Database Application for Changing Data Models in Environmental Engineering

Dr. Ulrich Hussels, Stephanos Camarinopoulos, Torsten Lüdtke, Dr. Georgios Pampoukis
RISA Sicherheitsanalysen GmbH, risa@risa.de

Abstract

Whenever a technical task is to be solved with the help of a database application and uncertainties regarding the structure, scope or level of detail of the data model exist (either currently or in the future) the use of a generic database application can reduce considerably the cost of implementation and maintenance.

Simultaneously the approach described in this contribution permits the operation with different views on the data and even finding and defining new views which had not been cosidered before.

The prerequisite for this is that the preliminary information (structure as well as data) stored into the generic application matches the intended use. In this case, parts of the generic model developed with the generic approach can be reused and according efforts for a major rebuild can be saved. This significantly reduces the development time. At the same time flexibility is achieved concerning the enviromental data model, which is not given in the context of conventional developments.

1 Introduction

As a Software company we are frequently faced with the challenge to develop a database application for an enviromental task (a technical information system) where the data model is not (yet) specified in detail. There are many reasons for the lack of a detailed data model. The fact is that during and after the development of a database constantly developing new insights arise that have an impact on the data model and would thus change an existing detailed specification. Particularly for new tasks, these findings often result from the use of the database application itself. In

many cases, it is almost impossible to define an exaustive specification of a data model without ever having worked with the data before. This is especially true if one want to avoid that "data tombs" arise instead of slim database applications. The approach to collect as much data as they could possibly be needed at some point, is not efficient. Such an approach also does not lead to transparent and efficient data models.

A further requirement frequently arising is to allow different views of the same data. A rigid approach does not offer this option.

The solution in these cases can be a parameterized, i.e. generic database application which is based on a common relational database management system (RDBMS). Such a generic database application can work with real data from day one, but it remains changeable.

Crucial for the efficiency is to parameterize the right attributes of the database application, and especially the right characteristics of the data model. Simultaneously however, the number of parameters must be kept as low as possible, because fewer parameters mean more transparency, performance and serviceability. To minimise the number of parameters, it is essential to have experience in the field of generic database applications and in-depth expertise in the task to be solved (e.g. [Lüdtke, 2012]).

2 Data Model

Since we have developed generic database applications and corresponding interfaces ([Hussels, 2006] , [Becker, 2002] , [Hussels, 2001]) for the last two decades, we know very well which parameters are useful and which are mainly useless ballast. Although theoretical reflections about the meta-model accommodating the application's parameters are beneficial, these theoretical thoughts however are without experience not efficiently applicable. For example, this relates to the modelling of time in the context of an information system. In order to map the time efficiently, considerable simplifications have to be made. Definition of simplifications meaningful for environmental issues requires experience and knowledge of the technical issues. This preliminary information is to be converted in a

parametrizable structure. In our opinion, the approach described subsequently represents the optimum for environmental databases.

Below, the following terms are distinguished:

- the underlying environmental data model,
- the meta-data model,
- and the overall data model of the generic database application.

The environmental data model represents the environmental context of the data, and it is configurable. In the meta-data model, all parameters required for the interpretation of the environmental data model are stored. The rest of the data model contains additional information, such as user management and role management.

2.1 Environmental data model

The basis for the modeling of the Envirometal data model in this approach is called the observation unit. The term “observation unit” is defined as a group of individual items which can be described by a common set of properties. In the database the observation unit is thus normally represented by a table. A typical observation unit in environmental context is a measuring point. The definition of the observation units are generally based on technical criteria, and is therefore not unique.

Many observation units have properties which can not be stored in a single table as the values of these properties stand in a 1: n relation to the individual. Already if some of the properties must be managed historically, there are (consecutively) different occurrences of a property. Here, too, groups of features can be formed in a common 1: n relationship to the individual. An example of this is historical master data management to a measuring point. In the historiography of master data mapping the time in the generic model is already playing a role. Each log record is stored with a validity start date and an expiration date.

