REPRESENTATION OF A 3-D CITY MODEL IN SPATIAL OBJECT-RELATIONAL DATABASES

G. Gröger [a,*], M. Reuter [b], L. Plümer [a]

[a] Institute for Cartography and Geoinformation, University of Bonn, Meckenheimer Allee 172, 53115 Bonn, Germany - (groeger, pluemer)@ikg.uni-bonn.de
[b] AED-SICAD, Mallwitzstr. 1-3, 53177 Bonn, Germany - reuter@aed-sicad.de

Commission IV, WG IV/1

KEY WORDS: Databases, GIS, Modeling, Three-dimensional, Performance, Query, Data Structures

ABSTRACT:

Three-dimensional city models become more and more important in many GIS applications. Examples are, apart from simple visualization, city and land use planning as well as telecommunication planning and disaster management. Currently, there are no 3-D GIS available, which are suitable for these applications. For managing large spatial data sets in an efficient and sustainable way, spatial databases are suitable, but in general restricted to two dimensions. The paper answers the questions how to represent a 3-D city model in an object-relational spatial database, and to what degree the modeling, analysis and query mechanism of the database can be used for 3-D models and applications. Our geometrical-topological 3-D city model is based on existing standards of the ISO and the Open GIS Consortium to ensure interoperability with other systems and data providers. Recent 'object-relational' databases support sophisticated object models closing the gap between relational databases and object-oriented models. Furthermore, spatial extensions are available for database systems. Based on the widespread commercial database system Oracle 9.2i Spatial, the functionality and efficiency of analysis, access and queries is examined. In our paper we will demonstrate which kind of 3-D spatial queries are feasible using Oracle Spatial, and how exact the query results are with respect to geometry, especially when using three-dimensional spatial indices. Likewise, the system's performance is considered by means of miscellaneous benchmarks and their dependency on individual factors such as the recursive aggregation of objects.

1. INTRODUCTION

Three-dimensional city models are important in many applications of geographic information systems (GIS). Examples are telecommunications planning, disaster management, or urban planning (Köninger & Bartel; 1998, Zlatanova, 2000; Zlatanova & Holweg, 2004). On one hand, most GIS currently available, however, cope only with two or two and a half dimensional data. Most systems from Computer Aided Design (CAD) or Computer Graphics (Foley et al., 1995) can handle 3-D data, but are limited since they do not handle topology and semantic properties adequately and do not offer the GIS functionality required for the applications mentioned.

Relational databases are, on the other hand, suitable for storing and managing data in an efficient and sustainable way (Ullman, 1988). Transaction mechanisms enable consistent updates of the database, and powerful access structures guarantee efficient execution of queries. Object-relational extensions close the gap between conceptual models and their implementation in a database. Add-ons to handle spatial data are available for the most commercial databases, for example the *Spatial Extender* for IBM's DB/2, *PostGIS* for PostgreSQL, *Spatial Data Blade* for Informix, or *Spatial* for Oracle. Most of these extensions are restricted to 2-D or 2.5-D data and offer only a few functionalities to handle 3-D data.

This paper addresses the question how to store, manage and query 3-D city models in object-relational spatial databases and to which degree the required 3-D functionalities are supported. In addition, the performance of 3-D queries is analyzed. The focus is on a widespread commercial system, Oracle 9i Spatial (Oracle, 2002a; Oracle, 2002b).

The 3-D city model on which the implementation is based (Kolbe & Gröger, 2003) is a multifunctional model, which may be used for analysis and simulation purposes as well as for visualization. To reach this objective, the model has explicit topologic relations between its geometric components, and a hierarchical structure to model aggregated thematic objects recursively. To support interoperability, it is based on international GIS standards, for example ISO 19107 'Spatial Schema' (Herring, 2001). It can easily be interchanged using the *Geography Markup Language* (*GML 3*) (Cox et al, 2003), which will be one of the most important GIS transfer formats in the future.

