INCREMENTAL DATA EVOLUTION - COMPLEMENTARY APPROACH TO SCHEMA EVOLUTION OF ORGANISATION'S LARGE OPERATIONAL GIS DATABASES

L. Salo-Merta

Institute of Business Information Management, Tampere University of Technology, P.O.Box 541, FIN-33101 Tampere, Finland – leena.salo-merta@tut.fi

Commission IV, WG IV/4

KEY WORDS: Application, Database, Design, Development, GIS, Management, Software, Spatial

ABSTRACT:

The paper addresses database development and schema evolution from the geographic information provider organisation's and system integrator's point of view. The applicability of incremental approach to database schema evolution is considered, and incremental data evolution (IDE) is introduced and proposed as a complementary method for schema evolution and data restructuring of large geospatial datasets in operational use. The hypothesis is that IDE combined with full data conversion strategy would provide a flexible approach in the schema evolution of large operational databases. The applicability of the IDE approach is considered with a real-world case.

1. INTRODUCTION

1.1 Development pressures in GI provider's evolving digital environment

According to European Union, geographic information (GI) is by far the largest type of public sector information in Europe, comprising 38,5 per cent of all public sector information (European Union, 2001). The computer penetration into the field of GI follows a general pattern: First it is adopted in the GI provider organisation for speeding up the production process. At this stage there are no significant changes in the end products. In the second phase the potential of computerized environment is exploited more widely and the market for GI in digital form starts to develop. In the third phase completely new products and activities arise from the digital culture.

The pioneer organisations that started to digitise GI early, have faced all the stages one-by-one. However, the stages are concurrent and ongoing, and are shown as layers of activities in Fig. 1. In the time being the 'geo–e-business' will be absorbed into the 'normal' service production layer, but at the moment it is feasible to examine it as a layer of its own. The project portfolios of organisations tend to grow into new topics, and the most appealing questions from the manager's or technologically oriented researcher's point-of-view may lie in the expansion of technology and new applications, rather than re-engineering the existing parts.

However, databases (DB) and data management are the true cornerstones of the whole and they take effect on each activity layer, directly or at least indirectly. Commercial solutions for geographic data management (GDM) have developed rapidly during the past few years, showing the evolution of architecture from rather closed monolithic systems towards client-server architectures with open, standardized interfaces. GI provider organisations have made great investments to establish their existing large datasets in digital form. Their concern for their asset is three-fold: Firstly, how to maximize the exploitation of the digital content. This is the driving factor for expanding into new dissemination technologies, like the WWW and mobile, and new types of products and services. These actions are very apparent and promoted. Secondly, the methods and techniques in the data production layer need improving for more economic and efficient data updating. To address these questions, incremental updating of databases has recently become an issue of research and development. Last but not least, user organisations are concerned as to how to keep up with the evolving data management technology and ensure the sustainable platforms for their information asset. Because GI datasets are typically *data-intense* both in size and complex structure, the solutions and new arrangements in data management have significant impacts on all actions. For user organisations of GIS, database re-design and schema evolution are challenging tasks in the development of their next-generation systems. They are more 'back-end' –processes, partly invisible and less promoted, but nevertheless significant and anything but trivial.

1.2 Databases in next-generation system projects

Next-generation GIS projects are current in many organisations. By next-generation projects we mean substantial development projects in an environment that already comprises a previous-generation GIS and digital datasets. Next-generation GIS projects usually include a database project, where data is translated from the existing data storage into a new database management system (DBMS). DBMS is typically a Commercial-Off-The-Shelf (COTS) component in an integrated system. A DBMS with spatial capabilities has been strongly associated to certain vendor specific client solutions and product families, but the situation is changing rapidly. Currently one has alternative clients, components and development tools to choose from. It is also possible to implement the 'business

intelligence', or should we say here: the 'geospatial intelligence' in different layers of the software architecture: the database server, the middleware or the client.

