ICA Workshop on Generalisation and Multiple representation; 20-21 August 2004 - Leicester

GENERATING AND USING A MULTI-REPRESENTATION DATABASE (MRDB) FOR MOBILE APPLICATIONS

M. Hampe & M. Sester

Institute of Cartography and Geoinformatics, University of Hannover, Appelstraße 9a, 30167 Hannover, Germany –
(mark.hampe, monika.sester)@ikg.uni-hannover.de

1 INTRODUCTION

The European project GiMoDig aims at serving topographic data for mobile users, equipped with small mobile devices like PDA or Smartphones. These maps will be compiled in real-time meaning at the moment the user requests the data. It is now possible for the user to select the content he or she is interested in as well as the preferred scale; the user gets an adaptive map meeting his or her individual requirements.
The customised maps processed in real-time open up the advantage to, on the one hand, fulfil exactly the certain needs of the user and on the user hand to balance the disadvantages of the small display of the mobile devices. Not only the selection of data but also the generalisation of data adjusted to the certain device in use can be carried out.
In the project we develop methods to generalise spatial data in real-time. One approach to realise this challenge is to fall back on a set of pre-generalised spatial data stored in a multiple representation database (MRDB). In this paper we describe the generalisation methods to derive a number of scale-layers in an MRDB. Furthermore the MRDB structure is used to increase the possibilities to customise and use the mobile maps. The user can choose not only the level of detail (LoD) for the map but also the LoD for certain objects in the map. In the paper three applications are presented that allow for completely new possibilities.

2 MOTIVATION TO INTRODUCE AN MRDB

An MRDB can be described as a spatial database, which can be used to store the same real-world-phenomena at different levels of precision, accuracy and resolution (*Devogele* et al., 1996; *Weibel & Dutton*, 1999).
There are two main features that characterise an MRDB (cf. Figure 1):

a) Different levels of detail (LoD's) are stored in one database and
b) The corresponding objects in the different levels are linked.

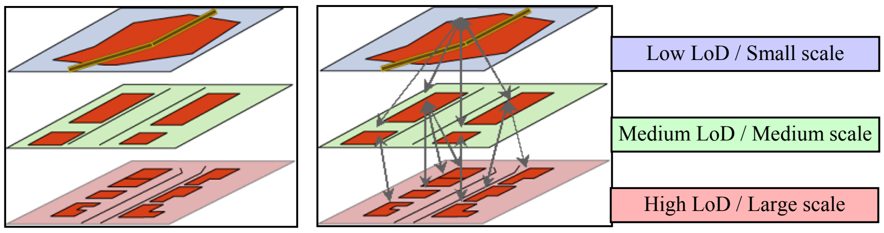

Figure 1. Characteristics of an MRDB: Store multiple representations (left), link corresponding objects (right).

There are several reasons for introducing an MRDB: On the one hand it allows for a multi-scale analysis of the data: Information in one resolution can be analysed with respect to information given in another resolution. Gabay and Sester (2002) present an example where topographic data is linked with cadastral data. A topographic data set of lower resolution containing only settlement areas is queried concerning the buildings in that area – an information that can be derived from a more detailed cadastral data set, whose objects are directly linked. A major reason for National Mapping Agencies to investigate and implement MRDB is the possibility of propagating updates between the scales: the appealing idea is that the actual information only has to be updated in the most detailed data set, this new information can then be propagated through the links in MRDB to all the other scales (Kilpeläinen 1997, Harrie & Hellström 1999). More details concerning the whole benefits of such kind of databases can be found in *Hampe & Sester* (2002).

In the EU-project *GiMoDig* (Gimodig, 2004) we aim at serving topographic data for mobile users in real-time. The data, coming from the databases of the national mapping agencies (NMA's) in Sweden, Denmark, Finland and Germany will be harmonised and generalised for the certain individual needs of a mobile user. That means the data have to be generalised within a few seconds. The intention in GiMoDig is on the one hand to develop methods for generalising the data in real-time and on the other hand to analyse the possibility to support this generalisation process by using the different levels of detail (LoD) stored in an MRDB. A multiple representation database has been set up to investigate the benefits of such a database.

When designing an MRDB two important cases can be discerned:

a) Linking existing datasets of different scale or thematic contents by matching procedures.
b) Creating new datasets from existing ones, which then form new layers in the MRDB.