The implemented 3-D city model was developed based on discussions within the "Special Interest Group 3D" (SIG 3D) of the initiative "Spatial Data Infrastructure North Rhine-Westphalia" (GDI NRW). In this group, municipalities, scientists and software developers cooperate to develop a unified approach for 3-D city models.

In the last decade, the suitability of databases for 3-D GIS models has been studied several times. (Molenaar, 1992; Rikkers et al., 1994) employ a database to implement a 3-D formal data structure. A pure relational model without spatial extensions is used. A similar approach is the prototype SOMAS (Pfund, 2002), which focuses on the thematic aspects and, in particular, 3-D city models. In contrast to our approach, recursive aggregates are not considered in both prototypes. (Arens et al., 2003) analyze the suitability of Oracle Spatial 9i

and propose the extension of this database by a 3-D primitive. Its topology, however, is only internal; there are no topological relations between primitives. (Stoter & van Oosterom, 2002) propose an implementation which is similar to ours, but they consider two different models: one using oracle spatial data types for Geometry, and another with topological relations. Our approach combines both in a single model. In addition, (Stoter & van Oosterom, 2002) do not deal with recursive aggregates.

This paper is organized as follows: In the second section, the spatial and object-relational properties of the database system Oracle 9i are discussed. The representation of a 3-D city model in this spatial object-relational database is the topic of the third section. The next section discusses how this model may be queried and to what extend these queries are suitable to consider the third dimension, including performance issues. The paper ends with some concluding remarks and a discussion of open questions.

2. SPATIAL OBJECT-RELATIONAL DATABASES - ORACLE 9I

Oracle 9i is a sophisticated, widespread commercial database system, which provides a spatial extension, called *Oracle Spatial*, and object-relational properties. Both are discussed in this section, which is based on (Oracle, 2002a) and (Oracle, 2002b).

2.1 Geometric properties

Oracle Spatial provides a data type, called SDO_GEOMETRY, for representing spatial data, and associated operators and functions, which allow storing, editing, updating and querying these data. Two mechanisms for indexing spatial data are used, *Quadtrees* and *R-trees* (Guttman, 1984). For querying 3-D data by Oracle Spatial, however, only R-trees may be employed.

The geometry types supported by Oracle Spatial are based on the 'OGC Simple Features Specification for SQL' issued by the Open GIS Consortium (Open GIS Consortium, 1999). According to this standard, a geometry may be a point or a multi point, i.e. a point cluster, a line string or a multi line string, a polygon or a multi polygon. In addition to the Simple Feature specification, arc line strings, arc polygons, and collections of arbitrary geometries are offered. A geometry is defined in a spatial coordinate reference system. According to the Simple Features specification, the explicit representation of topological relations between geometric objects is not provided. The coordinates of the geometric objects may be two or three dimensional, thus allowing polygons and lines positioned arbitrarily in 3-D space. The polygon boundaries may be non-planar, but the specification of interpolation rules for their interiors is not provided. For the representation of 3-D solids according to the Boundary Representation (B-Rep) (Foley et al., 1995; Mäntylä, 1988), no data type is offered by Oracle Spatial.

A database table may contain more than one column of type SDO_GEOMETRY. Thus it is possible, for example, to assign different levels of detail to a single object. This property is crucial for managing 3-D city models.

For manipulating and querying spatial data, two mechanisms are offered, which differ in particular in their 3-D properties: operators and functions.

Operators retrieve spatial data from the database according to geometrical criteria, using the R-tree index. One specific operator, called SDO_FILTER, selects geometries, which interact with a given fixed geometry, or pairs of geometries, which interact pair-wise. The first case is called *window-query*, while the second is a *join-query*. The operator does not consider the exact geometry of objects, but approximates it by a minimal bounding rectangle or a minimal bounding box, depending on the dimension of the geometries. The rectangles and boxes are parallel to the x-, y-, and z-axis of the coordinate reference system.

The approximation of geometries by bounding boxes, however, yields inexact results. Consider, for example, the two geometries in Figure 1b), which are disjoint. The bounding boxes overlap, thus the operator SDO_FILTER recognizes that both are not disjoint. In Figure 1a), the operator SDO_FILTER is able to identify the two geometries' bounding boxes as disjoint.

