CENTRE FOR
COMPUTING AND BIOMETRICS

There are bugs in our spreadsheet!

Designing a database for scientific data

Clare Churcher and Peter McNaughton

Research Report No: 98/02
March 1998

RESEARCH REPORT

ISSN 1173-8405

LINCOLN
UNIVERSITY
Te Whare Wānaka O Aoraki

Centre for Computing and Biometrics

The Centre for Computing and Biometrics (CCB) has both an academic (teaching and research) role and a computer services role. The academic section teaches subjects leading to a Bachelor of Applied Computing degree and a computing major in the BCM degree. In addition it contributes computing, statistics and mathematics subjects to a wide range of other Lincoln University degrees. The CCB is also strongly involved in postgraduate teaching leading to honours, masters and PhD degrees. The department has active research interests in modelling and simulation, applied statistics and statistical consulting, end user computing, computer assisted learning, networking, geometric modelling, visualisation, databases and information sharing.

The Computer Services section provides and supports the computer facilities used throughout Lincoln University for teaching, research and administration. It is also responsible for the telecommunications services of the University.

Research Report Editors

Copyright

Correspondence

This paper represents work to date and may not necessarily form the basis for the authors' final conclusions relating to this topic. It is likely, however, that the paper will appear in some form in a journal or in conference proceedings in the near future. The authors would be pleased to receive correspondence in connection with any of the issues raised in this paper. Please contact the authors either by email or by writing to the address below.

Any correspondence concerning the series should be sent to:

The Editor
Centre for Computing and Biometrics
PO Box 84
Lincoln University
Canterbury, NEW ZEALAND

Email: computing@lincoln.ac.nz

There are bugs in our spreadsheet!

Designing a database for scientific data

Clare Churcher and Peter McNaughton

Centre for Computing and Biometrics
Lincoln University
Canterbury
New Zealand

Abstract

When designing a database it is necessary to have an intimate understanding of the data in order to provide accurate and versatile information over the long term. A case study is used to demonstrate the design process in setting up a database for a large and long-term scientific project.

1. Introduction

Long term scientific studies produce large amounts of data. A considerable amount of time, money and expertise is involved in the design and implementation of experiments and also in the statistical analysis of the results. Often, however, very little expertise or effort is expended in deciding on the appropriate methods for entering, storing and retrieving the data. It seems unfortunate that the considerable efforts put in by scientists and statisticians might be undermined by a poorly designed data storage system.

Many end users feel more at home with a spreadsheet than a database, and will prefer to store data for a new project in a spreadsheet. While spreadsheets are excellent for performing calculations they have serious limitations for storing related sets of data accurately, and they lack versatility for retrieving subsets of data. As the project grows the inadequacies of a spreadsheet become more serious but by then inertia has often set in. One of the compelling motivations for moving data from a spreadsheet to a database is encountering problems as the amount of data grows. The quick solution is often to set up tables in the database that look very much the same as the spreadsheet. This solves the size problem but there are more serious problems to do with accuracy and versatility that will remain.

Designing a database that will deliver the required information over the long term requires a very precise understanding of the data and the way in which they are likely to be used. The database designer needs a meticulous understanding of how different sets of data are related and needs to second guess the uses to which the data will be put. The process is an exercise in communication between the designer and the scientific expert.

This report uses the design of a database for part of the Selwyn Stewardship Monitoring Scheme to illustrate the level of understanding required and the types of questions that need to be asked in order to achieve that understanding.

Section 2 gives an overview of the types of data that are being collected for the project, section 3 looks at problems with the initial spreadsheet and section 4 describes some very basic database techniques and terminology. The remainder of the report describes how to recognise some of the problems and traps that occur and demonstrates how they may be overcome.

2. Overview of the project

The Selwyn Stewardship Monitoring Scheme is a long-term project to monitor the environment in the Selwyn District. One aspect of this project is monitoring numbers of certain insect species that act as indicators of the "health" of the environment. A number of fields in the district have been nominated as sites where the counts will take place. To be able to draw any conclusions in the future it is necessary to record as much data as possible about the conditions at the time the counts are made.
The requirements of this database are much the same as for any database, scientific or commercial:

- The data need to be stored in a way that guarantees consistency
- The database must be able to evolve gracefully as the requirements change
- The database must be able to provide accurate answers to any question that may reasonably be posed

2.1 What type of data are going into the system?

The most difficult thing about designing a database is trying to establish what data are involved and what information is going to be required in the long term. In this case, very broadly speaking, the data involved are the numbers of certain insects found on various farms and fields under different conditions. Therefore it is necessary to allow for the storage of data that include:

information about the farm (location, etc.),
type of field (modern arable, organic arable, etc.),
soil type,
dates of visits,
weather conditions,
plant cover,
stock,
transects (whereabouts in the field a sample was taken),
a history of spraying and other field treatments,
the number and species of the insects collected, and
the collection method.

This list is not exhaustive and **never will be**. The project is certain to evolve and additional types of data will become important. The design must allow for further information to be added at later stages without causing monumental disruption to the existing data.

2.2 What information is required out of the system?

Over the term of the project the types of question that might be posed are varied and numerous. Given the types of data mentioned in the above list it is reasonable to be prepared for questions such as:

- Show me the average counts per transect of each insect species for each different field type over the last 5 years.
- Show me the counts of springtails for all fields excluding those that have been sprayed less than a week before sampling.

If the database is properly designed and the query correctly implemented then it will be a routine job to export the results of such questions into a statistics package, spreadsheet, graphing application, word processor or whatever is suitable for the subsequent round of processing.

Thinking of the types of questions that may be asked, now or in the future, is crucial for deciding how the data should be stored.

3. Why not use a spreadsheet?

There are two main problems with using a spreadsheet for storing complex data. One is that inconsistencies inevitably occur in the data. Another potentially more serious problem is encountering difficulties in accurately retrieving information. The data for the insect monitoring were originally stored in a spreadsheet as shown in Figure 1.

