Proceedings of the 6th AGILE
April 24th-26th, 2003 – Lyon, France

USABILITY OF GEODATA FOR MOBILE ROUTE PLANNING SYSTEMS[1]

Martin Breunig and Wolfgang Bär

Research Center for Geoinformatics and Remote Sensing
University of Vechta
P.O. Box 1553
49364 Vechta
Germany

e-mail: (mbreunig,wbaer)@fzg.uni-vechta.de

1. INTRODUCTION

The impact of tele- and mobile information technology will increase the need for route planning systems drastically [1,2]. However, hitherto many problems concerning the wireless access to geodata are not yet solved. First experiences with existing systems in the leisure time sector show that geodata cannot be accessed continuously, because the connection is timed out or because data is not locally available. The requirements of route planning systems to GIS technology mainly concern the modelling and management of graph-based geodata. The efficient storage of the routes and the supply of location-based database queries are new challenges to meet. Many database queries should be supported locally at the mobile client device, because they only need access to data which is spatially located near the user. That is why such queries should be processed without connection to the server DBMS. However, today's commercial database systems for mobile devices do not support spatial database queries.

In comparison to today's static route planning systems, a route planning system with mobile devices should not only support the preparation of routes being realized by graphical or textual route descriptions. But it should also support the user online during the tour, ride or walk, with guide features.

2. REQUIREMENTS OF MOBILE ROUTE PLANNING SYSTEMS TO GIS TECHNOLOGY

First experiences at our research centre with bicycle route planning systems [3] showed that a route planning system with mobile devices should provide the following GIS functionality separated into server and mobile client:

Server functionality:

- Computation, storage and visualization of different route types (e.g. shortest routes between two arbitrary points, roundtrip route for a given point set);
- generation of routes under consideration of semantic information (e.g. tourist information, roads with certain points of interest).

Mobile client functionality:

[1] This work has been supported by KFN, University of Vechta, Germany.

- Database support for the location-based visualization of the route and environmental data;
- offline data management of route data and tourist information (e.g. points of interest);
- offline support of smaller changes of the route to reserve the user the possibility to react flexibly on road works.

The offline support of tourist information leads to the transfer of large data sets to the mobile device. Additionally to routes being generated by the server system, neighbouring roads are candidates for possible updates of the route or detours. Tourist information near the used route have to be stored in the mobile client DBMS. Hence a DBMS developer can choose between two extreme possibilities:

1) Loading the complete road network at the beginning of the database session from the server database to the client database and processing it further at the mobile client.
2) Managing the road network at the server. Queries to the database are transferred each time when needed ”on demand” from the client to the server. The result of the database query is transferred back to the client.

The drawback of the first possibility presented above is that the data sets are too large (e.g. the bicycle network of Germany) to be transferred, managed or to be presented graphically on today´s mobile devices. The second possibility, however, has the disadvantage that a continuous connection between client and server has to exist almost without disconnection. The frequent communication between server and client can easily appear as the ”bottleneck” between the application and the database management system. Therefore, the following compromise seems to be a promising approach:

The primary route computation is done on the server database management system and the computed route with the essential data set is transferred to the client database management system. During the usage of the computed route, finer grained parts of the road network and the tourist information are dynamically loaded – dependent on the current position – for the further processing of location-based queries on the mobile database.

3. MODELLING AND MANAGEMENT OF GRAPH-BASED GEODATA

3.1 Requirements

For the modelling and management of road networks and routes, obviously graphs are the best suited data types [4,5,6,7]. Additionally, in route planning systems different levels of detail have to be distinguished to support the modelling and storage of hierarchies [6]. Using hierarchical graphs, sub-graphs can be inserted into new nodes and edges of a graph. With this extension nodes of different levels of detail can be connected with each other. Furthermore, the navigation between nodes of different levels of detail can be realized. The storage of graph-based geodata has to be organized in a way that graph algorithms like the shortest way search [8] are supported efficiently. Furthermore, algorithms on hierarchical graphs consisting of several sub-graphs and the distance search for routes of different levels of detail have to be supported efficiently. A special problem is to find points of interest which are outside the graph. To solve the problem additional context information has to be represented in the database. To provide efficient route planning, the following types of database queries on graphs have to be supported:

- Shortest way queries and context-specific queries (considering the amount of points of interest, the degree of slopes, user profiles etc.);

- navigation in graphs (predecessor, successor nodes and edges);
- search in hierarchical graphs (distinction between different levels of detail);
- spatial selection of sub-graphs (contains and intersects region query).

If using the route planning system without connection to the server DBMS, the route data and relevant additional information have also to be managed in the DBMS of the mobile device. The consequence is that large data sets have to be managed by the mobile client DBMS. Furthermore, the access to local audio guides (e.g. to explain the landscape or points of interest in a region) and to internet-based services like weather forecast services, accessible via external wireless connections are of special interest [9]. The mobile client DBMS should allow specific location-based spatial and non-spatial queries. The server DBMS should provide kernel functionality like the support of region queries and the navigation on nodes in a graph.

3.2 Using an XML-based DBMS

State-of-the-art

We follow Waterfeld [10] speaking of an XML database management system, if and only if the database management system understands XML completely as its data model. The schema then is defined in XML and the DBMS can store arbitrary XML instances. Furthermore, the XML database management system returns the XML instances with an XML-based query language. This implies the use of modern standards like XML Schema and XPath. XML Schema and XPath have been developed for the description of schemes and for the hierarchical navigation in nodes of a document (so called path expressions).

Comparable with the introduction of object-relational and object-oriented DBMS in the eighties, XML database management systems can also be divided into two categories [10]:

1) Extensions on top of existing database management systems;
2) ”native” XML database management systems.

The first approach uses existing DBMS technology adding a new XML layer to model XML data. The second, however, contains a totally new DBMS for the management of XML data. Obviously, a third approach – the extension of existing relational or object-oriented DBMS ”from the side” – is not pursued by today’s DBMS vendors. In this approach it would be possible to extend the DBMS at different layers like query language, access structures etc. Most of the existing DBMS today have realized layer-based solutions on top of relational or object-relational DBMS. However, there are also native XML database management systems.

DBMS of the first category are IBM DB2® and Oracle 8i®. The advantage of these systems is that proven database technology can be used. However, this advantage is paid with the price of losing information during the mapping to other data models. This can lead to bad performance, especially when very large data sets are processed. A representative of the second category of DBMS is Tamino DB® [11]. This DBMS stores XML documents in an optimized way without using other data models. It also provides XML database structures for the physical storage of the data. The XPath Standard has been extended in Tamino in a way that it can be used as query language for sets of XML instances. With the release of XQuery [12], current working draft status of W3C, in the next versions of Tamino DB also a standardized query language for XML documents will be supported. First prototypical implementations based on this specification already exist.

® DB2 is a registered trademark of the International Business Machines Corporation IBM
® Oracle 8i is a registered trademark of Oracle Corporation
® Tamino is a registered trademark of Software AG, Darmstadt, Germany

Using an XML-based DBMS for the management of geodata

To the knowledge of the authors, hitherto no solutions of XML-based DBMS for the management of (graph-based) geodata exist. Generally, the following two approaches could be used:

1) The extension of the DBMS directly for geodata (indexing, query) in the DB kernel;
2) a layer-based extension for geodata realized in the specific API of the DBMS.

The first approach is more efficient, because the extensions can be built in at a lower system level. Furthermore, a ”toolbox” for different classes of geo-applications can be realized in the DBMS (e.g. for 2D/3D applications). The advantage of this solution is that the classes do not have to be implemented from scratch for each application again (re-use of the code). The disadvantage, however, is that the DBMS developers are dependent on the further proceeding of specific geo-standards. That is why the geo-extensions often have to be updated and adapted. The second approach is more flexible, because an adaptation of the XML standard for geodata only has to be carried through for the current application class. However, the portability of other application classes is more difficult. In the following we give an example for the second approach (see also [13,14]).