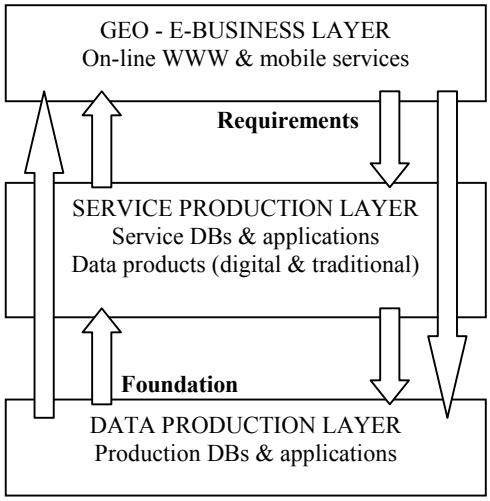

Figure 1. Activity layers of a GI provider organisation

In a next-generation GIS project, the user organisation needs a practical solution for database management, and it needs it now. Although the latest technology always seems to be three months ahead, one just has to try to make the best out of what is available here and now (Kucera et al., 1999). According to our experience a practical solution can be defined with three keywords: Economy, Maintainability and Technical requirements. *Economy* stands for both investment cost and operational costs. *Maintainability* stands for the wish to choose a provider and platform with a good probability to survive and be competitive and technologically advanced in the future. *Technical requirements* include general and specific database capabilities, and issues concerning architecture, interfaces, scalability, extendibility and programming facilities. All of these need to be examined in the context of the user organisation, to determine the 'customer need'. As the DBMS is a central component of the system, major development actions invoke needs to develop the other subsystems as well. In the most dramatic case it means the replacement of existing clients with new ones.

1.3 Content of this study

The focus of this paper is on primary schema evolution, as it is seen from the system integrator's point of view. By primary schema evolution we mean structural changes to user's data tables. The fundamental questions are: Is primary schema evolution possible with existing large geospatial datasets? Could it be carried out without significant disturbance for production? Can we allow pluralism in schema definitions and storage structures as a permanent, or at least long-lasting state, and how do we support pluralism?

First we introduce taxonomy of DB development activities and consider the factors that influence these activities in organisations. Then we consider the applicability of incremental approach to DB schema evolution. A method called *incremental data evolution* (IDE) is introduced. The hypothesis is that IDE combined with full data conversion strategy would be a flexible approach in the schema evolution of large production databases. The IDE approach is considered in a case study, where the possibilities of interrupting the production use of the DB for data conversions or any other reason are limited. The principle of the needed DB interface is introduced.

In conclusions the advantages and disadvantages of IDE approach are considered. Finally the question is raised whether we actually need primary schema evolution and for what reason? And if we don't, for how long are we able to survive?

2. TAXONOMY OF DATABASE DEVELOPMENT ACTIVITIES

Based on experience, we find it feasible to study the development of a system's DBMS environment as three basic types of activities: *Database product change*, *Version migration* and *Schema evolution* (Fig. 2).

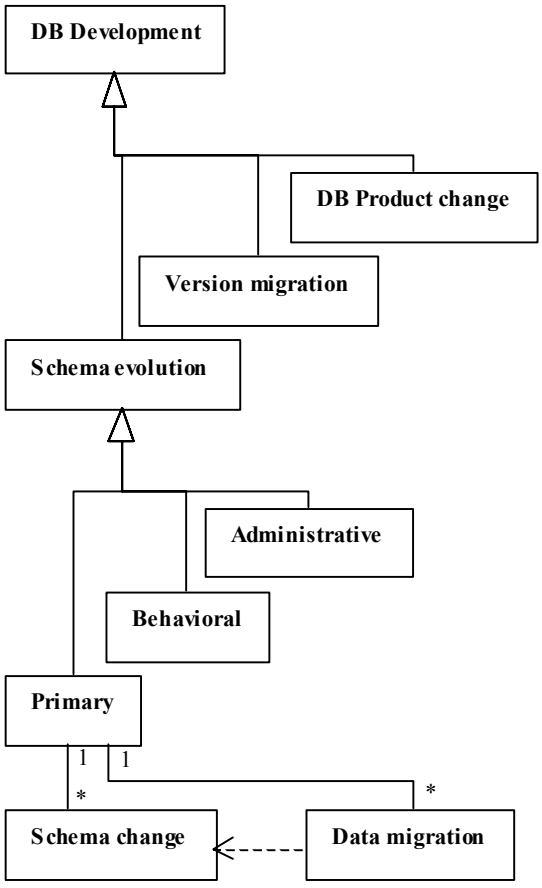

Figure 2. Taxonomy of database development activities

The database product change projects include:

- Design and implementation of the new DB schema
- Preparation of data translation
- Data translation
- Data validation
- Further data re-structuring (if included at this stage)

It is noteworthy that because of the history of proprietary DBMS solutions for geospatial data, considerable effort may be required for other software activities in database product change

projects. It may be necessary to change the client applications as well or to re-write the interfaces to the existing ones. For this reason the activities concerning the database itself are probably only a part of a large software project, and the pressure to minimize the size of the project by altering the data as little as possible at this stage may be substantial.

Version migration means upgrading the system to a higher version of the same DBMS product. The changes are product driven: It may include internal changes in data storage, indexing, partitioning etc. On the logical level there may be changes in system tables, e.g., metadata table structures. With new DBMS functionality and capabilities version migration may inspire the customer to further development, but as is, it includes no customer-driven changes in the logical structure of the database. DBMS vendors usually provide migration tools with their products to facilitate the shift into the latest technology and guarantee the evolution in this sense.

Schema evolution means altering the structure of the database, here especially the customer-driven changes. Basically it means changes in table definitions, but as other types of objects (e.g., views, triggers, DB functions and procedures) are stored in the DB as well, it needs to be concerned in a wider sense. However, the consequences of committing a behavioural change (e.g., inserting a new function into a package of DB functions) are quite different from the attempts to commit structural changes in stored data. Purely administrative tasks dealing with physical storage and performance (e.g. table space definitions and indexes) can be considered as another group of tasks. Therefore three subtypes of schema evolution activities are identified: *Administrative changes*, *Behavioural changes* and *Primary changes*. Primary changes concern structural changes to user's data tables. If the data tables are populated, primary changes mean two things: *Schema change* and *Data migration*. We use primary database evolution in a broad sense to cover all actions that are needed to change the schema and propagate the changes to the data, regardless of the existence and capabilities of DBMS's schema evolution facilities. If the DBMS's schema evolution facilities do not support the actions needed, they have to be carried out by other means, e.g. by creating new or temporary tables and implementing ad-hoc migration modules.

3. FACTORS OF DATABASE DEVELOPMENT ACTIVITIES

The factors that influence database development activities are discussed below. They are: Data conversion cost, Evolution strategy, Developing DBs in operational use and Pluralism in DBs.

3.1 Data conversion cost

Converting the existing data into the new system as quickly and painlessly as possible may be a key question in an organisation's DBMS implementation project. Therefore, commitment to the old design may become a critical factor in the design of the application data model. (Salo-Merta et al., 2001). For a very large database with objects with complex structure, which geographic objects appear to be, each extra step included in the conversion process may have a dramatic effect in the throughput time, if it causes a need for operator intervention. Data quality may become the bottleneck in the conversion process, if the target system's validation rules are tighter than the source system's. Relatively small numbers of exceptions that need to be handled manually may turn into vast number of working hours, if we consider millions of objects.

3.2 Step-by-step evolution

All the user requirements can seldom be satisfied at once, and they keep on changing. Yet the data acquisition for digital geo-spatial datasets is expensive and time consuming, there is a strong motivation to preserve the existing data and pass it on to the next system generation. We claim that the lifetime of a dataset can exceed the lifetime of the DBMS that is used to maintain it (Salo-Merta et al., 2001). Therefore the development of the data system's database environment gets a step-by-step evolutionary nature.