File Edit View Insert Format Tools Data Window Help

E534

	A	B	C	D	E	F	G	H
1	Farm	Field	Date	Rep.	Springtail	A.S.W.	Fungus Beetle	Ladybird la
268	1	ADhc	Aug-96	1	2	0	0	
269	2	ADhc	Aug-96	2	2	0	0	
270	1	ADhc	Aug-96	3	7	0	0	
271	1	ADhc	Aug-96	4	3	0	2	
272	1	ADhc	Aug-96	5	3	2	0	
273	1	ADhc	Aug-96	6	3	1	9	
274	1	ADhc	Aug-96	7	2	0	1	
275	1	ADhc	Aug-96	8	6	1	1	
276	1	ADhc	Aug-96	9	2	0	1	
277	1	ADhc	Aug-96	10	5	0	3	
278	1	ADhc	Aug-96	11	0	0	0	
279	1	ADhe	Aug-96	1	0	1	6	
280	1	ADhe	Aug-96	2	1	1	1	
281	1	ADhe	Aug-96	3	5	0	2	
282	1	ADhe	Aug-96	4	0	0	5	
283	1	ADhe	Aug-96	5	0	3	3	
284	1	ADhe	Aug-96	6	1	0	2	
285	1	ADhe	Aug-96	7	0	0	3	

visit number inc / field number fixed / co

Figure 1. Original Excel spreadsheet for storing the invertebrate data.

Each row on the spreadsheet represents the insect counts from a particular transect of a field on a given day.

- Column A identifies the farm number(values 1 – 6 with the matching details of farmer's name, type of farm etc being recorded elsewhere)
- Column B identifies the field with a unique 2-4 letter code
- Column C records the date the sample was collected
- Column D the transect number from which the sample was taken (typically there were 11 transects defined for each field)
- The next 15 columns record the counts in the sample for each of the insect species under consideration.

The weather conditions, plant cover and other information relevant to a particular visit are for the most part kept in a notebook.

The first and most obvious problem is that the data aren't all in one place, so that, for example, *excluding all counts taken in a strong wind* cannot be done without a considerable amount of manual manipulation. Arguably this could be just a matter of moving the data from the notebooks to some extra columns in the spreadsheet. However this will merely compound the more serious problem of having inconsistent data.

The fact that field ADhc on Farm 1 was visited in August 1996 is repeated for each of the eleven transect samples. This inevitably leads to inconsistencies such as that between rows 268 and 269. This is not an isolated example of an avoidable inconsistency even in this relatively small spreadsheet. If more information (weather, soil type, spraying regimes and so on) is stored in this fashion then this problem will recur to the extent that there must be doubts about the accuracy of any results that are inferred from the data.

Inconsistencies such as these are not a problem of data entry: they are a problem of database design and can be very easily avoided.

4. An overview of elementary database design

In this section we provide an informal description of some basic database concepts in order to be able discuss some of the problems that can occur. The reader is referred to one of the many textbooks on the subject for a more comprehensive and formal coverage (Elmasri & Navathe 1994, O'Neill 1994, Date 1995).

While it is impossible to ensure that data are always entered accurately, it is possible to prevent many errors and inconsistencies. To avoid inconsistent data, it is necessary to ensure that information is stored only once. Data about a farm should be stored once regardless of the number of fields involved; information about a field should be stored once regardless of how often it is visited. We also need to acknowledge the fact that fields and farms are related. This is most easily depicted in diagrammatic form using Entity Relationship (ER) Diagrams as in Figure 2. Very briefly, Figure 2 depicts the following information:

In our database we will keep information about farms and fields involved in the study. We will identify them by some unique (*) code (***FieldID, FarmID***).

Means that a farm can have many fields that are involved in the study. The circle indicates that it does not have to have any. (We may want to keep information about a farm that might provide a field in the future perhaps).

Means that a field must belong to exactly one of the farms recorded in the Farm table. (Farm) shows that we keep the FarmID of this farm in with the other information describing a field. The parentheses indicate it must be a farm that we know about. This establishes the connection between fields and a farm.

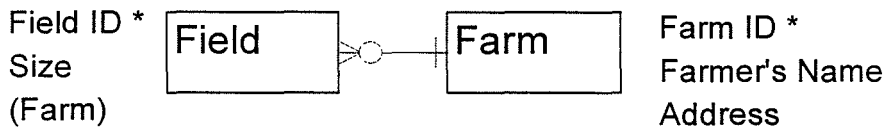


Figure 2. Entity Relationship Diagram of Fields and Farms

In a database management system (DBMS) such as Paradox or Microsoft Access this ER diagram will be implemented by creating two tables as in Figure 3.

FIELDS		
FieldID	***Size***	***Farm***
ADhc	0.8	1
ADhe	1.2	1
Etc,		

FARMS		
FarmID	***Name***	***Address***
1	Jones	West M..
2	Smith	South J..
3	Brown	West M..

Figure 3. Tables corresponding to the ER diagram in Figure 2.

The attributes ***FieldID*** and ***FarmID*** will be set up as key fields, which ensures that the values will be unique (i.e. no two farms will have the same ***FarmID***). By imposing a constraint known as referential integrity we make a connection between the two tables that ensures the ***farm*** attribute in the **Field** table refers to one of the farms that exist in the **Farm** table. These constraints are set up at design time and apply throughout the lifetime of the database.

The querying facilities in a DBMS will allow us to join the tables back together again, nominate criteria for which rows we are interested in and choose which columns we wish to see.

We can quite simply ask question such as:

Show me the sizes of all fields in West M

Show me all fields with size greater than 1.0 ha together with the farmer's name and address.

We have also solved the inconsistency problem that was evident in the spreadsheet by recording just once the fact that Field ADhc belongs to Farm 1.

5. Ensuring that the appropriate questions can be answered

It is, sadly, very easy to record information in such a way that it is impossible to use it sensibly in the future. A good example is keeping information about spraying, cultivating and other activities that are carried out on a field. It is important to ask "Why are you keeping this information? What questions might you want to ask?"

If you are storing dates and activities then it is reasonable to be prepared for instructions such as

1. Exclude all fields sprayed with insecticide within the last month for this analysis
2. Compare counts of insects for fields that use herbicides and those that don't.

The first approach is often to include another attribute (or column) in the **Field** table where these activities are recorded in free form text.

FIELDS			
FieldID	***Size***	***Farm***	***Treatments***
ADhc	0.8	1	May 1st: Sprayed with insecticide. Applied Round - Up on 3rd Apr Etc...
ADhe	1.2	1	
etc			

Figure 4 Treatments stored as free text

All the data are safely stored, but extracting the information requires us to sift manually through every record. The DBMS will not be able to recognise the dates in this format so will be unable to make any date comparisons, nor will it be able to distinguish one treatment from another.

An improvement is to consider treatments as an additional entity (table) and separate the type of treatment and the date into two separate attributes. See Figures 5 and 6.