An extension of Tamino API for the management of XML geodata

A straight-forward way to extend native XML DBMS is to extend their API used for the access to the DBMS. In the following we describe an extension of the Java API of Tamino DB. The extension can be divided into two parts. The first part contains the implementation of a spatial R-tree based index [15] with a simple support of transactions. This implementation is independent of the used DBMS. The second part is the extension of Tamino API with interfaces and classes for the management and the access to XML geodata.

The definitions of the Tamino API interfaces and classes being responsible for the access to XML data could be taken over 1:1 into the geo-extension, just adding the denotation ”Geo” at the beginning of the interface-, class-, and methods signatures. The implementation of the interfaces of the geo-extension has been realized by using a container around the existing access classes. Therefore within the classes of the geo-extension a pre-processing of the data takes place. Finally, the data are transferred to the implementation of the wrapped class of the original Tamino API. In the current realization all XML documents that contain a defined element ”BoundingBox” are inserted into a spatial index. Furthermore, the currently used XML query language of Tamino has been extended within the geo-extension by the query types `intersects`, `contains` und `nearestOf` to be used in spatial queries.

Currently, the storage of GML documents is not possible with Tamino DB, because the XML Schema elements needed by GML are not yet completely supported in Tamino Schema Language.

Modelling of graph-based geodata in an XML DBMS

There are several different possibilities for modelling graph data structures in an XML-based DBMS. We enter into the following two ways of modelling:

1) Modelling graph parts as XML documents with their nodes, edges and pointers to the neighbour graph parts;
2) modelling edges and nodes as single XML documents with pointers to their neighbour edges and adjacent nodes.

The first possibility partitions the geometric region given by the graph data into equal parts. Every part is modelled as an own XML document with pointers to the connected edges/nodes of the neighbour graphs parts. A positive aspect of this approach is that spatially selected parts of the graph can be loaded with only one transfer call from the database management system. However, a disadvantage is the larger size of the XML documents, as most XML-based DBMSs (Tamino, Xindice) are designed for use with a large number of small documents stored.

From that point of view modelling every edge and node of the graph as single documents with pointers to their connected edges/nodes is the better modelling design, because it keeps the size of the XML documents small. This design is also preferable, if we need a database with high concurrency access to the node/edge data, since for example Tamino DB's locking mechanism is based on the document level.

In order to achieve spatial access to the graph elements, the design has to follow the restrictions given by the extension of Tamino's API for the management of XML geodata described above, i.e. inserting a "BoundingBox" element.

3.3 Using an OODBMS

Obviously, object-oriented database management systems (OODBMS) are well suited to model and manage network structures as they are needed in route planning systems. That is why we decided to test also a representative of an OODBMS as database management system for our route planning application. The advantage of such a system is to work with references which are well suited to model graph data. However, the OODBMS technology still is a research field, which means that the tools provided for data input, output and management are less comfortable than those of relational or object-relational DBMSs. On the other hand, extending OODBMS with special access path (like spatial access paths for geodata in arbitrary dimensions) is much easier to realize.

State-of-the-art

An OODBMS [16,17] provides the object-oriented data model completely as its data model. Its schema is defined as classes and the OODBMS can store class instances, i.e. objects. The OODBMS returns objects with an OQL-based query language or through "walking" along references from named database roots. Index support is directly provided for classes and their attributes. Following Heuer [17] we can distinguish between the two following types of OODBMSs:

1) Newly developed object-oriented database systems (e.g. O_2®, ORION/ITASCA, COCOON);
2) object-oriented database programming languages (e.g. Objectivity/DB®, ObjectStore® etc.).