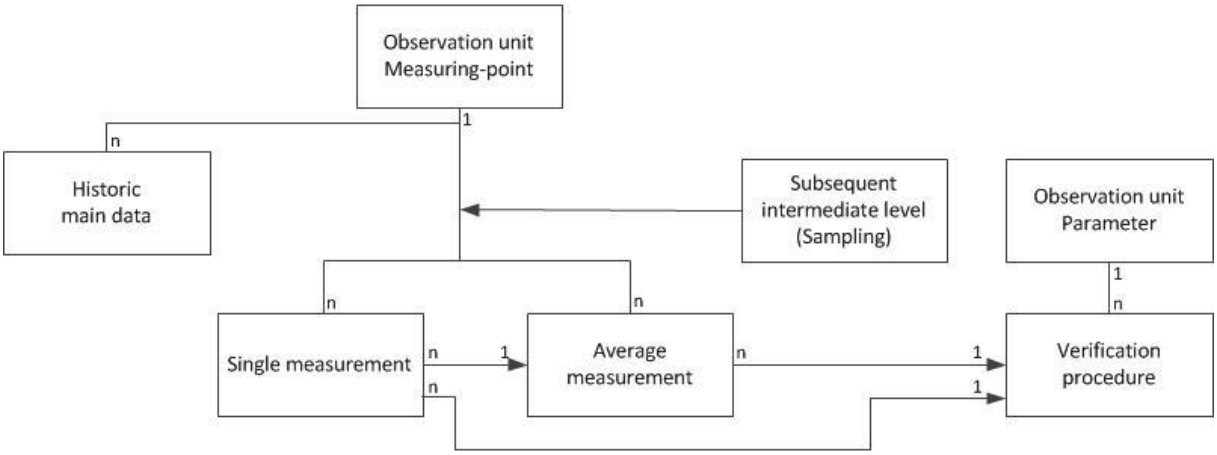

Figure 1: Generic data object measuring point with relation to the parameters

Groups of properties whose values stand in a common 1:n relationship to an individual of an observation unit are referred to as observation sub-units. Historically managed master data of an observation unit thus constitute already an observation sub-unit. An observation unit may have any number of sub-units, for example different sets of master data with their own histories. Another type of observation sub-units of a measuring point is the measured value. Initially one tends to forget that between the measurement point and the measured value sampling should be considered. With the presented approach this level will be introduced later if necessary.

If the concept for the observation sub-unit is pursued systematically, also observation sub-units may in turn have observation sub-units of a lower level. When considering sampling, these sub-units are the individual measurements and the average measured values for the sample. We call the hierarchical structure of an observation unit and its sub-units a technical data object.

The properties of observation units have at most one value for each individual, which is represented by a record in the table. Each value has a data type. In addition to the standard data types of the popular RDBMS the generic application provides the data type "pointer". For a "pointer" the value consists of a reference to any (other) record of an observation unit or sub-unit. E.g. for measurements, pointers to parameters (type of measurement) and pointers to units (of measurement) are stored into the measurement records. Parameters and units are observation units of their own. Only

the numeric value is a "real" date stored physically in the observation unit “measurement”.

Thus, the essential modeling rules become obvious. All managed data needs to be given as values of observation units or sub-units. Relations can be stored via pointers within and beyond the borders of multiple technical data objects. Thesauri are implemented using pointers to observation units.

To reflect the different substantive meaning of observation units, these are divided into groups. Typically, such groups are technical (environmental) data, catalogs and selection lists, in special cases also legal requirements.