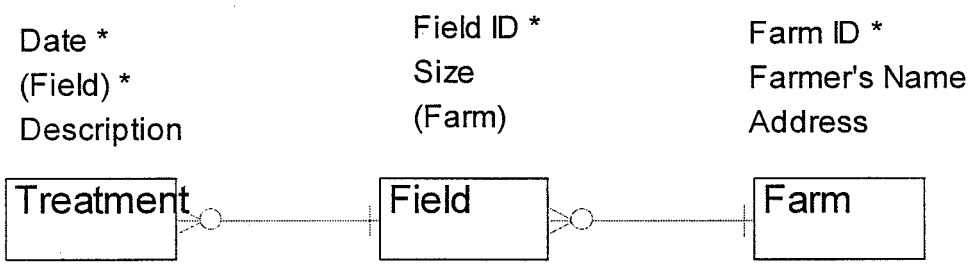


Figure 5. Treatments included as a separate entity.

TREATMENT		
Date	***Field***	***Description***
3/4/98	ADhe	Spray with herbicide
1/5/98	ADhe	Spray w RoundUp

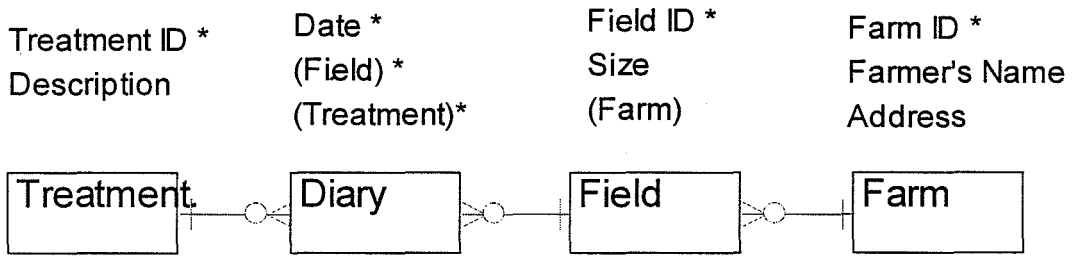
Figure 6 Table corresponding to the Treatment entity in Figure 5.

Now, by imposing a suitable restriction on the ***Date*** field, we can respond to instructions such as

Exclude fields that have had any treatments within the last 6 months.

However, we still have trouble extracting particular types of treatments because of the free form text in the ***Description*** attribute (column). We cannot expect the database to equate "spray with herbicide", "herbicide spraying" and all the misspellings that are inevitable.

A solution is to enable the data entry personnel to choose from a list of treatments. The most versatile way to do this is to have another entity or table of Treatments. Over the lifetime of the experiment the user can then add new treatments to the table just as they would add any other data. See Figure 7.



TREATMENT	
Treatment ID	***Description***
1	Spray w herbicide
2	Spray w insecticide
3	....

DIARY		
Treatment	***Date***	***Field***
1	1/5/97	ADhe
1	5/6/97	ADhe
2	3/7/97	ADhc

Figure 7 ER diagram and two of the corresponding tables for a general solution to storing treatments carried out on a field

The **Diary** table records which treatments have been applied to a particular field on a particular date.

A query can be quite routinely constructed in the DBMS to join the tables together on their common columns and to extract for example *all fields sprayed with herbicide since 1/6/97*

The two instructions we originally thought likely are both accounted for by this design.

The user will have to make some pragmatic decisions about the level of detail required in recording the different possible treatments in the **Treatment** table. They will need to decide whether a general treatment "Spray w herbicide" is sufficient or whether there should be separate entries for each different herbicide used. This can be decided only by the scientists involved, using their expert knowledge of how the data will be used. It needs to be thought about before too much data are entered, of course, but the design allows for that decision to be made by the users at any time they please.

It might also be sensible to allow additional columns for information such as the concentration of active ingredient and so on. Each time another column is added, however, the above process needs to be completed:

- Why are we recording this data?
- What types of question might be reasonably expected?
- Will the data stored in this way allow these questions to be answered?

6. All about Keys

When you create a new table, any self-respecting relational DBMS will insist that you nominate one or more of the attributes as the primary key. This is essential if your tables are going to be able to be joined together successfully to answer questions. Equally important for the design process, deciding on primary keys is a very useful way of finding out how well you understand the data.

Each row in a table records a unique piece of information (e.g. a farm, or an application of a treatment to a field). It is essential to be able to distinguish one row from another and to determine which is of current interest. If you nominate particular attributes as the primary key then the DBMS will ensure that the combined value of those attributes is always unique. For a farm this means that declaring ***FarmID*** as a key field ensures that we will never have two farms with the same ID. The value of this key attribute is then used in the **Field** table to specify the unique Farm that each field belongs to.

In the ER diagram in Figure 7 the key attributes for the **Farm**, **Field** and **Treatment** entities (tables) are quite obvious. (At this point, it is probably wise to question whether the rather obscure codes currently in use for field Ids are sensible in the long term.) What about the table **Diary** that relates fields and treatments. The attribute ***Field*** won't do because a field may appear several times with different treatments. Similarly a ***Treatment*** will appear several times for different fields. We could consider the combination ***Treatment*** and ***Field.***
To decide we must ask
Could a field have the same treatment more than once?
To which the answer is "yes". So that won't do as a possible key since it is not unique.
What about ***Treatment, Field,*** and ***Date?***
Could a field have the same treatment more than once on the same day?

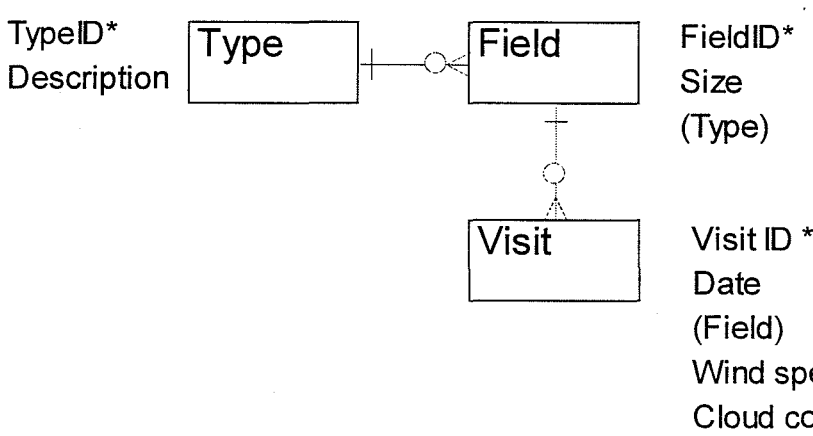
To which the answer is, for this problem, "no". This combination is therefore an appropriate key. If for some reason it was necessary to distinguish, for example, two sprayings with herbicide on a particular field on the same day, then we would need to go through the routine of asking

Why is it important?
What questions require this level of detail in order to be answered?
Does the present design allow these questions to be answered?