Concerning the first option, matching operations have to be developed. With regard to the second option new datasets have to be created from existing ones. In order to do so, a function has to be known that allows this creation. In the case of deriving a small scale dataset with a high resolution from a higher resolved one, generalisation operations can be applied. The function immediately establishes also the links between corresponding objects. Consider for example the aggregation of two adjacent buildings to a new combined building in the lower resolution dataset: links will be established between the high resolution buildings to the newly created one.
Both, an implementation for matching existing datasets as well as the derivation of several levels of detail and the links between those levels out of one existing high LoD by generalisation methods will be described in the following chapter. That chapter also portrays the data model which describes the links between the corresponding objects.

3 IMPLEMENTATION OF GENERALISATION AND MATCHING ALGORITHMS

In our research work we concentrate on buildings respective built-up areas. The aim was to set up an MRDB representing settlement structures starting at a very high LoD up to a very low LoD, say 1:1 Mill. In our case only data with a high level of detail were available for our purpose. That means that we had to derive more layers representing lower levels of detail of the same real world objects out of this first layer. The source data containing buildings, serve for a scale range of 1:1k up to 1:5k.

3.1 The development environment: JTS/JCS/JUMP

The main common development platform in our project is a Java-environment called *JTS/JUMP* (JTS Topology Suite, JUMP Unified Mapping Platform) (Vivid Solutions, 2004) which provides certain functionalities to import, handle and manipulate spatial data. The *JTS Topology Suite* is an API of 2D spatial predicates and functions. It conforms to the Simple Features Specification for SQL (SFS, 1999) published by the Open GIS Consortium. That means it is possible to import data which are following these specifications from the database. It also provides a complete, consistent and robust implementation of fundamental 2D spatial algorithms.
JUMP provides an API giving full programmatic access to functions, including I/O, feature-based datasets, visualisation, and all spatial operations like buffer or overlay. The provided classes allow for including own functionalities as an item in the menu bar or as cursor tools in the *JUMP* graphic user interface (GUI). These own functionalities can be easily integrated by copying the classes or java-archive (jar) files into a certain directory of the *JUMP* environment. The whole package including *JTS*, *JUMP* and *JCS* (Conflation Suite) is free to download and use.
During the development phase of the prototype this environment is used as an authoring tool as well as for visualising the processed data.

3.2 Generalisation methods

As mentioned in the chapters before our intention to develop and implement offline generalisation algorithms was to create several MRDB-levels out of one existing dataset. The offline generalisation processes are time consuming generalisation methods which sometimes can't be treated 100 percent automatically. For that reason the data are processed in beforehand and the results are stored in the database which can be accessed at the moment the data are requested. The online or real-time generalisation processes on the other hand will be computed at the moment the user asks for a map with a certain scale. The source data or the data pre-calculated by offline generalisation will be requested and generalised within a few seconds serving for the destination scale.

We integrated functionalities for building simplification, amalgamation, enhancement as well as typification and displacement into the *JUMP* environment. Although intended for offline generalisation these algorithms can also be used online, when applied only to small extracts of a spatial scene.

In this first phase we just implemented the generalisation algorithms separately. That means non of the implementations described below takes into account the other processes. In the near future a combination is thinkable. The algorithms described below serve also for different scale ranges. As the simplification and also the amalgamation of buildings is used for larger scales, the typification is used for smaller scales and also displaces the buildings by building symbols. It is not yet investigated for which scales steps the algorithms have to be used for generating maps for mobile devices. This will be also one of the future works.

3.2.1 Building simplification

The principle of the algorithm used for the building simplification is described in detail in *Sester* (2000). The idea is that three cases can occur in which a building has to be simplified:

- Offset
- Bulge
- Edge

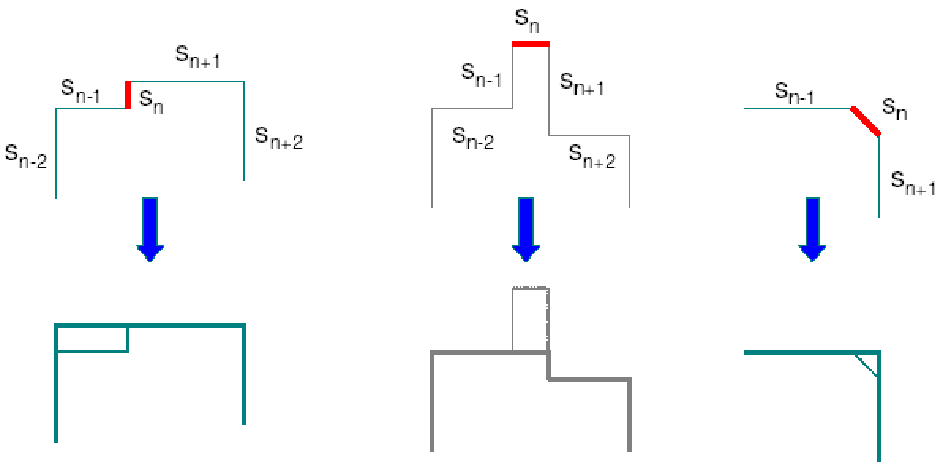

Figure 2. Cases for building simplification: Offset (left), bulge (middle) or edge (right). *Sester* (2000)

For every object, meaning every polygon or multipolygon, the length of every edge is the base operation. If an edge S_n is found which is shorter than a specified threshold value the algorithm has to find out which of the three cases mentioned above applies. This can be found out by comparing the angle between the edge before S_{n-1} the actual one S_n and the following edge S_{n+1}. Depending on the case the offset, bulge or edge will be eliminated. The results of this process can be seen in Figure 3. The process described above is quite fast, so that it can also be used for a real-time generalisation process. In our tests a number of 2.000 buildings have been processed in less than one second.

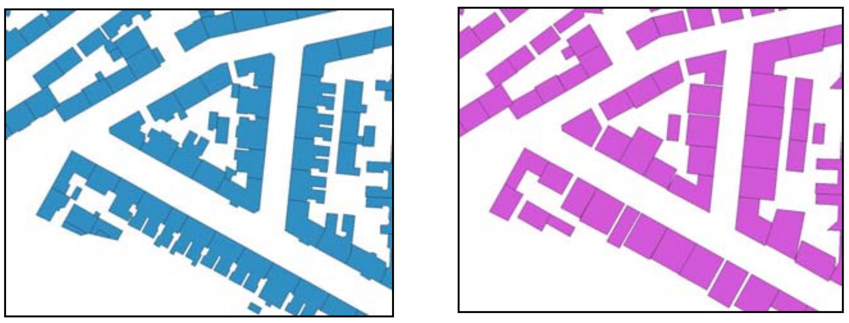

Figure 3. Simplification process: Original situation (left). Results of the simplification process (right).

As every building is treated on its own without any respect to the neighbouring objects the building is simplified as expected but after the simplification new gaps occur between neighbouring objects when eliminating parts of a buildings or the whole building. In the latter case the typical structure of the settlement would be destroyed, which is mainly critical in dense urban areas. Furthermore because of the same reason buildings aligned in a row now have offsets. Both can be seen in Figure 3. Furthermore the process can cause overlapping objects again because the new geometry does not take the neighbouring object into account. This lack can be equalled by processing amalgamation described below after the process of simplification. The combination of these two processes or to express it in a more common way, the combination of several generalisation methods into a unique network has not been investigated yet but has to be taken into account in the near future.

3.2.2 Building amalgamation

If the space between two objects is smaller than a certain minimum, depending on the scale, this gap is not visible for the user of the map. In this case the distance should either be expanded to be visible or should be zero, which leads to an amalgamation of the objects involved.
Our approach needs two steps to combine two neighbouring objects to one geometry. We enlarge the objects in a first step, combine all the now overlapping objects and in a third step these new geometries will be downsized again to their original size.
A by-effect of this enlarging and scaling down is a simplification of the objects, as it can be seen in Figure 4. These morphological operations are borrowed from the field of image segmentation, described by *Scott* (1998): The two principal morphological operations are dilation and erosion. Dilation allows objects to expand, thus potentially filling in small holes and connecting disjoint objects. Erosion shrinks objects by etching away (eroding) their boundaries. These two basic operations, dilation and erosion, can be combined into more complex sequences. The most useful of these for morphological filtering are called opening and closing. Closing consists of a dilation followed by erosion and can be used to fill in holes and small gaps. This combination of dilation and erosion is used in our case for amalgamating adjacent objects.