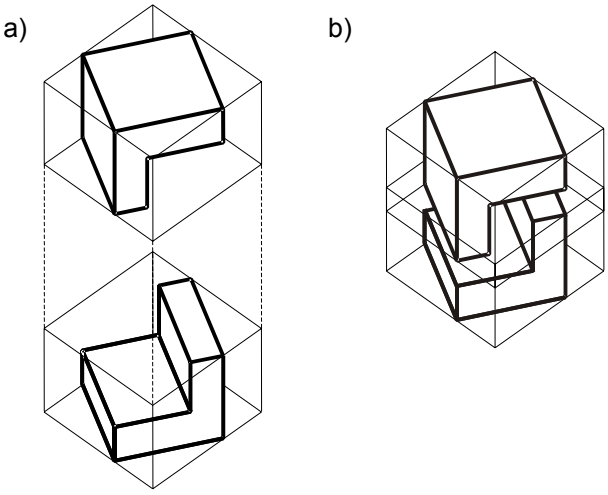

Figure 1: Approximation of geometries by minimal bounding boxes, yielding inexact results. In Figure a), both objects can be distinguished from each other, while in b), the two geometries do not touch each other, but their bounding boxes do so.

The other operators apart from SDO_FILTER select geometries within a given distance, nearest neighbor geometries, or geometries with topological relations according to the well-known 4-intersection model (Egenhofer & Herring, 1990). These operators are evaluated using a so-called 'two-tier model', which applies SDO_FILTER first, and the more exact operator to the result afterwards. All operators apart from SDO_FILTER may not be applied to data with more than two dimensions, and thus are not discussed any further.

In contrast to operators, *functions* do not use a filter step and a spatial index, and are applicable to 3-D data, but they ignore the z-coordinate. Oracle Spatial provides functions to select geometries according to the 4-intersection model, to compute areas, distances, or to construct convex hulls, centroids, buffers, and so on. In addition, the union, difference or intersection of a pair of geometries may be derived.

Figure 2 depicts the different spatial relations distinguished by the function SDO_GEOM.RELATE, which implements the 4-

intersection model. Note that the z-coordinate is ignored by the function. If, for example, one geometry A is above a geometry B and both are disjoint, then the operator SDO_GEOM. RELATE recognizes erroneously that A and B are not disjoint.

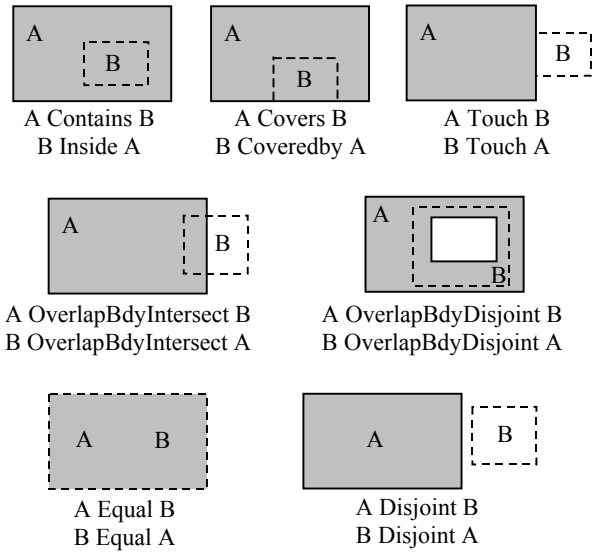

Figure 2: Topological relations differentiated by the function SDO_GEOM.RELATE, which implements the 4-intersection model (Oracle, 2002b).

Although functions do not use the R-tree index, both concepts may be combined in a 3-D query. The operator SDO_FILTER is employed first to pre-select a superset of the desired geometries in an efficient way, using the R-tree index and bounding boxes. This set is further refined by a 2-D function afterwards. In this way, a two-tier query model may be realized. An example of such a query will be given in section 4.1.