7. Ensuring that historical data is kept accurately

Often in the initial stages of a project some attributes are placed in the wrong table because it has been overlooked that the value may change in the long term. This can cause some grief if the mistake is not noticed for some months or even years by which time it can be difficult to repair.

As an example consider the type of a **Field** (i.e. whether it is a modern arable, organic arable, sheep and beef etc). For the same reasons that we created additional tables for **Treatments,** we have a table for **Types** so that more types can be added by the user at a later stage. At first glance it is quite reasonable to think that **Type** is directly related to **Field** as in the fragment of the ER diagram in Figure 8, where a **Field** has exactly one type associated with it. We also show that a field may be visited many times (in order to take the samples) and that we can record some of the weather conditions at the time of the visit. (For convenience we have introduced an automatically generated attribute ***VisitID*** in order to prevent the concatenated keys becoming too long as the problem becomes larger. The fields ***Date*** and ***Field*** would have served as a key just as well as a key for **Visit**.)



FIELD			
FieldID	**Size**	**....**	**Type**
ADhc	x		MS/B
ADhe	y		MS/B
etc			

TYPE	
TypeID	**Description**
MS/B	Modern Sheep and Beef
MA	Modern Arable
OA	Organic Arable

Figure 8 ER diagram and corresponding tables for recording the type of a field.

Field ADhe has type MS/B (Modern Sheep and Beef). Some years later with a change of ownership, however, this field might become MA (Modern Arable). If we change the value of the ***Type*** attribute in the **Field** table to MA we have lost the information that all the visits previously (and the samples taken) were from a field being farmed as a MS/B. This is obviously a hopeless situation but the design as it is at the moment does not really allow us to do anything else. A quick (but unsatisfactory) fix is to add a few extra columns to the **Field** table to allow room for some more ***types*** and possibly some dates (of the change over). This will not work in practice as it becomes ungainly to ask questions about which type of a field on any particular visit.

What has gone wrong here is that the ER diagram is incorrect. It says that a field has exactly one type. In fact a field has exactly one type **at a time**. Over the lifetime of the experiment a field may have many types.

To guard against this it is necessary to ask for each attribute of an entity:
Will the value for this change over time, and is it important to keep the historical data?
and to be somewhat pragmatic about the answer.

Some examples:

Might the type of a field change over time?

Emphatically "Yes" and we will see how to fix this in a moment.

Might the owner of a farm change over time?

"Yes." But do we care? Are we ever likely to want to ask questions such as *Show me all the fields that have ever been owned by Smith*? For the purposes of this particular database probably (or pragmatically) not. In that case the attribute ***owner's name*** in the **FARM** table should really be ***current owner's name*** because that is the reason we are keeping this particular piece of data, and we will forego keeping track of previous owners.

Maintaining a history of the farming practices of a field can be dealt with in two ways depending on why it is important to know this information. One is to keep a complete history of the field (since it entered the study, or even before if the information is available) keeping track of all the changes with start and finish dates. The other way is to keep the information about the type of field with the visit, i.e. at the time of this visit the field was *sheep and beef.* We prefer the second method in this case. It is simple to record, the queries are easier to formulate, and the only possible loss of information would be if a field underwent several changes in practice between visits (which is extremely unlikely and probably not of sufficient importance to warrant the extra effort). The modified ER diagram is shown in Figure 9.

Designed in this way it is possible to accurately fulfil a request such as:

Differentiate the counts taken while fields were being farmed organically.

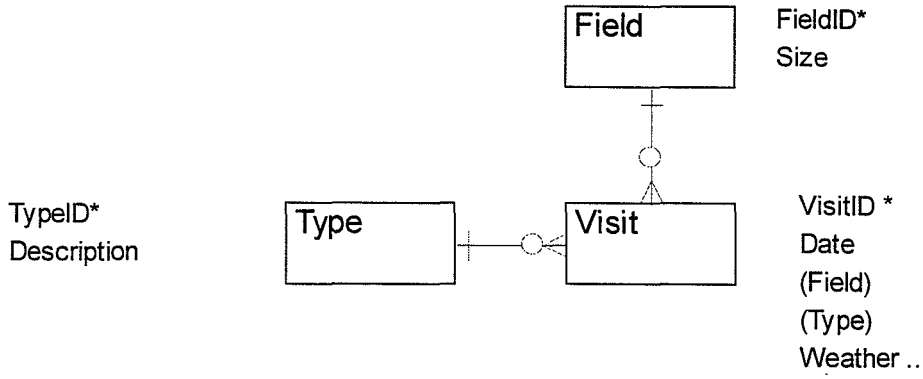


Figure 9 ER diagram showing Field type being connected to a visit in order to maintain historical data.

8. Avoiding "missing" data

A scientific project that will eventually be subjected to statistical analysis depends very much for its success on the data being complete. Where a piece of information is missing it could be because

The measurement wasn't applicable in this case
The measurement is applicable but for some reason wasn't taken
The measurement was taken but had a "null" value

For some data, it is important to distinguish these cases and the database can be designed to enforce this distinction.

It is possible to place a constraint on an attribute so that a value must always be entered. It is important to be sparing in applying this constraint or the restriction will cause aggravation. For example, it would be nice to know a farmer's phone number but it is not essential for any subsequent analysis and we do not want to be prevented from entering other data about the farm because we don't happen to have the phone number on hand. Most often it is the connections between tables that are the most important to enforce. For example, if we record information about a visit, it is absolutely essential that we know which field was visited. On an ER diagram, optional connections have a circle while compulsory ones do not.

An example relevant to the monitoring study is the importance of maintaining information about the plants in the field at the time the samples are taken. There are two considerations. One is to keep a record of the crops (possibly none, one or many) in the field and the stage of their development (height, coverage etc). The other is from which particular plant type the sample was taken. The monitoring experiments also include a quadrat survey of the all the vegetation present, but this has not been included in the database at the moment.