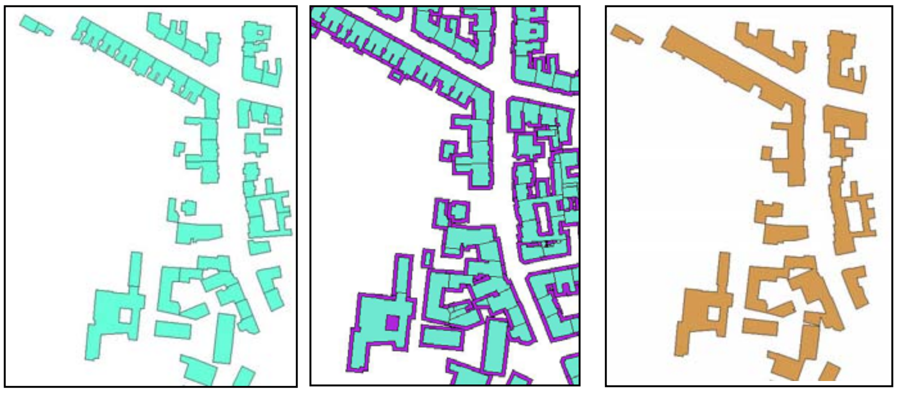
Figure 4. Amalgamation process: Original situation (left), enlarged objects (middle), resulting objects (right)

A time consuming part of this process is to compare every object with all other objects to find out, if the actual one is overlapping another object. Two ways to speed up this procedure have been followed. First the *JTS* provides functionalities to use spatial indexes for the test of overlapping. The second approach is to analyse only objects located inside the same settlement area. The road network serves as the basis for this approach. The roads which consist of a number of linestrings will be polygonised, meaning that closed polygons are computed from this linear network (see Fig. 5). This functionality can be simply integrated as it is already implemented in the *JTS* environment. So the input datasets are street polygons, which have to be calculated in beforehand and the buildings. Alternatively we can use the MRDB structure which already stores information about built-up areas, which have the same function as the polygonised streets in this case, and the linked buildings. Amalgamating only the objects located inside one polygon i.e. a certain settlement area also avoids combining buildings located over the road.

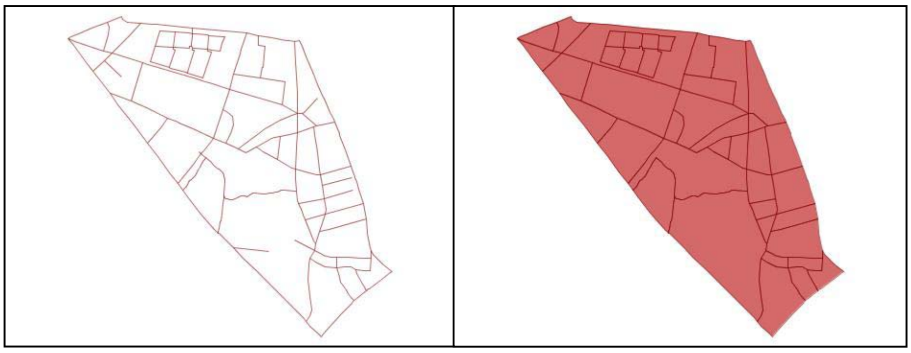
Figure 5. A road network consisting of linestrings (right); the same road network after polygonisation (left).

The input parameter for this function is the minimal visible space between two objects, which is dependent on the scale. A rule of thumb is a minimum of 0,15mm as a distance between two objects (*Hake & Grünreich*, 1994) which leads to a threshold value of 1,5m for a destination scale of 1:10k or 15m for 1:100k.
Compared to the simplification algorithm presented above computing the amalgamation takes more time leading to the conclusion that it can't be used for the realtime generalisation process. The section shown in Figure 3 containing 1200 buildings and 50 settlement polygons takes more than 30 seconds. A time span a mobile user is probably not willing to wait for his map.