2.2 Object-relational properties

The database Oracle offers an object-relational extension of the relational data model, closing the gap between the object-oriented conceptual model, which often is formulated using the *Unified Modeling Language UML* (Booch et al., 1997), and the database. Object-relational features in Oracle are object types and object views, which specify attributes and methods. They are implemented using one or more object tables, being similar to relational tables. Object methods, which can be used to manipulate the data, are stored with the table definition. The concept of inheritance allows the specification of super- and subtypes, which will be mapped on relational tables by the database management system. An object table may have references to other object types, allowing navigational access and replacing the relational concept of foreign keys.

In the relational data model, the columns of tables are restricted to contain simple data types. In contrast, the columns object tables may be whole objects or whole tables, which are called nested tables. *Nested tables* provide a convenient means to represent m:n-relations between object tables. In relational tables, the representation of these relations requires an additional table, too. But a nested table is embedded into an object table, simplifying the handling of the relation. An example for a nested table embedded in an object table is depicted in Figure 3. The first row of the object table has relations to three objects, which are identified by A11, A12 and A13.

The efficiency of nested tables may be improved by using indices. The database arranges the order of the rows according to the order of the corresponding rows in the superior object table.

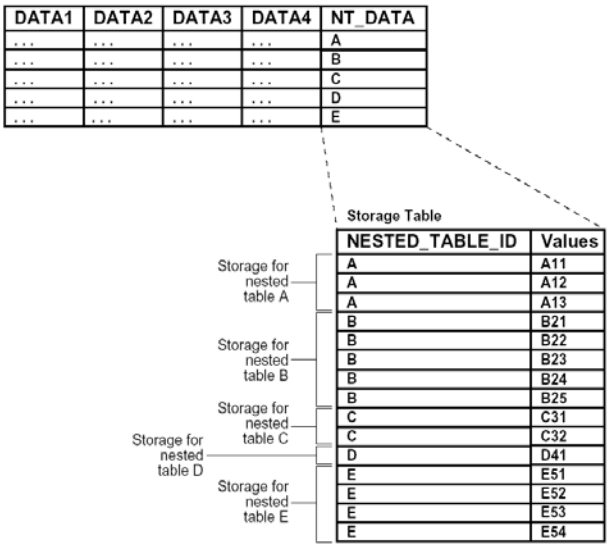

Figure 3: Example for a nested table, embedded in an object table. Each row of this object table corresponds to several rows in the nested table, representing a m:n-relation The figure is taken from (Oracle, 2002a).

3. REPRESENTING 3-D CITY MODELS

This section describes how the multi-functional 3-D city model presented in (Kolbe & Gröger, 2003) was implemented in Oracle 9i Spatial and which difficulties occurred due to some deficiencies of this database. First, the geometry data types of Oracle Spatial do not provide explicit topological relations between geometric components, which are crucial for representing 3-D city models. Thus topology had to be modeled using standard object-relational, non-spatial tables. To get benefit from the efficient spatial index structure, the spatial data types are used in addition, obtaining a double representation of spatial properties, which is redundant to a certain degree. In the following, the topological level of the database schema is discussed first, the aggregation level afterwards, and finally the representation of the geometry using the Spatial extension of Oracle. The complete database scheme is depicted in Figure 4.

At the bottom of Figure 4, the primitives vertex, edge, face and solid as well as their topological relations are represented, according to the well-known Boundary Representation (Foley et al., 1995; Herring, 2001). Each primitive is stored in a single table. Rings are employed to differentiate outer boundaries from inner boundaries of faces, i.e. holes. The various m:n-relations between the topological primitives are implemented by the object-relational concept of nested tables. For example, a solid is bounded by several faces. Thus one column of the solid table is a nested table called *FaceListNestedTab*, each row of which contains a reference to a bounding face of the solid. In a similar way, the relations between a face and its interior rings

