Handling Spatial Objects in a GIS Database -Relational v Object Oriented Approaches

Paul Crowther[1] and Jacky Hartnett [2]
[1]*Sheffield Hallam University,*
School of Computing and Management Sciences,
United Kingdom.
[2]*University of Tasmania,*
School of Computing,
Australia.
e-mail P.Crowther@shu.ac.uk
J.Hartnett@utas.edu.au

Abstract. One of the foundations of spatial analysis is the object. Objects can be many things varying from an actual feature which can be extracted from a scene to more abstract entities which are associated with those features. There are a variety of database structures which can be used to store data about spatial features. These include RDBMS (Relational Database Management Systems) OODBMS (Object Oriented Database Management Systems) and ORDBMS (Object Relational Database Management Systems). All of these have retrieval systems based on SQL (Structured Query Language).

RDBMS systems store data in tables linked by foreign keys, data fields which two or more tables have in common. These form the basis of the majority of database systems in use and include products such as Oracle and Microsoft Access. They can be used with GIS via ODBC (Open DataBase Connectivity) and handle spatial objects by extending the data types available.

OODBMS and ORDBMS such as PostgreSQL also store data in tables, but each of the tables represents an object (or object class) which spatial users are more comfortable with. The tables are arranged in a hierarchy with objects lower in the hierarchy inheriting attributes from those at higher levels. It is also possible to have objects which are composed of other objects. The dominant development methodology for object oriented applications is based on UML (Unified Modelling Language).

The aim of this paper is to compare the implementation of a GIS (Geographic Information System) database using RDBMS and OODBMS developed by normalisation and UML methodologies. To this end database schema of both types will be presented. These have been developed for an agricultural domain involving crop and field monitoring.

4. INTRODUCTION

The Northwest of Tasmania is one of Australia's prime cropping regions, producing large quantities of poppies, pyrethrum, potatoes, peas and onions for processing as well as other crops for the fresh vegetable market. There have been several studies (Barrett et al 2001) conducted in this area using satellite imagery for agronomic purposes (Figure 1). For these studies a database has been developed to store information about paddocks and the crops grown within the paddocks.

Independently, there have been a number of operational databases developed for specific crops and groups of crops by processors who would like to link these to spatial information via a GIS in order to aid with activities such as planning crop management (efficient scheduling of contractors for planting, spraying and harvesting for example), identification of areas suitable for a specific type of crop, yield prediction and crop health.

Most of these applications require the use of a database consisting of multiple tables and, ideally, a linked GIS.

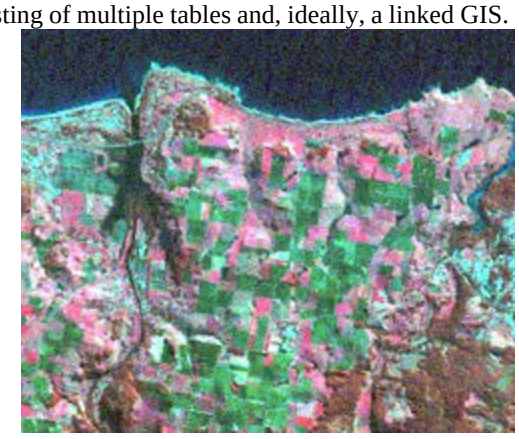

Figure 1: section of a stacked satellite image set over Ulverstone, Northwest Tasmania for the 1999 – 2000 growing season

The aim of this paper is to compare database design methodologies for developing efficient schema with a spatial dimension using the agricultural domain as a basis.

2. RELATIONAL DATABASES

2.1 Relations

The majority of databases in use today are relational database management systems (RDBMS). These consist of structured tables of related data which are linked to each other. These tables are known as entities and are linked to other tables by relations. Graphically the structure can be shown as an entity-relationship diagram (figure 2).