3.2.3 Building typification and displacement

Building typification is needed, when smaller scales have to be generated (<1:30k). Then, no longer individual buildings are represented, but symbols of buildings that are representatives for the group of original buildings. To this end, a Neural Net approach was implemented, that has the effect that after the reduction of the number of objects, the spatial distribution and density of the original buildings is still preserved.
Displacement is needed, when other generalisation operations – especially enhancement or simplification – or larger symbol sizes lead to spatial conflicts. An approach based on least-squares adjustment allows to solve the spatial conflicts by displacing and/or deforming objects according to a given priority order and given specifications.
Details on the typification as well as the displacement algorithm can be found in *Sester* (2004). The algorithms are implemented in C++; they are integrated in the Java-environment, by system calls.
The following Figure 6 shows the result of the typification. Input parameters are the original dataset, the percentage of reduction as well as the destination scale for amplifying the resulting geometries.

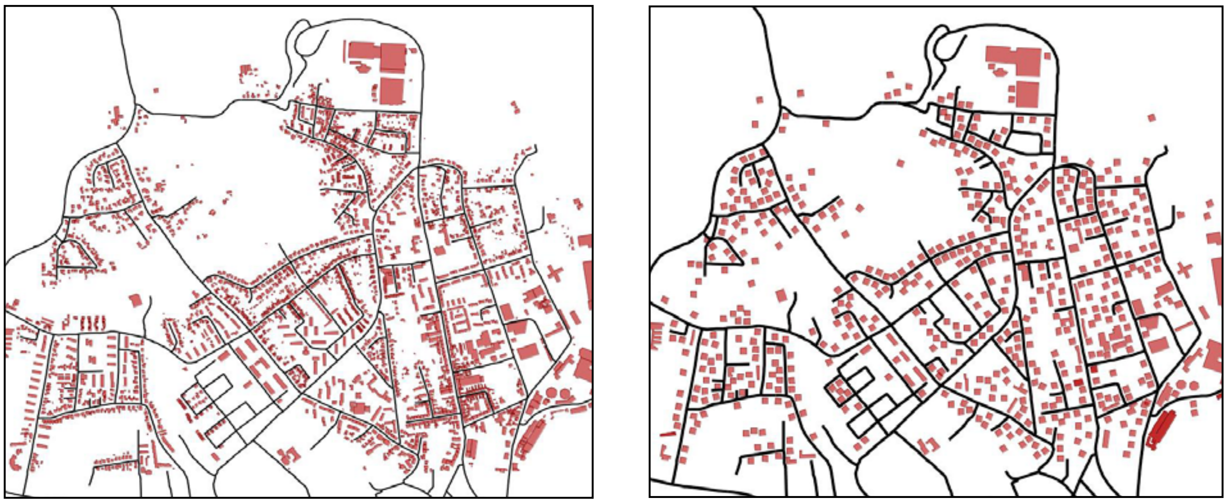
Figure 6. Original dataset (right); typified and displaced data (left).

3.3 Deriving the links between corresponding objects

Beside the feature to maintain several levels of detail of real world objects in a database, a second feature is to maintain also the links between objects representing the same real world phenomenon. The links between

corresponding objects may be useful for the use cases mentioned in chapter 2. In section 4 we introduce some test cases in which these links are essential.
As we maintain an object-relational database the links between the features are realised using the object-ID of them. In our case we prefer an one way link (see Figure 7). That means every object knows all its corresponding objects in the smaller scales, i.e. objects in the largest scale have links to all other representations stored as an

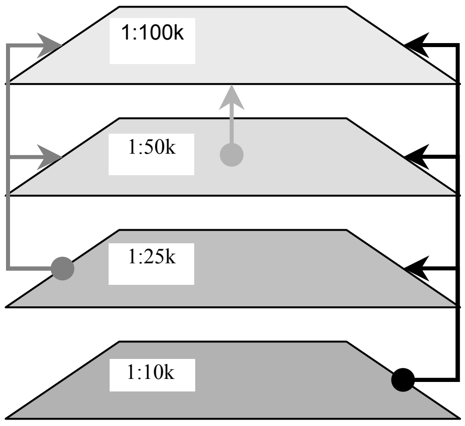

Figure 7. Connections stored in of the MRDB

Table 1:50k

ID	geometry	link_100k
5012	POLYGON ((352 ...	9858
5013	POLYGON ((352 ...	9578

Table 1:25k

ID	geometry	link_50k	link_100k
2513	POLYGON ((352 ...	5012	9858
2514	POLYGON ((352 ...	5048	9987