The first type of OODBMS are newly developed database management systems, based on completely new designed object-oriented data models having neither the relational model nor an object-oriented programming language data model as conceptual basis. In contrary to this approach, object-oriented programming languages like C^{++} or Java implicitly support object-oriented modelling. What they fail in is an extension to store objects persistently. Thus the second type of OODBMS – the object-oriented database programming languages - extend the transient object-oriented programming languages by data handling operations. In chapter 4 we present evaluation results with Objectivity/DB®, a representative of the second type of OODBMS.

Using and extending an OODBMS for the management of graph-based geodata

® Objectivity/DB is a registered trademark of Objectivity Inc., USA

Like with the XML-based approach, in an OODBMS we again have to choose between the extension of the database kernel and the extension of the specific APIs of the DBMS for the management of graph-based geodata. As before, the second approach has been selected to extend the OODBMS by management operations needed.

In the OODBMS the retrieval of arbitrary region based graph parts is needed. Furthermore, efficient spatial retrieval of node and edge data or of different geocoded tourism information data have to be supported. The different dimensional extents of node (0 D) and edge data (1 D) leads to the need of a spatial access method for zero- and multidimensional data.

The spatial access method extension to Objectivity/DB is based on an R*-tree [18] with algorithms for nearest neighbour search. This R*-tree is modelled – according to the above second approach of extending a DBMS – as application objects of the OODBMS, implemented in the application programming language Java. Thus the OODBMS is responsible for transaction processing and concurrent access to the objects modelling the R*-tree.

The graph structure in Objectivity/DB is represented as single node and edge objects with connecting references. Every object in the database is indexed twice. Once by the spatial access method for efficient region based access to graph parts and once by a B-tree data structure for an ID based access for start and end nodes.

3.4 Using other DBMSs

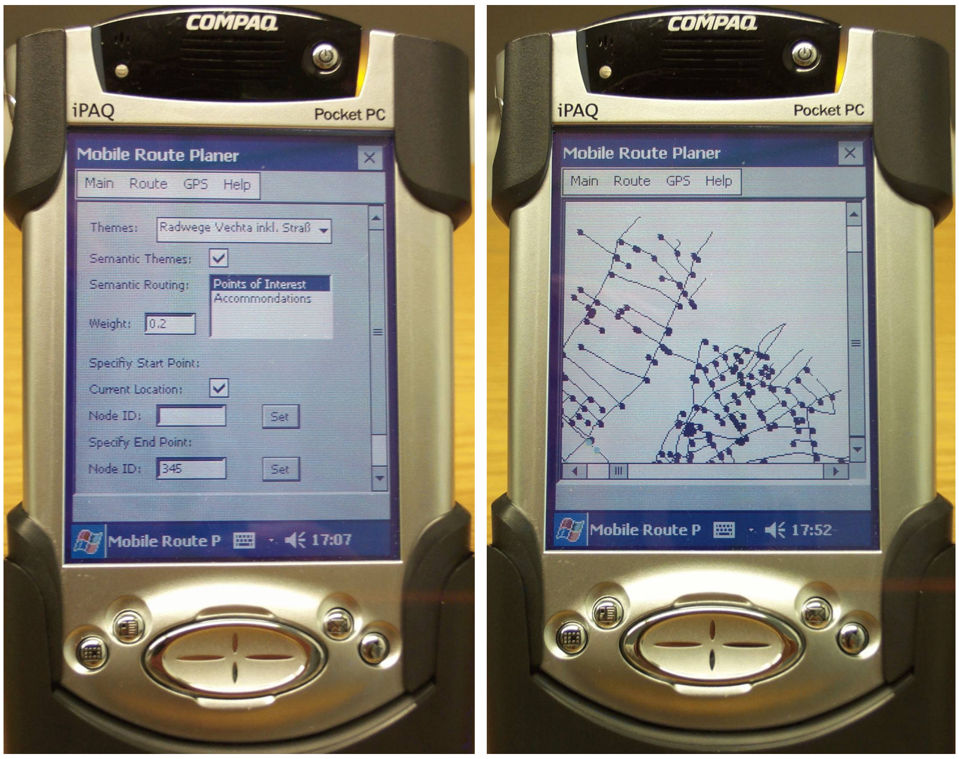
There are also other candidates for the management of graph-based geodata like object-relational database management systems (ORDBMSs). They also allow the modelling of object hierarchies and networks as they are needed for the management and processing of graph-based databases. ORDBMSs are based upon well known and proved relational database technology, i.e. the query processing can be optimized by the Relational Algebra and advanced tools for the input and output of data are available. The prize, however, we have to pay by using ORDBMSs is that object hierarchies and networks are mapped into “flat” relational tables. This leads to poor performance, if very large data sets are involved. Nevertheless, further experiments with a representative of an ORDBMS and an OODBMS, respectively, will follow in our future work.