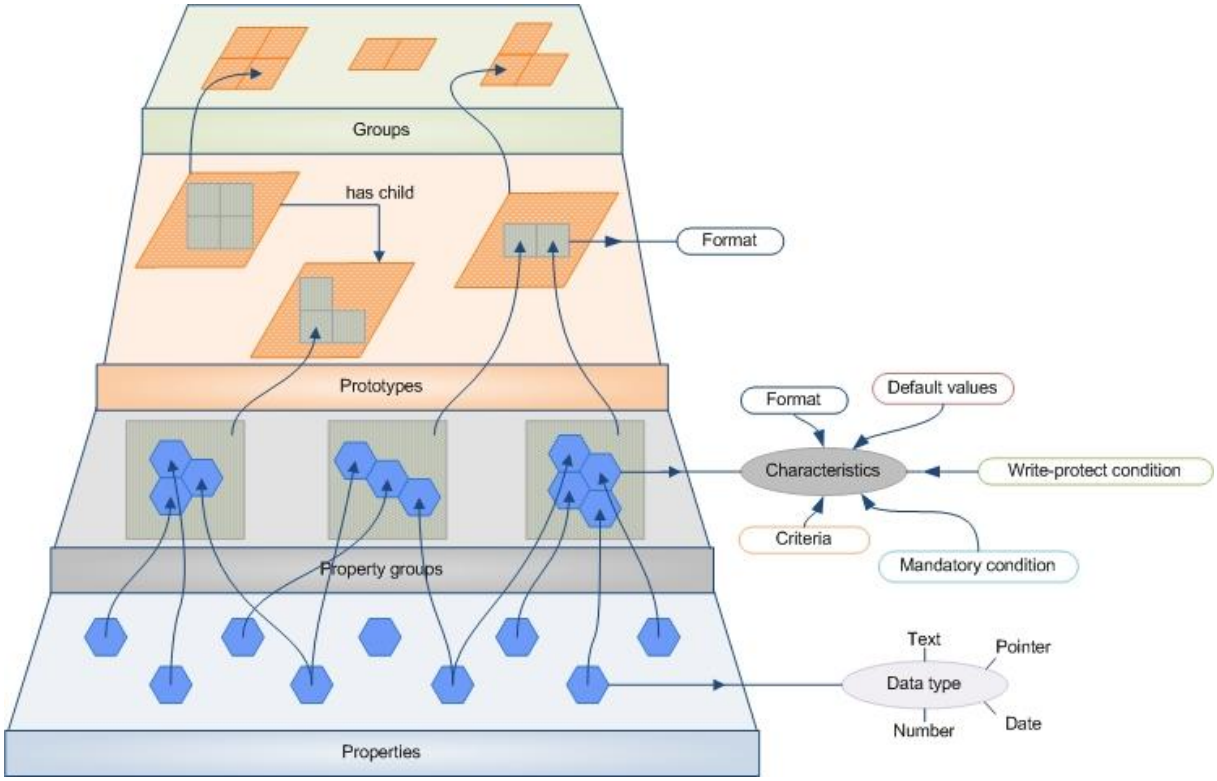

Figure 2: Basic structure of a generic application

This type of modeling results in database tables, which represent the environmental data model in a completely transparent way. The table names correspond to the (sub-) units considered. Apart from the pointers all table contents are directly readable by humans.

These properties of the tables facilitate both the subsequent change of the technical data model as well as data migration. Even structural modifications (such as later insertion of a observation sub-unit in an existing technical data object) is possible without much effort. If for example it is later found that certain master data has to be

managed historically, this observation sub-unit will be inserted into the technical data object. If necessary, the hierarchy immediately below the observation sub-unit considered needs to be "reassigned". Outside this part of the data model, this change has no effect. Data migration is trivial: For each existing record of the item under consideration a historically managed master data set with date of entry "open" and expiration date "open" is created and appended to the existing record. Subsequently, the data are transferred to the observation sub-unit and such properties (columns) of the observation unit which are not required any more, are deleted. Finally, the existing data records of observation sub-units, which are now located below the historically managed master data from the previous record of the observation unit in the new master record are "reassigned" to point to the new master record. This process can be completely automated.