Table 1:10k

ID	geometry	link_25k	link_50k	link_100k
1012	POLYGON ((352 ...	2513	5012	9858
1013	POLYGON ((352 ...	2599	5189	9858

Table 1. Structure of the database tables '1:50k', '1:25k' and '1:10k'

attribute reflecting the ID of the linked object.
Table 1 shows an example of a database table. The table '*1:10k*' stores the links to the tables '*1:25k*', '*1:50k*' and '*1:100k*' in the columns '*link_25k*', '*link_50k'*, '*link_100k'*. For example the building *1013* is represented by the object *2599* in the scale 1:25k, e.g. a simplified building, is also linked to the object *5189*, e.g. an amalgamated building in the scale 1:50k, and is part of the built-up area *9858* in the representation at scale 1:100k.
But objects don't know their corresponding representation in a higher LoD. The reason for this strategy is the assumption that we have an n:1 or n:0 relation from large scale to small scale. There are relations thinkable where an n:m relation is thinkable with both, *n* and *m*, are variable and larger than one. In the cases mentioned above, the simplification, displacement and typification we assume an n:1 or n:0 relation.
To store the relation to one or zero objects in a table is easy as we can add this link as an extra attribute meaning an extra column in the database table. If there would be more than one object linked to another one we need more than one extra column to store this link. As the number of linked objects would be variable from object to object we need a variable number of columns which leads to a high number of empty cells. The back door is to store only links to one or zero objects which leads us to the beginning, that we can store only links from large scale to small scale. In the case of an *n*:*m* relation we have to introduce extra tables where the links can be stored in.

As the links are only stored for one direction nevertheless requests are possible for both directions. For instance one can request for the built-up area the building *1013* is located in, a request from high LoD to low LoD, as well as one can ask for the buildings located inside a the built-up area *9858*, a request from low LoD to high LoD.
Let us assume that in the layer *1:10k* buildings are stored and in the layer*1:100k* we maintain built-up areas. Following the linking structure shown in Figure 5 and Table 1 a request for the built-up area the building *1013* is located in would be like: "*Get the attribute of the column 'link_to_100k' from the object 1013 stored in table '1:10k'.*" The result would be the number *9858* indicating the ID of the built-up area the building *1013* is located in.
Getting all buildings located inside the built-up area *9858* the following request would be used: "*Get all the objects where the column 'link_to_100k' has the value 9858.*" The resulting set of data will be all objects located inside the built-up area, in the example the buildings *1012* and *1013*.

As mentioned above, two ways are possible to derive the links between corresponding objects. One can use matching tools to link two or more existing datasets or while generalising the data both, the original as well as the generalised object is known during the generalisation process. This knowledge can be stored in the database.
Both ways to derive the links between corresponding objects have been implemented. As the simplification and amalgamation algorithm has been newly implemented we integrated a functionality to store the ID of the generalised object as an attribute added to the source object.
For the typification and displacement functionalities we used existing C++-coded functions that do not produce the link structure. For displacement the links are straight forward as the objects only may shift their position, for

typification a geometric matching functionality was realised to match the original and the generalised objects. Every object respective building searches for the closest object in the dataset of typified or displaced buildings and stores the ID of this object as an additional attribute.

4 EXAMPLES FOR USING MRDB

As mentioned in chapter 2 we aim at demonstrating the benefits of a multiple representation database and for that applications using an MRDB have been implemented.

All spatial data stored in an object-relational database will be accessed via a WFS-connection (Web Feature Service, WFS 2004). This WFS is contacted by an http-request specifying the table and the attributes of the features. The WFS gets the data from the database and provides GML (Geography Markup Language, GML 2004) data as a result. These GML data are imported into the certain application which transforms the geometries into features and feature collections containing the geometries and the attributes. These feature collections can be visualised in the JUMP-GUI. Alternatively, after processing the data the result can be transformed into the GML format again for further computation, e.g. for a transformation into an SVG image (Scalable Vector Graphics, SVG 2004).
The architecture for accessing the MRDB and to implement the test cases mentioned in the following chapter is similar to the whole GiMoDig system architecture but leaving out some parts which are not needed here. The whole GiMoDig system architecture is described in Lehto (2003).