In a step-by-step evolution strategy one proceeds with a series of development steps to achieve the desired state. For example, in the Next-Generation System Project of the Topographic Data System, The National Land Survey of Finland used this approach to restructure the data. The primary data loading into the new database included basic validation of attributes and topological conditions, and polygon reconstruction. Further data restructuring was carried out as a later stage. In this case it contained the merging of objects that had previously been fragmented by map sheet's edges. (Salo-Merta et al., 2001).

3.3 Developing databases in operational use

Perhaps the migration to a new DBMS is made with minimum changes to the structure of the data, with the intention to evolve the schema later in the future. However, the same principal problem, - how and when to propagate the structural changes to the data, - arises again if we need database schema evolution. For very-large production databases this may be a real life constraint in the system development and design, because the operational use of the database should not stop during a lengthy conversion phase. It may not be possible to stop the production for a full conversion. Instead, other alternatives have to be considered.

3.4 Pluralism in databases

Kucera (1999) discussed pluralism in geospatial databases and suggested the acceptance of data heterogeneity in contrast to a monolithic database, which reconciles any pluralism as part of the update process. In a pluralistic system a geographic feature may have several different representations stored 'as-is', as they existed in provided datasets, without attempts to integrate schemas, generalise, resolve spatial discontinuity, or otherwise encourage consistency in representation. Pluralistic data management and representation require specific techniques: e.g. versioning, metacontent description, feature linking with same-as links, on-line schema mapping etc.

4. INCREMENTAL DATA EVOLUTION APPROACH

4.1 Origin of incremental approach

The incremental approach originates from the software industry, where it has been applied for incremental development and compiling of software modules. In the context of GIS, it has previously been proposed for generalisation (Kilpeläinen & Sarjakoski, 1995).

Incremental updating of databases has recently become an issue of research and development. The International Cartographic

Association established a working group on Incremental Updating and Versioning in 1999. The working group's main interest relies on the management of updates between the base dataset supplier and the value-added dataset provider. Research issues include: bi-directional, multi-level, historical and temporal updating, planning for future changes, database maintenance, feature identifiers, modularity (dimension, context, layer, theme and size), inconsistent updating and simultaneous updating by field teams (Cooper & Peled, 2001). The implementation of new features to support incremental updating and versioning are likely to put pressure on product change or schema evolution on existing systems as well.

4.2 Production discontinuity problem

Traditionally database schema evolution is carried out by *full data conversion strategy* using the following sequence:

- Stop production
- Change the schema / Create a new schema
- Migrate all data to the new structure defined in the changed schema
- Continue production with the new schema

Here we assume that schema evolution includes such structural changes to the data, that a specific software module is required for migration (Fig. 3). This migration module may be a SQL-script, a database program, an application program or a translation process with external transfer files. We do not consider the implementation technology here. The characteristic of the migration process is the discontinuity that it causes to normal production. Normal production is carried out with software A using DB interface a. Data that conforms the new schema, has to be accessed with an updated interface a'. Therefore two versions of the DB interface are needed.

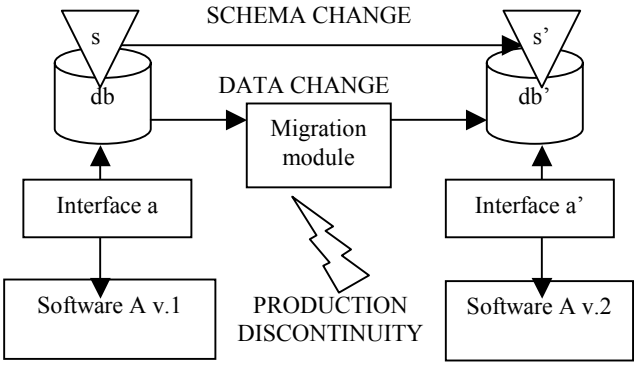

Figure 3. Migration module propagates schema changes to data. Migration causes discontinuity in production, because application program's DB interface needs updating to conform the new schema.

This approach can be described as one very-long transaction. From the production applications' point of view, the database is in inconsistent state during the migration. Since the migration starts, the whole database is not accessible for client A 1 anymore, but on the other hand, only the migrated objects are accessible to client A 2.