The likely instructions could be:

Differentiate all counts where lucerne was present as a crop.
Differentiate those samples taken from clover.

The results of instructions such as these will be pointless if there is a number of samples where the plant cover of the field has not been recorded **and** we cannot distinguish whether the field had no crops at the time, or we don't know what crops were present (old data perhaps).

We can design the database so that, whenever samples are taken, the field **must** have a crop associated with it. We can include "not recorded" and "bare ground" as possible crops to deal with the exceptional cases. This forces a user to make the distinction at data entry time. A field may have many crops at one time (especially for organic farming) and it is necessary to be able to record the growth of each. The three entities **Visit, Visit/Plant** and **Plant** in Figure 10 enable us to record this information.

9. Getting the detail correct

It is not always obvious where a particular attribute really belongs. The above discussion of fields and plants is a good example.

A field may have many crops on it and this has been taken into account in the previous section. In addition, where there is more than one crop on a field, each transect from which a sample is taken may have a different plant associated with it. The expert scientific advice was that this was also important to record. Now there is some uncertainty about what we really mean by the height of a crop. Do we mean some sort of average height of the crop over the entire field or do we want to know the possibly different heights on each of the transects. This depends only on how the scientists intend to use the data. This is where the communication between scientist and designer becomes critical. The designer is obliged to ask the question and to ensure the expert's requirements are understood. The consensus at the time of writing was that only one height was required to describe the growth of the crop. This means height should stored once per crop per visit in the **Visit/Plant** table as in Figure 10. If the decision had been that the difference in heights on each transect was important then the heights would need to be stored in the **Transect** table.

The ER diagram in Figure 10 also shows that where samples are taken the associated plants must be recorded. It is possible to record data about a visit and not include information about the plants (the circle denotes this option). However, if we record any counts then they must be associated with a transect, which in turn must be associated with a plant.

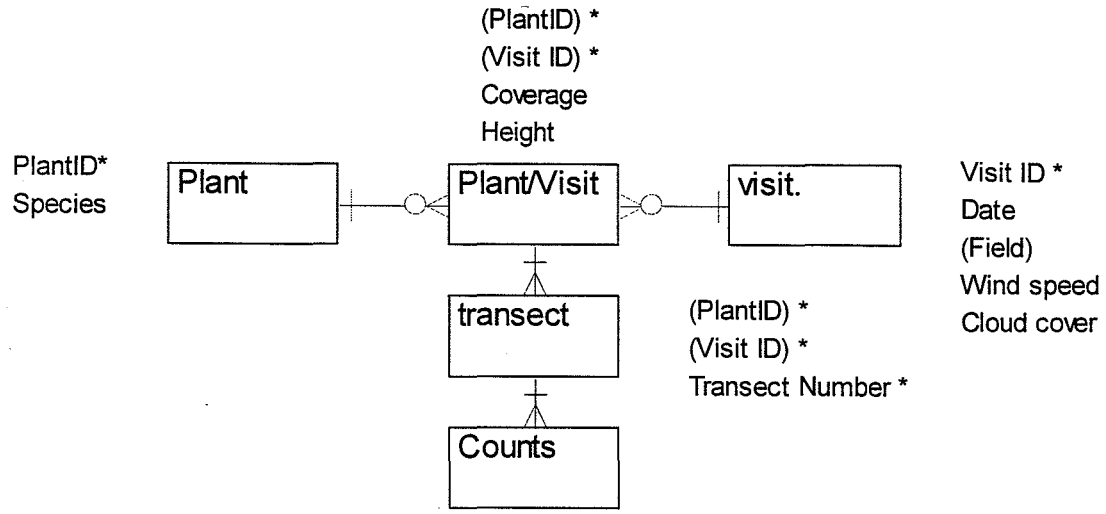


Figure 10. ER diagram representing the data involved with recording plant coverage

10. Making it useable

The final thing to consider now that the initial design has been undertaken is to implement the database and make it easy for a user to enter data. Certainly, the data entry personnel need never know how many tables are involved or how they are connected. A good DBMS will have the facility to create easy-to-use data entry forms that can simultaneously and quite transparently enter data into several tables at once. These forms should be based on the collection methods used in the field as much as possible in order to minimise data entry errors that cannot be trapped by the design. It is quite conceivable that the data could be entered directly into a laptop in the field, and the design of the interface needs to accommodate this. If data are to be entered on paper forms in the field, then these need to be redesigned to make transcription into the database easy but, more importantly, they need to be able to capture the new found complexity of the problem. For example, the existing paper data forms for this project made it difficult to record information where a field had more than one crop.

Designing and implementing a user interface to the database is another task that deserves care and thought. Users will not be bothered to use a database with an interface that is clumsy, irritating or doesn't fit their conception of how the data are collected or stored. This stage of the project also requires a considerable amount of communication between designer, experimenter and data entry personnel in order to produce a prototype that can be tested and modified until everyone is satisfied. Modifying the interface will not affect any of the underlying tables or data, so can be carried out over the lifetime of the project.

A prototype design is currently implemented in Paradox and some of the present data entry forms can be found in the Appendix.

11. Conclusions

It should be clear that designing a database is not a job to be undertaken hurriedly. It is a valuable exercise for both the designer and the scientist. The scientist is forced to confront questions that may not have arisen so far in the experiment. They may not have considered

That fields may have more than one crop
That a farmer may sell a field
That different machines may be used to collect the data on a single field visit
Whether they may want to be able to select plant cover by species or family or both
How they may wish to use the data about weather
That the weather may undergo a drastic change part way through a visit

All of these things and a multitude more need to be considered and some decision needs to be made whether they will be important to the long term experiment.

The current state of the data model at the time of writing is presented in the Appendix. It is certain to undergo many more changes but extra tables may be added quite easily without altering the existing data. For example counts are also being taken of worm and skylark populations and new tables recording the relevant information can be simply connected to the existing **Visit** table if that seems appropriate (after some careful thought, of course).

The database has come a long way from the original spreadsheet and notebooks. The main advantages are:

Many avoidable inconsistencies have been eliminated,
The database will be able to provide answers to the questions that are likely to be asked,
The database can evolve as more measurements and experiments are added to the project.