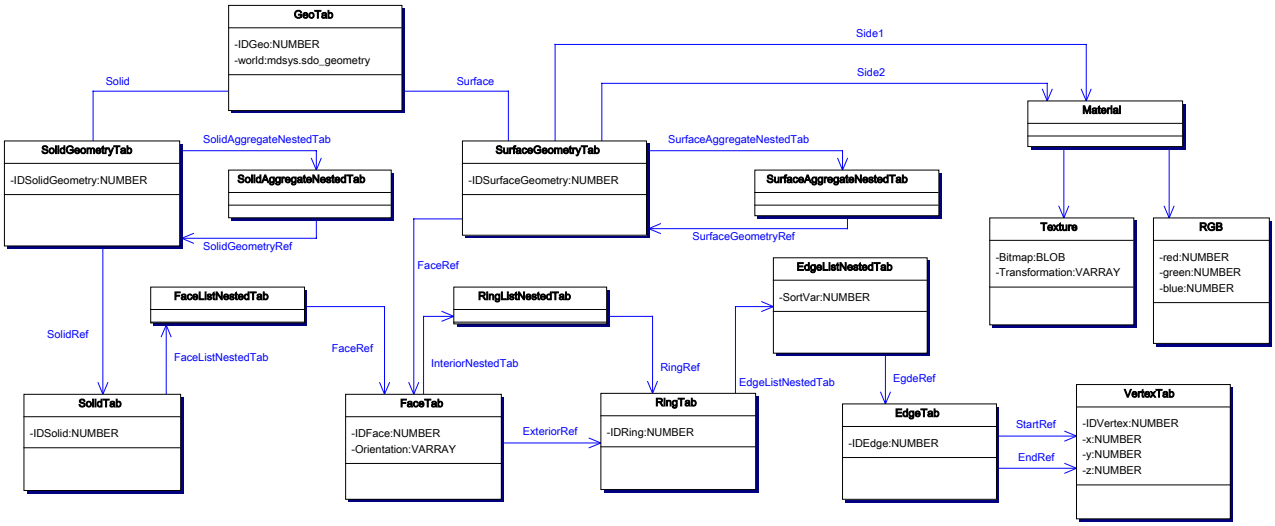

Figure 4. Object-relational database schema for the 3-D city model

and between a ring and its edges are realized, using the nested tables *RingListNestedTab* and *EdgeListNestedTab*.

Based on the topological primitives, aggregates can be build recursively. This enables the representation of arbitrarily nested building structures. An example is one university campus, which consist of several complex buildings. One complex building consists of parts, and these parts again are a composition of a main part, chimneys and balconies, and so on. An aggregate is represented by a row in the *SolidGeometry* table in Figure 4. The relations to the parts of the aggregate are defined by the nested table *SolidAggregateNestedTab*, which references several rows in the *SolidGeometry* table. Since these rows in the *SolidGeometry* table may have parts on their own, a nesting of arbitrary depth may be achieved. Finally, the *SolidGeometry* rows, which have no parts, are related to a *Solid* defining the geometry and topology of the *SolidGeometry* row.

Faces may be aggregated to *SurfaceGeometries* analogously. In particular, this is necessary to map textures on surfaces, which is very important for the visualization of 3-D city models. Attaching textures to aggregated *SurfaceGeometries* provides more flexibility than relating it to a face, since often textures correspond to whole walls covering more than a face of a single building. Textures are represented by a row in the *Material* table, which may alternatively be just a color value when no texture is available.

On the top level of the database schema in Figure 4, the *GeoTab* table represents the geometry of objects by a value of Oracle Spatial's SDO_GEOMETRY type, which was already discussed in section 2. A row in the *GeoTab* table may be related to a *SolidGeometry* or to a *SurfaceGeometry*, but not to both. In both cases, the SDO_GEOMETRY value is a collection of polygons. It must form a closed solid, if it is related to a *SolidGeometry*. The database, however, provides no standard mechanisms to check this property.

Using a collection of polygons to represent a solid is only an approximation. This is due to the fact, that Oracle Spatial does not offer geometry types for solids. The semantics of a solid is different to the semantics of a collection of polygons forming a closed solid. For example, the question whether a point is completely inside a solid may be answered by a solid model, but not by a polygon approximation of a solid. In such a polygon model, the notions of 'inside' and 'outside' are not defined.