If the basic problem is the duration of data change and we assume that it cannot be solved reasonably, we should consider ways of managing the production and the migration simultaneously. Could it be possible to use versions 1 and 2 of client A and run the migration process in combination? In theory this could be possible in certain conditions. We need to be able to partition the data. That means recognizing and isolating meaningful and appropriate subsets of data for processing. In some data systems feasible partitioning criteria can be found, e.g. in working area based map production, but that is not always the case.

4.3 Principle of incremental data evolution

It may be a characteristic to a data system that the updates are made object-based and on-request. The requests scatter all around the dataset and no partitioning criteria for isolation can be found. In that case we suggest the acceptance of pluralism in schema definitions of object classes and consider means to manage it. We introduce an approach called *incremental data evolution* (IDE) with the following definition: Incremental data evolution is an approach where only those objects that are touched as part of the normal production process are migrated to the new structure defined in the changed schema.

We accept the fact that a feature may have alternative schema definitions as a permanent or at least long lasting state of the DB. To manage this kind of pluralism, we suggest certain changes in the DB's read/write –interface and embedding the migration module in the updating client (Fig. 4).

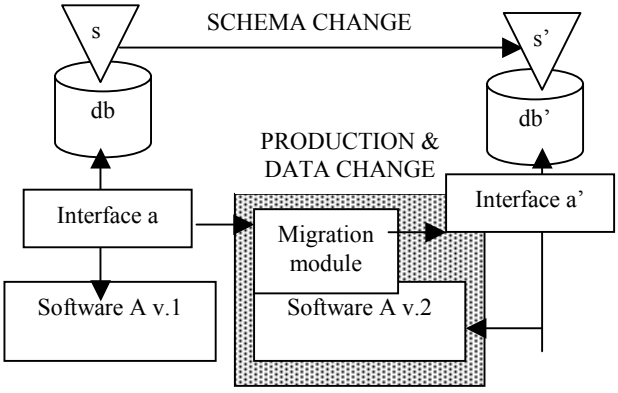

Figure 4. Migration module embedded in the production software module.

We call this approach incremental object evolution, because the schema evolution is propagated to the data incrementally, as the objects are touched in the normal production. It is not forced to the whole dataset, but only to those objects that are most actual.

4.4 Implementation aspects

We try to encapsulate the read/write –interface between the DB and applications to hide the pluralism and to encourage the data conversion from the old structure to the new. The database interface needs to be modified so that 'read' operations retrieve objects primarily from the new structure, and secondarily, for the objects that were not found, from the old structure (Fig. 5). Depending on the application case, the result sets for read operation may be separate or merged. 'Write' operations always save the objects only into the new structure, freeing the old

structure. Therefore every processed object will be converted from the old structure into the new structure. If there are any problems in the merging, conversion or validation, they will be solved by the operator during the normal production work.

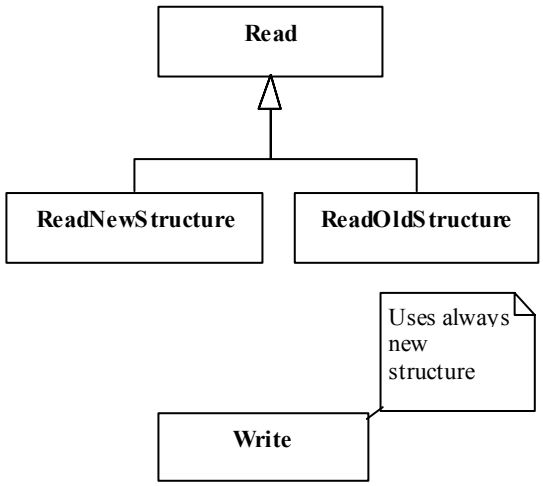

Figure 5. Principle of DB read/write interface implementation

5. CASE STUDY

The suitability of the incremental object evolution approach is considered on the Finnish Land Parcel Identification System (FLPIS). FLPIS is a national database for controlling the farmers' parcel-area-based subsidies that come from the European Commission. FLPIS is a subsystem of the Ministry of Agriculture and Forestry's data system for managing the farmer's subsidies. All activities on FLPIS have been contracted, currently to Genimap Ltd. The contractor is responsible for both the production work and the production environment, including system integration.