4.1 Emphasise landmarks and points of interest

The advantage of the non-printed mobile maps is to take the special needs of the user into account while creating the map. The user is interested in certain objects more than in other objects. For instance while navigating the user is interested in landmarks, eye-catching objects or points of interest located within a certain distance from the users position.
To emphasise these objects in the map several possibilities are cogitable. In order to visualise one individual building in the generalised data set a possibility is to use graphic variables like colour, size or the shape of the object. At this point the problem occurs that in a certain scale the objects are generalised which e.g. means the buildings are for instance amalgamated which in turn means that we can't localise a single object to point out a landmark or a point of interest (PoI).
A combination of this individual building and the generalised background buildings is desirable, given in Figure 8 (right) with detailed buildings and aggregated built-up areas.

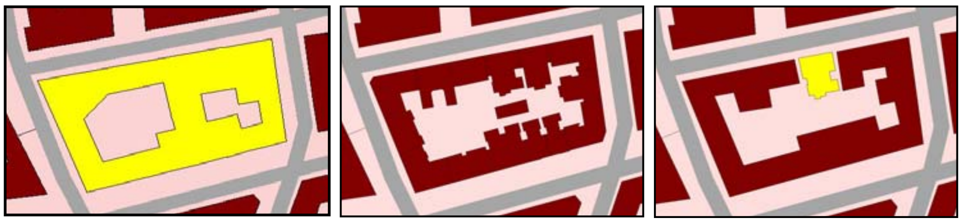
Figure 8. Emphasise PoIs using MRDB: Ungeneralised buildings (left). PoI in full detailed shape, re-generalised remaining buildings (right).

On the one hand the user gets detailed information of the objects of interest with a high level of detail and on the other hand he or she gets generalised spatial information of the remaining objects which are adequate for the actual scale and the display size.
These kinds of applications can be realised using the structure of the underlying MRDB. Because only a low number of objects has to be transferred and generalised the whole process of adapting the map can be done in real-time. The extraction and visualisation of landmarks in combination with an MRDB is described in detail in *Elias* et al. (2004).

4.2 Information drilling

Another application which benefits from the MRDB-structure is the "information drilling" scenario. The mobile or non-mobile user may be interested in a certain area or a certain object in the map or even in certain attributes which are stored not explicitly with the certain feature but with the feature linked to this.

4.2.1 *Drilling for geometry*

In the case of drilling for geometry the user can for instance request the buildings which are located inside a certain built-up area or a single building which is part of an amalgamated object in the actual map.
From the users point of view it is only necessary to click on the object or position he or she wants to drill for more information. This click activates a request sending the ID (identification) with the number of the built-up area. A servlet passes this request on the WFS by requesting only those objects which are linked to the requested object. The user can determine the deepness of drilling which can go down to a detailed building plan linked with a certain building.

4.2.2 *Drilling for attributes*

The MRDB structure allows to get information which are not connected directly to the objects in the actual map. One example may be the name or the number of inhabitants of a city which is not typically an attribute of the buildings within the map of a certain scale. However if we go down to smaller scales in the MRDB the object built-up area or city owns the attribute "name of the city" and thus the buildings linked with this object can access this information.
If the user is interested in information about a city with respect to an individual building, the necessary sequence of operations is the following: From the buildings table, the corresponding built-up area the building belongs to is requested. Using the example from chapter 3.3. for building *1013* this is built-up area *9858*. From there the desired information about name and number of inhabitants can be returned.
That means the information the user can get is not limited to the attributes stored with a certain feature. Because of the links between the objects in the database the user has indirect access to all the information stored in the database.