4. EVALUATION WITH BICYCLE ROUTE DATA

Using the data of an existing local bicycle routing software [19], the given concepts have been prototypically implemented. As server DBMSs, Tamino DB with the extension for the management of XML geodata and Objectivity/DB have been used. At the mobile device a simple spatial object store, based on the Java serialization feature, has been implemented for prototypical experiments. Figure 1 shows two screenshots of the mobile client application implemented in Java (PersonalJava Specification) on a Compaq iPAQ device.

The data set of the local bicycle routing software consists of 40,908 nodes connected through 51,800 edges. The edges are weighted by their Euclidean distance connecting their nodes bidirectional, so that there are no one-way streets. The whole routing area has an extent of approximately 60 km horizontal, 80 km vertical and is located in the Oldenburger Münsterland (near Bremen) in Lower Saxony, Germany.

For the prototype of the bicycle route planning system a multi-tiered system architecture has been chosen. The layer for the primary route generation implementing a modified version of the A*-algorithm with a heuristic for Euclidean graphs [13,20], is based on a data broker component which dynamically loads needed graph data from the underlying server DBMS.



Themes: Radwege Vechta inkl. Straß

Semantic Themes: ✓

Semantic Routing: Points of Interest
Accommondations

Weight: 0.2

Specifiy Start Point:

Current Location: ✓

Node ID: Set

Specify End Point:

Node ID: 345 Set

Fig. 1 Screenshots of the mobile client application.

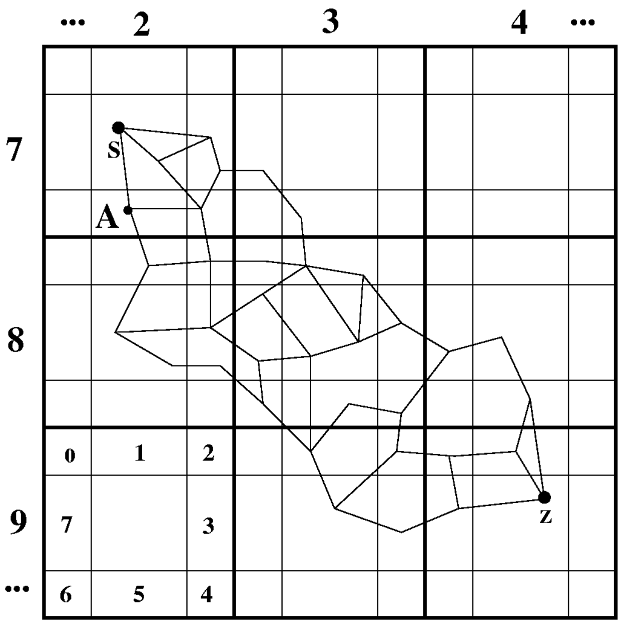


Fig. 2 Partitioning of the routing area with fringe-areas.
(S - start node, Z - end node, A - example node – see text for explanation)

We focused our attention on comparing the run time behaviour between the “brusque-force method” of loading the whole graph once into main memory and a non-optimized version of a graph partitioning method which loads sub-graphs on demand into main memory as they are needed by the way searching of the routing algorithm (fig. 2). Obviously, the first method can be used for small graphs. However, it will fail to be used for very large graphs. For the second method, however, a spatial cluster strategy should be used for further optimization.