5.1 General description

Annually 20 000 farmers' data are updated by the contractor, according to requests from farmers and the controlling authorities. The production year is tightly time-scheduled with severe economic delay sanctions.

The system comprises the farmer information, orthophotos of the country and the parcel information including boundaries in vector form. The main functions of the system are to digitise the parcel boundaries, identify the parcels and calculate the parcel areas. The system preserves history of each project year's final states, which can be queried, viewed and compared to previous and later states.

The database is updated both based on the announcements from the farmers, and the requests from the controlling authorities. Both the farmers and the controlling authorities are served with massive document output, including maps.

The database can be divided into two parts: the core with the actual parcel data, and the extension for managing the production. There are client applications for:

- Data exchange (with Ministry's system)
- Document registration and management
- Digitising
- Validation
- Quality control (internal and external)
- Document output
- Reporting the production
- Extracting data on request

The database environment is Oracle Spatial 8.1.5. The client applications are implemented with a variety of tools including MapInfo Professional, MapBasic, Visual Basic, PL/SQL and Oracle Reports.

5.2 Motivation for change

The evolution of Ministry's data system for managing the farmer's subsidies follows the general pattern shown in Fig. 1. First the data production layer was established, with limited services: First paper documents only, then file extracts on request.

In 2000 the Ministry started the development of an intranet/extranet service pilot for the controlling authorities. The service is based on a service database. In the beginning it was a read-only –service, but new features for managing digital update requests with graphics from the authorities are developed and tested. The ultimate goal is to offer an up-to-date on-line service to the farmers as well, and reduce the need for paper documents.

The role of the production database has changed from a rather isolated system with only a few off-line interactions a year with the Ministry's systems, to a core dataset that needs to be replicated to the service database continuously. However, only those changes that have passed the two-phased quality control, are allowed to show. This generates new requirements to long transaction and version management. The major schema evolution challenges concern the modernisation and enhancement of the spatio-temporal capabilities of the system. More intelligence concerning the management of geographic objects would be implemented in database instead of the client application.

5.3 Applicability of incremental data evolution

The IDE approach is appealing to FLPIS for the following reasons:

- The project year is tightly scheduled. The scheduled breaks last only a couple of weeks, and causing longer breaks in production for conversions is not possible.
- The amount of historical data in the system is high, since the production started in 1996. The storage and management of historical data could remain as-is.
- IDE in combination with full conversion of subsets selected on area-basis would make an ideal processing method for FLPIS, because of it's flexibility.

On the implementation level, a new digitising client with embedded migration module and read/write DB interface would be needed. The data exchange module would need updating, but with streight-forward changes. The changes for document output client would be manageable.

6. CONCLUSIONS

This paper considers the applicability of incremental approach to database schema evolution, and proposes a complementary method for schema evolution and data re-structuring of large GI databases. It is proposed and considered as a resolution to a practical problem of schema evolution of a production database, in a situation where the possibilities of interrupting the production are limited. The options are either to arrange the data conversion concurrently with the ongoing production, or to forget about the schema evolution. The applicability of IDE approach is considered in a case study, but so far there is no implementation and therefore no proof of concept.

The advantage with incremental object evolution is that one does not have to run down the production line for the migration process. The schema evolution can be propagated to the data 'on-line' during normal production. The combination of IDE and full-conversion of selected subsets offers a flexible processing method.

There are risks in the IDE approach. If we consider a schema evolution step with data conversion as a very-long transaction, there is no reasonable 'rollback', if the conversion takes place embedded in the normal production. On the other hand, do we ever have a real rollback option in system change projects?

The database schema becomes more complicated as pluralism is accepted. But the pluralism does not have to be forever – the old structures may be deleted when there is no need to access them anymore. The pluralism within object definitions may be hidden by using DB views and encapsulation in the read/write interfaces, e.g. by database programming.