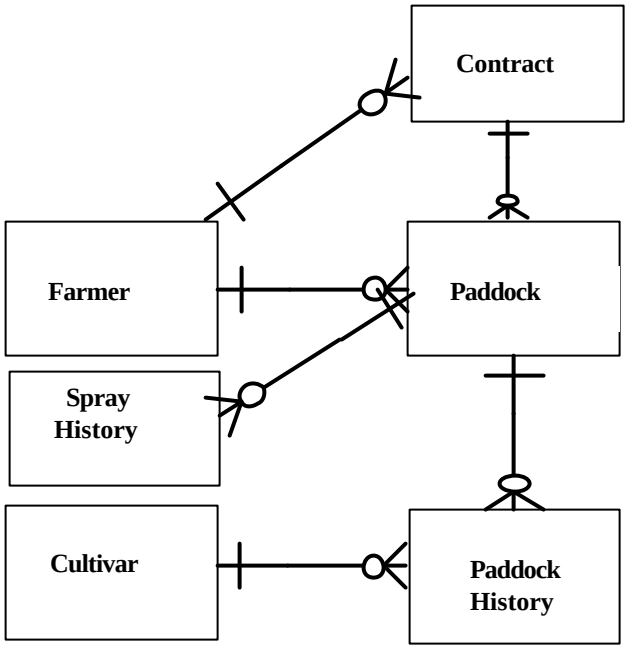

Figure 2 Example of and entity-relationship diagram. The boxes are entities. The linking symbols indicate a 1 to 0 or more relationship

The entities are composed of attributes, (fields or rows of data) for example, figure 3 shows a Microsoft Access table used for storing paddock data. This particular table shows data fields which have a spatial dimension (GPS location)

Figure 3. Table taken from a screen shot of an Access database used in an agricultural domain

The challenge in developing this kind of database is dividing up data into tables and secondly, identifying links between tables. The methodology used is called to facilitate this process is normalisation.

2.2 Normailsation

Normalisation is a process which produces a database with minimal redundancy suggested by Codd (1970). This follows a well defined set of steps. The first stage is to identify what needs to be stored in the database. If this already has some grouping involved, that grouping should be maintained in the first instance and given a unique identifier if one does not already exist.

As a starting point for a new system, a list of what needs to be stored, and what needs to be retrieved can be developed. Anything which is stored but not retrieved should be investigated as should any data which needs to be retrieved, but is not being stored. This also includes spatial data items.

The first step in the normalisation methodology is to remove any repeating data from the initial groupings (Kendall and Kendall, 1999), for example, if fields describing a grower and a paddock were grouped together, the paddock fields should be removed as a separate grouping. This is because a farmer may own many paddocks. The paddock data is therefore repeating.

A link between the two is maintained by storing the unique identifier (key) of the original entity in the extracted group of fields which become a new entity. In the example, the unique farmer identifier (farmer id) would be stored in the new paddock table. If the paddock identifier (paddock id) was not unique over the whole problem domain (something which could be caused by farmers using their own paddock identifiers), the key of the paddock table would consist of both the farmer id and paddock id. This structure is called a composite key. The farmer id field serves as the link or relation between the two tables.

Failure to do this step properly results in major inefficiencies in the database including, in this case, duplication of all the grower information. If this step is done correctly, most of the problems of relational database design are solved.

The next two steps are checks to make sure attributes are attached to the appropriate entities. There is also a check to make sure data which can be calculated is not stored. The final result is a set of relations which are in Third Normal Form. This can then be translated into an entity-relationship diagram such as shown in figure 2.

2.3 Problems with Normalisation

Normalisation is a well developed methodology, which if followed, produces a schema design which can be directly converted into tables in RDBMS such as Access and Oracle.

However it is very easy to make a mistake in the process and develop too many tables, or assign attributes to the wrong table. Users also find the process difficult to understand, particularly when tables are created who's only purpose is to avoid many to many links in the design. An example of this is as follows. A paddock can be used to grow many different crops, but a crop type can be grown in many different paddocks. To implement this a linking table is created as part of the normalisation process. Sometimes this has a purpose in its own right (in figure 2, for example, it is implemented as a history table), but in other cases it may consist only of linking fields.