Note that a solid may not be approximated by a multi-polygon. According to the 'Simple Feature Specification' (Open GIS Consortium, 1999), two polygons being part of one multi polygon may touch only in a finite number of points. In a solid boundary, two polygons meet in a common edge and thus touch in an infinite number of points.

The database schema for the 3-D city model is accompanied by a set of integrity constraints, which express relevant properties of the model explicitly, and which are important for many applications using the model. One constraint, for example, states that two solids must be disjoint and may touch at least at their boundaries. In this case, the area where both solids touch must be a face, which is contained in the boundary of both solids. More details about integrity constraints may be found in (Kolbe & Gröger, 2003).

4. QUERYING 3-D CITY MODELS

Based on the analysis of the 3-D query capabilities in the second section, now it is discussed how the 3-D city model presented in the last section may be queried.

4.1 3-D Queries

As discussed in the second section, Oracle Spatial's operators apart from SDO_FILTER are not applicable to 3-D data, while functions neglect the z-coordinate and treat them as zero, respectively. To obtain a efficient two-tier query model, the SDO_FILTER operator, which is suitable for 3-D bounding boxes, may be combined with 2-D functions, enabling a few standard applications for 3D city models. This combination is achieved by a nested SQL command; SQL is the standard query language for relational databases (Ullman, 1988).

An example for a nested query is given in Figure 5. It selects the identifiers of those objects, which are related to solids having the relation 'inside' (see Figure 2) to the bounding rectangle or window given by the two coordinate pairs (3446733.79, 5549996.27) and (3445133.79, 5539196.27). In the nested select-from-where-statement, which is included in the from-part of the outer query, the SDO_FILTER operator is applied. It selects all geometries interacting with the 3-D

bounding box given by the two coordinate triplets, using the R-tree index. The result of this sub-query is passed to the outer from-part, where the exact relate function is applied. Note that the x- and y-coordinates are considered only. The query is illustrated in Figure 6.

```
SELECT s.IDGeo
FROM (
  SELECT r.IDGeo, r.world
  FROM GeoTab r
  WHERE MDSYS.SDO_FILTER(
    r.world, MDSYS.SDO_GEOMETRY(
      3003, NULL, NULL,
      MDSYS.SDO_ELEM_INFO_ARRAY(1,1003,3),
      MDSYS.SDO_ORDINATE_ARRAY(
        3446733.79, 5549996.27, 142.8,
        3445133.79, 5539196.27, 104.8)
    ),'querytype=window'
  ) = 'TRUE'
) S, SolidGeometryTab v
WHERE MDSYS.SDO_GEOM.RELATE(
  S.world, 'INSIDE', MDSYS.SDO_GEOMETRY(
    3003, NULL, NULL,
    MDSYS.SDO_ELEM_INFO_ARRAY(1,1003,3),
    MDSYS.SDO_ORDINATE_ARRAY(
      3446733.79, 5549996.27, NULL,
      3445133.79, 5539196.27, NULL)
)
  ),0.5
) = 'TRUE' AND
v.IDSolidGeometry = S.IDGeo;
```

Figure 5. Nested SQL query, which applies the SDO_FILTER operator first and then refines the resulting set by applying an exact 2-D function, selecting those geometries inside a given rectangle.

To summarize the query, the filter operates in 3-D, but yields no exact results (see Figure 1). The relate function provides exact results regarding two dimensions, but neglects the z-coordinate.

These query capabilities offered by Oracle Spatial are sufficient for many 3-D city model applications, where the vertical relation between objects is not relevant. But other queries, in particular those which implement consistency constraints, are not supported by Oracle Spatial. For example, such a constraint states that solids must be pair-wise disjoint. To cope with these queries, special functions must be implemented, which analyze the result of the filter operator exactly and includes the third dimension.