References

Date, C.J. 1994. *An Introduction to Database Systems* Addison Wesley

Elmasri, R and Navathe, S.B. 1994 *Fundamentals of Database Systems* Benjamin-Cummings

O'Neill, P. 1994 *Database: Principles, Programming, Performance* Morgan Kaufmann

Appendix

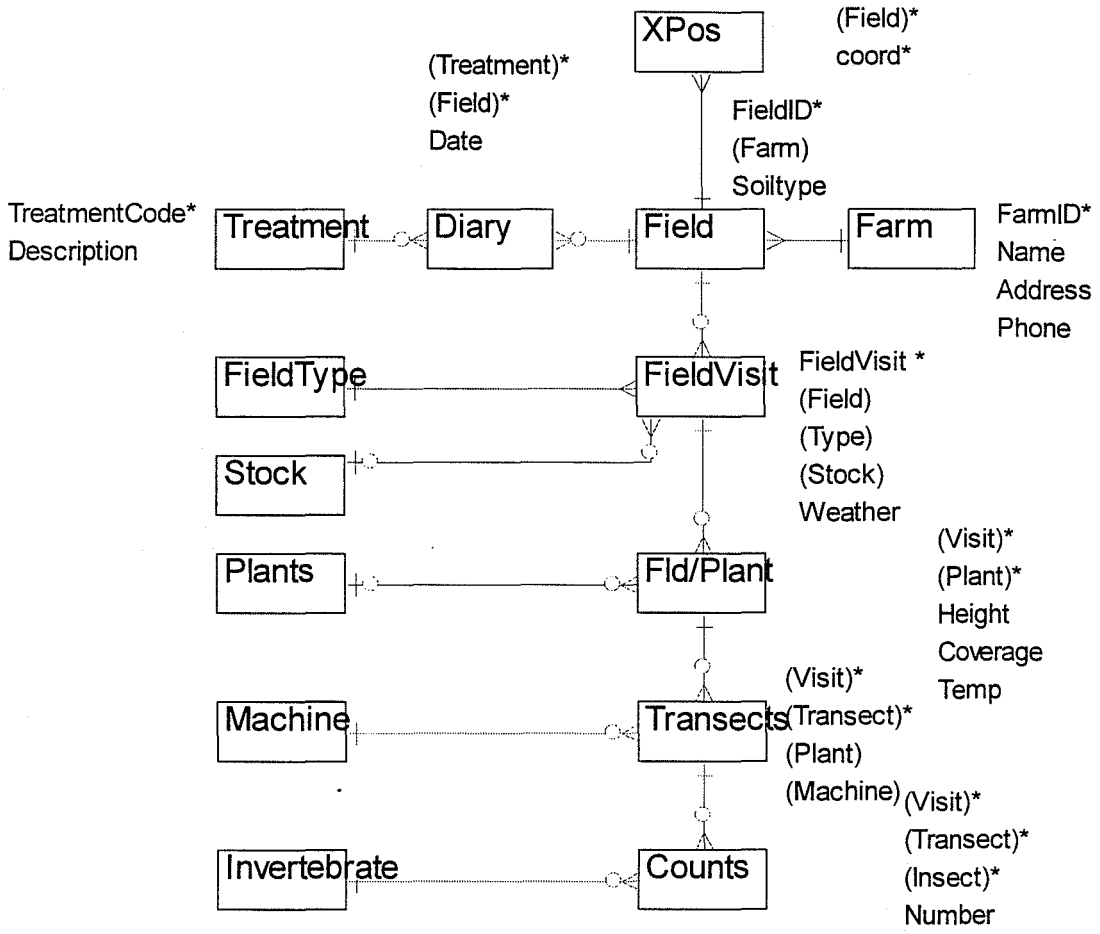


Figure 11 The current data model

Figure 12 Opening Screen for the Selwyn Monitoring Scheme Database. The buttons take you to the main data entry forms (shown below). The menu item *lookup tables* take you to the less used forms based on a single table eg invertebrates (see Figure 16), stock, plants etc.

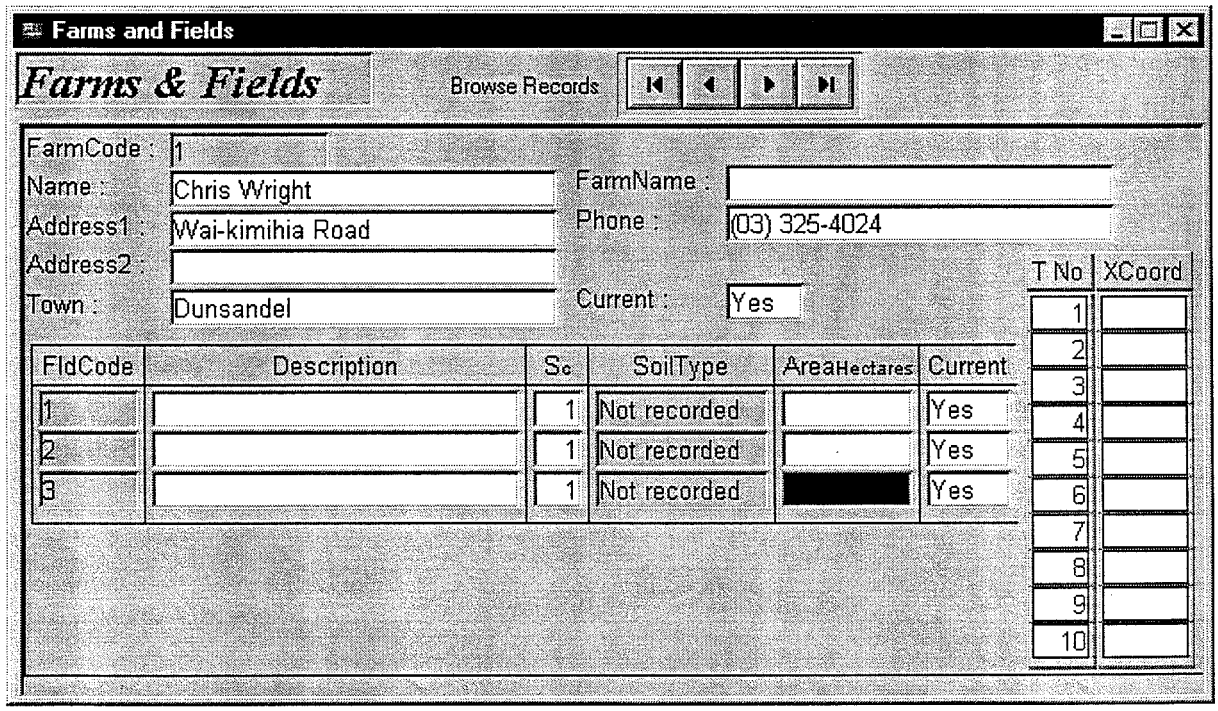


Figure 13 Form to record Farms, Fields and the transect positions on each field. Where a farm has more than one field the transect – x coord values will adjust to show the values associated with the highlighted field.