5 SUMMARY AND OUTLOOK

The main focus of this paper was on the development and implementation of generalisation algorithms to produce several layers of an MRDB and the presentation of applications that exploit the MRDB-structure. We also pointed out that the introduced Java based development environment JTS/JUMP provides a fundamental basis to support the development of such kind of processes.
Generalisation algorithms have been presented that can and have been used to interactively generalise several representations of a certain settlement area. Furthermore we showed that the links between the representations can be obtained automatically during the generalisation process. We presented tests where this structure is also useful to provide new applications, mainly for mobile devices with their small displays. On the one hand the MRDB will be used to support the process of generating individual maps in realtime. A combination of online generalisation applications and MRDB access is desirable. Furthermore the user of the map has access to all the information linked to a certain object.
The future work will concentrate on the development of the functionalities of the MRDB and the connected architecture. It will be extended to a more flexible tool for different kinds of mobile users. One application will be to combine the MRDB with an online generalisation service as mentioned above. Investigations have to be accomplished to determine which generalisation procedures can be processed in realtime and which need support from the MRDB, maintaining pre-generalised datasets. There are two cases in which an online generalisation would fail. The first would be a too time consuming generalisation process. The user is not willing to wait for the requested data as long as it takes. After a time span of 10 to 15 seconds he or she cancels the request. The second reason for pre-processing is that there occur some cases in which the generalisation can't be done completely automatically. The development of an intelligent system which decides what can be done online and when an MRDB should be contacted has to be developed in the future.

6 REFERENCES

Devogele, T., Trevisan, J. & Raynal, L., 1996. Building a Multiscale Database with Scale-Transition Relationships. Proc. of the 7th Int. Symposium on Spatial Data Handling, Advances in GIS Research II. Delft, The Netherlands, pp. 6.19-6.33.

Elias, B., Hampe, M. & Sester, M., 2004. Adaptive visualisation of landmarks using an MRDB. Accepted for publication in: Map-based mobile services – Theories, Methods and Implementations. Springer Verlag, 2004.

Gabay, Y. & M. Sester, M., 2002 Forming and utilizing communication between two spatial representations at different scales - a demonstration. Submitted to Geoinformatica.

GML, 2004. Geography Markup Language Implementation Specification, Version 3.0. http://www.opengis.org/docs/02-058.pdf (accessed 19 April 2004).

Gimodig, 2004. http://gimodig.fgi.fi (accessed 01. June 2004).

Hake, G., Grünreich, D. and Meng, L., 2002. Kartographie, Gruyter Verlag, Berlin, Germany.

Hampe, M. & Sester, M. 2002. Real-time integration and generalization of spatial data for mobile applications. Geowissenschaftliche Mitteilungen, "Maps and the Internet", Wien, Heft (60):167-175.

Harrie, L. & Hellström, A.-K. 1999. A Prototype System for Propagating Updates between Cartographic Data Sets. The Cartographic Journal, Vol. 36, No. 2, pp. 133-140.

Lehto, L., 2003.Architecture specification. GiMoDig-project, IST-2000-30090, Deliverable D4.4.1: Final system architecture, Public EC report, 41 p. An electronic version available at http://gimodig.fgi.fi/deliverables.php (accessed April 23, 2004).

Kilpeläinen, T., 1997. Multiple Representation and Generalization of Geo-Databases for Topographic Maps. PhD Thesis, Helsinki University of Technology, Kirkkonummi, Finland.

Scott E. Umbaugh, 1998.Computer Vision and Image Processing, Prentice Hall PTR.

Sester, M., 2000. Generalization Based on Least Squares Adjustment, International Archives of Photogrammetry and Remote Sensing, Vol. XXXIII, Part B4, Amsterdam, pp. 931-938.

Sester, M., 2004. Optimizing Approaches for Generalization and Data Abstraction, accepted for publication in: International Journal of Geographic Information Science, 2004.
SFS, 1999. Simple Feature for SQL Specification. http://www.opengis.org/docs/99-049.pdf (accessed 20. April 2004).

SFS, 1999. Simple Feature for SQL Specification. http://www.opengis.org/docs/99-049.pdf (accessed 20. April 2004).

SVG, 2003. Scalable Vector Graphics 1.0 Specification. http://www.w3.org/TR/SVG (accessed 15 April 2004).

Vivid Solutions, 2004. JTS, JCS, JUMP. http://www.vividsolutions.com (accessed 15 April 2004).

Weibel, R. & Dutton, G., 1999. Generalising spatial data and dealing with multiple representations. Geographic Information Systems – Principles and Technical Issues, volume 1. pp 125–155.

WFS, 2004. Web Feature Service Implementation Specification, http://www.opengis.org/docs/02-058.pdf (accessed 19 April 2004).