The data broker component partitions the routing area given by the geometric location of the start and the end node (fig. 2), into equal partitions. Each partition has 8 fringe-area parts associated with their neighbour graph partitions. Even numbered fringe-area parts are associated with all three touching neighbour partitions. Fringe-area parts with odd numbers are only associated with the direct neighbour partition. If the routing algorithm requests a graph element (edge/node) from the data broker which is lying inside one of the fringe-areas (like node A in fig. 2), the associated graph partition(s) are loaded asynchronously into main memory.

The dynamical loading guarantees that only the graph parts just needed for route computation are loaded into main memory. Thus the memory consumption of the server DBMS is significantly reduced.

4.1 First tests with an XML-based DBMS

Table 1 shows the performance of this primary routing operation on the XML-based server DBMS Tamino. In the different columns, the time needed for the route computation, the number of nodes and edges visited by the algorithm and the number of nodes of the retrieved route are given, respectively.

As we can see, the use of the XML-based DBMS for this routing application is not yet acceptable due to the impedance mismatch between storage structure and the needed object representation of the routing algorithm. Despite that, the presented extension of Tamino API for the management of XML geodata reduces the time needed for spatial access to the XML documents drastically, especially with small result sets.

Routing time (seconds)	Nodes (count)	Edges (count)	Nodes on route (count)	Routing time/Nodes on route (ms per node)
127.7	14442	18533	334	382.4
62.1	7140	9172	226	274.5
30.1	835	1088	66	456.4

Table 4 Performance of primary routing operation for the XML-based DBMS

4.2 First tests with an OODBMS

In tables 2-4 the query results for Objectivity/DB as a representative of an OODBMS are given. The graph data is modelled as node/edge objects with references (see section 3.3). In every approach the start and end node is retrieved by a B-tree index. For the two sub-graph approaches the routing data is accessed only through the DataBroker component loading the graph parts on demand into main memory. The different tables stand for the following approaches:

Sub-graph approach with ID-network:

With this approach (see table 2) the node/edge objects are still provided for the routing algorithm by the DataBroker using an ID-based data structure for references between nodes and their edges. This approach is similar to that used in the XML-based DBMS which let us compare the routing times between that different types of DBMSs. Obviously, the query times are by a factor of 2-3 faster than with the XML-based DBMS. However, there still is needed too much time for the loading of the sub-graphs. These times are caused by the mapping of the database objects and by the expense needed for the management of the objects in the cache. Furthermore, the overhead caused by the database representation of the objects has to be considered.

Routing time (seconds)	Nodes (count)	Edges (count)	Nodes on route (count)	Routing time/Nodes on route (ms per node)
47.0	14442	18533	334	140.7
32.0	7140	9172	226	141.6
27.0	835	1088	66	409.1

Table 1 Performance of primary routing operation for the OODBMS sub-graph approach with ID-network

Sub-graph approach with reference network:

That approach uses - in contrary to the sub-graph approach with ID-network - the references between node and edges objects of the graph for providing the needed objects by the DataBroker. This means that in this approach the algorithm is also waiting until the sub-graph is loaded into main memory before proceeding with the algorithm. Therefore the same number of sub-graphs has to be loaded like in the “ID-approach” (first approach in table 2). The time profit of the references in comparison to the ID-based data structure in total are 1-1.5 seconds in average.