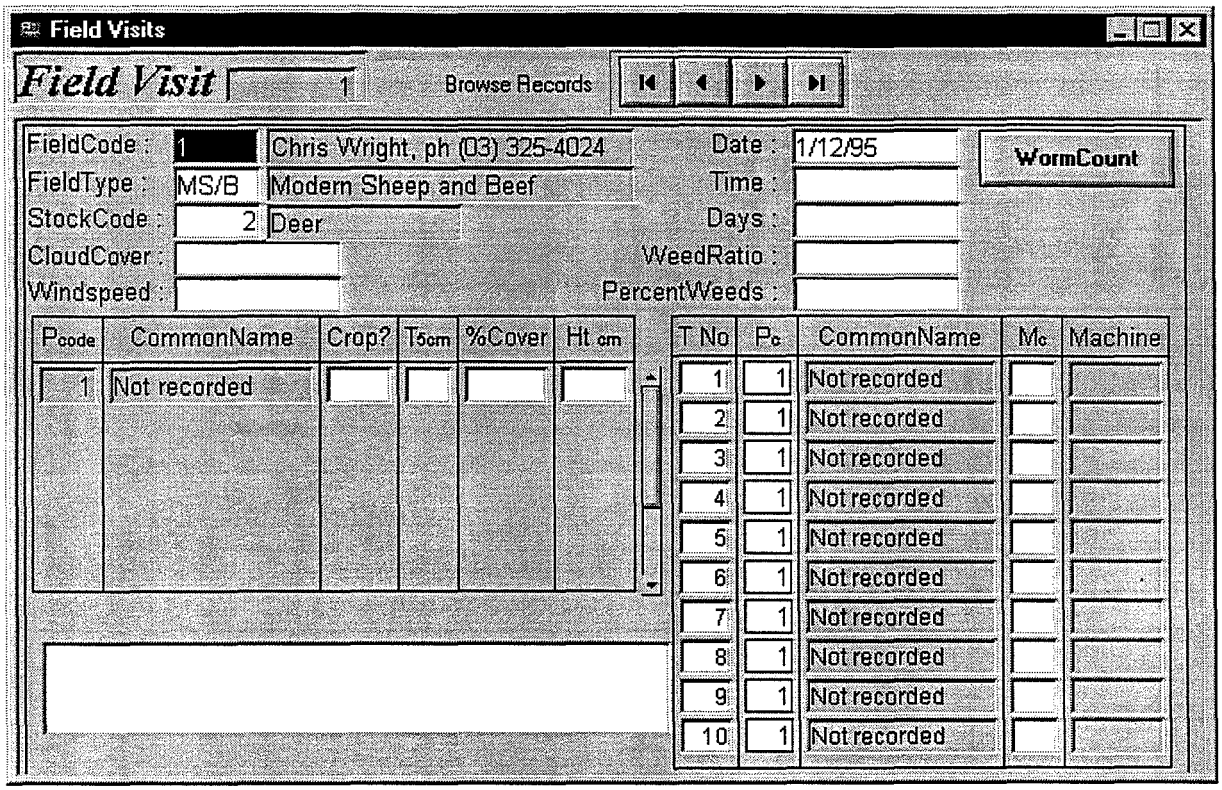


Figure 14. Form to record informatioin obtained on a field visit: Weather, plant cover, plants on a transect, machine.used to sample transect and so on. Values for attributes such as plant, stock, field which must exist in another table can be selected off drop down lists for ease of use. The default value for the required plant attributes is set to "not recorded"