There are methods where relational database design can be checked, for example in many agricultural systems the input requirements for recording grower, crop and paddock history details are very similar for a large number of applications. This lends itself to a highly structured approach to development based on a series of elementary structures as identified in a more general way by Kennedy (2000).

Figure 4. Example where paddock and grower details have been included in the one table, however other tables are linked to it via the site code

A more typical example is where normalisation has either not been carried out or has been done badly. Figure 4 shows an example that was initially developed as a flat file (a single table with no links to other tables) for a prototype system. The system then grew, but no attempt was made to redesign the tables. Other tables were later linked to this table via the site code attribute.

The result is that every time a new paddock is entered, all the grower details have to be re-entered, or, alternatively a routine written to locate a grower, copy the record, clear paddock fields, then allow paddock entry. A further complication in this example was poor analysis resulting in a site code with only a single digit restricting each farmer to 9 paddocks, or having a farmer with 2 records. As mentioned this site code was also used as a foreign key (relational link) to several other tables. This could not easily be rectified because the database management system software involved (Microsoft Access) will not allow modifications to such fields without removing all the data stored in them.

3. OBJECT METHODOLOGIES

Objects are something which most users are happy to work with. For example, in an agricultural domain such as we are discussing here, there are a number of objects which can be identified. Farmer, paddock and crop are all objects. In this methodology, rather than object, these would be called object classes, or simply classes. An individual farmer would then be regarded as an instance of that object class.

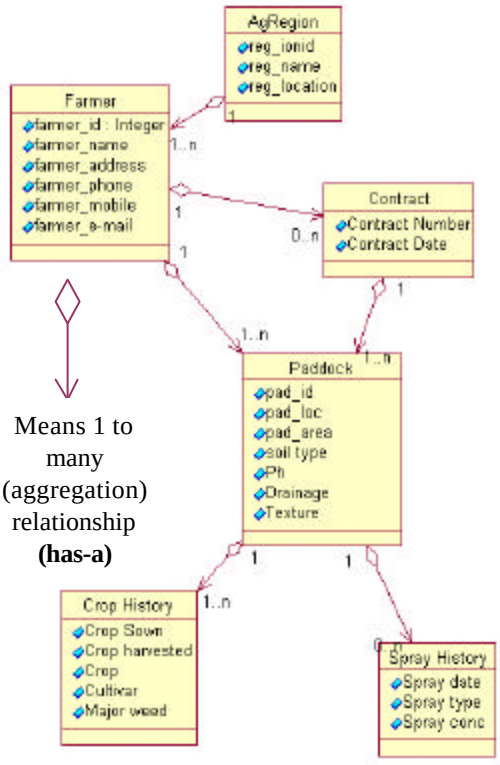

Figure 5. An object oriented class structure of an agricultural application

In pure object oriented methodologies, object classes not only contain data fields, they also contain procedures to process that data. The instances that are stored are called persistent objects (as distinct from transient or temporary objects) and form the basis of the database.

Unlike a relational database, the links in an object database are either of the form ***is a*** or the form ***has a***. The first case allows you to build hierarchies and allow inheritance. The second allows structures which show aggregation, or how one object is composed of other objects

In figure 5, the class diagram could be read as a region *has a* farmer (s), a farmer *has a* contract(s), a farmer has a paddock(s), a paddock *has a* crop history and a paddock *has a* spray history. Unfortunately there are currently few commercial database systems which can implement the ***has a*** structure directly. Rather this is still implemented as a one to many relationship using relational database techniques, that is storing a foreign key in each of the subclass. Hence the crop History class would also contain the paddock id field.

3.1 Object-Relational databases

Although Object Oriented Databases Management Systems (OODBMS) are rare, a hybrid, Object-Relational Database Management Systems (ORDBMS) are more common, for example Oracle 8, which can be used with Oracle Spatial for geographic applications. Although this does not allow aggregation (has a) structures, it does allow generalisation (is a) structures. This allows object classes lower in a hierarchy to inherit attributes from classes higher in the hierarchy.