Routing time (seconds)	Nodes (count)	Edges (count)	Nodes on route (count)	Routing time/Nodes on route (ms per node)
46.0	14442	18533	334	137.7
30.0	7140	9172	226	132.7
26.0	835	1088	66	393.9

Table 2 Performance of primary routing operation for the OODBMS sub-graph approach with reference network

References-only approach:

In this approach the routing algorithm exclusively runs on the references of the graph element objects in the database loading only the node/edge objects needed by the routing algorithm into main memory. As we see in table 4, this approach shows the best results for the used route data sets. Only few objects, about by factor 6 less than for the other approaches, had to be loaded. Whereas in the sub-graph approaches 700,000 objects had to be loaded, in the references-only approach only 120,000 objects were loaded into main memory through the routing algorithm processing (numbers are inclusive database specific overhead objects).

Routing time (seconds)	Nodes (count)	Edges (count)	Nodes on route (count)	Routing time/Nodes on route (ms per node)
8.4	14442	18533	334	25.1
5.7	7140	9172	226	25.2
3.2	835	1088	66	48.8

Table 3 Performance of primary routing operation for the OODBMS references-only approach.

4.3 Interpretation of the results

Taking a look at the routing times for the different approaches, the references-only approach seems to be best suited for route computation in an OODBMS. However, in the context of a mobile routing application we not only intend to transfer the route itself, but also the adjacent graph data in a reasonable range for orientation or detour planning to the mobile client database. That is why not only the needed server side routing time has to be taken into account. Whereas the two sub-graph approaches are loading the needed graph data already into memory, the references-only approach has to do that time consuming job afterwards before transferring the data to the client applications database system.

It should be noticed that the presented query results are limited to three single route queries with different distance ranges between start and end node. The presented sub-graph approach is likely to lead to better performance results if applied to very large graphs or in a routing system with a large number of simultaneous routing queries. In that case already loaded sub-graphs from other routing queries reduce the loading time for the needed sub-graphs significantly. For such applications, the spatial clustering of sub-graphs in the DBMSs and "intelligent" swapping strategies for sub-graphs in main memory cache will play a central role. Further research with optimizing spatial cluster strategies for the sub-graphs has to be done in order to reduce the running time of the dynamical graph loading component. Using spatial clustering methods, the loading of spatially neighboured cells will reduce the number of needed database pages. For very large graphs, also hierarchical graph representations should be used and pre-computed routes should be provided as "stored procedures" in the DBMS. Our first experiments with the object-oriented database management system show that these systems seem to be well suited for this type of application.

5. CONCLUSION AND OUTLOOK

In this paper we have introduced the modelling and management of graph-based geodata to be used for mobile route planning systems. We showed the use of an XML database management system and an object-oriented DBMS, respectively, in an application of a mobile bicycle route planning system. Extensions of Tamino DB and Objectivity/DB have been implemented and evaluated by a local bicycle routing software. In our future work, we intend to test object-relational and object-oriented DBMS to be used for mobile geological applications in modern geoservices. The challenges in the new project are the acquisition, visualization and management of geological data by mobile client applications. Experiences from earlier work with OODBMSs [21, 22] and geological 3D modelling systems [23] will be helpful for these studies.

6. BIBLIOGRAPHICAL REFERENCES

[1] Brinkhoff, T., *Requirements of Traffic Telematics to Spatial Databases.* Proceedings of the 6th Intern. Symposium on Large Spatial Databases, Hong Kong, China. Lecture Notes in Computer Science No. 1651, 365-369, 1999.

[2] Zipf, A., Strobl, J., *Geoinformation mobil,* in German, Herbert Wichmann Verlag, Heidelberg, 2002.

[3] Schneeweiß, H., Jung, S., *"Fahrradies" – Planning Bicycle tours in the Internet.* In German, German ESRI User Conference, Munich, Germany, 8p., 2001.

[4] Agrawal, R., Jagadish, H., *Algorithms for Searching Massive Graphs.* IEEE Transactions on Knowledge and Data Engineering. 6 (2), 225-235, 1994.

[5] Becker, L., Güting, R.H., *The GraphDB Algebra: Specification of Advanced Data Models with Second-Order Signature.* Techn. Report No. 183 – 5/1995, Dept. of Computer Science, FernUniversität Hagen, Germany, 36p., 1995.