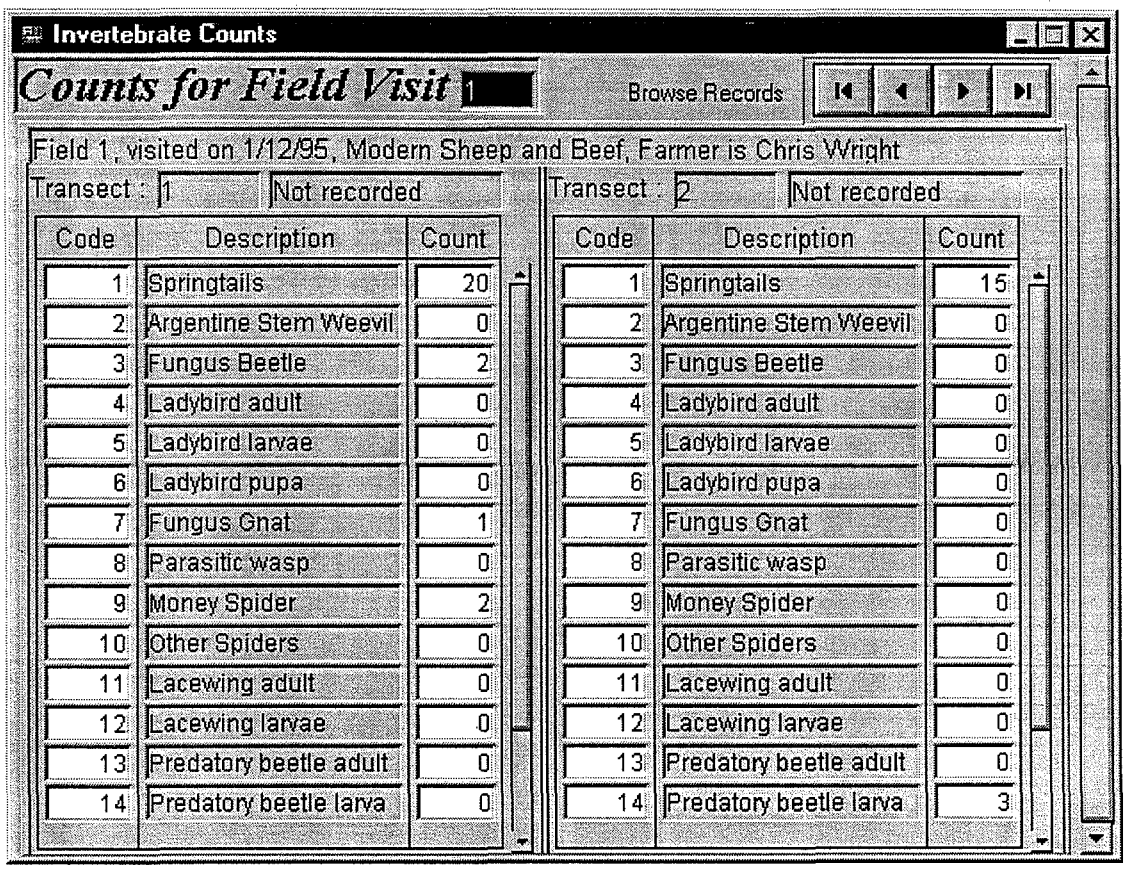


Figure 15 Form for recording the counts from a sample. The form defaults to show all the insects currently being considered in the study. This will update automatically if insects are added to the insect table (or denoted as no longer in the study). See also Figure 16.

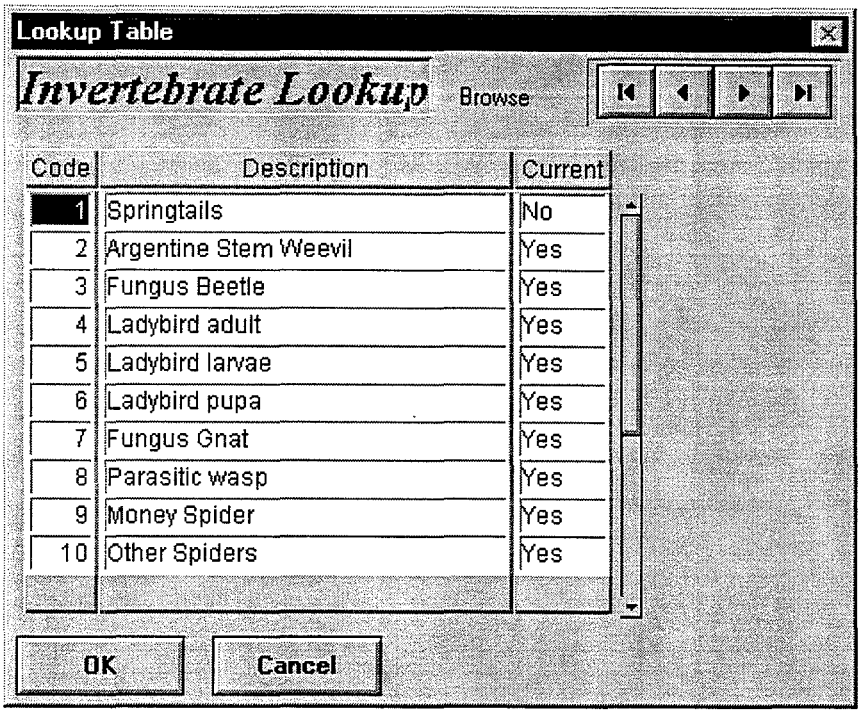


Figure16 Form to update insects being considered in the study. Insects cannot be deleted if any samples include counts for them, but they can be marked as no longer current to the study. Similar forms exist for plants, soil types, filed types, and so on.

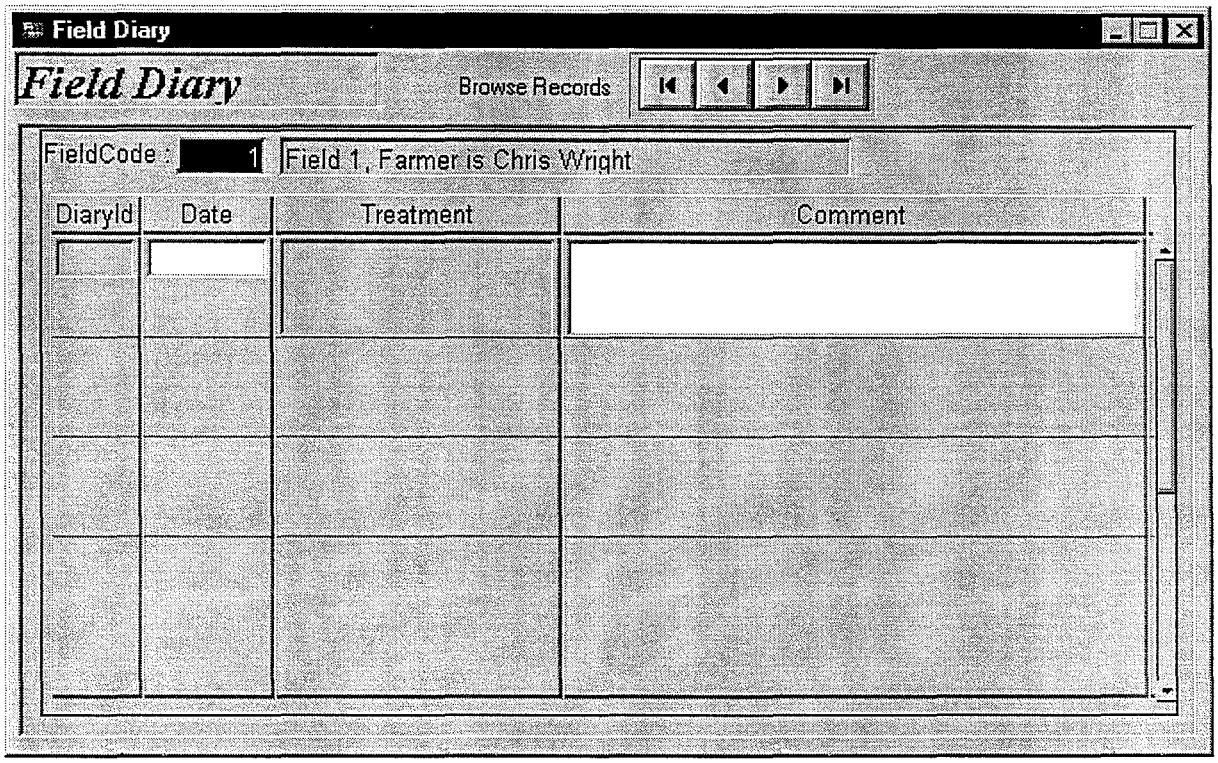


Figure 17 Form to record treatments to fields. The value of the treatment attribute can be selected off a drop down list of values currently in the treatment table.