is the standard decimal place of the amount of the component in this feed, ValueType, Formula with which the aggregated value has been calculated, ModificationDate and autoREmarks, some text about this value and how it has been obtained.

'Formulas' entity set represents an expression, where kgConverter is used to convert the obtained result given, in the unit represented by tokens, in to kilograms. These are used to calculate aggregated values with measured values. Each Formula is uniquely identified by it expression.

"Datainput' The entity set 'datainput' does not have any connection with data requirements. The system uses 'datainput' table to collect an input from the user: the user fills in a record into the table, then the system reads the record and redistributes the input to other tables in the database and, then the record is deleted. Each entity has an identifying key ID_DataInput and a bunch of additional information: SourceType, Value, Range, decimalplace, timestamp, SD (standard deviation) and NumberOfValues. This also has the entry ref_terms.

3 Introducing temporal information.

The measurements are chemical investigation done on samples of feed taken from the different attachment locations. Those investigations are done sporadically. After an investigation, the labor fills an excel table which looks like the following:

Feed	location	arrival	TSL	RA	RLB	RLBT
Orge en grains	Schweiz	22.10.1997	875.9	24.66034935	28.542071	
Orge en grains		22.10.1997		29.20913284		30.1201389
Orge expandée	Westschweiz	22.10.1997		28.62027131		32.5355685
ORGE APLATIE 1		11.12.1998	901.30	22.9668257		

In this excel table one line represents one sample taken somewhere at a certain date, which are identified by the first attributes: Feed, arrival and Location. Then, different tests can be done on this sample to decide the amount of one nutrient in this sample, those are then filled in the corresponding line like TSL, RA or RLB, which is just a small Outline of all nutrients which could be tested. For one sample not all nutrients are tested, so the excel table has a lot of blanks. And also some sample may be tested twice or more for the same nutrient, which demands a hole new line.

In the current design of the database, different measures are taken with which they calculate a mean value, which is then stored in the 'feedcomponents' entity set. In addition to the value we have some attributes, like the numbers of measures, which tells how good this value corresponds to reality.

Important to mention is that the attribute Location isn't represented at all in the database.

The next step is to distinguish all samples and measurement in the new design of the database and to introduce information about the location.

For this issue we need a new entity set 'measurement', which has the attribute 'value', which is the value obtained by the experiment. Then, 'measurement' is in relation with 'Component' and 'Sample', 'Component' represents

which component has been tested and 'Sample' should represent the real samples. 'Sample' has the attribute 'arrival' which is the date the sample has been harvested and brought to the laboratory. Then, 'Sample' has a relation with the new entity 'Location' and with 'Feed'. 'Location' has the attributes 'name' and an 'ID_Location'. And naturally each entity is uniquely identified by it's ID.

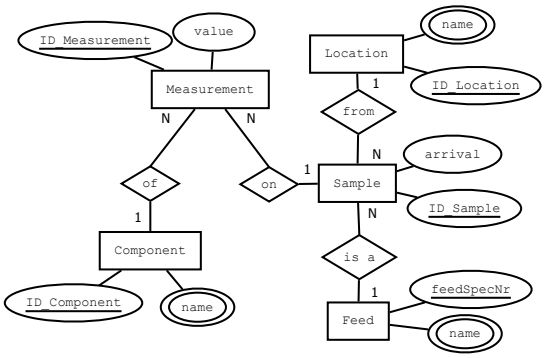

Figure 3: measure relation set

It is important to know that we omitted to list all the attributes of components which are not of importance in this picture.

Now in this design our example data would look like this:

measurement			
ID_Measurement	Value	ID_Sample	ID_Component
1	875.9	1	1
2	24.66034935	1	2
3	28.542071	1	3
4	29.20913284	2	2
5	30.1201389	2	5
6	28.62027131	3	2
7	32.5355685	3	5
8	901.30	4	1
9	22.9668257	4	2

Component	
ID_Component	name_Component
1	TSL
2	RA
3	RLB
5	RLBT

Feed	
feedSpecNr	name
1	Orge en grains
2	Orge expandée
3	ORGE APLATIE 1

Sample			
ID_Sample	arrival	ID_Location	feedSpecNr
1	22.10.1997	1	1
2	22.10.1997		1
3	22.10.1997	2	2
4	11.12.1998		3

4 Improvement to the overall design.

Analysis ER diagram. The following points are the weakest parts of the database design.