[6] Buchholz, F., Riedhofer, B., *Hierarchische Graphen zur kürzesten Wegesuche in planaren Graphen.* Techn. Report No. 1997/13, Institute of Computer Science, University of Stuttgart, 12p., 1997.

[7] Car, A., *Hierachical Spatial Reasoning: Theoretical Consideration and its Application to Modeling Wayfinding,* Geoinfo Series, Vol. 10, Dept. of Geoinformation, Technical University of Vienna, published Ph.D. thesis, Viennna, Austria, 1997.

[8] Dijkstra, E.W., *A Note on two Problems in Connection with Graphs.* Numerische Mathematik (1), 269-271, 1959.

[9] Giguère, E., *Mobile Data Management: Challenges of Wireless and Offline Data Access.* Proceedings of the 17th. Intern. Conference on Data Engineering, Heidelberg, IEEE Computer Society, Los Alamitos, CA, 227-228, 2001.

[10] Waterfeld, W., *Realization aspects of an XML database management system,* in German, Proceedings Datenbanksysteme in Büro, Technik und Wissenschaft (BTW), Oldenburg, Informatik aktuell, Springer, Berlin et al., 479-484, 2001.

[11] Tamino, *Tamino XML Server,* Version 3.1 - http://www.softwareag.com/tamino/., 2002

[12] W3C, *XQuery 1.0 - An XML Query Language,* W3C Working Draft (November 2002), URL: http://www.w3.org/TR/xquery/, 2002.

[13] Bär, W., *Design and implementation of software-components for a route planning system with mobile end-user devices,* in German, Diploma thesis, Department of Geoinformatics, University of Vechta, 112p., 2002.

[14] Breunig, M., Brinkhoff, T., Bär, W., Weitkämper, J., *XML-based techniques for location-based services.* In German. A. Zipf and J. Strobl, Geoinformation mobil, Wichmann Verlag, Heidelberg, 26-35, 2002.

[15] Guttman, A., *R-Trees: A Dynamic Index Structure for Spatial Searching.* Proceedings of the Annual Meeting ACM SIGMOD, Boston (MA), 47-57, 1984.

[16] Atkinson, M., Bancilhon, F., Dewitt, D., Dittrich, K., Maier, D., Zdonik, S., *The Object-Oriented Database Manifesto*, 1989.

[17] Heuer, A., *Objektorientierte Datenbanken,* Addison-Wesley, Bonn, 724p, 1997.

[18] Beckmann, N., Kriegel, H.-P., Schneider, R., Seeger, B., *The R*-tree: An Efficient and Robust Access Method for Points and Rectangles.* Proceedings ACM SIGMOD, Atlantic City, N.Y., 322-331, 1990.

[19] Fahrradies, http://www.fahrradies.net, 2002.

[20] Sedgewick, R.; Vitter, J. S., *Shortest Paths in Euclidean Graphs.* Algorithmica 1 (1986), No. 1, pp. 31-48, 1986.

[21] Balovnev, O., Breunig, M., Cremers, A.B., *From GeoStore to GeoToolKit: The Second Step.* Lecture Notes in Computer Science *No. 1262*, Springer, Heidelberg, 223-237, 1997.

[22] Breunig, M., Cremers, A.B., Müller, W., Siebeck, J., *Examination of Database Supported Spatio-Temporal Intersection Queries.* Proceedings of the 6th Intern. AGILE Conference on Geographic Information Science, Palma de Mallorca, Spain, April 25th -27th, 195-204, 2002.

[23] Mallet, J.-L., *GOCAD: A Computer Aided Design Program for Geological Applications.* Turner, A.K. (Ed.), Three-Dimensional Modeling with Geoscientific Information Systems, NATO ASI 354, Kluwer Academic Publishers, Dordrecht, 123-142, 1992.