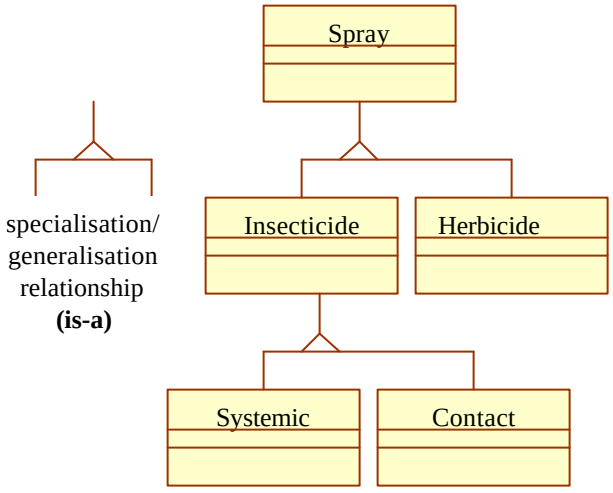

Figure 6. Generalisation/specialisation of class structure of spray

Hence in figure 6, the class *insecticide* would inherit all the attributes of the more general class *spray*, while all its attributes would be inherited by the more specialised class *systemic*.[1] The class *spray* would be linked to the structure in figure 4 with an aggregation link to *spray history* (spray history *has a* spray(s)).

3.2 Object Oriented Development Me thodologies

A problem with object oriented development is a lack of a standard methodology. Currently there are several, the most popular being Object Modelling Technique (OMT) (Rumbaugh,et al, 1991) which forms the basis of the Unified Modeling Language (UML) (Booch et al, 1999).

The method of developing an object class model is to first identify a series of Use Cases. These are a user view of the system and present a description of what the user wants to enter and retrieve. An analysis of the use cases will reveal a mixture of classes and instances. To move from instances to classes requires identifying instance with similar attributes (Bennett et al, 1999).

Since users tend to think in terms of objects, this is generally a straight forward process. The next step is to refine the objects definition, particularly in regard to attributes. In a database sense, the procedures or services are not important at this stage. Once classes have been identified, the associations and links between objects is established.

3.3 Advantages of Object Models

The major advantage of using objects as the basis of database design is the ease of using them for both design and user verification. Secondly the object model can be directly implemented in a relational form, particularly the aggregation or *has a* relationship. In the example used it should be noted that an extra object 'region' was identified from the use cases. This entity was missed in the original relational design because of the emphasis on attributes rather than objects.

Relational designs are not suitable for hierarchy's as shown in figure 6. In a relational system these would be shown as entities with one to many links extending down. That is a spray can have many insecticide sprays associated with it, and an insecticide spray can have many systemic sprays.

4. APPLICATIONS AND COMPARISONS

In the short term, most applications are going to be implemented in either relational or object-relational database management systems. Some of these can be directly linked to GIS such as Oracle 8 to ArcInfo, or via ODBC (Open DataBase Connectivity) drivers (for example Microsoft Access to a variety of GIS products). Likewise the access language is standardised around SQL (Structured Query language).

[1] A systemic spray kills insects by poisoning them when they eat the plant, while a contact spray kills on contact.

The main consideration, as already mentioned, is the design of the database schema in the first instance. The first point is that there must be some design methodology used. Failure to use any methodology will almost certainly lead to a poor structure with later problems with data entry and schema maintenance.

Given that a methodology must be used, the current choices are normalisation and object modeling technique. As already mentioned, normalisation is a well developed methodology which will produce a sound solution. It is however a technique which requires training and is difficult for users to understand. Some practitioners go further and regard normalisation "as an anachronism…" (Blaha and Premerlani, 1998, p 273). Object modeling technique on the other hand uses user information in the form of uses cases, which becomes the basis for the class structure. This makes the structure and the naming of objects in the structure easier for users to understand and hence verify. However, as already shown there are object structures which cannot be implemented in relational systems.