The initally classification of the environmental data model into multiple data objects represents only one possible view on the data. Basically, this classification is not unique. For other problems, other classifications can be made. This is done by “dissolving” the technical data objects into individual observation units and then another observation unit is chosen as the starting point for the formation of technical data objects. By reverse tracking of the pointer the observation sub-units of new technical data objects can be determined. A prerequisite that an observation unit can be a observation sub-unit of another observation unit is that all records of the potential observation sub-unit under consideration have a pointer to the parent observation unit. An interesting structure arises, for example, if instead of the measuring point the parameter is selected as the starting point for the formation of a technical data object.

In that this process can be automated, it is possible to have multiple technical data objects searched through a program and to generate views which have not been previously considered. This is a type of query that conventional databases can not offer and which can bring insights that go beyond a standard database query. Nevertheless conventional queries are also possible with this model .

2.2 Meta-data model

In the meta-data model information on the structure of the technical data model from data objects, observation units and properties as well as their data types is stored. It

also contains information on the interpretation of times. For this purpose properties that represent absolute time points may be identified as the beginning or end of the validity of a data set. Furthermore, absolute time points may also reperesent an event time for the record. An observation unit can only contain a maximum of one of the two specifications. Based on this meta-information management histories (e.g. historically managed master data) and time series (such as measurements) are implemented generically.

2.3 Total Data Model

Besides the actual data model and the here described meta-data model tables are required, which contain parameters for the database application. These are tables for the status, users, roles and rights management, as well as tables for the management of the layout and the terms of the user interface.

3 Conditions, status and roles and rights management

The concept is that the same data set can be viewed in different conditions. This means that it will be included in the database multiple times in different conditions, and in each case it may have other values. This property can be used for different problems. The simplest is the version control system, to be able to fall back on older work states. In order to use the condition administration in a most general way, the possible conditions and condition transitions are stored in a directed condition graph. This may reflect, for example, different conditions of a system or environmental conditions (e.g. summer / winter) or a process chain. After Login it is decided which condition is assumed. If the condition graph forms a process chain, then the condition is the station of the process.

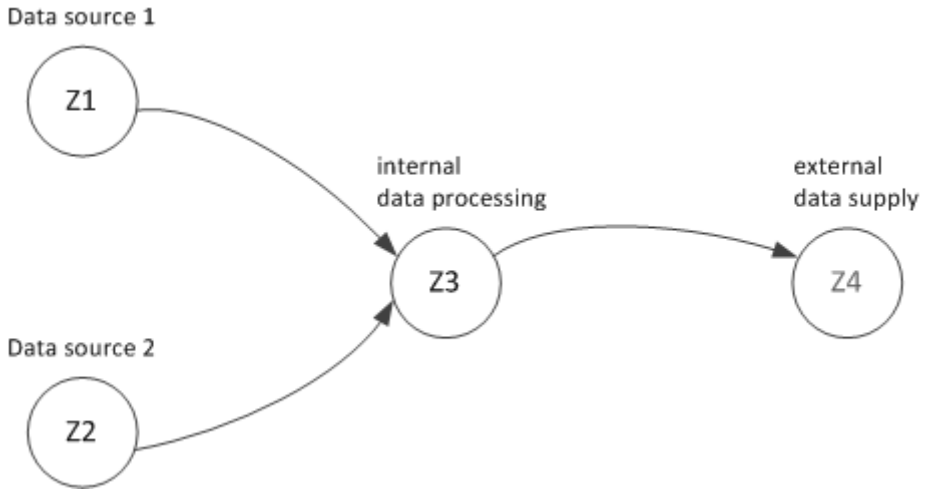

Figure 3: Condition graph for an external data provision

Furthermore, the concept envisages that each record has (in any condition) a status. The list of possible status can be expanded. Fixed intended status are "Active" and "Deleted", as deleted items often can not be reused and must remain in the database. The erased information can be indicated by the status information via interfaces and also exchanged between different installations.

The roles and rights management is also solved generically. Any number of roles can be assigned. Users are assigned to these roles.