1. There is information about the web page Layout in the database since this is functional requirement this shouldn't be in the database;

2. Units shouldn't be attached to the measurement because the unit will always be the same for a component;

3. In the current design the 'componentgroup' entity set just stores a name and information for the layout. So this doesn't store change in time. Whereas the different components are grouped in to different groups for

the different diets of the different animals and this groups aren't definitive so each can change or disappear during time, but some times one would like to know which component was in a group at a certain time.

4. There are different groups which have the same attributes so one could group them, this means to make one abstract entity set group which would represent all groups and so there would be less redundancy in the database;

To improve the design first all attributes about the web page Layout have been removed.
Then units is now in relation with components.
And then the 'componentgroup' entity set has been changed in function to store the needed information. And to correct the last part. We take the new 'componentgroup' set in order to make an abstract group and let inherit the groups like 'unitgroup', 'FeedGroup' and 'AttribGroup'. All this is represented in the next diagram:

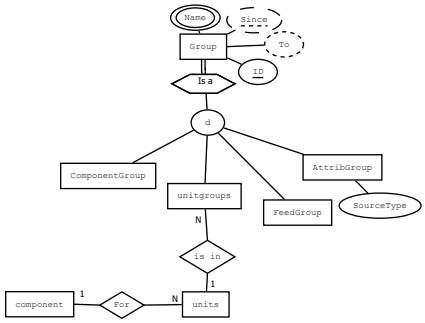

Figure 4: Modification for enhancement

5 Complete ER Diagram

Now the resulting design of the database is represented on the following diagrams. The colored attributes and Entity sets are the new one, and the uncolored are those that haven't changed.

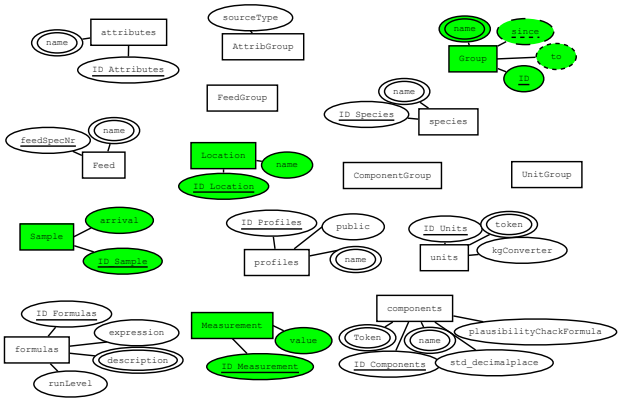

Figure 5: attributes

This time we count 14 entity sets, 27 attributes and 13 relations. For the entity sets three are gone and have been replaced by three others, the attributes are reduced because we removed all layout information and finally owing to the time information the relations incremented by three.