4.2 Performance

Due to the R-tree index, 3-D spatial queries using the filter operator are very efficient. As an example, the diagram in Figure 7 shows the query time of the query given in Figure 5. The database contains 9934 buildings. The query time is depicted for several queries, differing in the number of buildings in the result set. For a small result set, the time is less than a second. This analysis was performed using an AMD Athlon XP 2100+ PC, with 1.800 GHz and 512 MB DDR-RAM. The operating system was Windows XP.

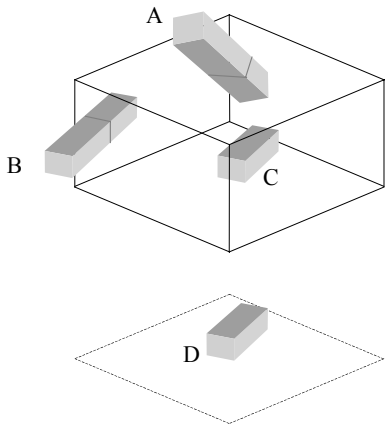

Figure 6. Example for an application of the query in Figure 5. The objects A, B and C interact with the bounding box and pass the filter, while D does not. The projections of A and C onto the x-/y-plane are inside the projection of the bounding box, thus both constitute the final result set. Note that only C is inside the bounding box.

The query time increases significantly, when the result set grows. But the most important observation is, that the query time does not increase significantly when the database size grows. This scalability is an important property of spatial databases.

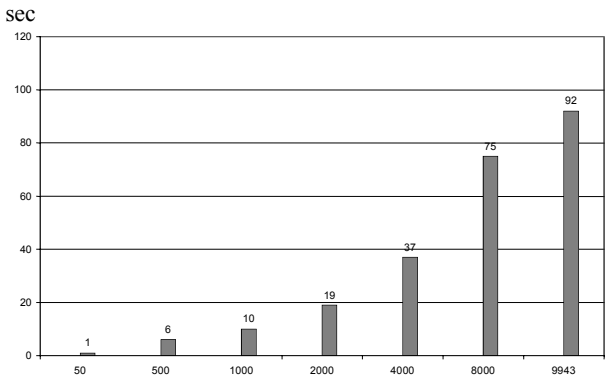

Figure 7. Query time of the spatial window filter in seconds for selected numbers of buildings in the result set. The database contains 9934 buildings.

5. CONCLUSIONS AND FURTHER WORK

This paper presents an approach to store, manage and query 3-D city models using a spatial object-relational database. Thus applications can benefit from the efficiency, consistency and sustainability of databases. Topology is represented explicitly, providing efficient access to neighboring objects and geometric consistency. In addition, the build-in functionality of the spatial extension of the database enables efficient geometric access to large 3-D data sets.

Some deficiencies of the database regarding its 3-D capabilities were identified. Topology is not modeled explicitly by the spatial data types, resulting in a redundant representation of

spatial properties. Furthermore, a spatial data type for the specification of solids is missing. Spatial queries are restricted to two dimensions, apart from the efficient filter operator, which considers three dimensions. These 3-D queries are sufficient for many applications of 3-D city models, but too restricted for other relevant queries, for example for checking the consistency of the model.

The next step is the extension of the model by a variety of thematic objects relevant for 3D city models, including the corresponding attributes as well as aggregation and generalization hierarchies. Efficient visualization and analyses will be obtained by allowing multiple representations of a single thematic object in different levels of detail (Kolbe & Gröger 2003). A further extension will be the integration of the relief structure using Triangulated Irregular Networks.

The long-term objective of the database is to extend it to a 3-D GIS prototype, which is a platform for various 3-D research projects. In the context of spatial data infrastructures, the database will provide services to get access to data in an interoperable way. An example is the provision of GML 3 data, thus extending OGC's Web Feature Service (Open GIS Consortium, 2002), which is limited to 2-D data currently.

REFERENCES

Arens, C., Stoter, J. & van Oosterom, P.J.M., 2003. Modelling 3D spatial objects in a GEO-DBMS using a 3D primitive, *AGILE conference*, April 2003, Lyon, France.