On the other hand, regardless of that rights are defined. The rights may have a parameter. The parameter allows to define specific write permissions to properties, acording to the value of the parameter at the given moment. The following rights are available:

- identify numbers (number of hits of a query)
- summation allowed (sum of the properties for a query)
- Read (single); When the read permission on a table by a record-dependent condition is defined, the user will see only the records that satisfy the condition. Is there a permission to read the table, but none of the attributes, blank records are displayed. So the existence of the records is well known.
- Create (complete data, where previously there were none; with property-related rights the parameter is the value that can be generated)
- Replace (overwrite data; with property-related rights the parameters is the allowable replacement value)

- Delete (sets of data or on individual properties override with NULL)
- Queries (all records)
- Export (related to specific functions and data formats)

The assignment of rights may relate to the structural elements "observation unit" (table) and "Property" (Column) and takes the form:

"If <Condition satisfied> then <right> to < structural element >"

The condition is a logical combination of comparisons, which can include not only the role of the user or the user themselves, but also properties. Thus, it is possible to define record- and impaired related rights. For example it is possible together with the parametric rights to allow certain roles only for certain state transitions.

4 Database Application COODEXX

We call the database application that implements the scheme described above, COODEXX (Configurable Object Oriented Database Engine with indexer and XML Interface).

The aim of this parameterized, i.e. generic database application, is the ability to work with real data from the beginning of the development of a specific application. That’s why the generic database application has the essential functions of many database applications:

- data explorer,
- data forms,
- query engine
- Import and export interfaces (xml and table-oriented)
- printing and reporting features,
- user management,
- roles and rights management,
- process management.

The layout of the user interface is determined by the model and additional technical data layout parameters. There is a setting which allows, without additional information to map each data model. At the same time it is important to use the terminology of the field instead of concepts from computer science. The texts of the user interface are therefore not fixed, but adjustable. Thus, the application can also be used in several languages, and this applies not only to the user, but also for the data. The application also supports the parallel management of data (limited to the text data) in several languages, but there must be a base language. Although the generic database application already covers a large part of the required functionality, in most cases there remain functions that need to be added individually. Through the existing framework of a functioning and filled with data application, it is much easier to specify and implement those functions, than without such an environment.

The database application can be used both as a client-server application and a web application. Individual units can be viewed on mobile devices, for example, for data acquisition or data editing.

5 Limits

The solution of an environmental task using a generic database application can only be successful if the environmental data model can be modeled effectively by means of the specific generic approach. E.g. the modeling of the time must suffice for the environmental task. An extension of the generic approach to implement specifics of an environmental data model is usually more complex than a specific solution.

6 Literaturverzeichnis

[Lüdtke, 2012] Lüdtke, Torsten: Effektive Softwarelösungen für Umwelt- und Anlagendaten an Chemiestandorten. Jahrestagung der NORDOSTCHEMIE, Rheinsberg, 11. Mai 2012

[Hussels, 2006] Hussels, Ulrich: AGXIS – Ein Konzept für eine generische Schnittstellenbeschreibung. Workshop des Arbeitskreises „Umweltdatenbanken“ 6.–7. Juni 2005 in Hannover,Umweltdatenbanken und Netzwerke, Texte 11/06; Dessau, März 2006

[Becker, 2002] Becker, Dierk; Hussels, Ulrich; Nagel, Janet: Vergleich zweier theoretischer Ansätze zur Realisierung generischer Datenbankentwicklungswerkzeuge. In: Simulation in Umwelt- und Geowissenschaften: Workshop Cottbus 2002, Aachen: Shaker Verlag, 2002, S. 137–147, ISBN 3-8322-0733-3

[Hussels, 2001] Hussels, Ulrich; Nagel, Janet: Vorstellung eines generischen Ansatzes zur Erstellung und Pflege von Umweltdatenbanken. In: Wittmann, J.; Bernard, L. (Hrsg.) : Simulation in Umwelt- und Geowissenschaften, Workshop Münster 2001. Aachen, Shaker Verlag, 2001, S. 173–178, ISBN 3-8265-9251-4