User requirements change in time, but so does the platform for database management. The advances in DBMS technology create a internal pressure for further development, as the changing user requirements create an external pressure. One has to migrate to new versions of DB products to keep one's system sustainable and watch out not to become obsolete. But do we actually need primary schema evolution that generates from technical advances? 'Why fix it if it ain't broken?'

The relational model is simple and provides a fairly low level of abstraction with tables, rows and columns. For geospatial data, a higher level of abstraction would be preferred, to make the investment cost of a new application reasonable, and to facilitate standards. Higher level data models for geometry and topology, that have been provided by proprietary GIS's, are coming into mainstream general DBMSs as well. Oracle has been developing the spatial concept step-by-step since the first release of Spatial Data Object (SDO) in 1994, including new features in each product version. However, the specific geospatial features provided with Oracle 9 Spatial are still far behind from some GIS vendor's DBMSs, e.g., there is no support for storing topology.

Let's assume an organisation that has implemented it's GIS on Oracle's BLOBs (Binary Large Objects) or first versions of spatial with the idea of getting a standard mainstream solution. However, Oracle's geospatial solution has not been adequate yet, and therefore plenty of proprietary features have been implemented in the database and the application programs by the system integrator, e.g. the management of topology and temporal dimension. As the DBMS's geospatial capabilities improve in the new product versions, they remain unexploited in the application, because the issues are handled already – with a proprietary, non-standard and probably complicated way. Under these circumstances, we may need to fix, although it wasn't actually broken, to survive and to gain the benefits of openness and standards. The tools for not only storing, but really managing geospatial data in the DBMS are in our hands, but do we want to take them to use?

Research and development on incremental updating and versioning of databases, generalisation and multiple representation, pluralism and new concepts that deal with spatio-temporality change our way of thinking. Technological advances are being made and new features become available, also in DB management. Therefore, we should raise a question: Do we actually need primary schema evolution? And if we don't, how are we going to survive and for how long? It depends on the case, but I should say that yes we do need primary schema evolution - if we can manage it. Managing the evolution is the challenge.

REFERENCES

Cooper, A., Peled, A., 2001. Incremental updating and versioning. In: *20th International Cartographic Conference, Conference Proceedings*, Beijing, China, Vol. 4, pp. 2804-2809.

European Union
http://www.cordis.lu/euroabstracts/en/june01/feature02.htm
(accessed 2 Feb. 2002)

Finnish Land Parcel Identification System FLPIS. Genimap Ltd., Ministry of Agriculture and Forestry in Finland. (Unpublished system documentation)

Kilpeläinen, T., Sarjakoski T., 1995. Incremental Generalization for Multiple Representations of Geographic Objects. In: Müller, J-C, Langrane J-P, Weibel R., (eds) *GIS and generalization*, Gisdata 1, ESF, Masser, I., Salgé, F. (series eds.), Taylor & Francis, pp. 209-218.

Kucera, G., 1999. Pluralism in spatial information systems. Paper on the IV International Conference on GeoComputation, USA, on 25-28 July 1999. http://www.geovista.psu.edu/sites/geocomp99/Gc99/060/gc_060.htm (accessed 19 Feb. 2002)

Kucera, H., Lalonde, B., Lafond, P. 1999. GEO-2001:Designing and Building a Spatial Information Warehouse. *19th International Cartographic Conference, Ottawa, Canada. (unpublished workshop material)*

Salo-Merta, L., Tella, A., Vanhamaa, I. 2001. Database Design in Migration from Traditional to Object-Oriented GIS – the Evolution Story of the Topographic Database of Finland. In: *20th International Cartographic Conference, Conference Proceedings*, Beijing, China, Vol. 2, pp. 1393-1400.

ACKNOWLEDGEMENTS

The author wishes to acknowledge Mr. Matti Vilander from the Ministry of Agriculture and Forestry and Mr. Juha Haapamäki from Genimap Ltd. for their support and co-operation, and Prof. Tuija Helokunnas from Tampere University of Technology for support and comments concerning this paper.