An example of a database developed using object methodologies was by March (1998). Although not specifically an agricultural system it was based around an object (or entity) called an asset. There were various types of asset, for example, land and facilities asset, plant and equipment asset and network asset, but all shared a number of attributes including an identifier, name, description and owner. The relationship *network asset is a asset* means *network asset* inherits all the attributes of the super class *asset*. If a paddock was substituted for an asset and there were different types of paddock, this could be implemented in the agricultural domain.

A further example is presented by Tennakoon and Bell, (1999) who developed a domain for crop rotation decisions. Although not a database, the information they used could be regarded as the basis of a use case analysis and could be easily converted into a object-oriented database design. A relational analysis of the same domain would not be so straight forward.

4. CONCLUSION

Although this paper has been very critical of the normalisation methodology, it should be stated that the ideal approach is build both an object model and a relational model, then compare the two. The results should be very similar, that is they serve as a check against each other.

The relational model summarised in figure 2 has been independently re-developed as an object-oriented model in figure 5. The structure is almost identical, but the development techniques used were different. There are differences, for example the extra class region, and the generalisation-specialisation of sprays shown in figure 6.

Differences between the models such as this must be examined, but it should be noted, differences do not always mean there is an error. Object models are capable of describing more complex structures than relational models. However most current database management systems are not capable of implementing all the class structures that can be modeled. Nevertheless, object models appear to be a more useful way of communicating analysis and design concepts to users.

5. ACKNOWLEDGEMENTS

The authors would like to acknowledge the contributions of David Sikk, Jiar Wei, Chih Ong and Robert Cox, third year students in the University of Tasmania's School of Computing who, under supervision, designed some of the database systems mentioned in this paper using the normalisation methodology. We would also like to acknowledge funding from the Horticultural Research and Development Corporation for the crop identification system of which one of the databases discussed in this paper was part.

REFERENCES

Barrett, R., Crowther, P., Laurence, R. And Lincolne, R. (2001), 'Remote Sensing as an Aid to Tasmanian Horticultural Crop Recording and Husbandry', *Proceedings of IGARSS 2001*

Bennett, S., McRobb, S and Farmer, R. (1999), *Object-Oriented Systems Analysis and Design*, McGraw-Hill.

Blaha, M. and Premerlani, W., (1998), *Object-Oriented Modelling and Design for Database Applications*, Prentice Hall.

Booch, G., Rumbaugh, J. and Jacobson, I. (1999), *The Unified Modelling Language User Guide*, Addison Wesley.

Codd, E.F., (1970), "A relational Model of Data for Large Shared Databanks", *Communications of the ACM*, Vol. 13, No. 6, pp 372-387.

Kendall, K.E. and Kendall, J. E., (1999), *Systems Analysis and Design (4th Edn.)*, Prentice Hall, pp. 609-611.

Kennedy, G., (2000), "Elementary Structures in Entity-relationship Diagrams as a Diagnostic tool in Data Modeling and a Basis for Effort Estimation", *The University of Otago Information Science Discussion Paper Series*, No. 2000/18.

March, B. R. (1998) “Spatially Oriented Strategic Asset Management Information System” *Proceedings of the International Conference on Modeling Geographical and Environmental Systems with Geographic Information Systems*, Hong Kong, Lai, P., Leung, Y. and Shi, W. (eds.) Vol. 1, pp. 396-403.

Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F. and Lorensen, W. (1991), *Object-Oriented Modeling and Design*, Prentice Hall

Tennakoon, S.B. and Bell, C.J. (1999) “Comparison of a knowledge-based system and a linear programming model for choosing crop sequences in dryland agriculture”, *The Application of Artificial Intelligence, Optimisation and Baysean Methods in Agriculture*, Abbass, H. A. and Towsey, M. (eds.), QUT Press, pp. 31 – 44.