Booch, G., Rumbaugh, J. & Jacobson, I., 1997: *Unified Modeling Language User Guide*. Addison-Wesley.

Cox, S., Daisy, P., Lake, R., Portele, C. & Whiteside, A., 2003: OpenGIS Geography Markup Language (GML3), Implementation Specification Version 3.00, OGC Doc. No. 02-023r4.

Egenhofer, M.J. & Herring, J.R., 1990. Categorizing Binary Topological Relations Between Regions, Lines, and Points in Geographic Databases. Technical report, Department of Surveying Engineering, University of Maine.

Foley, J., van Dam, A., Feiner, S. & Hughes, J., 1995: *Computer Graphics: Principles and Practice*. Addison Wesley, 2nd Ed.

Guttman, A., 1984. R-trees: a dynamic index structure for spatial searching. *SIGMOD Record* (ACM Special Interest Group on Management of Data), 14(2):47-57(1984).

Herring, J., 2001: The OpenGIS Abstract Specification, Topic 1: Feature Geometry (ISO 19107 Spatial Schema), Version 5. OGC Document Number 01-101.

Kolbe, T. H. & Gröger, G., 2003: Towards Unified 3D-City-Models. In: *Proc. of. ISPRS Commission IV Joint Workshop on Challenges in Geospatial Analysis, Integration and Visualization II*, September 8 - 9, Stuttgart, Germany.

Köninger, A. & Bartel, S., 1998: 3D-GIS for Urban Purposes, *Geoinformatica*, 2(1):79-103(1998).

Mäntylä, M., 1988, *An Introduction to Solid Modeling*, Rockville, Maryland: Computer Science Press.

Molenaar, M., 1992: A topology for 3D vector maps. ITC Journal 1992-1, The International Institute for Aerospace Survey and Earth Sciences, The Netherlands.

Open GIS Consortium, 1999. OpenGIS Simple Features Specification for SQL, Revision 1.1, OpenGIS Doc. No. 99-049.

OpenGIS Consortium, 2002. Web Feature Service Implementation Specification. Version: 1.0.0, OpenGIS Doc. No. 02-058

Oracle, 2002a. Oracle 9i Application Developer's Guide – Object-relational Features, Release 2 (9.2).

Oracle, 2002b. Oracle Spatial – User's Guide and Reference, Release 9.2.

Pfund, M., 2002. 3D GIS Architecture, A Topological Data Structure, *GIM International*, Vol 16. Feb. 2002.

Reuter, M., 2003. Implementierung eines 3D-Stadtmodells in einer objekt-relationalen Datenbank am Beispiel von Oracle Spatial (Implementation of a 3-D city model in a object-relational database, considering Oracle Spatial as example). Diploma Thesis. Institute for Cartography and Geoinformation, University of Bonn, Germany (in German).

Rikkers, R., Molenaar, M. & Stuiver, J., 1994. A query orientated implementation of a topologic data structure for 3-dimensional vector maps, *Int. J. Geographical Information Systems*, 8(3):243-260(1994).

Stoter, J.E. & van Oosterom, P.J.M., 2002. 3D Data Modelling in a Geo-DBMS In: *Proceedings GIScience*, Boulder, Colorado, USA.

Ullman, J.D., 1988: *Principles of Database and Knowledge-Base Systems*, Vol. 1, Computer Science Press.

Zlatanova, S., 2000: *3D GIS for Urban Development*. PhD Thesis, ITC Dissertation Series No. 69, The International Institute for Aerospace Survey and Earth Sciences, The Netherlands.

Zlatanova, S. & Holweg, D., 2004: 3D Geo-information in emergency response: a framework. In: *Proceedings of the Fourth International Symposium on Mobile Mapping Technology (MMT'2004)*, March 29-31, Kunming, China.

ACKNOWLEDGEMENTS

We thank Thomas H. Kolbe, Viktor Stroh and Ingo Petzold for discussions, which helped to generate and clarify the ideas described here.