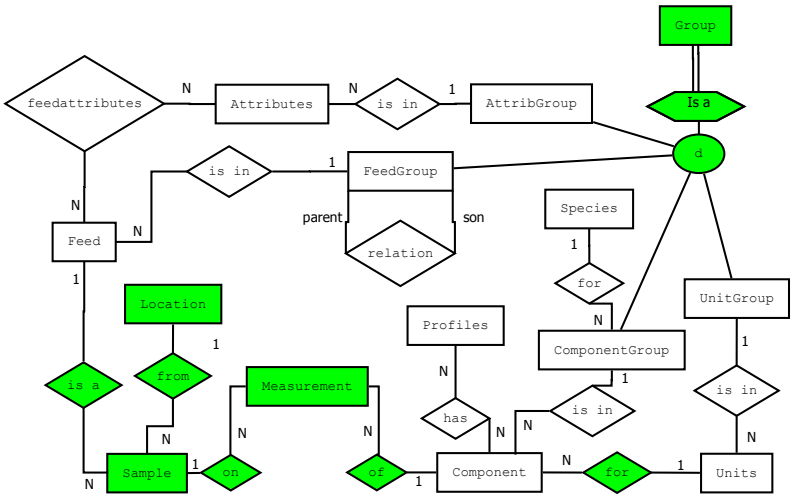

Figure 6: relational diagram

6 Implementation and Testing

In order to test the efficiency of the new design the part of the database described in the section Introducing temporal information has been implemented in postgresql. Then, dummy data has been filled in namely 62 component's, 600 feed's, 100 location's, two sample's per feed per year, 30 measurement's per sample and this for the years 1990 to 2011.

Now to test how good our design is let us try some query on it.

Let us, in order to decide where we want to get the next sample, look at which component has been tested from which feed how many times in each location in the last few years.

select Feed.name Component.name Location.name **count(*)**
from component **natural join** measurement **natural join** Sample **natural join** Location **natural join** Feed
where Sample.arrival **between** '1.1.2009'**and** '31.12.2010
group by ID_Component ID_Feed ID_Location

In the next table we see how long it takes to give an answer to this query for different time intervals:

first query	
time interval	time in ms
1.1.2009-31.12.2010	3660
1.5.2000-31.12.2011	23145
1.1.1990-31.12.2011	44593

Since this should be a fast query we want to speed it up by putting some indexes, but after some attempt's it seemed like this query can't be easily enhanced. Since the group by produces many groups. Maybe this could be achieved with some views. However, let us concentrate on a second query.

Assume we want to know the mean measured value of a certain component in a certain feed. This could be reached by the following query.

select avg(Value)
from component **natural join** measurement **natural join** Sample **natural join** Feed
where Feed.Name = 'potatoes' **and** Component.name = 'Sugar'
and arrival >= '1.1.2008'

For the test we take some of the random names that have been generated and look at different time ranges.

second query	
since	time in ms
1.1.2008	218
1.1.2000	217
1.1.1990	228

By putting an index on the foreign key 'sample_ID' in the 'Measurement' set we get the new times

second query	
since	time in ms
1.1.2008	25
1.1.2000	15
1.1.1990	25

So for further enhancement we would need more data because we wouldn't notice differences since the query is very fast by now. So after removing the indexes we are able to try a third and last query. We could make one in order to see changes in sugar in all nutrients over the years.

select distinc Feed.name avg(Value)
from component **natural join** measurement **natural join** Sample **natural**

join Feed
where Component.name = 'Sugar' **group by extract(year from** Arrival**)**

This query took 754ms, after introducing an index on the foreign key component_ID in 'Measurement' we can reduce it to 610ms, other indexes just affects the time lightly so again we would need a bigger amount of data to see differences.

7 Conclusion

What we have done so far, we took an existing database more precisely it's SQL source code and derived the ER-diagram from it. Then we tried to understand how it works and what could be changed to make it perform better and took anything away which should not be in a database. Then we took a sample of the excel table which contains the information which is filled in the database and looked for information which is not in the database. So we introduced Time information and altered a few things so that we now have a new database. To insure us that the new design of the database is working fast enough we implemented parts of the new design and filled in some dummy data. So we let act some query's on this database and checked if we could reduce the duration of the query by introducing some indexes.

Now to continue, one should make some more experimentation on the new design till we are sure that it will fill the tasks. Then it would be time to implement the hole database. Like we have seen there is a few information missing in the excel table, so before filling in the data in the database one should complete these information. Then time would come to fill in the old data followed by the new one and so one could make the wanted statistical analysis.

In this project the hardest part was to derive the database design just having a SQL code and the excel table, after that there was a lot of work to do for the new design. Then to get satisfied by the new design there was also a lot of details which had to be solved, so there was also a lot to do. For the implementation I felt like it was not very hard work, but in fact time was also running during this part maybe it's because that was the part I learned the most that I felt like it was not too hard working.